Discovery of Delay-Tolerant Networking Endpoint Elements

Alex McMahon

July 26, 2013

Declaration

Alex McMahon

A succinct summary of the methods used in this dissertation follows:

1. We review the main causes and consequences of a node's inability to exploit an opportunistic contact in a broad range of contexts, in order to derive a set of protocol and context specific requirements
2. Basing our review on the requirements, we analyse in detail how they are fulfilled by standard discovery protocols.
3. Basing our design on the analysis, we describe our hypotheses, which is captured in the definitions of an application discovery protocol and a set of conceptual models. We then describe our discovery protocol system architecture, which is captured in a set of modules developed specifically in order to test the DTN discovery concept. Next we define the evaluation system architecture, which is captured in simulation and analysis modules.
4. Basing our implementation on the design, we implement the system architecture modules.
5. Basing our evaluation on the requirements and the 2009, 2010 and 2011 N4C trial data, we describe a set of context specific tests and results that use the modules in order to demonstrate:
   (a) Our discovery approach is viable.
   (b) In some cases our approach, can be used to discover facts to exploit a contact, in order to transfer more ADUs than a node using facts discovered by other approaches.
6. Finally, we consider how well our approach fulfils the set of context specific requirements.

In summary, this dissertation makes the following contributions to the area of DTN:

1. TDP, which as part of an approach can be used by DTN nodes to learn their endpoint elements and attributes
2. A set of conceptual models as part of an approach to organise endpoint elements and attributes to provide a (delay-tolerant) network of associations.
3. The architectural constraints, exemplified by the TDP system and evaluation system architecture modules, that, when applied as a whole, emphasise scalability of loosely coupled systems.
4. The evaluation of how TDP can exploit an opportunistic contact in a broad range of contexts for the Internet.

Acknowledgements

I dedicate this thesis to my father, Michael McMahon who is always in my thoughts and heart. Thanks to my wife Rachel, my son James, my mother Patricia and my brothers Larry and Bobby and their wives Dee and Laura for making life great and always being there. Thanks to Dr. Stephen Farrell, Dr. Stefan Weber and Dr. Kevin Fall for your friendship, sharing your knowledge, criticisms, and helping me find my way.

Thanks to everyone in DSG and the DTNRG for your support.

Alex McMahon

University of Dublin, Trinity College

July 2013

Abstract

In this thesis, we study discovery problems in heterogeneous networks. Internet applications generally involve contemporaneous low-latency connectivity oriented around hosts and addresses. The discovery standards, protocols and models used by these applications are focused on near real-time, low overhead, discovery of network information. These approaches are unsuitable as a generic solution to discover information that can be used to drive the process of message delivery in heterogeneous networks.

Delay tolerant networking (DTN) protocols support communications that are subject to link asymmetry, long propagation delays, disruption and packet loss. Anything used in DTN can be named in order to support heterogeneity, a major tenet of DTN. The DTN architecture uses endpoints, identifiers, and a wide variety of existing communications protocols for the opportunistic exchange of information.

There are several limitations of existing discovery approaches. Firstly, they do not have the scalability of loosely coupled systems. The majority of existing approaches feature discovery messages that are tightly coupled with underlying transport mechanisms. Such coupling means the structure of the information is specific to the infrastructure of the discovery mechanism. Furthermore, they mostly discover a subset of facts about a specific context. This might be reasonable in the constrained and secure environments in which web resources are typically deployed, but not for heterogeneous networks where it is necessary to decouple information from its producer in order to exchange it between different consumers or producers. We require the scalability of loosely coupled systems.

Secondly, they do not provide a generic solution. Existing approaches deal with context-specific aspects of discovery. They do not provide generically applicable information about structural and functional components of a system to work with network elements. Furthermore, their discovery messages are inflexible in terms of the discovery data and data structure. We require a generic solution.

Thirdly, existing approaches are based on a large number of assumptions. They assume how the discovery information is to be used, or configuration prior to conducting discovery. Approaches sometimes assume operator intervention, synchronous clocks or the ability to make a request. However, there may not be an operator or mechanisms required to maintain synchronous clocks, or align slot boundaries. Nodes might not have a complete knowledge of their context and might not be able to reach a consensus on how to behave, or even to request information. We need to an approach that makes as few assumptions as possible.

Finally, they do not offer explicit support for the discovery of network elements, i.e., offer laws that network elements can use to describe themselves for the purpose of discovery. Learning the presence of peers is not sufficient to facilitate informed decision making. Higher layer discovery is needed to correlate network elements. We need models capable of representing any network element to structure data, provide a portable representation for their transfer, and describe how to persist them to storage.

In this thesis we propose the Thing Discovery Protocol (TDP) and the TDP conceptual models that provide a node with the information to exploit an opportunity to communicate, in order to transfer ADUs. The hypothesis of our work is that a node can use our models to exploit a contact, in order to transfer more ADUs than a node using other models. The design of models to organise endpoint elements and attributes, in order to provide a heterogeneous network of associations, is the substantive contribution of this thesis to the area of DTN.

TDP can be used by DTN nodes to report their endpoint elements, attributes and identifiers. TDP, along with the TDP conceptual models, defines the methods for a DTN node to learn a description of the classes and relationships of its elements. DTN TDP things are associations of names with any element of a DTN endpoint that can be verified directly.

To evaluate our hypothesis, we have implemented TDP. Our evaluation demonstrated cases where nodes can use information provided by TDP to exploit contacts of the 2009, 2010 and 2011 Networking for Communications Challenged Communities (N4C) trial data, in order to transfer ADUs. It shows cases where in the event of a contact, data transfer applications, exemplified by “TDP-decider”, can transfer more distinct ADUs than applications that use existing discovery protocols.

Published Material

Publications related to this Ph.D.

- Delay- and Disruption-Tolerant Networking (McMahon and Farrell, 2009, Internet Computing, IEEE 13(6), pages. 82-87)
- The Endpoint Discovery Protocol (EDP) (McMahon and Fall, 2010, Internet Draft, `http://tools.ietf.org/id/draft-mcmahon-dtnrg-dtn-edp-00.html` (Accessed 04/04/2013))
- Report on dtn applications during arctic summer 2010 trial (McMahon, Farrell, Weber, Hartnett, and Lynch, 2011, International Workshop on Opportunistic and Delay/disruption-Tolerant Networking in conjunction with the 14th International Symposium on Wireless Personal Multimedia Communications (WODTN 2011 (WPMC 2011 Workshop)))
- DTN Trials and Router Updates (McMahon, Farrell, Lynch, Weber, and Hartnett, 2011, ExtremeCom 2011, `http://www.cl.cam.ac.uk/~fb375/extremecom/2011/program.html` (Accessed 04/04/2013))
- N4C DTN Router Node: 2009 Results, 2010 Plans (McMahon, Farrell, Weber, Meehan, and Hartnett, 2010, ExtremeCom 2010, `http://www.cl.cam.ac.uk/~fb375/extremecom/2010/FarrellExtremeCom10.pdf` (Accessed 04/04/2013))
- Report on an Arctic Summer DTN Trial in 2009 (McMahon, Farrell, Meehan, Weber, and Hartnett, 2011, Wirel. Netw. 17(5), pages 1127-1156)
- An N4C DTN Router Node Design (McMahon, Farrell, Weber, Meehan, and Hartnett, 2009, ExtremeCom 2009, `http://www.cl.cam.ac.uk/~fb375/extremecom/2009/extremecom2009_Farrell.pdf` (Accessed 04/04/2013))

Contents

Chapter 1

Introduction

"Die Welt ist die Gesamtheit der Tatsachen, nicht der Dinge" (Wittgenstein, 1922).

The translation of the rationale observed by Wittgenstein (1922) is "the world is the totality of facts, not of things"(Sullivan, 2000). In this thesis, we study approaches used to discover facts to provide a node with the information to exploit an opportunistic contact in order to transfer application data units (ADUs). Our hypothesis is that a node can use our approach to discover facts to exploit a contact, in order to transfer more ADUs than a node using facts discovered by other approaches. Our approach is captured in the definitions of a set of conceptual models, modules, and the Thing Discovery Protocol (TDP), a Delay-Tolerant Networking (DTN) (Cerf et al., 2007) application discovery protocol. We validate our hypothesis using our approach that was developed specifically to test it.

In this chapter we first describe the motivations for our work. Then we describe its key concepts and give background information on the challenges of communications in heterogeneous networks. Following this we give a brief overview of existing work in the area of heterogeneous networks in which naming and tolerance are a primary areas of focus. Next we outline our approach, goal, contributions, and scope and describe the layout of the document. Finally, we present a summary of the chapter.

1.1 Motivations

Here we investigate how to exploit less predictable communications opportunities. We also consider how mechanisms related to handling less predictable links relate to other, especially routing, protocols at other layers. We use the term context to describe the environment, situation, state, surroundings, task, and other characteristics as described by Schmidt et al. (1999). Discovery of the facts of network elements and their relationships can be used by a node to make intelligent routing decisions, trigger message forwarding or make effective scheduling decisions. Routing schemes and data transfer applications can use the information provided by application discovery protocols. A node that can consume and extract information, such as the duty schedules of it's peers, can use it's mobility in order to transfer a message. This thesis describes such discovery protocols that support the use of Internet and opportunistic exchange.

1.2 Background information

This section first defines our concept of association and then describes our notion of a thing. Then we provide an overview of our structure of information, and describe our concepts of network, node, identifier and opportunistic contact.

1.2.1 Associations

Our concept of an identifier is based on the Oxford English Dictionary definition of identifier (Pro, 2010b):

"a sequence of characters used to identify or refer to a program or an element, such as a variable or a set of data, within it."

The notion of identity as an instrument used by an entity in order to provide information about itself features in much of the current literature (Torres et al., 2012). In contrast, we use semantic free identifiers, which consequently, cannot be used to provide information. As the definition is particular to computing we simplify it by removing the notion of program, variable and data. Additionally, a set of data is too restrictive as an identifier for a null set may be required. As the definition also has a self referential nature we replace the notion of being "used to identify", with the state of being associated for which we adopt the Oxford English Dictionary definition of associated (Pro, 2010a):

"(of a person or thing) connected with something else."

So our definition of identifier is:

"a sequence of characters associated with a set of zero or more elements."

In the following, the term thing is used instead of resource, entity, object or element so that the simple principle is not obscured by language. Our intention is not to provide an analysis of the nature of the things being described by the language, or to suggest any particular processing model. We adopt the Oxford English Dictionary definition of thing:

"an object that one need not, cannot, or does not wish to give a specific name to."

So by our definition a thing needn't be named. Nevertheless, when it is, a name is associated with the thing, that is, we create an association.

1.2.2 Structure of information

A named thing is an association represented as a set of zero or more elements that is identified by at least one universally unique identifier. Our data model uses thing identifiers to determine the facts about a thing such as what, or who a thing is associated with (see figure 1.1).

The Association lists (URL, 1) of LISP are used to record a mapping from keys to values. We adopt the Entity Attribute Value (EAV) structure, which was first used in the form of Association lists (Winston, 1984) to base our definition of a thing data model (TDM) which we formulate as:

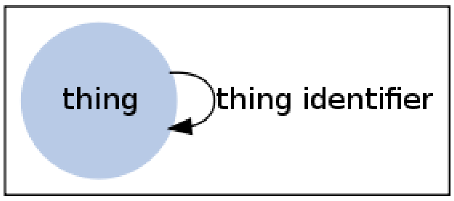


Figure 1.1: A named *thing* is an association

"a set of tuples $\{(T), (A), (V)\}$ where T and A are identifiable by at least one universally unique identifier and T is the set of zero or more elements, A is the set of zero or more element attributes and V is the set of zero or more attribute values."

A thing attribute can be associated with one or more attribute identifiers, which can be associated with one or more thing identifiers. The value associated with an attribute of a thing (see figure 1.2) is a TDM value.

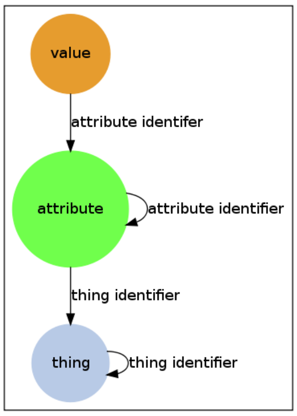


Figure 1.2: Value associated with an attribute of a thing

Every element of a network can be associated with a universally unique name, denoted a thing identifier. Any association can have a thing identifier.

1.2.3 Contact

A node can be represented by set of zero or more identifiable network elements. Elements can be physical or virtual. We represent a node as a set of thing identifiers, identified by at least one thing identifier (see figure 1.3). Consider a network composed of V potential mobile nodes populating a rectangle s. Movement of potential nodes is restricted within s (closed population). The opportunity to communicate is denoted a contact as in Fall (2003), which is parametrised by a source, destination, direction, time, capacity, and a propagation delay. Propagation delay is assumed to be constant during the duration of the contact. For a node to exploit an contact it needs to have facts about elements of itself and the peer node(s). For this purpose, a node can use data provided by lower and higher layers.

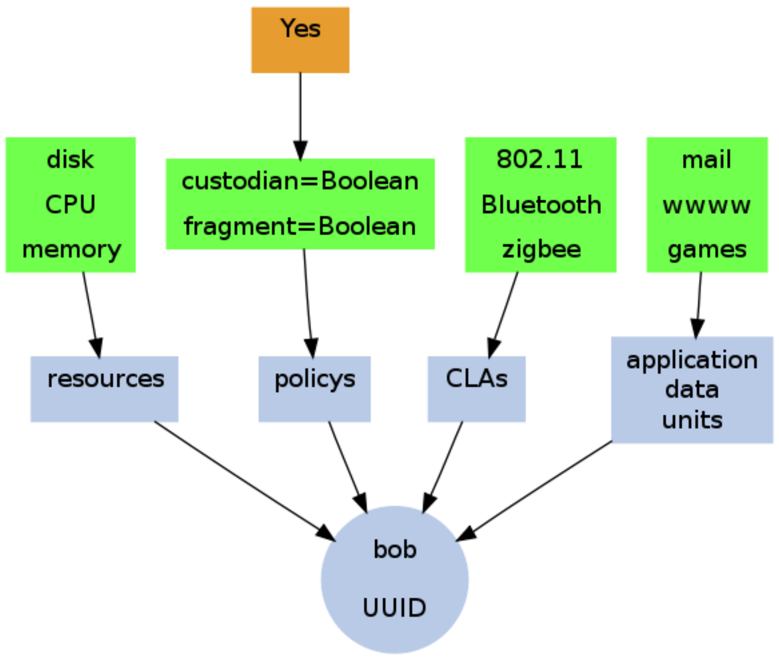


Figure 1.3: A node *bob* identifiable by at least one universally unique identifier (*UUID*). A node is a set of zero or more identifiers

.

1.3 Context

In this section we provide some background information about the evolution of Internet applications and how new problems have necessitated new Internet protocols. Next, we outline the characteristics of systems that limit their application domain. Finally, we give a brief overview of existing work in the area of heterogeneous networks in which naming is a primary area of focus.

1.3.1 Evolution of Internet applications

The "Internet and its architecture have evolved from modest beginnings"(Carpenter, 1996). Internet applications generally involve contemporaneous low-latency connectivity orientated around hosts and addresses. The World Wide Web has been iteratively developed through "a series of modifications to the standards that define its architecture"(Fielding, 2000). However, use of the Internet is predominantly oriented around data and services, and it is often of little consequence from which host, or by what protocol, the data arrives (Demmer et al., 2007).

The Internet Architecture faces challenges in order to scale to a greater number of users (Atkinson et al., 2010). Abstraction is a property that has contributed to scalability of structures, for example, the "layered, hierarchical, and federated structure of the Internet"(Cerf, 2013). The scalability provided by using names has been demonstrated in previous work (Shaw, 1990; Hodges and Morgan, 2002; Fielding et al., 1999; Itu, 2005; Tim Bray, 2009).

Internet applications designed to handle data and services directly as associations can help to decouple data and services from the mechanisms of the underlying infrastructure that provide them. The Internet architecture already supports use of named associations, for example DNS (Mockapetris, 1987) and HTTP (Fielding et al., 1999) redirect are mechanisms are used to ensure persistence, for example to ensure persistence of an association remains valid as long as the thing with the identifier is available. However, neither work without operator intervention to ensure persistence of data moving across domains.

1.3.2 Novel applications, problems and protocols

New applications for the Internet have necessitated new protocols to support reliable access to mobile nodes. A trend towards association-based networking, whereby new applications are designed to use higher layer Application Programming Interfaces (APIs), or frameworks, instead of classic sockets is underway. How we can provide access to things and achieve persistence, availability, and authentication is a major issue. New approaches have required specification of protocols to support access to named data, mobility, multi-homing, wireless technologies and opportunistic contacts. For the most part, these approaches assume that the network infrastructure will provide discovery capabilities.

Such approaches include Content Delivery Networks (CDN), Representational State Transfer (REST), peer-to-peer (P2P) frameworks, DECoupled Application Data Enroute (DECADE), Simple Object Access Protocol (SOAP), CDN Interconnection (CDNI)(Niven-Jenkins et al., 2012), Java, JavaScript, HTTP 2.0 (HTTPBIS), and WebSocket API. Approaches have been taken to provide an API which decouples applications from the underlying communications mechanisms, such as the Service-Orientated Device Architecture (SODA) (Deugd et al., 2006), an adaptation of a service-oriented architecture (SOA) (Erl, 2004). A focus of REST, an architectural style for networked systems (Fielding, 2000), is to reduce interaction latency, enforce security, and encapsulate legacy systems through placing constraints on connector semantics instead of component semantics. The simple and scalable semantic of REST has lead to its wide adoption. Demmer et al. (2007) argue the need for a generic communications API that decouples time, space, and synchronisation between producers and consumers of information and applications from the underlying communications mechanisms. There is deployment of abstract higher layer networks that includes peer-to-peer (P2P) frameworks and Content Delivery Networks (CDN). Trends toward decentralisation of information can be observed in the use of the structured key-based routing approaches of the Freenet project (URL, 2). Additionally, research on identifier / locator split architectures has provided Distributed Hash Table (DHT), such as Chord (Stoica et al., 2001), CAN (Ratnasamy et al., 2001) and Pastry (Rowstron and Druschel, 2001), based locator / ID separation mechanisms. P2P caches provide in-network storage(Xia and Muppala, 2010). CDN achieves scalability and performance by operating dedicated caches close to access networks (Ahlgren et al., 2012). The development of CDNs and P2P networking has been attributed to the increasing demand for mass distribution and replication of large amounts of resources (Ahlgren et al., 2012). In-network caching, on-demand replication are research challenges to be addressed to bring the Information-Centric Networking (ICN) (Ahlgren et al., 2008) approach to life (Bari et al., 2012). Research into the deployment of the Internet in space and other challenged contexts has characterised the communications problems encountered, and furthermore, generalised them as delay related.

Our applications generally assume that the network infrastructure will provide contemporaneous low-latency connectivity. However, propagation delay or packet loss may have a much more serious effect on TCP traffic due to TCP backoff / slow start. We require new protocols to provide us with access to things in such challenging communications contexts.

1.3.3 A limited range of contexts

We can use applications can transfer ADUs in a limited range of contexts if they are tightly coupled with infrastructure or communications mechanisms. Systems with complex interactions and tight coupling are likely to have unforeseen failure states and less flexibility in recovering from failure states (Bush and Meyer, 2002). A wider range of technology and protocol options can be seen, for example by global use of the Simple Mail Transfer Protocol (SMTP), to translate into a wider range of real world deployments. The tight coupling of time, space, and synchronisation between producers and consumers of TDM data and the tight coupling of applications with the underlying communications mechanisms can lead to a reduced range of technology and protocol options and in some contexts mean that communications are not possible.

1.3.4 The context

The information that nodes have about their peers varies significantly over time and space. A number of network architectures for the exchange of messages between heterogeneous nodes have already been designed.

DTN is a generalisation of an architecture designed for an Interplanetary Internet (IPN). Anything used in DTN can be named in order to support “radical heterogeneity”, a major tenet of DTN (Fall, 2012). The DTN architecture can work with an ADU and it’s source or destination, as a first-class abstraction. The binding between the name and an associated thing can change. To enable a DTN node forward ADUs, the binding of a destination identifier might occur at the source, during transit, or possibly at the destination. The latter two contexts are referred to as late binding. Storage in DTN is primarily used for persistence and disruption tolerance. Identifiable ADUs can be fragmented into more identifiable ADUs and distributed throughout the network. DTN is capable of proactive, i.e., mainly used when contact volumes are known, or predicted, in advance, and reactive fragmentation. i.e., to fragment a bundle cooperatively, of identifiable ADUs (see Cerf et al., 2007, section 3.8).

While the communication of information is central to a DTN, it is not bound to particular infrastructure or communications convergence layer. DTN protocols enable communications over asymmetric links, over links subjected to long propagation delays, disruption, bandwidth asymmetry and/or packet loss. While the focus of DTN has not been on forwarding performance, DTN protocols have been demonstrated to work with contemporaneous low-latency connectivity, for example using store-and-forward Voice-over-IP in rural telemedicine networks (Scholl et al., 2009).

The areas of focus in DTN are forwarding in heterogeneous networks, naming, and handling disruption (Fall, 2012). In contrast, a focus of our work is to work with any element as a first-class abstraction in heterogeneous networks with moderate-to-well, or poorly performing links that are subject to disruption. We can constrain the relationships of network elements in order to derive a set of network architecture properties. Network architectures that provide for communications in a subset of communications contexts, consequently provide communications capabilities in a limited range of contexts. Support of a wide range of communications characteristics enables transfer of ADUs in a wide range of communications contexts

We can associate each DTN ADU with an identifier, as in the Internet Indirection Infrastructure (i3)

(Zhuang et al., 2005), that can be used to define an indirection point used by the receiver to obtain the ADU. Multiparty communication through replication, use of distributed cache / storage and publish-subscribe mechanisms are possible in a broad range of contexts using DTN protocols.

1.4 Approach

We review the main causes of a node's inability to exploit an opportunistic contact in in heterogeneous networks in order to derive a set of protocol and context-specific requirements in Sections 1.4.1 through 1.4.7. The context-specific requirements, herein called radically heterogeneous networking contexts (RHNC), is the set of communications characteristics that our nodes must support. At the end of each section we summarise the main requirement to address the section problems as a sentence in italics, along with a table of the derived set of requirements that address the sub problems. Table 1.1 provides a summary of the causes and the RHNC, that give rise to our set of networking requirements.

Basing our review on the set of networking requirements derived in Sections 1.4.1 through 1.4.6, we analyse in detail how they are fulfilled by standard discovery protocols in Chapter 2. We describe the hypotheses on which our discovery protocol builds in Chapter 3. We capture these hypotheses in the definitions of our application discovery protocol, called the Thing Discovery Protocol (TDP), and a set of conceptual models for naming, data, data storage, data interchange and network. We define the TDP system architecture, which we capture in a set of modules, called *TDP-module*, *TAV-maker*, and *TDP-decider*, developed specifically in order to test our discovery approach. Next we define the evaluation system architecture, which is captured in two ns-3 simulator modules, called *ns3_configer* and *ns3_lxc_configer*, and an analysis module, called *Analysis*. Basing our implementation on the design, we implement the system architecture modules in Chapter 4. Basing our evaluation on the requirements and the 2009, 2010 and 2011 N4C trial data, in Chapter 5 we describe a set of tests and results in order to demonstrate: a) our discovery approach is viable, and b) that in our approach, can be used to discover facts to exploit a contact, in order to transfer more ADUs than a node using facts discovered by other approaches. Finally, we consider how well TDP fulfils the RHNC.

1.4.1 The context-specific requirements

In our design of Internet applications, we cannot always assume that the network infrastructure will provide for contemporaneous low-latency connectivity, or provide mechanisms for security, discovery and name resolution. We cannot always assume that end-to-end loss is relatively small or that all nodes support the TCP/IP protocols, or use IP addressing. Networks may be subject to partition and significant latency, i.e., long transit delays, on the order of a few seconds or more. It is not always the case that an opportunistic contact will be orientated around host addresses. Nor is it always the case that applications need not worry about communication performance or that endpoint-based security mechanisms are sufficient. In the RHNC communications may be over asymmetric links, be subject to long propagation delays, disruption, bandwidth asymmetry and/or packet loss. Packet loss, significantly reordered packets and acknowledgements received within a period of time based on the measured round-trip time can wrongly indicate congestion to TCP (Bush and Meyer, 2002). Furthermore, we cannot assume that messages can be exchanged between heterogeneous nodes without common specifications and protocols.

There are potentially trillions (Bari et al., 2012) of identifiable things for which assignees must be capable of generating identifiers. The structural components of a potential naming facility must scale at least on the order of trillions and possibly more to accommodate future growth (Bari et al., 2012). Scalable mechanisms for searching and exploiting vast numbers of associations efficiently face similar problems to those of the routing in the Internet. We face severe scalability problems for an Internet-scale deployment like all of the currently prominent ICN projects reviewed by Bari et al. (2012). Potential routing table size scalability issues of the Internet are known and attributed to the number of networks connecting to the Internet, unaggregatable IP address allocations, and prefix aggregation due to multi-homing and traffic engineering (Meyer et al., 2007).

Coupling is intimately related to synchronisation and loosely coupled systems are said to have more flexibility in time constraints, sequencing, and environmental assumptions than do tightly coupled systems (Bush and Meyer, 2002). Systems with complex interactions and tight coupling are likely to have unforeseen failure states and less flexibility in recovering from failure states. Tight coupling of a message to particular modes of communication, discovery, name resolution or message transfer mechanisms can inhibit the transfer of that message.

We cannot assume a node has structured information to exploit network resources Our design must not assume how our discovery information is used: any applications should be capable of decoding the structured information independently of the specific mechanisms of the information producer. In order for it to be used by heterogeneous nodes we must specify the structure of the information, an appropriate portable representation to transfer it, and how to persist it to a data store. We cannot simply turn on inquiry periodically (Bohman et al., 2004) or use pull mechanisms that require operator intervention in the RHNC. Synchronous discovery mechanisms, which are required to maintain synchronous clocks or align slot boundaries, might not work in the RHNC (Yang et al., 2009).

Internet applications, “should use names rather than addresses” (Carpenter, 1996). Routing strictly by address increases the chance of delivery failure and data loss (Clare et al., 2010). A single route selection between sender and receiver might not always be sufficient for achieving acceptable communication performance. Wang et al. (2012) describe an aggregation-aware routing scheme called named Aggregation-aware Inter-Domain Routing (AIDR) that leverages the path diversity, and takes prefix aggregation into account in the BGP best route selection. We cannot assume that nodes are stationary in the RHNC. The location-independence of flat names has been shown to aid mobility and eliminate the management burden of location based address assignment (Singla et al., 2010). Many proposals suggest routing on location independent flat names (D. Farinacci, 2011; Clark et al., 2003). *Communications protocols that support the contexts can be used to exploit a contact in the contexts.*

1.4.2 Resource consumption

Resource consumption is a performance bottleneck on opportunistic data transfer (Oliver and Falaki, 2007). Many problems attributed to consumption of limited power, memory, storage and processing resources have been described (Kushalnagar et al., 2007; Dohler et al., 2009; Pister et al., 2009; Brandt et al., 2010; Martocci et al., 2010). The undesirable consequences of resource consumption are more evident in opportunistic networking scenarios (Yang et al., 2009). As the resources of a node are consumed incoming messages may be dropped. Khabbaz et al. (2012) demonstrated that resource consumption causes the dropping rate to increase and adds network overhead as the forwarder inefficiently ends up consuming precious bandwidth to transmit messages that are subsequently dropped. In such cases electing not to transmmit a message is better, i.e., an informed decision not to transmit exploits the contact. Nodes within opportunistic networks have access to resource information (Grundy and Radenkovic, 2010a). *Knowledge of time varying peer resource consumption can be exploited to transfer ADUs.*

1.4.3 Peer capabilities

During periods in which nodes are joining and leaving a network, it may be possible for the transmitting or requesting node to obtain a more capable peer while a transmission or request is outstanding. Node connectivity, computation and storage capabilities vary enormously (Ping et al., 2009). Oliver and Falaki (2007) demonstrate the effect of control parameters capabilities, such as varied bundle sizes by a factor of up to 60, on opportunistic data transfer. An opportunistic network should be open-loop and requires in-network congestion control capabilities (Grundy and Radenkovic, 2010b). The peer selection process can have a direct impact on throughput (Gurses and Kim, 2008). In order to transfer data within some network contexts we must discover peer capability information (Schmidt and Butt, 2009). *Knowledge of time varying peer capabilities can be exploited to transfer ADUs.*

1.4.4 Network element relationships

Enabling "discovery of the relationships between resources, is an important factor in the design of today's LAN equipment" (URL, 5). However, physical layer information might not be a good indicator of actual network performance. There is a need for a method of discovering network resources that goes beyond radio interface characteristics by enabling nodes to exchange information in a peer to peer manner (Pedrasa and Seneviratne, 2011). It is necessary sometimes to establish a trade off between discovery beaconing, i.e., broadcasting messages that contain discovery information, and power usage. The facts that describe the relationship between power usage and message transmission can be used to establish optimal energy efficient message transmission. Optimal energy efficient discovery schemes often require the exchange of information to establish the trade off Yang et al. (2009). It is important to discover when contacts occur and their durations for efficient energy consumption, as these can be difficult to accurately predict (Yang et al., 2009). Some data might be more critical than other data. An inability to distinguish or learn the nature of the data and its relationships to a nodes mechanisms, For example whether data is more or less important than other data queued on a link for transmission, can impede a nodes ability to select a suitable set of mechanisms. The need for discovery is "imperative" to learn the "nature of networks", and "up-to-date node discovery is imperative for achieving efficient data transfers" (Schmidt and Butt, 2009). *Knowledge of time varying network element relationship information can be exploited to transfer ADUs.*

1.4.5 Routing information

Prior knowledge of node mobility and connectivity, are assumed by several routing schemes to perform message transfers (Abdulla and Simon, 2007). A node might make forwarding decisions using measurements based on, for example, the known state of other nodes, information on resource utilisation, and the probability of an encounter. Store-and-forward routing schemes also use information on node contacts, location, and future movement (McMahon and Farrell, 2009). A node generally bases forwarding decisions on locally held information, for example, to remove failed paths, determine the best next-hop, time to forward, and highest delivery probabilities for each message. However, contact information must be known, or discovered in by nodes to form the basis for routing (McMahon and Farrell, 2009).

Routing schemes developed for heterogeneous networks use various mechanisms, including packet replication, discovery of the meeting probabilities among nodes, and network coding. There are many specified routing schemes for heterogeneous networks, including Delay-Tolerant Link State Routing (DTLSR) (Demmer and Fall, 2007), Contact Graph Routing (CGR) (Burleigh, 2007), the Resource Allocation Protocol for Intentional DTN (RAPID) (Balasubramanian et al., 2007a), and the Probabilistic Routing Protocol for Intermittently Connected Networks (PRoPHET).

Using PRoPHET a prediction of a future viable paths might be informed by calculating the probability of link availability, based on history. Through prioritisation, based on prediction, of both the schedule of packets transmitted to other peers and the schedule of packets to be dropped, MaxProp is demonstrated to outperform protocols that have access to an *oracle* that knows the schedule of meetings between peers (Burgess et al., 2006). Distance vector, link state, and source routing approaches are used to determine an immediately available path from source to destination. The path computation of DTLSR takes prediction into account when discovering viable paths (Demmer, 2008). The convergence layer informs DTLSR on link availability. Routing protocols are generally not responsible for providing neighbour discovery. DTLSR requires a discovery mechanism to provide connectivity information which it distributes throughout a network. The DTLSR implementation in the DTN reference implementation (DTN2) uses CLA specific discovery protocol mechanisms is for discovery of proximate nodes, which issue calls to the routing layer when connectivity is detected, or lost, between nodes. RAPID discovers network resources via a control plane that helps nodes in acquiring complete information about network state. In RAPID each node exchanges such information as the number and location of replicas, and average size of past transfers (Khabbaz et al., 2012).

Probability prediction requires information. An optimal probabilistic forwarding protocol is demonstrated to maximise the delivery rate of each message when mean inter-meeting times between all pairs of nodes is known (Liu and Wu, 2009). Previous works have proposed a variety of long term metrics, including social pattern similarity (Daly and Haahr, 2007) and encounter patterns (Lindgren et al., 2003) which routing protocols can exploit. One advantage of long-term delivery metrics is that they are relatively stable once generated from historical connectivity information or prior knowledge on the contact pattern of nodes, avoiding the cost associated with frequent updates. On the other hand, “many real objects have cyclic motion patterns, and therefore, it is possible and valuable in practise to increase the accuracy of a delivery metric by allowing it to be time-variant” (Liu et al., 2009). Delegation forwarding (Erramilli et al., 2008) forwards the messages based on different delivery probability metrics. Delegation forwarding maintains a forwarding threshold when contact of node pairs occurred. Forwarding threshold indicates the quality of node such as cost, delivery rate and average delay. Node mobility, contact, and delivery metrics are a subset of network elements that can be structured using an appropriate mechanism and stored using appropriate data storage models. Such elements can be extracted and made accessible in an appropriate generic interchange format to local routing protocols, or made available to the routing protocols of remote entities.

The DTN Publish / Subscribe Protocol (DPSP) (Greifenberg and Kutscher, 2008) is a probabilistic multicast routing protocol for opportunistic networks. DPSP routers do not try to maintain a view of the network topology and select an optimal path. Instead, the routers replicate bundles to their neighbours in order to get the bundle delivered by multiple hops using store-and-forward techniques. Greifenberg and Kutscher (2008) assert that epidemic routing and *spray-and-wait* approaches have the advantage that they can be applied without any knowledge about the network, and do not require any kind of global coordination. The goal of Epidemic routing is “to deliver a message (update) with high probability to a particular host", and a key issue is "deciding" whether to transmit a message in the event of a contact (Vahdat and Becker, 2000). Such decisions might be based on the facts about a message.

A general destination endpoint identifier mapping function is required that can be used by a number of routing schemes; endpoint discovery protocols may be a good vehicle for developing such a scheme (Farrell, 2010). Routing protocols could make use of information learned from endpoint discovery protocols. In particular, by supplying the attributes of endpoints to epidemic protocols (Vahdat and Becker, 2000), such as Context-aware Adaptive Routing (CAR) protocol (Musolesi and Mascolo, 2009) which derives delivery probabilities from context information and defines context as the set of attributes that describe the aspects of the system that can be used to drive the process of message delivery.

The routing scheme and simulation of Yang and Chuah (2006) assume neigbour discovery, that is, configurable contact probing, i.e., devices must probe their environment to discover other devices as described by Wang et al. (2009), or beaconing is assumed. Discovery is required to enable a source node use routing protocols to attempt to probabilistically transfer data packets to a destination node (Kim et al., 2008). When a network node moves the network topology changes. A consequence of node mobility is that route discovery packets are re-broadcasted, consuming more control packets, in order to rebuild or repair a broken path; a critical challenge in building routing protocols (Chuang et al., 2012).

To exploit a contact a data transfer application or routing protocol should be capable of reordering or prioritisation of messages, for transfer, and preemption of transfer opportunities. While protocol studies generally assume that two nodes will exchange all the information they possibly can while the contact last, routing protocols should not just determine the order in which to replicate messages but also when to stop and turn to a potential next peer, without knowing if such a peer is present (Pitkanen et al., 2012). *Knowledge of time varying routing information can be exploited to transfer ADUs.*

1.4.6 Distributed resources

Routing approaches have been demonstrated to assume network knowledge (Zhao et al., 2004; Abdulla and Simon, 2007). A routing approach that distributes copies amongst the potential relays requires the utility of both the nodes must be known to distribute copies (Jindal, 2006). In order to guarantee eventual message delivery using some routing approaches, a subset of nodes must have buffer space equal to the maximum number of messages that are in flight at any given time (Vahdat and Becker, 2000). Using some routing approaches, a "system must balance the conflicting goals of maximising message delivery and minimising resource consumption ", such that in order to maximise the likelihood that a message is eventually delivered, a message should not consume buffer space at all nodes to ensure its most timely delivery (Vahdat and Becker, 2000). *Knowledge of time varying distributed resource information can be exploited to transfer ADUs.*

1.4.7 Security and policy mechanisms

While the issues of security and policy are not in the scope of this thesis, we provide a brief overview. The discovery of network elements may be perceived as orthogonal to securing the network elements. Discovery mechanisms are a potential source of network congestion and contention and careful consideration should be made to the frequency and lifetimes of discovery messages. Security concerns have been raised such as the routing path discovery vulnerability in overlay networks (Beitollahi and Deconinck, 2008) (Castro et al., 2002) (DePaoli and Mariani, 2004), where an adversary may possess real-time knowledge of the specific nodes through which a client is routing traffic (Beitollahi and Deconinck, 2009). Nevertheless, in some contexts we must determine who generated or published discovery information. DoS attacks can result in resources, such as the bandwidth capacity being consumed on all nodes involved (Farrell and Cahill, 2006b) in a contact. For resource contrained nodes there is a high risk of physical attacks to unattended sensors. In overlay networks, an adversary can discover connectivity information of routing path easily, and consequently, prevent communication by attacking only the specific nodes (Beitollahi and Deconinck, 2009). In the Internet, host-to-host security protocols, such as SSL, are used when host authenticity and data integrity services are required. In contrast, content-oriented security models, in which content is signed by the producer, allow validity of the content to be ascertained by the consumer through verification of the signature. A security mechanisms for heterogenous networks, the Bundle Security Protocol (Symington et al., 2011), protects the infrastructure from unauthorised use by allowing for policy based discarding of traffic as quickly as possible, and "indications are that these items can be combined to enable a type of multi-layer security for content storage, discovery and distribution"(Ivancic, 2010). *Knowledge of time varying security and policy mechanisms information can be exploited to transfer ADUs.*

1.4.8 Summary of the problem

The main causes of why it is difficult for a node to handle an opportunistic contact to transfer ADUs, and do so in the RHNC have been presented. The causes and consequences of a node's difficulty to handle an opportunistic contact in order to transfer data in the RHNC are summarised in Table 1.1.

No.	**Cause**	**Section**
1	Use of communications protocols that don't support RHNC are unable to exploit a contact in the RHNC	Section 1.4.1
2	Ignorance of time varying resource consumption	Section 1.4.2
3	Ignorance of time varying peer capabilities	Section 1.4.3
4	Ignorance of time varying network element relationships	Section 1.4.4
5	Ignorance of time varying routing information	Section 1.4.5
6	Ignorance of time varying distributed resources	Section 1.4.6
7	Ignorance of time varying security and policy mechanisms	Section 1.4.7

Table 1.1: Causes of a node's inability to exploit an opportunistic contact to transfer ADUs within the bounds of RHNC

1.5 Discovery protocol requirements summary

A set of protocol requirements that are specific to this context can be derived through review of the problems presented. The protocol requirements can be used to examine how existing discovery protocols fulfil these requirements. A goal of this thesis is to use existing Internet protocols that fulfil these requirements where possible. Table 1.2 lists all of the discovery protocol requirements together for ease of referencing. The application discovery protocol must be capable of handling an opportunistic contact in the RHNC described in section 1.4.1.

The protocol must address problems outlined in section 1.4.2 through 1.4.7 to be capable of exploiting an opportunistic contact to transfer ADUs within the bounds of the described contexts.

Three requirements for this work are a structure of information, the exchange of time varying information, and support of the RHNC. The structure of information must represent network elements and the relationships between them. In particular, the discovery ADUs must be transferred despite potential communication disruptions, and new discovery information must be made available to the nodes in order that they may adapt their behaviour within defined time bounds.

No.	Requirement
R.1	Protocol must support the context-specific requirements (RHNC)
R.2	Protocol must exchange knowledge of time varying resource consumption,
R.3	peer capabilities,
R.4	network element relationships,
R.5	routing information,
R.6	distributed resources, and
R.7	security and policy mechanisms

Table 1.2: Summary of requirements for a node to exploit an opportunistic contact in order to transfer ADUs, and do so in the RHNC.

1.6 Goal and contributions

Our design of the TDP conceptual models, TDP and the TDP modules, is the substantive contribution of this thesis to the area of DTN. The DTN Endpoint Discovery Protocol (EDP) (McMahon and Fall, 2010) can be used by DTN nodes to report their endpoint identifiers (EIDs) and EID registrations. TDP can be used by DTN nodes to report their endpoint elements, attributes and identifiers. TDP, along with the TDP conceptual models, defines the methods for a DTN node to learn a description of the classes and relationships of its elements. DTN TDP things are associations with any element of a DTN endpoint. TDP enables the direct verification of an association.

An ultimate goal of this investigation to make a better Internet through eventual incorporation of TDP, the TDP conceptual models, and the TDP modules features and capabilities into the standard suite of Internet protocols. This thesis seeks to develop protocols that endorse the global testing of those protocols and represents another iteration of a series of additions to the protocols that define the Internet.

The contributions of this thesis are fourfold:

1. A discovery approach which can be used by a node to discover facts to exploit a contact, in order to transfer more ADUs than a node using facts discovered by other approaches.

2. A set of conceptual models as part of an approach to organise endpoint elements and attributes to provide a heterogeneous network of directly verifiable associations

3. The architectural constraints, exemplified by the *TDP-module* system and evaluation system architecture modules, that, when applied as a whole, emphasise scalability of loosely coupled systems.

4. The evaluation of how TDP can exploit an opportunistic contact in the RHNC.

1.7 Scope

This thesis defines an application discovery protocol that can be used to exploit an opportunistic contact in order to transfer ADUs in the RHNC. This work also describes models used by nodes to organise, transfer and persist the facts about themselves. We describe a systematic process to use the models to create a network of associations. To facilitate this process, a tool is presented that guides thing developers through the stages required to use the model, and automates the stages. Finally, we evaluate and assess the generality of the protocol through its implementation and application to several emulated contexts as exemplified by opportunistic contacts described by the 2009, 2010 and 2011 N4C trial data. To facilitate this process, we present a tool that uses TDM data in order to make decisions relating to the transfer of ADUs during an opportunistic contact. However, the algorithm for deciding what information to provide is out of scope.

This work, however, does not present a formal definition of the thing protocol, nor a formal proof of its correctness. The development process allows individual elements of networks to be mapped to the TDM. If these requirements are met, exchange of information by the protocol will be ensured. Ensuring these requirements might require nodes to use specific architectures or algorithms e.g., compression algorithms, memory and storage capacity. How to ensure that nodes fulfil these requirements is outside the scope of this work. This work does not present a formal semantic theory of the models, nor a formal proof of their correctness. The chief utility of such a theory is to provide a technical way to determine when inference processes are valid (URL, 102) in order to provide the maximal freedom for implementations while preserving a globally coherent notion of meaning. The definition and use of the protocol and models is illustrated with a number of examples from communications challenged contexts. However, the described contexts are not the subject of this thesis, and is only one of the possible application domains of this work.

The discovery protocol and models have been designed to support the radical heterogeneity of the RHNC, not just networks with moderate to well performing links. That is, the constructs defined facilitate reasoning about opportunistic contacts of physical nodes within heterogeneous networks and would not be efficient for generic networks with moderate to well performing links, and consequently for underlay discovery mechanisms. Note also that while the TDM can be used to design things for many contexts, interaction with the things are assumed within the described context and their interactions with other context are not covered by this work. In addition, the radical heterogeneity of network elements that constitutes the RHNC aspect of our discovery application is crucial for such a protocol, and this is also a defining assumption for our work. Finally, it should be noted that the issues of security and policy are not in the scope of this thesis.

1.8 Document layout

The remainder of this thesis is organised as follows. Architecting a discovery application to enable a node make best use of an opportunistic contact in the RHNC requires an understanding of its requirements, as we shall review in the state of the art provided in Chapter 2.

Chapter 3 presents the hypotheses on which TDP builds. These hypotheses are captured in a naming model which we describe in Section 3.3, a data model that is described in Section 3.4, a storage model described in Section 3.5, and a network model which we describe in Section 3.6.

Next, the system architecture is captured in the analysis and design of the TDP module, *TDP-module* in Section 3.7, the TDM module *TAV-maker* in Section 3.8, and the module which makes decisions based on TDM data, *TDP-decider* in Section 3.9. The design of the modules that build and use ns-3 as a real-time network emulator, which corresponds to the network model, and that can be interconnected with the *real* world is specified in Section 3.10. The design of the module used to analyse cases where in the event of a contact, *real*-world data transfer applications that utilise information provided by TDP are enabled to transfer more application data units than applications that utilise information provided by other sources is specified in Section 3.11.

Chapter 4 describes an implementation of TDP, TDP related models, and suitable modules for their application, that we described in Chapter 3, as the modules *TDP-module*, *TAV-maker* and *TDP-decider*.

Chapter 5 describes the evaluation of TDP using the modules designed in Chapter 3 and subsequently implemented in Chapter 4. The hypothesis on which TDP builds is evaluated using the the *TDP-module*, *TAV-maker*, and *TDP-decider* modules, implemented respectively in Sections 4.3, 4.4, and 4.5, by way of the ns-3 modules, *ns3_configer* and *ns3_lxc_configer*, implemented in Section 4.6, and the automated *Analysis* module for analysis and evaluation, implemented in Section 4.7. The *ns3_configer* and *ns3_lxc_configer* modules use the ns-3 discrete-event network simulator as a real-time network emulator, to enable ns-3 to be interconnected with the *real* world, in order to allow use of appropriate existing *real*-world protocol implementations.

Chapter 6 draws conclusions and describes planned and potential future work.

1.9 Summary

In Chapter 1 we outlined the goals and scope of our work described in this thesis; our definition of a discovery protocol to provide a node with the information to exploit an opportunistic contact in order to transfer ADU. The chapter began by presenting the basic motivation for the work described in this thesis, i.e., the need for discovery mechanisms that can be used by nodes to exploit an opportunistic contact in the RHNC to transfer ADUs despite the limitations of such communications opportunities. We defined the problem in more detail by examining the limitations of an opportunistic contact in the RHNC, availability of time varying node data, and their effect on a nodes ability to transfer ADUs.

The main challenges that arise from this problem were outlined, and a brief overview of existing work demonstrated that while there have been attempts to fulfil discovery requirements with discovery applications, the main challenges that it is difficult for a node to exploit an opportunistic contact in the RHNC to transfer ADUs prevails. There has not yet been a demonstration of the enablement of a node to exploit an opportunistic contact in the RHNC to transfer more ADUs than a node using other discovery protocols. This is the particular question that this thesis attempts to answer. Chapter 1 concluded by detailing our approach, and the goals and contributions, as well as the areas that are outside the scope of the work.

Chapter 2

Related work

Discovery mechanisms have been studied in networking, engineering, and computer science domains but this work has mostly been divided. In particular discovery mechanisms have become an integral part of modern networked systems (Meshkova et al., 2008). Detection of the presence of peer systems in communications range is fundamental to intermittently connected networks. In this chapter we review work on discovery from different communities, focusing on frameworks, standards and protocols and the contexts for which they were designed. At the lower layers, the mechanisms most relevant to our work are predominantly specified by the Institute of Electrical and Electronics Engineers Standards Association (IEEE-SA) (URL, 4). While the IEEE-SA specifications do not specify the entire set of mechanisms, analysis of the body of work that describes those mechanisms provide an appropriate body of knowledge of our problem space.

In the following words in italics indicate IEEE-SA terms. First we shall review the general area of discovery within *IEEE 802* networks in Section 2.1. Section 2.2 explores structured information mechanisms used by those discovery specifications. Section 2.3 reviews mechanisms for naming structured information. Fourth, we will discuss the *IEEE 802* based RHNC in Section 2.4. Work on DTN that is relevant to this thesis is investigated in Section 2.5. DTN discovery mechanisms are reviewed in Section 2.6. In Section 2.7 we outline the requirements that a mechanism needs to fulfil to ensure discovery, compare the discovery mechanisms using these criteria, and analyse their approach to discovery. In Section 2.8 we provide a summary of Chapter 2.

2.1 Discovery

“Discovery has become an integral part of modern networked systems“ (Meshkova et al., 2008). Discovery mechanisms play a fundamental part in Web service architectures, and in the process of creating new virtual networks. Meta-data discovery services are widely recognised as a core ingredient of the Internet of Things (IoT) architecture (Roussos and Chartier, 2011). Resource discovery (RD) is a fundamental step in the process of creating new virtual networks (Belbekkouche et al., 2012).

2.1.1 Higher layer characteristics

Rodrigues et al. (2012) cite identification, interoperability, and aggregation of services as the three main issues that current service discovery (SD) protocols should resolve to provide SD in the large. In their state of the art on resource and service discovery mechanisms in general, Meshkova et al. (2008) focused on RD mechanisms and SD frameworks and observed that the majority of the systems developed involve peer-to-peer overlays with a distributed architecture, and thus far have been targeted towards large-scale use in the fixed Internet. In contrast, the work in this thesis caters for discovery of mobile network elements.

In a comprehensive survey of RD protocols in mobile ad-hoc networks, Su and Guo (2008) categorised discovery of services as being performed in two modes; push and pull. In push mode, servers send unsolicited advertisements which clients receive and select the services they are interested in. In pull mode, clients request a service and servers that offer the service answer the request. The authors assert there are two ways to classify SD protocols. One is based on the existence of a directory, and the other is based on whether it is implemented at the application, or network layer. A directory partially or entirely stores the description information about services available in the network so as to enable SD and service invocation. Su and Guo (2008) describe a directory-based architecture, in which discovery protocols first select backbone nodes, and then discover the paths between them, to form a distributed virtual backbone. In a directory-less architecture, there is no directory server, and some, or all of the services are located in a local service cache that is maintained on some node.

Service discovery is described as being performed in the application layer or in the network layer. Su and Guo (2008) compare protocols in terms of how they address the problems of implementation layer, directory service, push mode, pull mode, service caching, service description, service access and power saving. Table 2.1 provides a simplified version of their comparison, in which the type of implementation layer, and whether or not directory service, push mode, an pull mode is provided.

RD methods are further divided by Sarvanko et al. (2010) into three main modes; sensing, push and pull. The sensing mode is used only for radio RD since sensing gives information about spectrum use at a particular time and in a particular location. Majumder et al. (2011) analysed the performance of these approaches under different scenarios and found that due to periodic updates of routing information, the “proactive”(Jonsson et al., 2000; Elizabeth et al., 2002) push approach shows better packet delivery performance than the “reactive”(Broch et al., 1999) pull approach.

There is a trade-off between contact duration and type of discovery. Pitkanen et al. (2012) assert that determining the presence of another node is not sufficient, and higher layer SD is needed to determine which services two peers have in common, which takes time to complete and shortens contacts. The

Protocol	**Layer**	**Directory**	**Push**	**Pull**
DEAPspace (Nidd, 2001)	Application	No	Yes	No
Konark (Helal et al., 2003)	Application	No	Yes	Yes
PDP (Campo et al., 2005)	Application	No	Yes	Yes
Allia (Ratsimor et al., 2002)	Application	No	Yes	Yes
GSD (Chakraborty et al., 2002)	Application	No	Yes	Yes
SANDMAN (Schiele et al., 2004)	Application	Yes	Yes	Yes
SSD (Sailhan and Issarny, 2005)	Network	Yes	Yes	Yes
DSDP (Kozat and Tassiulas, 2004)	Network	Yes	Yes	Yes
AODV-SD(Garcia-Macias and Torres, 2005)	Network	No	No	Yes
LSD (Li and Lamont, 2005)	Network	Yes and No	Yes	Yes
M-ZRP(Ververidis and Polyzos, 2005)	Network	No	Yes	No
ODMRP-SD (Cheng, 2005)	Network	No	Yes	Yes

Table 2.1: Ad hoc SD protocols comparison of Su and Guo (2008).

authors assert that as long as SD mechanisms can use physical WiFi hotspots, such hotspots can provide a supplement for ad-hoc interactions between mobile nodes, as they solve issues essential for effective SD: efficient address allocation, and preservation of communication among multiple peers at once.

Discovery mechanisms can be symmetric or asymmetric. In some ad-hoc networking scenarios it is not sufficient to simply turn on inquiry periodically since intervals between two devices might not align well, and consequently fail, or take too long (Bohman et al., 2004). When using access points in infrastructure mode wireless networks, it is usually the client device that tries to discover the access points by using the inquiry procedure, a pull mode, while the access points announce their presence by periodically turning on inquiry scan mode, a push mode.

Evolving network element relationships can used by discovery mechanisms to sucessfully interact with their environment. Research on the design of energy conscious, adaptive contact probing algorithms, which trade-off energy consumption and the probability of missing a contact, conclude discovery should be done adaptively in some contexts, by choosing the probing interval based on the state of the environment (Wang et al., 2009). A discovery protocol called Sleeper, which uses proxied advertisement and discovery to dynamically offload SD workload from power-limited devices has been described (Buford et al., 2006).

A RD framework for network virtualisation is proposed to discover the resources available in the underlying physical infrastructure (Houidi et al., 2009). Belbekkouche et al. (2012) survey previous work on, and describe challenges of, RD and allocation in network virtualisation. The authors conclude that future work on RD should focus on efficient monitoring techniques that consider non-functional time-variant attributes, as architectures that consider only functional attributes are less practical for RD. While this work can be used to design RD mechanisms that can support network virtualisation, the network is assumed capable of catering for efficient monitoring techniques, an assumption with a real-time aspect that is not possible for our work.

Evdokimov et al. (2010) compare the characteristics of "prominent approaches" for implementing Discovery Services for the Internet of Things (IoT), and outline the functional requirements for for the IoT Discovery Services (DS). The authors assert that should the IoT visions of large-scale actuator and sensor networks be realised, none of the existing, well-established discovery systems would be suitable due to the resource constraints and potentially complex queries. The approach taken by ZigZag is based on protocol translation to enable SD protocols initially designed for local area networks to work across highly heterogeneous networks targeting the needs of Future Internet. ZigZag makes the assumption that all SD protocols use a multicast group address and a UDP/TCP port. In contrast, like DTN2 discovery which can use ethernet beacons to discover an Ethernet convergence layer, the work in this thesis, does not assume all SD protocols use IP addressing or a UDP/TCP port. A modular overlay approach has been taken (Berger et al., 2004) in which discovery protocols, access models and authentication mechanisms are represented as software components, that are registered with, and coordinated by a software framework. In the design the agent answers query and advertisement requests on behalf of mobile and other applications and applies authentication and access control policies.

In an ICN inspired IoT approach (Marias et al., 2012), scopes, which can be can be hierarchically organised, are used to structure their information. A scope is a meta-service for attribute queries which it forwards a resolution service. In that architecture, "companies" attach information about an object to scopes. A user can then query the scope that corresponds to the desired context, which will forward the query to a service in order to acquire appropriate data. In that approach, objects are associated with attributes which are coupled with values. Every object is uniquely identifiable and associated with multiple access controlled $< attribute, value >$ tuples assigned by its stake holders. The IP addresses of scopes are used are their identifiers. The operations of this discovery mechanism are described in Table 2.2

Operation	**Description**
Scope Creation	new entity or sub entity scope created. sub level entities subject to inheritance of parent scopes.
Attribute Advertisement	The forwarding tables of a scope are populated by the stakeholders. Each entity can set access control policies to the attributes it specifies.
Attribute Subscription	A user requests the value of an attribute
Value Forwarding	a value being forwarded from the look-up service to the user

Table 2.2: Architecture operations of an ICN based discovery mechanism for the Internet of Things

Greenberg et al. (2005) describe a complete clean slate re-design of the Internet management and control planes, called the 4D architecture. Greenberg et al. (2005) describe three key principles as a basis for a new architecture: network-level objectives, network-wide views, and direct control. The 4D architecture has four planes: decision, dissemination, discovery, and data. 4D sets up a separate dissemination channel for control and management activities through link layer self discovery mechanisms. The discovery and dissemination planes of the the 4D architecture, depends on network wide broadcasts. Decisions are taken to implement dynamic configurations, based on centralised topology information and organisational policy inputs. The 4D architecture completely decouples an autonomous system's decision logic from protocols that govern the interaction among network elements. In contrast, the work in this thesis is used by network elements to exploit themselves.

2.1.2 Lower layer characteristics

A fairer channel contention and better channel utilisation has been demonstrated by Bong and Yeo (2011) that uses ND in DTN with media access control based on Multiple Access with Collision Avoidance for Wireless (MACAW). Discovery has be shown to be important in DTN Wireless Sensor Networks (DTWSNs) (Nayebi et al., 2009) and each Haggle Connectivity object, of which there is one instance per instance of a network interface on a node, must support a well-defined interface that includes functionality for neighbour discovery (Scott et al., 2006).

Opportunistic spectrum access (OSA) is a networking approach to improve the utilisation of the available spectrum, and enhance the availability of wireless ad-hoc networks. OSA networks with dynamic channel hopping are referred to as cognitive personal area networks or CPANs (Misic et al., 2012). Low power and Lossy Networks (LLNs), which are made up of many embedded devices with limited power, memory, storage and processing resources, require simple service discovery network protocols to discover, control and maintain services provided by devices (Kushalnagar et al., 2007; Dohler et al., 2009; Pister et al., 2009; Brandt et al., 2010; Martocci et al., 2010). The open wireless channel nature of communication of LLNs means that generally anyone can get the packets and consequently ID and key management mechanisms are required. However, in many LLN contexts it is not possible to configure them in advance as location is not predefined and devices are moving.

We review the discovery mechanisms used in Networks considered in the family of *IEEE 802* standards of the IEEE-SA. These discovery mechanisms may be roughly categorised as defined by the standards and protocols of the IEEE-SA, or by the Internet Engineering Task Force (IETF) (URL, 7) and organisations associated with the IETF, or by other bodies, i.e., standards not from organisations associated IEEE-SA or IETF. Firstly we review the work of the IEEE-SA in Section 2.1.3. Secondly we review the work of the IETF and organisations associated with the IETF in Section 2.1.4. Thirdly we review the work of other bodies in Section 2.1.5. Finally, we provide an analysis of the discovery mechanisms in Section 2.1.6.

2.1.3 IEEE-SA

The IEEE-SA has standards for networking, engineering, computer science and electronics. The IEEE-SA discovery mechanisms that are particularly relevant to our work are provided in Table 2.3. A summary of the characteristics of these follows.

Standard	**Working Group**	**Reference**
IEEE 802.2	Logical Link Control (LLC)	(802, 1989)
IEEE 802.1	Higher Layer LAN Protocols	(802, 2009a)
IEEE 802.11	Wireless LAN (WLAN)	(802, 2012b)
IEEE 802.15	Wireless Personal Area Network (WPAN)	(802, 2005)
IEEE 802.16	Broadband Wireless Access	(802, 2012a)
IEEE 802.21	Media Independent Handover Services	(802, 2009b)

Table 2.3: *IEEE 802* work particularly relevant to our work

IEEE 802.2

IEEE 802.2 specifies the services of the Logical Link Control (LLC) sublayer. The protocol data unit (PDU) structure for data communication systems and three types of operation for data communication between service access points are defined. All classes of the LLC sublayer standard for all *IEEE 802* networks include mandatory support operations, that enable stations to discover the existence of other stations, without the need for the prior establishment of a logical link between peers. In layer-2 mechanisms, usually beacon frames are periodically transmitted, and responded to with probe response frames. *IEEE 802.2* provides point-to-point, multicast, or broadcast data transfer for a pull mechanism.

IEEE 802.1

IEEE 802.1 defines the media-independent Link Layer Discovery Protocol (LLDP), and deals with the Management Information Base (MIB) elements, and the transfer of this structured information via a push mechanism.

IEEE 802.11

A mobile station of an *IEEE 802.11* network has to cope in the RHNC. These contexts include heterogeneous wireless device deployments, characterised by overlapping frequencies, different traffic load and high interference. The contexts presented cannot be precisely predicted by a mobile station in motion, and consequently require an appropriate scanning algorithm. When a mobile station initialises or moves, it needs to discover its radio frequencies, the points of attachment of its neighbours, and available services. Each time a mobile *IEEE 802.11* station moves, it needs to discover, and join new service sets. During the *IEEE 802.11* scanning phase, a mobile station uses management frames to actively scan a channel and discover the points of attachment. In infrastructure mode, a mobile station initiates a discovery process on each Layer 2 handover in order to join a new basic service set, and in ad-hoc mode, a mobile station initiates the discovery process to form an independent basic service set with its direct neighbours. Castignani et al. (2011) analyse and evaluate the discovery process on *IEEE 802.11* devices through experiments with an emphasis on ascertaining how long a mobile station must wait before receiving a response from a point of attachment. Heterogeneous scenarios may have different optimal timer values to achieve an optimal trade-off between full scanning latency and full scanning failure. The authors demonstrate the importance and the efficiency of using an adaptive strategy, when faced with heterogeneous scenarios as the discovery process must dynamically adapt to them. The authors inform that the experiments and conclusions can be extended to non-*IEEE 802.11* based discovery systems. A significant contribution of the *IEEE 802.11* working group are the reporting mechanisms, comprised of various WLAN radio measurements. Reports are in the ASN.1 encoded MAC, and use the PHY MIB specification detailed in the (802, 2012b) standard. Another significant contribution are the Wireless Network Management (WNM) protocols, to enable *IEEE 802.11* stations learn about the topology and state of the network. The mesh facilities, provide reporting enhancements for mesh wireless stations. Tunnelled direct-link setup (TDLS) discovery information allows TDLS peer wireless stations to discover their capabilities which can also be accomplished by using higher layer protocols.

IEEE 802.15

A number of types of physical channel are defined by *IEEE 802.15.1* in which all *IEEE 802.15.1* devices use the inquiry procedure to discover devices or to be discovered by devices. Two *IEEE 802.15.1* devices use a shared physical channel for communication, and a *IEEE 802.15.1* device can use only one of these physical channels at any given time. A *IEEE 802.15.1* device that is connected to a channel is synchronised to the timing, frequency, and access code of a physical channel. A discoverable device listens for inquiry requests on its inquiry scan physical channel and then sends responses to these requests. An example of use of the inquiry procedure to discover services and service attributes is provided by the Service Discovery Protocol (SDP).

IEEE 802.16

All *IEEE 802.16* devices can obtain topology information using an *IEEE 802.16* discovery mechanism. The pull mechanism for neighbourhood discovery/measurement can be used during initial network entry, during periodic intervals or whenever a multihop relay base station requires this information. IPv6 mechanisms were conceived with the assumption that they would run over *IEEE 802.3* MAC protocol and the *IEEE 802.11* MAC protocol, however, the structure of *IEEE 802.16* differs greatly from the *IEEE 802.3* or *IEEE 802.11* protocol. *IEEE 802.16* supports multicast and broadcast to transmit *IEEE 802.16* MAC management messages, not IP data. An IPv6 Neighbor Discovery message on *IEEE 802.16* cannot be delivered to neighbouring hosts by means of multicast. However, there are proposals on how to operate IPv6 NDP over an *IEEE 802.16* network (Jeon and Jee, 2006). A model in which neighbouring nodes cooperate was demonstrated (Du et al., 2012) to reduce delays in the neighbour discovery process.

IEEE 802.21

The *IEEE 802.21* Media Independent Handover Services defines extensible *IEEE 802* media access independent mechanisms that enable the optimisation of handover between heterogeneous *IEEE 802* networks and facilitates handover between *IEEE 802* networks and non *IEEE 802* networks. Link-layer intelligence and other related network information is provided to upper layers to optimise handovers between heterogeneous networks. The intent of *IEEE 802.21-2008* is to provide generic link-layer intelligence independent of the specifics of mobile nodes or radio networks. A network can include a mixture of cells of radically different sizes, such as those from *IEEE 802.11*, *IEEE 802.15* and *IEEE 802.16* with overlapping coverage. The handover process can be initiated by measurement reports and triggers supplied by the link layers on the mobile node and measurement reports can include metrics such as signal quality, synchronisation time differences, and transmission error rates.

Summary

The family of *IEEE 802* standards and their architectures assume continuous connectivity and negligible latency, i.e., transit delays, on the order of a few milliseconds or less. In contrast, the work in this thesis, tolerates network partition and significant latency i.e., transit delays, on the order of a few seconds or more.

While the problems addressed by the IEEE 802.2 working group are similar to those tackled in this thesis, the work in that community provides point-to-point, multicast, or broadcast data transfer for a pull mechanism. Therefore, that work is not directly applicable to work on autonomous mobile entities in partitioned, or constrained network contexts that require a push mechanism.

LLDP enables an agent to learn information from adjacent stations on all *IEEE 802* networks via the *IETF PTOPO MIB*, which if available provides network management that can be used to exploit a contact. The framework that makes this information accessible, however, it is designed for managing TCP/IP networks. the work in the *IEEE 802* community mostly deals with objects in the MIB which have highly complicated encoding rules and long addresses, that when transferred, consume substantial parts of each message. Therefore, that work is not directly applicable to work in the RHNC, which require the discovery of elements of heterogeneous, resource constrained networks.

To fulfil our requirements we can make use of the various the *IEEE 802.11* WLAN radio measurements, to enable participants of an opportunistic contact make intelligent decisions about the most effective way to exploit the available spectrum, power, and bandwidth for communications. Similarly, approaches can use information from discovery mechanisms of the WNM protocols, mesh facilities, and TDLS. However WLAN radio measurement reports from remote nodes take the form of a MIB, and use other mechanisms specific to the *IEEE 802.11* type. Therefore, this work is not directly applicable to work in the RHNC, which require the discovery of heterogeneous, and resource constrained network elements.

All *IEEE 802.15* devices use the asymmetrical inquiry procedure, a pull mechanism, to discover nearby devices or to be discovered by devices in their locality. *IEEE 802.16* also uses a pull mechanism. Like the pull mechanism of the LLC sublayer standard, the pull mechanism of *IEEE 802.15* and *IEEE 802.16*, don't completely solve the problems presented in the RHNC.

The *IEEE 802.21-2008* media independent information service (MIIS) for all *IEEE 802* networks provides a query/response type of pull mode mechanism for transfer of information elements (IEs). It also supports a push mode that requires operator intervention. Pull mode mechanisms may not be applicable in contexts with long round trip times, in which types of push mode mechanisms are required. Additionally, such operator intervention may not be possible and consequently prohibit use of push mode mechanisms. In contrast, the work in this thesis uses push mode mechanisms that do not require operator intervention.

The main goal behind the *IEEE 802.21-2008* Information Service is to allow mobile node and network entities to discover information that influences the selection of appropriate networks during handovers. This information is intended to be primarily used by a policy engine entity that can make effective handover decisions based on this information. Dynamic information describes different access networks, such as current available resource levels, state parameters, and dynamic statistics. In contrast, a goal of the work in this thesis is to provide appropriate elements of IEEE 802.21-2008 information, and any other appropriate information, to enable a policy engine entity make effective handover decisions.

There are several elements loosely defined in *IEEE 802.21-2008*, that are particularly significant to our work, but whose specification are considered outside the scope of *IEEE 802.21-2008*. Information can be present in an information server from where the media independent handover function in the mobile node accesses it. However, the definition of and indexing of such a local database, as well as the regime for maintaining it or accessing it, are outside of *IEEE 802.21-2008*'s scope. In some contexts, information cannot be accessed at layer 2, or the information available at layer 2 is not sufficient to make an intelligent handover decision. In such cases information can be accessed via higher layers yet how this information is developed and deployed is not specified as it also falls outside of the scope of the standard. In contrast, this thesis provides the structure, portable representation, and the storage of such structured information. Assumptions are made by some IEEE-SA standards that limit the capabilities of the technology itself. For example, in *IEEE 802.15.1-2005*, a formalisation of Bluetooth, it may be possible to exploit devices to connect simultaneously to more than one physical channel in order to transfer ADUs, however, *IEEE 802.15.1-2005* assumes a device is capable of connecting to only one physical channel at any time.

Table 2.4 summarises the capabilities of the IEEE-SA discovery mechanisms in relation to our requirements. The table provides the IEEE-SA number, title, and six columns R.1 through R.6, which are used to indicate the relationship of each specification to our protocols requirements that are provided in Table 1.2. An X in column R1 indicates support of the RHNC. An X in column R.2 through R.6, indicates the type of discovery information.

Number	**Title**	**R.1**	**R.2**	**R.3**	**R.4**	**R.5**	**R.6**
IEEE 802.2	Logical Link Control (LLC)		X	X	X	X	X
IEEE 802.1	Higher Layer LAN Protocols		X	X	X	X	X
IEEE 802.11	Wireless LAN (WLAN)		X	X		X	X
IEEE 802.15	Wireless Personal Area Network (WPAN)		X	X	X	X	
IEEE 802.16	Broadband Wireless Access					X	
IEEE 802.21	Media Independent Handover Services		X	X	X	X	X

Table 2.4: IEEE 802 discovery summary. Columns R.1 through R.6, are used to indicate the relationship of each specification to our protocols requirements

2.1.4 IETF and associated organisations

The IETF develops new Internet standard specifications. The Internet Research Task Force (IRTF), an organisation associated with the IETF, focuses on longer term research issues related to the Internet while the IETF, focuses on the shorter term issues of engineering and standards making.

The standards and protocols of the IETF and IRTF relating to discovery in the contexts of *IEEE 802* networks are reviewed in the following.

IETF and IRTF discovery review

First, we provide a broad overview of the framework and methods of the Network Configuration Protocol (NETCONF). Configuration of networks of devices is a requirement for network operators. Operators have developed bespoke mechanisms or have used vendor specific mechanisms to transfer configuration data to and from a device and to examine device state information which may impact the configuration. Each of these mechanisms may be different in various aspects, such as session establishment, user authentication, configuration data exchange, and error responses. NETCONF provides a standard framework and a set of standard methods to manipulate the configuration of a network device. The NETCONF protocol contains several standard operations which operate on one or more conceptual configuration databases. NETCONF uses structured schema-driven data. There are a core set of operations that must always be supported by the server. All NETCONF operations are carried out within a session, which is tied to the transport layer connection.

In the following we provide tables to summarise the mechanism capabilities, which the specifications that address discovery from 1990 to 2012 detail, in relation to our requirements. Each table provides the RFC number or Internet Draft ID, title, and six columns R.1 through R.6, which are used to indicate the relationship of each specification to our protocols requirements that are provided in Table 1.2. An X in column R1 indicates support of the RHNC. An X in column R.2 through R.6, indicates the type of discovery information. Only standards with an X in two or more columns are provided. A summary of the characteristics of the IETF RFCs and Internet Drafts is provided in Section 2.1.4.

Standards from 2009 to 2012 are summarised in Table 2.5, standards from 2007 to 2009 are summarised in Table 2.6, and standards from 1990 to 2007 are summarised in Table 2.7.

Number	**Title**	**R.1**	**R.2**	**R.3**	**R.4**	**R.5**	**R.6**
RFC6408	Diameter Straightforward-Naming Authority Pointer (S-NAPTR) Usage			X		X	
RFC6186	Use of SRV Records for Locating Email Submission/Access Services			X		X	
RFC6121	Extensible Messaging and Presence Protocol (XMPP): Instant Messaging and Presence		X	X	X	X	X
RFC6120	Extensible Messaging and Presence Protocol (XMPP): Core		X	X	X	X	X
RFC6116	The E.164 to Uniform Resource Identifiers (URI) Dynamic Delegation Discovery System (DDDS) Application (ENUM)			X		X	

Table 2.5: IETF Standards Track discovery mechanisms 2009 - 2012

Number	**Title**	**R.1**	**R.2**	**R.3**	**R.4**	**R.5**	**R.6**
RFC5121	Transmission of IPv6 via the IPv6 Convergence Sublayer over IEEE 802.16 Networks			X		X	
RFC4972	Routing Extensions for Discovery of Multiprotocol (MPLS) Label Switch Router (LSR) Traffic Engineering (TE) Mesh Membership					X	X
RFC4861	Neighbor Discovery for IP version 6 (IPv6)			X		X	
RFC4848	Domain-Based Application Service Location Using URIs and the Dynamic Delegation Discovery Service (DDDS)			X		X	
RFC4610	Anycast-RP Using Protocol Independent Multicast (PIM)					X	X

Table 2.6: IETF Standards Track discovery mechanisms 2007 - 2009

Number	Title	R.1	R.2	R.3	R.4	R.5	R.6
RFC4605	Internet Group Management Protocol (IGMP) / Multicast Listener Discovery (MLD)-Based Multicast Forwarding (IGMP/MLD Proxying)					X	X
RFC4604	Using Internet Group Management Protocol Version 3 (IGMPv3) and Multicast Listener Discovery Protocol Version 2 (MLDv2) for Source-Specific Multicast					X	X
RFC4204	Link Management Protocol (LMP)			X		X	
RFC4095	Attaching Meaning to Solicitation Class Keywords			X		X	
RFC3958	Domain-Based Application Service Location Using SRV RRs and the Dynamic Delegation Discovery Service (DDDS)			X		X	
RFC3822	Finding Fibre Channel over TCP/IP (FCIP) Entities Using Service Location Protocol version 2 (SLPv2)			X		X	
RFC3810	Multicast Listener Discovery Version 2 (MLDv2) for IPv6					X	X
RFC3590	Source Address Selection for the Multicast Listener Discovery (MLD) Protocol					X	X
RFC3404	Dynamic Delegation Discovery System (DDDS) Part Four: The Uniform Resource Identifiers (URI)			X		X	
RFC3403	Dynamic Delegation Discovery System (DDDS) Part Three: The Domain Name System (DNS) Database			X		X	
RFC3402	Dynamic Delegation Discovery System (DDDS) Part Two: The Algorithm			X		X	
RFC2710	Multicast Listener Discovery (MLD) for IPv6					X	X
RFC2608	Service Location Protocol, Version 2			X		X	

Table 2.7: IETF Standards Track discovery mechanisms 1990 - 2007

The Experimental standards and protocols of the IETF and IRTF relating to discovery, the structuring of information and the exchange of structured information are reviewed next. The IETF Experimental standards that address same are outlined in Table 2.8.

Number	Title	R.1	R.2	R.3	R.4	R.5	R.6
RFC5780	NAT Behavior Discovery Using Session Traversal Utilities for NAT (STUN)			X	X	X	
RFC5523	OSPFv3-Based Layer 1 VPN Auto-Discovery			X		X	
RFC4624	Multicast Source Discovery Protocol (MSDP) MIB					X	X
RFC3832	Remote Service Discovery in the Service Location Protocol (SLP) via DNS SRV			X		X	
RFC3618	Multicast Source Discovery Protocol (MSDP)					X	X

Table 2.8: IETF Experimental discovery mechanisms 2000 - 2012

Discovery mechanisms defined by the The Internet Drafts of the IETF or other organisations associated with the IETF are reviewed in the following. Table 2.10 and Table 2.9 provide the set of Internet Drafts under review.

Number	Title	R.1	R.2	R.3	R.4	R.5	R.6
draft-lynn-core-discovery-mapping-02	CoRE Link-Format to DNS-Based Service Discovery Mapping			X		X	
draft-kist-alto-3pdisc-01	Third-Party ALTO Server Discovery (3pdisc)			X		X	
draft-ietf-p2psip-base-23	REsource LOcation And Discovery (RELOAD) Base Protocol			X		X	
draft-irtf-dtnrg-ipnd-02	DTN IP Neighbor Discovery (IPND)	X		X		X	
draft-cao-mif-srv-dis-ps-01	Service Discovery in a Multiple Connection Environment: Problem Statement			X		X	
draft-ietf-behave-nat64-discovery-heuristic-13	Discovery of the IPv6 Prefix Used for IPv6 Address Synthesis			X		X	
draft-ietf-alto-server-discovery-06	ALTO Server Discovery			X		X	

Table 2.9: Internet Draft discovery mechanisms (first table)

Number	Title	R.1	R.2	R.3	R.4	R.5	R.6
draft-zheng-sdn-discovery-00	SDN Discovery			X		X	
draft-vanderstok-core-dna-02	CoRE Discovery, Naming, and Addressing			X		X	
draft-jimenez-p2psip-coap-reload-02	Constrained Application Protocol (CoAP) Usage for REsource LOcation And Discovery (RELOAD)			X		X	
draft-bertrand-cdni-footprint-discovery-01	CDN Footprint Discovery			X		X	

Table 2.10: Internet Draft discovery mechanisms (second table)

While NETCONF provides a generic framework, and a set of generic methods to manipulate the configuration of a network device, it does so using Extensible Markup Language (XML) structured schema, and mechanisms are bound to the transport layer connection. In contrast, the application domain of the work in this thesis, calls for a generic format that is suitable for the portable representation of structured data and mechanisms that are not coupled to the transport layer.

The discovery protocols reviewed are defined by a variety of terms, such as use, operations, mechanisms, information, persistence and exchange. The protocols generally define a specific use. Multicast Listener Discovery (MLD) is used by IPv6 routers to discover the presence of multicast listeners. The Dynamic Delegation Discovery System (DDDS) is the method used to discover an authoritative server. The MANET Neighborhood Discovery Protocol (NHDP) is specified in terms of its objectives. The reviewed specifications generally detail a set of mechanism operations, such as procedures, processes, methods and exchanges, that deal with a specific problem. Such operations include the procedure to discover the ID of a remote Simple Network Management Protocol (SNMP) engine. The operation of IPv6 and its discovery mechanisms over the *IEEE Std 802.16* air interface described by RFC 5121. A method for provisioning a shared key from the Access Router to the Mobile Node. Constrained Application Protocol (CoAP) usage involves three basic functions. The Aggregated Service Discovery Protocol (ASDP) process is described. A procedure for third-party Application-Layer Traffic Optimization (ALTO) server discovery is described. Use of SD in a multiple-interfaces (MIF) environment is described in terms of the general problems presented and the requirements to solve them. The ALTO server discovery is described in terms of the procedure. The process for finding Fibre Channel over TCP/IP (FCIP) Service Discovery is described. An overview of the Service Location Protocol version 2 (SLPv2) process is described. An option is defined for SLPv2 operations which are defined primarily for interactive use. The method by which DHCP Messages are exchanged between clients and servers is described. Resolution operations are described, such as those of RFC4861, draft-cao-mif-srv-dis-ps-01, draft-cheshire-dnsext-multicastdns-15. Addressing operations are described, such as those of RFC6775, RFC6059, RFC5121 and RFC6496. Naming operations are described, such as those in RFC4171, draft-cheshire-dnsext-dns-sd-11, RFC4095, and RFC3972. Operations for message transfer described include the "pull" operation, such as that provided by an RFC2710 MLD "querier" solicitation, or an RFC6690 "GET" request for a resource made for "/.well-known/core" or the client configuration method of Decoupled Application Data Enroute (DECADE). The "push" operations are described, such as an RFC6690 "POST", the client configuration method of DECADE, Internet MANET Encapsulation Protocol (IMEP) broadcasts, and the presence element of Extensible Messaging and Presence Protocol (XMPP).

Protocols generally deal with a specific problem. The set of mechanisms of the Neighbor Discovery protocol (NDP) for IP Version 6 are devised to support interaction between nodes attached to the same link. The Diameter base protocol specifies mechanisms to advertise Diameter nodes and the supported transport protocol. Multicast Source Discovery Protocol (MSDP) is a mechanism to connect multiple IP Version 4 Protocol Independent Multicast Sparse-Mode (PIM-SM) domains together. The RD mechanisms utilised by the Web Proxy Auto-Discovery protocol (WPAD) are described. RFC4761 described the mechanisms for the auto-discovery of the endpoints of a Virtual Private LAN Service (VPLS) and for signaling a VPLS.

The types of structured information described includes the XML stanza, objects. The basic protocol data unit in XMPP is an XML stanza. XMPP is for enabling asynchronous, end-to-end exchange of structured data by means of direct, persistent XML streams. The Internet Storage Name Service (iSNS) objects are described. The iSNS protocol messages are defined and iSNS Attributes are defined in groups as described. An overview of the building blocks for discovery of information on the set of end-users, to which content can be delivered in Content Distribution Network Interconnection (CDNI) is provided. The "Hello" messages of NHDP are used to determine the presence of, and connectivity to neighbours.

The information exchange formats described include messages, beacons, frames and packets. These formats are mostly used to solve a set of specific problems. The five different The Internet Control Message Protocol (ICMP) packet types NDP provides for solicitation advertisement and redirect are described. The beacon messages of the DTN Internet protocol neighbor discovery provide to learn of the existence, availability, and IP addresses. A format is specified for the Diameter Straightforward-Naming Authority Pointer (S-NAPTR) to provide discovery of the supported applications without doing Diameter capability exchange. Multicast Router Discovery consists of three messages for discovering multicast routers which are described.

Persistence is generally described for specific data, such as storage via DNS as detailed by RFC6116 , the intelligent storage provided by iSNS, or relationship information that is stored or retrieved in draft-cao-mif-srv-dis-ps-01. NDP hosts maintain an NDP Neighbor Cache. A MSDP speaker caches MSDP messages.

NDP was conceived with the assumption that it would run over *IEEE 802.3* MAC protocol and the *IEEE 802.11* MAC protocol, which is like the *IEEE 802.3* protocol. However, the structure of *IEEE 802.16* differs greatly from the *IEEE 802.3* or *IEEE 802.11* protocol. Problems regarding NDP over *IEEE802.16* are described elsewhere (Lee et al., 2006).

IETF and IRTF discovery summary

Discovery standards of the IETF and IRTF are classified in Table 2.11 in order to provide an overview of the relationship of the standard, in terms of it's contribution to our requirements.

The discovery protocols reviewed are defined by their use, mechanisms, operations, information, persistence and information exchange format. Specifications share common characteristics. Specifications generally detail a set of mechanism operations, mechanisms, types of structured information, information exchange formats and approach to persist structured information in order to deal with a specific problem, in a specific context. Most specifications detail configuration, neighbour, resource, service, DNS mapping and route types of discovery which correspond to the capabilities and routing discovery requirements "R.3" and "R.5".

While the various discovery specifications of the IETF and IRTF provide information that contributes to aspects of the requirements "R.2" through "R.6", only the discovery type "streaming of structured information" which XMPP represents, is capable of fulfilling all aspects of these requirements. Additionally, "DTN Neighbour discovery" which "draft-irtf-dtnrg-ipnd-02" represents, is the only specification that fulfils the requirement "R.1".

Table 2.12 shows how our requirements map to the discovery mechanism types of the IETF and IRTF standards.

2.1.5 Alternatives to IETF work

In the following we review the work of other organisations. First we provide a broad overview of the relationship of prominent wireless networking technologies to the *IEEE-SA* standards that we have reviewed. Then we review the approach used to discover information.

WiFi refers to interoperable implementations of the *IEEE 802.11* family of wireless-networks standards certified by the Wi-Fi Alliance (URL, 9). Bluetooth wireless technology defines a wireless communication link, operating in the unlicensed industrial, scientific, and medical (ISM) band at 2.4 GHz using a frequency hopping transceiver. In 2002 IEEE approved the *IEEE 802.15.1* specification. ZigBee and *IEEE 802.15* combined efforts to provide the low-rate wireless personal area network (LR-WPAN) standard called *IEEE 802.15.4*. WiMAX refers to interoperable implementations of the *IEEE 802.16* family of wireless-networks standards ratified by the WiMAX Forum (URL, 8). The WiMAX Forum is an organisation that certifies the compatibility and interoperability of broadband wireless products based upon *IEEE 802.16* standard.

Table 2.13 presents the discovery mechanisms reviewed in this section, which are broadly summarised in the following.

The Link Layer Topology Discovery (LLTD) protocol, OpenFlow Discovery Protocol (OFDP), Cisco Discovery Protocol (CDP), Nortel Discovery Protocol, and Extreme Discovery Protocol are Link Layer multicast based discovery protocols used to discover neighbours and topology. MikroTik Neighbor Discovery protocol (MNDP) described a UDP based broadcast ND protocol. The Oasis Identity Provider Discovery Service Protocol (IPDSP) defines centralised SD that encompasses three steps, including two message exchanges. The Jboss community (URL, 12) Mod Cluster Discovery Protocol (MCDP) provides node discovery using a pull mechanism. The discovery management interfaces of Jini for Java objects. Really Simple Discovery (RSD), Boot Service Discovery Protocol (BSDP), multicast based Local Peer Discovery protocol (LPDP), and the link layer based Discovery and Discovery and Basic Configuration Protocol (DCP) provide configuration discovery. ADDP provides UDP response based device discovery. Intelligent Platform Management Interface (IPMI) is designed to provide platform resident discovery mechanisms and defines the commands for discovering IPMI-based systems. The IPMI specification defines the messages and system interface to platform management hardware comprises three specifications; the Intelligent Platform Management Interface, the Intelligent Platform Management Bus (IPMB) and the Intelligent Chassis Management Bus (ICMB).

Universal Description Discovery and Integration (UDDI) defines a universal method for enterprises to dynamically discover and invoke Web services. The Universal Plug and Play (UPnP) architecture is operating system, programming language, or network technology agnostic and uses XML to structure information to support Zeroconf and automatic discovery. XMPP is a protocol for streaming XML elements.

The Web Services Dynamic Discovery (WS-Discovery) protocol specifies multicast based service location discovery. Bluetooth Service Discovery Protocol (SDP), part of the Bluetooth Host stack announces and specifies all the other upper layer features such as the RFCOMM channels or port names. Presence is a SD/announcement protocol. The Yadis Protocol provides an HTTP based SD, in which the response is one of four. Bonjour and Avahi, implementations of Zeroconf provide SD.

Other organisations summary

The discovery protocols of other organisations reviewed are defined by a variety of terms, such as use, operations, mechanisms, information, persistence and exchange

LLTD, OFDP, CDP, NDP, EDP and MNDP provide vendor specific link layer discovery protocols, and are therefore not generally applicable. This also applies to GDP, which provides for vendor specific router discovery. Centralised SD such as that provided by the IPDSP do not work when the network is partitioned. Because of this and other limitations of ad-hoc wireless communication, described in Chapter 1, centralised solutions requiring all the necessary information to be available at a single place are not considered further, and only decentralised solutions are considered. Pull mechanisms, such as MCDP, RSD, BSDP, LPDP, DCP and ADDP, requiring information to be provided in response to a probe or request, may not work well in the constrained or partitioned networks that are the application domain of this thesis, and push approach has been shown to provide better packet delivery performance (Majumder et al., 2011). The IPMI sub-system operations are not applicable to the operating system, and are consequently capable of providing limited information.

XML is a format derived from SGML, in order to be "more suitable for Web use"(URL, 15). Protocols that are not for Web use, which use XML to structure information, such as UDDI, UPnP, XMPP may be difficult to implement and manage. Furthermore, protocols which stream XML are subject to real time constraints. Service discovery protocols such as WS-Discovery, and UPnP are unable to discover services if the producer moves to another network. The majority of the SD or RD systems reviewed involve peer-to-peer overlays with a distributed architecture that use response or pull based mechanisms, that are largely targeted towards large-scale use in the fixed Internet. In contrast, the work in this thesis, caters for ad-hoc networking scenarios in which it is not sufficient to simply turn on inquiry periodically.

While a large proportion of the work in this area deals with specific (aspects of) discovery, the goal of this thesis is to provide a generic solution to the discovery of autonomous mobile network elements Finally, none of these mechanisms, offer explicit support for the discovery of network elements, i.e., offer laws that network elements can use to describe themselves for the purpose of discovery.

Discovery standards of the other organisations are classified and mapped to our requirements in Table 2.14.

2.1.6 Analysis

While the problems addressed by the work reviewed are similar to the ones dealt with by this thesis, the work in these communities deal mostly with discovery mechanisms focused on near real-time, low overhead, discovery of network elements, i.e., the provision of appropriate standards, protocols and models that assume the characteristics of communication provided by *IEEE 802* networks to provide continuous end-to-end connectivity and negligible latency, i.e., transit delays, on the order of a few milliseconds or less. Therefore, this work does not completely solve the challenges in providing interoperable communications with and among extreme and performance-challenged environments where continuous end-to-end connectivity cannot be assumed.

A notion formalised by the this thesis is the discovery of the relationships between things. The IEEE state that the discovery of the relationships between objects is an essential concept, as discovery of the relationships of network elements facilitates informed decision making. A significant proportion of this work specified discovery of representations of information that are tightly coupled with underlying communications mechanisms, which might be reasonable in the constrained and secure environments in which web resources are typically deployed, but would not be appropriate for heterogeneous networks where it is necessary to decouple information from its producer in order to exchange it between different consumers or producers.

Other significant contributions that that are capable of representing any element so as to fulfil our requirements "R.2" through "R.6" and provide a means to exchange structured information between any two network endpoints, might not work well in contexts they were not intended. For example XMPP is designed to stream XML over TCP in close to real time. While the close to real time constraint is not appropriate to our requirements, neither is XML as it introduces a tight coupling between the representation and the processing program. While XEP-0030, the XMPP protocol extension, is not tighlty coupled with TCP it uses XML. IPNDs specification states the principles and basic mechanisms used may also be expressed in terms of other datagram protocols. However, ultimately IPND specifies a ND protocol whose principles and basic mechanisms are described in terms of IP bound CLAs and datagrams that contain IP information. Beaconing methods are expressed in terms of IP unicast, broadcast and multicast. In contrast, the work in this thesis caters for a discovery framework designed to exchange structured information, that is independent of producer or consumer, between any two heterogeneous network endpoints, and over any convergence layer. Several SD protocols such as SLP, WS-Discovery, and UPnP are unable to discover services if the producer moves to another network.

Discovery mechanisms span a wide range of domains. We class the discovery mechanisms in terms of the type of information they contribute to our consumption, capability, relationship, routing and distribution requirements “R.2” through “R.6” and whether they support the RHNC “R.1”. Table 2.12 shows the types of information contributed by the reviewed discovery mechanism to our information requirements and indicates the mechanisms capable of supporting the RHNC.

The discovery mechanisms can be symmetric or asymmetric and discovery methods are push and pull. The push and pull modes use layer 2 broadcast or IP broadcast, unicast, and multicast mechanisms. In lower layer mechanisms, usually beacon frames are periodically transmitted and responded to with probe response frames. All mechanisms specify a structure of information. Structures provided range from a simple frame capable of representing a specific element only to formalised frameworks capable of representing any element so as to fulfil our requirements “R.2” through “R.6”. All mechanisms specify a set operations to act on a structure of information. Table 2.15 provides a summary of the characteristics of reviewed mechanisms.

An example of how these characteristics are used to analyse protocols is provided in Table 2.16.

Discovery approaches, based on the the proactive push approach have shown better packet delivery performance than the reactive pull approach. Learning the presence of another node is not sufficient and higher layer SD is needed to correlate common services, which consume contact volume. The TDP conceptual models, TDP and the TDP modules, provide a push mode, asymmetric mechanism, that provides a set operations to act on a simple structure of information, that is capable of representing any network element.

In this section we reviewed the general area of discovery within *IEEE 802* networks.

Discovery Type	**Requirement**	**RFC / Draft**
Endpoint discovery	R.5	RFC4761
Device discovery	R.3	RFC4171
Server discovery	R.5	draft-reddy-pcp-server-discovery-00
Node discovery	R.5	draft-dhody-pce-bn-discovery-ospf-05, draft-dhody-pce-bn-discovery-isis-05
Router discovery	R.5	RFC4286
DTN Neighbour discovery	R.1, R.3, R.5	draft-irtf-dtnrg-ipnd-02
Neighbour discovery	R.5	RFC5380, RFC6775, RFC3684, RFC5614, RFC4429, RFC3122, draft-kaiser-nd-pd-00, draft-korhonen-mif-ra-offload-05, draft-smith-6man-mitigate-nd-cache-dos-slnd-05, draft-chakrabarti-nordmark-6man-efficient-nd-01, draft-ietf-manet-zone-zrp-04
Secure neighbour discovery	R.5	RFC3972, RFC3971 RFC6496, RFC6495, RFC6494, RFC5269, RFC5195
Proxy discovery	R.5	RFC3608, RFC4389, draft-cao-core-pd-02, draft-ietf-wrec-wpad-01, draft-he-core-energy-aware-pd-01,
Detection using discovery messages	R.5	(RFC4957) RFC6059
Path computation element discovery	R.5	RFC5089, RFC5088
Link layer discovery	R.3, R.5	RFC5121, RFC4861, RFC4204, draft-ietf-manet-imep-spec-01
Neighbourhood discovery	R.5	RFC6130
IP discovery	R.5	RFC1256, RFC4066
Multicast listener discovery	R.5, R.6	RFC3810, RFC2710, RFC3590
Multicast reciever interest discovery	R.5, R.6	RFC4604
Multicast source discovery	R.5, R.6	RFC3618, RFC4610, RFC4624
Identifier discovery	R.3	RFC5343
Membership discovery	R.5, R.6	RFC4605, RFC4972
Configuration discovery	R.3	RFC2131, RFC6074, RFC6610, RFC6440, RFC6153, RFC5678, draft-pentikousis-decade-discovery-00, draft-ietf-zeroconf-reqts-12
Capabilities discovery	R.3	RFC4579, RFC5073
Resource Discovery	R.3	RFC6690
Resource and location discovery	R.3, R.5	draft-ietf-p2psip-base-23, draft-jimenez-p2psip-coap-reload-02, draft-vanderstok-core-dna-02, draft-lynn-core-discovery-mapping-02
Service discovery	R.3, R.5	RFC3822, draft-cheshire-dnsext-multicastdns-15, draft-cheshire-dnsext-dns-sd-11, draft-ietf-p2psip-service-discovery-06, draft-daboo-aggregated-service-discovery-01, draft-cai-ssdp-v1-03, draft-cao-mif-srv-dis-ps-01
Service location discovery	R.3, R.5	RFC2608
Location discovery	R.3, R5	draft-ietf-geopriv-res-gw-lis-discovery-04, draft-jones-simple-web-discovery-04
Route discovery	R.5	RFC3561, RFC4728, RFC5252, RFC6625, RFC5790, draft-ietf-roll-p2p-rpl-14
Session discovery	R.5	RFC2974
Behaviour discovery	R.2, R.3, R.5	RFC5780
Footprint discovery	R.3, R.5	draft-bertrand-cdni-footprint-discovery-01
Path MTU discovery	R.5	RFC4821, RFC1981, RFC1191
Streaming of structured information	R.2, R.3, R.4, R.5, R.6	RFC6120, RFC6121
DNS mapping	R.3, R.5	RFC6186, RFC6116, RFC5523, draft-zheng-sdn-discovery-00, RFC6408, RFC4095, RFC3403, RFC3404, RFC3402, RFC4848, RFC3958, RFC3832 draft-kist-alto-3pdisc-01, draft-ietf-behave-nat64-discovery-heuristic-13, draft-ietf-alto-server-discovery-06

Table 2.11: Classification of discovery types of the IETF and IRTF between November 1990 to October 2012. Requirements columns relates discovery type to a requirement number

Requirement	Discovery	Discovery type
R.1	RHNC	DTN Neighbour
R.2	Consumption	Behaviour, Streaming of structured information
R.3	Capability	Device, Link layer, Identifier, Configuration, Capabilities, Resource, Service, Service location, Location, Resource and location, Behaviour, Footprint discovery, Streaming of structured information, DTN Neighbour
R.4	Relationship	Streaming of structured information
R.5	Routing	Endpoint, Server, Node, Router, Neighbour, Secure neighbour, Neighbourhood, Proxy, Detection using discovery messages, Path computation element, Link layer, IP, Multicast listener, Multicast reciever interest, Multicast source, Membership, Route, Service location, Session, Behaviour, Footprint, Path MTU, Streaming of structured information, Resource and location, DTN Neighbour
R.6	Distribution	Multicast listener, Multicast reciever interest, Multicast source, Membership, Streaming of structured information

Table 2.12: Mapping discovery types of the IETF and IRTF standards (between November 1990 to October 2012) to the requirements

Title	R.1	R.2	R.3	R.4	R.5	R.6
UDDI			X		X	
SDP			X		X	
WS-Discovery			X		X	
XRD		X	X	X	X	X
IPDSP			X		X	
Jini			X		X	
Cache Discovery Protocol (CDP)			X	X	X	X
RSD			X		X	
Presence			X		X	
Yadis			X		X	
XMPP		X	X	X	X	X

Table 2.13: Other discovery mechanisms

Discovery Type	Requirement	Protocol
Service discovery	R.3, R.5	
Information discovery	R.2, R.3, R.4, R.5, R.6	XEP-0030
Configuration discovery	R.3	RSD , IPMI, DCP, ADDP, Local Peer Discovery protocol
Topology discovery	R.5	CDP, NDP, GDP, LLTD, OFDP, MNDP, Extreme Discovery Protocol
Resource and relationship discovery	R.2, R.3, R.4, R.5, R.6	XRD
Node discovery	R.5	MCDP
Cache Discovery	R.3, R.5, R.6	Cache Discovery Protocol

Table 2.14: Classification of discovery types of other organisations. Requirements columns maps discovery type to a requirement number

Derived characteristcs	**Example**
Operations	pull mode (i.e, probe beacons, inquiry, request, subscribe, register, deregister, etc..), push mode (i.e., probe response, inquiry scan, advertise, announce, publish, etc..)
Structured information	MIB, XML, XRD, etc..
Mechanisms	Naming facility (resolution, addressing, assignment etc..)
Persistence	Directory, storage, cache, databases, etc..
Exchange format	Beacon, probe, packet, message, etc..

Table 2.15: Characteristics of discovery mechanisms

	XMPP	**IPND**	**NDP**
Operations	Push (Message) Push (Presence) Pull (Request-response),	Push (UDP Advertisement / Announcement)	Push (Advertisement) / Pull (Solicitation)
Structured information	XML stanza	Service block	ICMP message types
Mechanisms	Setup, Teardown Channel encryption Authentication Error handling Network availability Request-response interactions etc ..	Service Definition(s) Service Block Tag-Length-Value (TLV) Neighborhood Bloom Filter (NBF) IGMP group membership etc . .	Router Discovery Prefix Discovery Parameter Discovery Address Auto-configuration Address resolution Next-hop determination Neighbor Unreachability Detection Duplicate Address Detection Redirect etc . .
Persistence	Persistent XML streams	Neighbour set	NDP Neighbor Cache
Exchange format	TCP message	UDP messages	ICMP packet

Table 2.16: XMPP, IPND and NDP are used to provide an example of how the derived characteristics are used in analysis discovery protocols.

2.2 Structured information

Several working and research groups of the IEEE, IETF, the World Wide Web Consortium (W3C), and other organisations have investigated frameworks for representing information. We first present work that addresses only the mechanisms for structuring information used by the specific discovery protocols reviewed in Section 2.1. We then provide an analyis of the characteristics of the mechanisms.

2.2.1 Review

Firstly we review the information structures of the W3C. Secondly we review the information structures of other organisations. Thirdly, we review the information structures of the IETF and associated organisations. Finally we review the information structures of the IEEE-SA specifications.

XML & RDF

The Standard Generalised Markup Language (SGML) (URL, 14) (ISO 8879) is a set of formalisms that facilitates the definition of descriptive markup languages for the purpose of encoding and transfer of structured information. HTML and XML are based on SGML. Information objects modelled through an SGML markup language are named and described using attributes and sub-elements. XML is a text format derived from SGML, in order to be more suitable for Web use.

The Resource Description Framework (RDF) is a framework for representing information in the Web. RDF can use values represented according to XML schema data-types for the exchange of information between RDF and other XML applications. RDF provides a common framework for expressing information so that it can be exchanged between applications without loss of meaning. Since it is a common framework, application designers can leverage the availability of common RDF parsers and processing tools. The ability to exchange information between different applications means that the information may be made available to applications other than those for which it was originally created. RDF is based on the idea of identifying things using Uniform Resource Identifiers (URIs), and describing resources in terms of simple properties and property values. This enables RDF to represent simple statements about resources as a graph of nodes and arcs representing the resources, and their properties and values. This abstract syntax is quite unique compared to the set of definitions for use in other specifications that need to refer to the information in an XML document, i.e., XML's tree-based infoset. RDF has an abstract syntax that reflects a simple graph-based data model, and formal semantics with a rigorously defined notion of "entailment" providing a basis for well founded deductions in RDF data. A semantic extension of RDF is RDF Schema (RDFS). RDFS extends RDF to include a larger vocabulary with more complex semantic constraints. There is a specification of a precise semantics, and corresponding complete systems of inference rules, for the RDF and RDFS (URL, 102). A URI reference within an RDF graph is an RDF URI. A blank node is a unique node that can be used in one or more RDF statements, but has no intrinsic name. A *Class* in RDF is a general concept, category or classification. Something used primarily to classify or categorise other things. Formally a *Class* is a resource of type "$rdfs:Class$" with an associated set of resources all of which have the class as a value of the "$rdf:type$" property. Classes are often called *predicates* in the formal logical literature (URL, 15). RDF distinguishes class from the set, although the two are often identified. Distinguishing classes from sets is said to allow RDF more freedom in constructing class hierarchies. URI references are used for naming things in RDF which also has a recommended XML serialisation form which can be used to encode the data model for exchange of information among applications. The RDF framework allows anyone to make statements about any resource. The underlying structure of any expression in RDF is a collection of triples, each consisting of a subject, i.e., an RDF URI reference or a blank node, a predicate or property, i.e., an RDF URI reference that denotes a relationship, and an object, i.e., an RDF URI reference, a literal or a blank node. A set of such triples is called an RDF graph. An RDF graph can be illustrated by a node and directed-arc diagram, in which each triple is represented as a node-arc-node link. "Any assertion of a graph implicitly asserts that all the names in the graph actually refer to something in the world" (URL, 102). A subgraph of an RDF graph is a subset of the triples in the graph. A triple is identified with the singleton set containing it, so that each triple in a graph is considered to be a sub-graph. A proper sub-graph is a proper subset of the triples in the graph. A ground RDF graph is one with no blank nodes. RDF does not impose any logical restrictions on the domains and ranges of properties; in particular, a property "may be applied to itself" (URL, 102). When classes are introduced in RDFS, they may contain themselves. An RDF expression X is said to entail another RDF expression Y if every possible arrangement of things in the world that makes X true also makes Y true. On this basis, if the truth of X is presumed or demonstrated then the truth of Y can be inferred. Facts that indicate a relationship between two things may be represented as an RDF triple in which the predicate names the relationship, and the subject and object denote the two things. A representation of a fact might be as a row in a table in a relational database. The table has two columns, corresponding to the subject and the object of the RDF triple. The name of the table corresponds to the predicate of the RDF triple. A further familiar representation may be as a two place predicate in first order logic (URL, 103). Relational databases permit a table to have an arbitrary number of columns, a row of which expresses information corresponding to a predicate in first order logic with an arbitrary number of places. Such a row, or predicate, has to be decomposed for representation as RDF triples. A simple form of decomposition introduces a new blank node, corresponding to the row, and a new triple is introduced for each cell in the row. The subject of each triple is the new blank node, the predicate corresponds to the column name, and object corresponds to the value in the cell. The new blank node may also have an "$rdf:type$" property whose value corresponds to the table name.

UDDI

Universal Description Discovery and Integration (UDDI) defines an XML schema (URL, 16). Template structures are used represent individual Web services to provides the technical information needed by applications to bind and interact with the Web service being described. A template can be decorated with meta-data that enable its discovery given a set of parameters and criteria. The "discoveryURL" structure is a simple container of one or more "discoveryURL" elements. A "discoveryURL" is a URL that points to Web addressable, via HTTP GET, discovery documents.

XRD

The Extensible Resource Descriptor (XRD) Version 1.0 (URL, 17) OASIS standard describes a simple generic format for describing and discovering resources. The XRD schema defines only the elements necessary to support the most common use cases, with the intention that applications will extend XRD to include other meta-data about the resources they describe. XRD provides an XML format for describing a resource. An XRD document may describe the properties of the resource itself, as well as the relations the resource has with other resources. XRD uses URI-based identifiers for describing resources as well as for describing the relations between resources.

ASN.1

Abstract Syntax Notation One (ASN.1), provides a data-definition language for defining protocol syntax and information. ASN.1 can be used to describe the information that will be exchanged independent of the way that information is represented on each of the communicating systems. ASN.1 is an international standard for representing data types and structures and provides a syntax for specifying application layer protocols and information in open systems. The module is the basic unit in ASN.1, which consists of types and value definitions. A type definition is used to define and name a new type by means of a type assignment and a value definition is used to define and name a specific value, when it is necessary, by means of a value assignment. Syntax for built-in simple and structured types is specified and exemplified. Tagging enables the receiver to distinguish and correctly decode values from various data types. Additional features of the notation allow specification of subtypes, handling of recursion, and defining new types and values with macros.

XEP-0030

XEP-0030 (URL, 18) describes an XMPP protocol extension for discovering information about other XMPP entities. Any protocol defined in XEP-0030 has been developed outside the Internet Standards Process and is to be understood as an extension to XMPP rather than as an evolution, development, or modification of XMPP itself. The extension enables the identity and capabilities of an entity, including the protocols and features it supports; and the items associated with an entity to be discovered. A standards-track protocol for service discovery, as abbreviated to "disco", is specified to discover the target entity's identity. In disco (URL, 18), an entity's identity is broken down into its category and its particular type within that category, the features offered and protocols supported by the target entity and any additional items associated with the entity, whether or not they are addressable as XMPP addresses (JIDs)

NDEF

The Near Field Communication (NFC) Data Exchange Format (NDEF) (URL, 19) that is specified by the The Near Field Communication Forum, defines a message encapsulation format to exchange information. NDEF is a lightweight, binary message format that can be used to encapsulate one or more application-defined payloads of arbitrary type and size into a single message construct. Each payload is described by a type, a length, and an optional identifier. Type identifiers may be URIs, MIME media types, or NFC-specific types. This latter format permits compact identification of well-known types commonly used in NFC Forum applications, or self-allocation of a name space for organisations that wish to use it for their own NFC-specific purposes. The payload length is an unsigned integer indicating the number of octets in the payload. A compact, short-record layout is provided for very small payloads. The optional payload identifier enables association of multiple payloads and cross-referencing between them. NDEF records can encapsulate documents of any type. An NDEF message is composed of one or more NDEF records. There can be multiple records in a NDEF message. A NDEF message is array of NDEF records. The number of records that can be encapsulates in a NDEF message depends on the application and the tag type. The NDEF generator encapsulates each document in NDEF records as payload or chunked payload, indicating the type and length of the payload along with an optional identifier. The NDEF records are then put together to form a single NDEF message. The NDEF message is transmitted across an NFC link to another NFC Forum Device where they are received and parsed, or as an intermediate step, the message is written to an NFC Forum Tag. An NFC Forum Device brought close to this NFC Forum Tag will read the NDEF message from this tag and hand it over to the NDEF parser. The NDEF parser deconstructs the NDEF message and hands the payloads to a potentially different user application. Each NDEF message is sent or received in its entirety.

ISMF

The specifications of the Internet Standard Management Framework (ISMF) consists of a data definition language, definitions of management information (the MIB), a protocol definition, and security and administration(Case et al., 2002). The framework was designed with a protocol-independent data definition language and MIB along with a MIB-independent protocol. The original ISMF SNMPv1 is comprised of three RFCs. SNMPv1 data definition language is comprised of three RFCs and are often collectively referred to as “SMIv1”. RFC 1157, describes the SNMPv1 protocol operations performed by protocol data units (PDUs) on lists of variable bindings and describes the format of SNMPv1 messages. Layering of SNMP on a connectionless transport service is also defined. SNMPv1 by itself is simply a protocol for collecting and organising information. Most tool-sets implementing SNMP offer some form of IP discovery mechanism. RFC 1157 defines a simple protocol by which management information for a network element may be inspected or altered by logically remote users. Together with its companion documents, which describe the structure of management information, along with the management information base, these documents provide an architecture and system for managing TCP/IP-based Internets and in particular the Internet. The ISMF SNMPv2 is comprised of six RFCs. SNMPv2 provides several advantages over SNMPv1, in terms of data types, improved efficiency and performance, event notification, error handling, sets and the data definition language. SNMPv2 data definition language is comprised of one RFC and is often referred to as “SMIv2”. The third version of the Internet Standard Management Framework (the SNMPv3 Framework) is derived from and builds upon both SNMPv1 and SNMPv2. The SNMPv3 Framework augments those specifications with specifications for security and administration for SNMPv3. All versions (SNMPv1, SNMPv2, and SNMPv3) of the Internet Standard Management SNMP Framework share the same basic structure and components. Furthermore, all versions of the specifications of the ISMF follow the same architecture. The organisation of SNMPv3 ISMF architecture follows SNMPv1 and SNMPv2. The management protocol is used to convey management information between SNMP entities such as managers and agents. Management information is viewed as a collection of managed objects, residing in a virtual information store, termed the Management Information Base (MIB). Collections of related objects are defined in MIB modules. These modules are written in the SNMP data definition language, which evolved from an adapted subset of ASN.1. MIB modules can contain object, event notifications and compliance statement definitions. MIB modules define the management information maintained by the instrumentation in managed nodes, made remotely accessible by management agents, conveyed by the management protocol, and manipulated by management applications. MIB modules are defined according to the SMI as supplemented by the related specifications. Generally management information defined in any MIB module, regardless of the version of the data definition language used, can be used with any version of the protocol. Three documents collectively define the data definition language, specify the base data types for objects, specify a core set of short-hand specifications for data types called textual conventions, and specify a few administrative assignments of object identifier (OID) values. The SMI is divided into three parts.

NETCONF

NETCONF uses structured schema-driven data, and provides detailed structured error return information. The device configuration data, and the protocol itself, are encoded with XML. Standard XML tools such as XML Path Language (XPath) are used to provide retrieval of a particular subset configuration data. All NETCONF messages are encoded in XML within XML Namespaces. NETCONF messages are encoded in XML, using the UTF-8 character set. The NETCONF protocol contains several standard operations which operate on one or more conceptual configuration databases. Configuration databases are represented (in XML) within NETCONF operations as either an empty element identifying a standard database, or a "$< url >$" element identifying a non standard database. There are three standard conceptual configuration databases.

PTOPO

A Management Information Base MIB is a virtual information store. Most often associated with SNMP, the term is also used in other contexts such as in OSI/ISO Network management model. While a MIB can refer to the complete collection of management information available on an entity, it can be used to refer to a particular subset, called a MIB-module. Objects in the MIB are defined using SMIv1 or SMIv2 a subset of ASN.1. The software that performs the parsing is a MIB compiler. The database is hierarchical and each entry is addressed through an object identifier (OID). A SNMP MIB search engine, such as mibDepot (URL, 13) can be used to search for a MIB by MIB object name, trap name, OID, or substring thereof. The Physical Topology MIB (PTOPO-MIB) (Bierman and Jones, 2000) is a MIB module specified to provides a standardised means of representing the physical network connections pertaining to a given management domain that is compliant to the Structure of Management Information Version 2 (SMIv2). PTOPO-MIB provides a standard way to identify connections between network ports and to discover network addresses of SNMP agents containing management information associated with each port. A topology mechanism is used to discover the information required by the PTOPO-MIB. The scope of the physical topology (PTOPO) mechanism is the identification of connections between two network ports. Network addresses of SNMP agents containing management information associated with each port can also be identified. The *802.1AB-2009* standard defines the LLDP to provides compatibility with the PTOPO-MIB. NHDP-MIB, a MIB module is defined which reports state, performance information, and notifications about NHDP. This additional state and performance information is useful to troubleshoot problems and performance issues during neighbour discovery. MGMD MIB module used for managing the IGMP and the MLD protocol. Specific elements of a SEcure Neighbor Discovery (SEND) MIB (SEND-MIB) are used to monitor and configure most of the elements related with SEND operation.

SDAT

A data structure called the Simple Detecting Network Attachment (DNA) address table (SDAT) is maintained by a host for each interface on which it performs Simple DNA.

ENUM

ENUM is designed as a way to translate from E.164 numbers to URIs using NAPTR records stored in DNS. The basic protocol data unit in XMPP is an XML stanza; essentially a fragment of XML that is sent over a stream. The root element of a stanza includes routing attributes, and the child elements of the stanza contain a payload for delivery to the intended recipient. The basic syntax and semantics of XMPP were developed originally within the Jabber open-source community.

NDP Neighbour Cache

The NDP Neighbour Cache is a data structure that holds the cache of a set of entries about individual neighbours to which traffic has been sent recently. Entries are keyed on the neighbour's on-link unicast IP address and contain such information as its link-layer address to IP mappings for connected nodes.

DDDS

The Dynamic Delegation Discovery System (DDDS) functions by mapping some unique string to data stored within a DDDS Database by iteratively applying string transformation rules until a terminal condition is reached. A DDDS database uses the DNS as a distributed data store of DDDS rules in which the keys are domain-names and the rules are encoded using the NAPTR Resource Record (RR).

JSON

JavaScript Object Notation (JSON) (Crockford, 2006) is a text format for the serialisation of structured data. It was derived from the European Computer Manufacturers Associations ECMAScript Programming Language Standard itself based on several originating technologies, the most well known being JavaScript (Netscape) and JScript (Microsoft) JSON defines a small set of formatting rules for the portable representation of structured data. JSON can represent four primitive types, i.e., strings, numbers, booleans, and null, and two structured types, i.e., objects and arrays. An object structure is represented as a pair of curly brackets surrounding zero or more name or value pairs. An array structure is represented as square brackets surrounding zero or more values or elements. A JSON text is a serialised object or array. JSON text is a sequence of tokens for which the set of tokens includes six structural characters.

MIME

Multipurpose Internet Mail Extensions (MIME), redefines the format of RFC 822 messages to allow features. Unlike other mechanisms reviewed here MIME is not used by the reviewed discovery mechanisms.

Nevertheless in the context of structuring network information MIME provides a mature mechanism to, in a standardised way, handle arbitrary types of data and provide hierarchical structuring of entities of different types in a single message. RFC 2045 specifies the various headers used to describe the structure of MIME messages. RFC 2045 describes use of the MIME-Version, Content-Type, and Content-Transfer-Encoding header fields, it is possible to include, in a standardised way, arbitrary types of data with RFC 822 mail messages. RFC 2046 defines the general structure of the MIME media typing system and defines an initial set of media types. RFC 2046 describes the five top level media types provide provide a standardised mechanism for tagging entities as audio, image, or several other kinds of data. The composite multipart and message media types allow mixing and hierarchical structuring of entities of different types in a single message. A distinguished parameter syntax allows further specification of data format details, particularly the specification of alternate character sets. Additional optional header fields provide mechanisms for certain extensions deemed desirable by many implementers. Finally, a number of useful media types are defined for general use by consenting user agents. RFC 2047 MIME specifies a protocol for the representation of non-ASCII text in message headers. RFC 2048 specifies various IANA registration procedures for MIME-related facilities, specifically media types, external body access types, and content-transfer-encodings. And RFC 2049, which describes MIME conformance criteria as well as providing some illustrative examples of MIME message formats and acknowledgements.

DNS-SD

DNS-Based Service Discovery (DNS-SD) defines a conventional method of configuring DNS PTR, SRV, and TXT resource records. DNS-SD service names are limited to 255 octets and are formatted as three elements. These three elements comprise a unique name for an SRV/ TXT record pair within the DNS subdomain.

COAP

A CoAP Function Set is a service subtype with a set of resources and behaviours that may be accessed through a REST interface. A Function Set is typically described by a base URI plus an interface definition. The interface definition may specify the naming patterns of subordinate resources and the methods that act on them. End-point grouping can be used to express that end-points are related by supporting a specific Function Set. The Function Set specified in a message can contain a collection of resources of the same subtype. Boundry Node (BN) of thee Open Shortest Path First (OSPF) routing protocol discovery information is composed of the IPv4 and/or IPv6 address.

RELOAD

RELOAD provides a set of generic mechanisms for storing and retrieving data in the Overlay Instance. The basic unit of stored data is a single "StoredData" structure. Each Resource-ID specifies a single location in the Overlay Instance. Each service provided in the RELOAD Overlay Instance has its own tree. The nodes in a Recursive Distributed Rendezvous (ReDiR) service discovery mechanism tree contain pointers to service providers. Each service provided in the overlay is identified by an arbitrary identifier, called its namespace. Each tree node in the ReDiR tree contains a list of Node-IDs of peers providing a particular service. Each node in the tree has a level. The root is at level 0, the immediate children of the root are at level 1, and so forth. There are eight nodes each of which is responsible for one eight of the identifier space. ReDiR tree nodes are stored using the dictionary data model defined in RELOAD base. The data stored is a type of Resource Record. All state in the ReDiR tree is “soft” and therefore can be regenerated or replaced as needed..

CoRE

CoRE defines a format for use by constrained web servers to describe hosted resources, their attributes, and other relationships between links. An RFC 5988 link is a typed connection between two resources identified by URI and is made up of a context URI, a link relation type, a target URI, and optional key/value pairs that describes the link or its target attributes. The CoRE Link Format extends the HTTP Link Header field. In CoRE the collection of URIs and attributes is itself a resource. A SWD response, by default, is a JSON object containing an array of URIs that point to where the principal has instances of services of the requested type.

DTN IPND

DTN IPND uses Beacon messages. IPND beacons advertise neighbour availability by including the DTN node’s canonical endpoint identifier. IPND beacons optionally include service availability and other parameters. In IEEE 802 networks beacons are transmitted as link-layer broadcast messages. IPND beacons optionally advertise a node’s available services while maintaining the ability to decouple node and service discovery as necessary. A service block is comprised of a “Number of services” field that contains the number of services described in the block and a “Service Definition(s)” field that is a list of service definitions. A service is a structure that represents an advertisement for a DTN-related resource available on the beacon source node. New services must be defined in such a way that external users may implement the same services if they wish to inter-operate. Service definitions should mirror those found in the IPND draft; that is, they should clearly define the set of child components that construct the service.

IEEE-SA

The *IEEE 802.11-2012* Wireless LAN radio measurements may be used to collect transition information used by stations to make informed decisions about the most effective way to exploit the available spectrum, power, and bandwidth for their communications. *IEEE 802.11* measurement reports contain information from various tables in the ASN.1 encoded MAC and PHY MIB. Information is structured in request/report measurement pairs. Reports contain information from a table in the MIB. The MIIS specifies a way of representing information by using a standardised format, such as XML or binary encoding. Higher layer mobility management collects information from different media types and assemble a consolidated view to facilitate its inter-media handover decision. Static and dynamic information are obtained directly from access networks. IEs are specified for each point of attachment and two methods for representing IEs are defined; binary and resource description framework (RDF).Information Service elements are classified into three groups. An architecture of a MIIS is proposed by Cicconetti et al. (2010) which detailes the procedures for a network-assisted handover, which are left unspecified by the standard.

2.2.2 Analysis

This section reviewed existing work on the structuring of network information. While the majority of this work relates to configuration and management of elements of networks in which contemporaneous end-to-end connectivity is assumed, a number of abstract frameworks to describe elements, their attributes, and their relationships have been specified.

A naming facility is used by a significant proportion of the work. An ASN.1 names types and values. Information objects modelled through an SGML are named. assertion of an RDF graph implicitly asserts that all the names in the graph actually refer to something in the world. Furthermore, they use structural and functional components of a naming facility. Facts that indicate a relationship between two things may be represented as an RDF triple in which the predicate names the relationship. The keys of DDDS rules are domain-names. Each service provided in the RELOAD overlay is identified by an arbitrary identifier, called its namespace. While the naming facility is a core component, it's characteristics and mechanisms are not provided. In contrast, our work must provide a systematic process for the functional components of the naming facility to use structural components, in order to name things.

A significant proportion of the work we reviewed specifies representations of information that are tightly coupled with underlying communications mechanisms, which might be reasonable in the constrained and secure environments in which web resources are typically deployed, but would not be suitable for heterogeneous networks where it is necessary to decouple information from its producer in order to exchange that information between different consumers or producers.

A significant proportion of the work reviewed assumes a directory-like structure of information that maps names to addresses of devices hosting services. This might be a reasonable assumption in environments in which the address of the producer is coupled with the services it provides. As stated in Chapter 1, the approach of using of existing Internet protocols where possible is a design goal of this thesis. Several mechanisms that use or update the directory like structure of DHCP and DNS have been described. The DNS-based NAPTR/SRV peer discovery mechanisms that have been described use DNS records for information storage and retrieval. DNS use includes service instance enumeration, service name resolution and instance names persistence. DHCPv4 and DHCPv6 options are specified that allow resolution of names and locations, such as the home agent IP address. However, universal deployment of a new DHCP option takes a considerable amount of time and often, networking equipment needs to be updated in order to support the new option. Ultimately DHCP is a protocol designed for configuring IPv4 and IPv6 hosts and DNS is designed to associate information with domain names. DHCP and DNS are coupled with their transport mechanisms and are therefore are not appropriate in the RHNC. In contrast, we require a structure of information that is capable of representing data, or fragments of a data, and how the data or fragments relate to other network element.

While the MIB is probably the most widely adopted mechanism of all working and research groups, we do not adopt the MIB as it does not fulfil our requirements completely. The MIB database is hierarchical and each entry is addressed through an OID. Objects in the MIB are defined using SMIv1 or SMIv2 of the ISMF. Using MIB objects and IETF ISMF or *IEEE LLC* mechanisms, *IEEE 802* stations can exploit their communications mechanisms. However, objects in the MIB have highly complicated encoding rules and large addresses, that when transferred, "consume substantial parts of each SNMP message" (Vilardi et al., 2013). SNMP may not be applicable in low power, lossy, resource constrained environments, as along with the long OIDs, the SNMP version, multiple length and data descriptors are transmitted in each message. Although SNMP is a contribution that is well suited to contexts with low latency contemporaneous end-to-end connectivity, SNMP is not a particularly efficient protocol. Ultimately the MIB is part of an architecture designed to structure network management information in order to manage TCP/IP-based Internets. In contrast, the work in this thesis must support communications in the RHNC.

Many reviewed mechanisms, structure link layer frames or UDP messages as *beacons* to advertise presence. The LLC protocol data unit (PDU) beacons provide a structure representing data link layer service access point addresses. UDP beacons are small UDP messages in the IP underlay that have no guarantee of ordering, delivery or duplicate detection, while a reliable data transport might be desirable in some contexts. Use of UDP may adequate in the RHNC. Although the practical limit on the Internet for the UDP packet data length is 512 bytes, i.e., a number of services such as DNS restrict the largest UDP packet to 512 bytes, IPv6 "Jumbograms" of size greater than 65,535 bytes are possible. However, a limit of 512 bytes limits the information, and therefore constrains the range of its application. In contrast, while the work in this thesis caters for IEEE 802 networks that use TCP, UDP or IP it also caters for the exchange of structured information in *IEEE 802* networks that do not use TCP, UDP or IP.

Several reviewed contributions, provide detailed specification of an XML schema. Some constrain the information represented to a specific set, while others provide a simple generic format for describing information. XML was derived from SGML and defines a set of rules for encoding documents in order to be more suitable for Web use. XML might be an excellent choice for generic interchange for Web use and the benefits of XML as a markup language are evident. However, its benefits as a generic format are less so. fXML introduces a tight coupling between the representation and the processing program and parsing implementations are complex in nature. Parsing XML is an expensive operation that can degrade XML processing performance (Lam et al., 2008). Describing many to many relationships are problematic in XML. In contrast, the work of this thesis requires the ability to represent many to many relationships and that the representation is decoupled from the processing program.

JSON and RDF are contributions from the IETF and W3C working and research group communities that are particularly relevant to our work. The lean set of formatting rules of JSON provide a generic format that is suitable for the portable representation of structured data we require. JSON is therefore adopted by TDP for this purpose. This abstract syntax of RDF is distinct from the set of definitions for use in other specifications reviewed here that need to refer to other information. RDF has an abstract syntax that reflects a simple graph-based data model. While the graph-based data model approach of RDF is appropriate to our requirements, the formal semantics with a rigorously defined notion of entailment providing a basis for well founded deductions in RDF data are to a lesser extent appropriate to our needs. Ultimately, RDF is designed as language for representing information about resources in the World Wide Web, in which low latency contemporaneous end-to-end connectivity can be assumed. In contrast, the work in this thesis caters for the discovery of autonomous mobile network elements even when faced with communication disruption.

In this section we reviewed the information structuring mechanisms used by the specific discovery protocols reviewed in Section 2.1.

2.3 Association

The relationships, classes and characteristics of associations can describe the context and interactions of network elements with other network elements, or be used to describe the state of network elements.

2.3.1 Naming scheme characteristics

An early analysis of naming schemes for distributed systems by Znati and Molka (1992) classified components of the naming facility as structural or functional. The authors classify the functional components as those that relate to communication, database management and name management. The structural components are viewed as the logical organisation of a naming facility approach that describe the relationship between the hosts and the named objects, which they classify as centralised, distributed and hybrid.

Torres et al. (2012) present the state of the art of Identity Management systems for the Future Internet that highlights the existing architectures, specific devices applied, challenges and future perspectives. The authors describe "Identity" as an instrument used by an entity, and an "Identifier" as that used to prove ownership of the identity and then show the relationship between entity, identity and identifier.

In contrast, a goal of the work in this thesis is to decouple named things from hosts. In our approach we use classifications of components of Znati and Molka (1992). However, we classify the structure of information, generic interchange mechanisms, data storage models, identifier properties and the association properties as the structural components. We classify the association mechanisms for resolution, assignment, encoding, ascertaining authenticity and ownership, validation and scalability as the functional components.

We have described work on the structure of information and generic interchange mechanisms in Section 2.2. The name scheme components under review in this section, include the identifier properties, the association properties, and the association mechanisms.

2.3.2 Review

In following we review identifier properties, association properties and association mechanisms in the RHNC .

Authenticity, validation and encoders

As described in Chapter 1, security and policy mechanisms are not in the scope of this thesis. However, the generation of associations is in the scope of this thesis. Therefore the structural and functional components of the naming facility, relating to the authenticity, validation, encoding and ownership of things using their identifiers, are discussed.

In content-oriented security models, content is signed by the producer, allowing validity of the content to be ascertained by the consumer through verification of the signature.

A "self-certifying name" is a name that along with some signed metadata allows direct verification of the association between the name and an associated object (Ali Ghodsi, 2012). For example, self-certifying names for mutable data, i.e., immutable data can be checked against a cryptographic hash of the content embedded directly in the name, could consist of a cryptographic hash of a public key of the identifier of the principal for the data, and a unique name.

Information-Centric Networking (ICN) architectures (URL, 41) work with information objects as a first-class abstraction. TRIAD (Cheriton and Gritter, 2000), which based identifiers on names and not end-to-end addresses has been attributed with introducing the concept of ICN (Bari et al., 2012). Ali Ghodsi (2012) cite naming as the fundamental difference between the architectures attributed to ICN, such as CCN (URL, 20), 4WARD (URL, 21), NDN (URL, 22), NetInf (URL, 23), SAIL (URL, 24), COMET (URL, 25), PSIRP (URL, 26), CBCB (Carzaniga et al., 2004), CONNECT (URL, 27), PURSUIT (URL, 28), CONVERGENCE (URL, 29), DONA (isb, 2008; URL, 30). ICN architectures focus on the attributes of first-class abstractions and their receivers to achieve efficient and reliable distribution of such objects. ICN architectures typically support communication through replication, distributed cache/storage and publish-subscribe mechanisms. The binding of data with names is the subject of in depth discussion in the Information-Centric Networking Research Group (ICNRG). However, in contrast, the work in this thesis must support working with any ADU, process, device and mechanism of a network as a first-class abstraction of information.The main abstraction of ICN, called the Named Data Object (NDO), is semantic free named data chunks that can have meta-data, such as author, creation date, or other data about the represented information, associated with it (Ahlgren et al., 2012).

Choi et al. (2011) present a comprehensive survey on content naming. The naming and routing mechanisms proposed by CBCB, DONA, NetInf, NDN and PURSUIT are reviewed in a recent survey of naming and routing in information-centric networks (Bari et al., 2012) in which the authors provide their perspective on suitable design choices for naming. Self-certifying flat names are proposed for their scalability, persistence, security binding, authenticity, and global uniqueness. Bari et al. (2012) assert it to be desirable for the name structure to have owner selected keywords assigned that are used by search engines to index them and enable end users to search for their desired content.

In the recent ICN architecture comparison by Ahlgren et al. (2012), two naming schemes are reviewed:the self-certifying scheme described earlier, and the hierarchical scheme, which has a structure rooted in a publisher prefix that enables aggregation of routing information, improving scalability of the routing system. Bari et al. (2012) propose existing ICN naming schemes to belong to three categories: hierarchical, flat, and an attribute-value pair based in which an attribute has a name, a type, and a set of possible values. Table 2.17 provides a summary of prominent ICN naming approaches.

ICN approach	**Namespace**	**Request routing**
CBCB	Attribute-Value (AV)	Name based
DONA	Flat / Structure	Name based
NetInf	Flat / Structure	Name or resolution based
NDN	Hierarchical	Name based
PURSUIT	Flat / Structure	Resolution based
CCN	Hierarchical	Name based
PSIRP	Flat / Structure (rendezvous and scope identifiers)	Resolution based

Table 2.17: Summary of prominent ICN naming approaches

An advantage of self certification is that two parties can authenticate each other without any third party such as a list of CAs who sign the keys. A classic problem is that when a public key changes so does the identity. This can be problematic as, for example, it makes keeping historical data difficult and credentials have to be acquired. DONA binds a self-certifying name to a real-world identity (isb, 2008) so that a public key is required for data authentication similar to today's public key repositories.

In their paper, Baugher et al. (2012) consider how self-verifying names might be used in ICNs. In their scheme, there is no public key needed for data authentication, and public key cryptography is optional for name authentication. Self-verifying names are proposed to improve delivery of read-only data by eliminating signatures on most cacheable data, simplifying integrity checks, and by allowing alternative means of authentication in addition to public key cryptography. Cai et al. (2006) realise a self-certifying object-based network storage system that uses a two-layer self-certifying path-name algorithm.

When provided with the name of a thing, one can reference the thing or get the thing unless the thing has ceased to exist. The lifetime of associations is an important factor to consider. A persistent association may be used as a reference to a thing beyond the lifetime of the thing. If an association expires then it is no longer possible to use it a as a reference to a thing. In the absence of an association, a name and a thing are unrelated. An association can be considered valid for a duration of time (lifetime or TTL value). When the TTL of a thing is equivalent to the TTL of an association of that thing, data validation schemes such the client-based cache consistency scheme for MANETs of (Fawaz and Artail, 2012), that rely on estimating the inter update intervals of data items to set their expiry time, are applicable.

We can observe existing mechanisms to ensure authenticity of the named data or service, and that the name remains valid as long as the data or service is available. In ICNs, the validity of the object is ascertained directly and not by verifying which host supplied the object (Ali Ghodsi, 2012). Using self-certifying names there are cryptographic checks used to verify an association, i.e., that a thing is associated with an identifier. A consumer using external trust mechanisms can require a producer of data to provide the necessary signatures of that association when it is first provided.

Identifiers can be generated from public keys. A method for binding a public signature key to an IPv6 address in the Secure Neighbor Discovery (SEND) protocol is described by RFC 3972. IPv6 addresses, for which the interface identifier is generated by computing a cryptographic one-way hash function from a public key and auxiliary parameters, are called Cryptographically Generated Addresses (CGA). The binding between the public key and the address can be verified by re-computing the hash value and by comparing the hash with the interface identifier. Using this method messages sent from an IPv6 address can be protected by attaching the public key and auxiliary parameters and by signing the message with the corresponding private key. The protection works without a certification authority or any security infrastructure.

Every node in the RELOAD (URL, 42) overlay is identified by a unique Node-ID. Each node has a certificate (Cooper et al., 2008) containing a Node-ID, which is unique within an overlay instance. Ownership, location and address are inherent properties of the RELOAD Node-ID.Uniform Resource Names (URNs)(Moats, 1997) are designed to serve as globally unique, persistent, location-independent, identifiers used for recognition, for access to characteristics of the resource or for access to the resource itself (Sollins and Masinter, 1994). An URN gives every entity in the network a global unique identity, and has global uniqueness. Consequently when used anywhere, and in any context, identifies the same resource(Sollins, 1998). Walfish et al. (2004) have shown a move away from host-based URLs as desirable.

A Uniform Resource Identifier (URI) is a locator, a name, or both. A generic URI syntax and a process for resolving URI references that might be in relative form, along with guidelines and security considerations for the use of URIs on the Internet is defined in RFC 3986 (Berners-Lee et al., 2005). The URI is a sequence of characters that identifies an abstract or physical resource. The URI syntax defines a grammar that is a superset of all valid URIs, allowing an implementation to parse the common components of a URI reference without knowing the scheme-specific requirements of every possible identifier. URIs enable uniform identification of resources via a separately defined extensible set of naming schemes. URIs have a global scope. A URI has a scheme part that refers to a specification for assigning identifiers within that scheme and a scheme specific part.

Uniform Resource Name namespace for UUIDs (Universally Unique IDentifier), also known as GUIDs (Globally Unique IDentifier) (Leach et al., 2005). A UUID is an identifier that is unique across both space and time, with respect to the space of all UUIDs. A UUID can be used for multiple purposes, from tagging objects with an extremely short lifetime, to reliably identifying very persistent objects across a network. The UUID consists of a record of 16 octets and must not contain padding between fields. The total size is 128 bits and it requires no central registration process.

While one UUID format uses IEEE 802 node identifiers, others do not. That no centralised authority is required to administer UUIDs is one of the main reasons for using UUIDs. The UUID generation algorithm supports very high allocation rates of up to 10 million per second per machine if necessary, so that they could even be used as transaction IDs. UUIDs are of a fixed size (128 bits) which is reasonably small compared to other alternatives. This lends itself well to sorting, ordering, and hashing of all sorts, storing in databases, simple allocation, and ease of programming in general. Since UUIDs are unique and persistent, they make excellent Uniform Resource Names. The registration process allows for UUIDs to be one of the URNs with the lowest minting cost. The generation of UUIDs does not require a registration authority for each single identifier. Instead, it requires a unique value over space for each UUID generator. This spatially unique value is specified as an IEEE 802 address[1], which is usually already applied to network-connected systems. This 48-bit address can be assigned based on an address block obtained through the IEEE registration authority. This UUID specification assumes the availability of an IEEE 802 address.

Identifier persistence considerations: UUIDs are inherently very difficult to resolve in a global sense. This, coupled with the fact that UUIDs are temporally unique within their spatial context, ensures that UUIDs will remain as persistent as possible.

Generic Security Service Application Programming Interface (GSS-API), which authenticates two named parties to each other is extended by the (GSS-API) Naming Extensions. RFC 6680 (Williams et al., 2012), extends the GSS-API naming model to include support for name attributes in order to provide a naming architecture that supports name based authorisation. The goal of GSS-API naming extensions is to make information modeled as *name attributes* available to applications. GSS names have attributes that may be authenticated. GSS name attributes' values are network representations. GSS provides a function to obtain objects of types associated with names and name attributes. An attribute is said to be authenticated when there is a secure association between the attribute, its values and the trusted source of the peer credential. Attributes are classified as asserted (non-authenticated) or authenticated. Authenticated implies a trusted third party has asserted the attribute as opposed to the attribute being asserted by the peer itself.

[1]An EUI-48, in which EUI is an acronym for Extended Unique Identifier.

Assignment

We can observe several existing identifier assignment approaches in a heterogeneous networks, such as proposed by (Abdelkader et al., 2012) for 4G environments, ANARCH which provides a node with a unique user-oriented name (Aoki et al., 2003), and the node address naming scheme for wireless sensor networks. In some scenarios mechanisms to determine who generated or published an association and the conditions under which a name is associated with a thing are required by producer and consumers. Torres et al. (2012) state of the art classifies the main features in existing Identity management initiatives of the Future Internet as privacy, security, mobility, interoperability, anonymity, federation and user-centric, and presents a comparison of support for these features by existing mechanisms.

Generating a UUID does not require that a registration authority be contacted. One algorithm requires a unique value over space for each generator. This value is typically an IEEE 802 MAC address, usually already available on network-connected hosts. The address can be assigned from an address block obtained from the IEEE registration authority. If no such address is available,

Scalability and resolution

The two global namespaces of the Internet are DNS names and IP addresses. DNS achieves scalability through hierarchy. Use of the DNS to name information has been shown to tightly couple information with administrative domains or network locations, and consequently it can be difficult to move service instances and data, as well as to replicate (Balakrishnan et al., 2004). Singla et al. (2010) assert that hierarchy requires location-dependent addresses, complicating management, mobility, and multihoming. The original Interplanetary Internet design was based on two-level hierarchical endpoint identification (URL, 62). Each endpoint identifier in the interplanetary Internet was a tuple consisting of a region and a region-specific-part. Partial addresses called *Regions* were intended to be topologically significant and useful for coarse-grained routing (Clare et al., 2010). The rationale behind two-tier hierarchical routing was that nodes generally need to know more about proximate things than about things far away (Clare et al., 2010).

Mechanisms for efficient search, and exploitation of vast numbers of associations face similar problems to those posed by routing in the Internet. Potential routing table size scalability issues of the Internet are known and attributed to the number of networks connecting to the Internet, unaggregatable IP address allocations, and prefix aggregation due to multi-homing and traffic engineering (Meyer et al., 2007). All of the currently prominent ICN projects reviewed by Bari et al. (2012), will "face severe scalability problems for an Internet-scale deployment" as name based routing tables need to handle around 10^8 routes, which is 4 orders of magnitude bigger than the biggest BGP routing table size. Nevertheless protocols such as the Distributed Compact Routing (Disco) protocol have been demonstrated, which provide distributed and dynamic routing in flat namespaces with guaranteed scalability and low stretch, i.e., the ratio of route length to the shortest path length (Singla et al., 2010). Any ICN name resolution scheme needs to scale at least on the order of trillions and possibly beyond to accommodate future growth. The routing scalability issue has been recognised as an important problem to be addressed in future Internet design. Ballani et al. (2009) describes an aggregation method that divides the address space. (Wang et al., 2012) describe an aggregation-aware routing scheme called named Aggregation-aware Inter-Domain Routing (AIDR) that leverages the path diversity, and takes prefix aggregation into account in the BGP best route selection. It has been argued that it is not the need for aggregation that drives the use of longest-prefix-match but the lack of global uniqueness of fragments (Ali Ghodsi, 2012). While hierarchical names help scalability by reducing the size and update-rate of the routing tables, Ali Ghodsi (2012) also argue that lack of inherent hierarchy provides greater flexibility in aggregation and propose routing using concatenations of flat names. Routing strictly by address increases the chance of delivery failure and data loss (Clare et al., 2010). Nevertheless, hierarchical naming has been shown to be naturally capable of scalable routing, while for a flat naming approach to reach an acceptable performance, a routing architecture and routing algorithms must be carefully designed (Chi et al., 2011). In their comparison of alternative architectures for achieving the functional goals of name oriented networking, the results of Baid et al. (2012) show that in certain scenarios, the hybrid flat names and network address approach may offer scalability and performance improvements over baseline ICN. In this approach routers operate with both flat names that are Globally Unique Identifiers (GUIDs) and network addresses, reducing routing table size and overhead.

Within the structured information reviewed it is often necessary to locate and manage identifiers associated with things e.g., the GOSSIP 2 NSAP Structure (Colella et al., 1994). Network addresses serve two purposes: first, to uniquely identify an interface, and second, to aid routing by identifying where an interface is on the network. An address is said to have "topological significance", and a name is an identifier (Clare et al., 2010). However a name that has topological significance or address that is expressed as a string is both a name and an address. Use of IP addresses as host identifiers is problematic for multihoming, security, mobility and routing on the Internet. Scaling of Internet routing and addressing has been complicated by the overloading of IP address semantics (Meyer et al., 2007). There have been many architectural proposals to decouple identifiers and locators over the past 20 years, for example the Host Identity Protocol (HIP) (URL, 31). Research on identifier / locator split architectures has provided Distributed Hash Table (DHT) architectures, such as Chord (Stoica et al., 2001), CAN (Ratnasamy et al., 2001) and Pastry (Rowstron and Druschel, 2001) based locator / ID separation mechanisms. The notion of locator/identifier split is designed to make Internet routing more scalable. Flat names are more suitable for DHT based lookup services (Bari et al., 2012). FIRMS (Menth et al., 2010), is mapping system to find the location of a host, that returns appropriate locators in response to requests for specific identifiers. Web data URLs are for the most part based on the host name that serves the resource. Approaches and potential approaches to introducing multi-ID and multilocator support into the Internet architecture have been reviewed in (Kafle and Inoue, 2012). Roussos and Chartier (2011) propose a novel extensible identifier locator resolution scheme for IoT that places no restrictions on the choice of identifier or locator specification.

(Walfish et al., 2004) described a layered architecture that caters for resolution of user-level descriptors, service identifiers, endpoint identifiers and IP addressed. Flat names are presented as a natural choice for the service and endpoint identifiers. Table 2.18 describes the basic design principles of Walfish et al. (2004) that are essential to the nature and use of Internet names.

Adjie-winoto et al. (1999) present the design and implementation of an intentional network naming architecture in which applications describe their intent. The authors argue that efforts in 1999 to efficiently enable new services such as mobility, group communication, resource discovery, service location and caching have been greatly hampered by the lack of a flexible naming system and that significant effort is spent in creating independent, but similar infrastructure for each situation. Adjie-Winoto et al. (2000) present the design and implementation of the Intentional Naming System (INS) (URL, 32) , a resource

No.	Principle
1	Names should bind protocols only to the relevant aspects of the underlying structure; binding protocols to irrelevant details unnecessarily limits flexibility and functionality.
2	Names, if they are to be persistent, should not impose arbitrary restrictions on the elements to which they refer.
3	A network entity should be able to direct resolutions of its name not only to its own location, but also to the locations or names of chosen delegates.
4	Destinations, as specified by sources and also by the resolution of service IDs and endpoint IDs, should be generalisable to sequences of destinations.

Table 2.18: Basic design principles of Walfish et al. (2004) essential to the nature and use of Internet names.

discovery and service location system for dynamic and mobile networks of devices and computers. The INS resolvers self-configure to form an application-level overlay network, which they use to discover new services, perform late binding, and maintain weak consistency of names using soft-state name exchanges and updates. INS is a naming system intended for naming and discovering a variety of resources in (future) networks of devices and services. INS uses an attribute and value (AV) tree for naming and implements late binding mechanism to integrate name resolution. A resource is named by a hierarchical arrangement of AV pairs such that an AV pair that is dependent on another is a descendant of it.

The notion of an identifier namespace being flat means that the identifiers used are unstructured and semantic free. In their design and implementation of Semantic Free Referencing (SFR), Walfish et al. (2004) provide the notion that the service identifier namespace be flat. HIP (Moskowitz and Nikander, 2006), which uses a flat, self-certifying identifier for the host, provides the notion that the endpoint identifiers be flat. Identifiers of flat namespaces can be resolved, resilient, self organising, extensible and distributed. Semantic free names are a goal of the Data-Oriented Name Service (DONS), as resource names which are not semantic-free from resource attributes restrict the flexibility and functionality of the network (Wu et al., 2009). The reflection of visible semantics in an association restrict the flexibility and functionality of the network. Nevertheless, whether an association can be easily transcribed, transported (without modification), and parsed or spoken by a human or computer are important considerations, for example in near realtime voice communications.

DHTs prove valuable for distributed architectures by providing distributed information lookup within flat namespaces. Luo et al. (2009) propose a one hop DHT-based identifier to locator mapping scheme with low lookup latency and reasonable bandwidth usage. Li et al. (2010) propose a similar on hop DHT scheme that can support mobility and has very low load and resolution delay. DHTs represent one possible solution for resolution of identifiers of flat namespaces. The basic functionality of the DHT name resolver described by Znati and Molka (1992), must include the support of at least two levels of names. The first level allows names to be easily manipulated by machines and the second level aims at allowing names to be generated in a user readable form.

Choi et al. (2009) provide a comprehensive and quantitative study on the two major infrastructure alternatives in substantiating content-oriented networking: a tree and a DHT. They found that if the network will be deployed with information about underlying physical network information, the tree structure is preferable due to its superior delivery efficiency. Otherwise, the superior scalability and better resiliency of the DHT structure is a better choice. Nguyen et al. (2012) propose a distributed attribute-value tree (AV tree) on DHT-based networks to substantiate multiple-attribute searching. Each resource is named by an AV tree and DHT keys are created from each AV branch of a resource name. Resource information is distributed to a DHT-based network by the use of these DHT keys. For their indexing system, the authors create a DHT key from each AV branch of an AV tree, which includes AV pairs on the path from root node to a leaf node. AV branches that share a subsequence of attribute / value pairs are mapped to a contiguous portion of the key space.

Name assignment is coordinated by established organisations for the management of hosts and gateways on the Internet. The Internet Assigned Numbers Authority (IANA), a function of Internet Corporation for Assigned Names and Numbers (ICANN) (URL, 33) , allocates and maintains unique codes and numbering systems that are used in the technical standards *protocols* that drive the Internet. SMI includes the provision for parameters or codes to indicate experimental or private data structures. These parameter assignments are coordinated by IANA (URL, 34).

2.3.3 Analysis

Any communications mechanism, data or service can be associated with or uniquely identified by a name. That is not to say everything should be given a name; "good abstraction is ignoring the right detail at the right times"(Shaw, 1990). It may be less meaningful to name (near) realtime elements, such as a voice call, which are not cachable than a file which is cachable. A significant body of work is inspired by work on the Future Internet, and typically offers fairly elaborate means for the binding of ADUs, or services with names, they typically do not decouple named objects, attributes and values from underlying communications mechanisms. In contrast, the work in this thesis must support autonomous mobile network elements and working with any element , attribute and value as a first-class abstraction of information.

Naming schemes for distributed systems have classified the structural components as the logical organisation of naming facility approaches that describe the relationship between the hosts and the named object. Such schemes have classified the functional components of the naming facility as those that relate to communication, database management and name management. In contrast, the work in this thesis seeks to decouple named objects from hosts and consequently the structure of information, generic interchange mechanisms, data storage models, identifier properties and the association properties comprise the structural components. In our approach the functional components are the association mechanisms for resolution, scalability, assignment, encoding, ascertaining authenticity, ascertaining ownership, and validation.

The majority of approaches to authenticity and validation of the Internet use the host authenticity and data integrity services that are built on the notion of securing contemporaneous communications channels between hosts. Heterogeneous mobile nodes subject to time constraints, however, might not be able to communicate appropriately with each other in order to secure a channel. Therefore, host authenticity and data integrity services based approaches can only offer guarantees in limited contexts where there is reliable communication and are not suitable for applications with unreliable communication. Amongst the other approaches to authenticity, validation and ownership, the content-oriented security models of ICN are particularly relevant, as they take into account the significance of the data authenticity rather than that of the channel allowing validity of the content to be ascertained by the consumer through verification of the signature.

Unstructured and semantic free identifiers are extensible and can be resolved and distributed. Routing architecture and routing algorithms must be carefully designed for a flat naming approach to reach an acceptable performance, nevertheless, routing in flat namespaces with guaranteed scalability has been demonstrated (Singla et al., 2010). For the most part, the ICN approaches reviewed use either flat or hierarchical naming approach. The pros and cons of hierarchical and flat namespaces have been discussed at length, but ultimately hierarchical and flat namespaces are not mutually exclusive. The need to split locator from identifier to make Internet routing more scalable might be seen as a more pressing issue. Hybrid flat and hierarchical namespaces may offer scalability and performance improvements. DHT-based networks and self certifying flat names have been predominantly adopted by most ICN approaches. However, as we have seen there are other architectures that support name-based authorisation and mobility of named elements. An architecture reviewed that supports name based authorisation that is of particular relevance to our work is the GSS-API Naming Extensions which make information modelled as *name attributes* available to applications. The application domain of this thesis calls for flat association identifier that can avail of network resolution, that do not restrict the flexibility of the network, and facilitate decoupling of information from administrative domains. This motivates our choice of flat names along with AV trees to facilitate multiple-attribute searching.

The majority of approaches to naming of autonomous network elements are built on, or some combination of, Uniform Resource Identifier (URI), Extended Unique Identifier (EUI) or Universally Unique IDentifier (UUID). There are billions of objects that are currently identifiable and possible candidates for membership to the IoT. In contrast, the association mechanisms described in this thesis must support scalability, at least on the order of trillions, and possibly beyond to accommodate future growth. EUI like the UUID is intended to be used within applications that require fixed size universal identifiers. The IEEE-RA 24 bit OUI assignment effectively limits the number of unique EUI values the assignee can generate to approximately 1 trillion (10^{12}). Although the EUI-64 format could be modified (as in IPv6 for example) to exceed this limit, in contrast the URI and UUID are designed with no such a limit. It is of particularly significant to our work that the UUID can be generated via hashing of a thing. The application domain of this thesis calls for association mechanisms that are designed for very high generation, via hashing, and assignment rates of identifiers, unique across both UUID space and time, that do not require centralised administration of each single identifier, and these are main reasons for using UUIDs in TDP.

The scaling of Internet routing and addressing has been complicated by the overloading of IP address semantics. When information is tightly coupled with administrative domains or network locations it can be difficult to move service instances and data. While hierarchical naming has been shown to be naturally capable of scalable routing by reducing the size and update-rate of the routing tables, the reflection of visible semantics in an association restrict the flexibility and functionality of the network. In contrast, the work of this thesis calls for identifier properties that include a syntactic structure, that is appropriate for fixed size universal identifiers, that facilitate partitioning of the namespace into identifier subsets, and support limited identifier rules. This thesis calls for association properties which describe extensible, non persistent, unique, associations. Such associations that are difficult to transcribe, do not structure a semantic free namespace and consequently decouple location from name. URIs, are extensible, have a global scope, facilitate parsing of identifier components without knowing the scheme-specific requirements and enable uniform identification of resources via a scheme part. UUIDs can be used as URIs and RFC 4122 (Leach et al., 2005) describes the correct scheme for rendering a UUID. To use a UUID as a URI a scheme is supplied, such as *urn:uuid:*, in addition to the URI content that is the UUID. In our approach we seek to decouple named objects from hosts and consequently the desired functional and structural components comprised of association mechanisms, identifier and association properties can be fulfilled in part through use of the UUID and URI are the main reasons for using the URI along with the UUID in TDP.

Amongst the other approaches, the ICN approaches to naming are particularly relevant, as they take into account metadata associated with named data objects. The application development support and semantics offered for such metadata, however, are very limited. In contrast, our work must provide a systematic process for the functional components of the naming facility to use structural components in order to name, and when required, authenticate associations. Our functional components must be capable of naming autonomous mobile network elements and providing a structure, generic interchange mechanism and data storage model for such metadata.

Our review of work that defines naming scheme characteristics used by the specific discovery protocols reviewed in Section 2.1, and information structuring mechanisms reviewed in Section 2.2, has reached its conclusion. The next section first reviews work on protocols applicable in a broad range of contexts that facilitate mobility and the opportunistic exchange of information.

2.4 Protocols for RHNC

New problems have necessitated new protocols to support access to named data, mobility, multi-homing, wireless technologies and opportunistic contacts. As described in Section 1.4.1, communications may be over asymmetric links, be subject to long propagation delays, disruption, bandwidth asymmetry and/or packet loss. Infrastructure and communications mechanism bound systems are sufficient in a limited range of contexts. Systems with complex interactions and tight coupling are likely to have unforeseen failure states and less flexibility in recovering from failure states (Bush and Meyer, 2002). A wider range of technology and protocol options can be seen to translate into a wider range of real world deployments. The tight coupling of time or space, and synchronisation between producers and consumers, and of applications with the underlying communications mechanisms, can lead to a reduced range of technology and protocol options, or in some contexts, mean that communications are not possible. A lack of support of the contexts described in Section 1.4.1 translates to an inability to exploit a contact in *the contexts*. Consequently, appropriate communications protocols must address these problems to be capable of exploiting an opportunistic contact to transfer ADUs within the bounds of the described contexts.

The limitations of an opportunistic contact in the presence of disruption and/or constrained network contexts mean that the information that nodes have about their peers varies significantly over time and space. In this section, we give a brief overview of existing work in the area of network architectures designed for tolerance of unreliable communications, and robustness in challenging communication scenarios, and review the underlying mechanisms used to deal with such contexts.

Caching, replication and store-and-forward techniques are used to improve the transfer of ADUs, through handling disruption and/or constrained network contexts. Store-and-forward techniques can be used to make unreliable communication links reliable. Baran (1964) discuss the feasibility of using low cost unreliable (even unusable) communication links, to form highly reliable networks by using store-and-forward techniques. Store-and-forward nodes often use long-term persistent storage, a technique that is not novel: a long range multi station store-and-forward message handling capability utilising a satellite, in which a tape recorder in the satellite is used to play back recorded messages, is documented by the courier-communication-satellite experiment (Siglin and Senn, 1961). Store-and-forward involves the forwarding of messages between a set of cooperating hosts. When an intermediate store-and-forward host receives a message, it determines appropriate routing for the message, and whether it can establish contact with next hop peer(s) via appropriate link(s). If it can, the message is forwarded, and if not, the message can be stored until a contact becomes available. link availability may be checked periodically or on-demand, and forwarding is initiated, or resumed, when a contact becomes available.

Caching can be used to exploit an opportunistic contact to improve the transfer of ADUs. If data is contained in the cache, then the data can be retrieved comparatively faster by accessing the cache. ICN combines caching at the network edge with in network caching (Ahlgren et al., 2012). ICN requests for NDOs can be satisfied by any node holding a copy in its cache, as all nodes potentially have caches. Various techniques can be used to ensure validity, invalidate or update a cached object. Cache coherence protocols (Stenstrom, 1990) are used to ensure that requests for a certain NDOs always return the most recent value; they solve the cache coherence problem. As a consequence of the principle described by the “Consistency Availability Partition tolerance (CAP) theorem” (Fox and Brewer, 1999), if network partitions are unavoidable, the cache coherency problem relates to a wider design trade-off between availability and consistency (Demmer, 2008). In contrast to caching, replication distributes data according to some policy. In some replication systems a set of cooperating nodes exchange data in a federated manner. In others, a one to many relationship between a set of cooperating nodes is used to exchange data. The development of CDNs and P2P networking has been attributed to the increasing demand for mass distribution and replication of large amounts of resources (Ahlgren et al., 2012). P2P caches provide in-network storage and CDN achieves scalability and performance by operating dedicated caches close to access networks. While CDNs and P2P mechanisms provide low-latency and reliable access to data and services, neither work to ensure availability, as both rely on application specific and ad hoc mechanisms. In-network caching and on-demand replication are research challenges to be addressed to bring ICN to life (Bari et al., 2012).

2.4.1 New designs

Architecture designs based on new design principles are expected to address the challenges of the Future Internet architecture. A main goal of Future Internet is to achieve a strong separation between control functions related to packet transmission and service delivery by providing service-related functionality, that are independent of lower layer technologies Torres et al. (2012). A survey on the research of Future Internet architectures (Pan et al., 2011) provides focus on a series of representative research projects as provided in Table 2.19. In this overall picture of the research progress on the future Internet architecture, the key research topics are said to be, content or data oriented paradigms, mobility and ubiquitous access to networks, cloud-computing-centric architectures, security and experimental testbeds. We can see the

Categories	**Project**
Future Architectures and Technologies	4WARD, TRILOGY, EIFFEL, SPARC, SENSEI, Socrates, CHANGE, PSIRP, etc.
Services, Software, and Virtualization	ALERT, FAST, PLAY, S-Cube, SLA@SOI, VISION Cloud, etc.
Network Media	3DLife, COAST, COMET, FutureNEM, nextMEDIA, P2P-Next, etc.
Internet of Things	ASPIRE, COIN, CuteLoop, SYNERGY, etc.
Trustworthiness	ABC4Trust, AVANTSSAR, ECRYPT II, MASTER, uTRUSTit, etc.
Testbeds	FIRE, N4C, OPNEX, OneLAB2, PII, WISEBED, G-Lab, etc.
Others	HYDRA, INSPIRE, SOCIALNETS, etc.

Table 2.19: EU research projects of Pan et al. (2011) on Future Internet of which a key topic is the N4C experimental testbeds we use to evaluate our approach

category "Testbeds" in Table 2.19, which refers to experimental testbeds. One of the experimental testbeds listed is the N4C project. It is important to note that we use the data from the N4C project testbed to evaluate our approach, and that this data is representative of research project of the Future Internet architectures.

The survey of Paul et al. (2011), provides a comprehensive coverage of relevant next generation networking research. The "more progressive" (Paul et al., 2011) research is classified in sections: security, content distribution mechanisms, challenged network environments, network monitoring and control architectures, and inter-networking layer design.

Inter-networking layer designs, such as Cloud-computing-centric architectures, provide new ways to provide global-scale resource provisioning. While robustness is a characteristic of cloud-computing-centric architectures, the focus is on interconnecting data, control, and management planes of well connected data centres. In contrast, a focus of the work in this thesis is on interconnecting data, control, and management planes of autonomous mobile network elements of partitioned networks. Consequently cloud-computing-centric architectures are not reviewed further.

Testbeds

There are various experimental environments for new protocols (see Table 2.20) which involve; simulation, emulation or virtualisation, actual implementation on the terrestrial Internet, actual implementations that use the terrestrial Internet as a gateway, and actual implementations in challenged areas without communications infrastructure that use the terrestrial Internet as a gateway.

Testbed	**URL**	**Description**
DTNBONE	`http://www.dtnrg.org/wiki/DtnBone`	An effort is underway to establish a network of DTN (mainly DTN2 and ION) nodes on the terrestrial Internet for interoperability testing, application deployment, and the develop of network management, naming, and routing conventions.
Planetlab	`http://www.planet-lab.org`	A worldwide interconnection of testbeds for networking technologies managed by Princeton University
Federica	`http://www.fp7-federica.eu`	an experimental network infrastructure for trialling new networking technologies. The infrastructure is intended to be agnostic as to the type of protocols, services and applications that may be trialled, whilst allowing disruptive experiments to be undertaken.
ns-3	`http://www.nsnam.org`	A discrete-event network simulator for Internet systems, targeted primarily for research and educational use

Table 2.20: Representative set of experimental environments for new protocols

Experimental testbeds composed of networks and nodes on the terrestrial Internet, such as DTNBONE, have been designed and implemented for interoperability testing, application deployment, and the develop of network management, naming, and routing conventions. The UMass Trace Repository provides network, storage, and other traces to the research community for analysis. DieselNet (URL, 106) was part of the UMass Amherst GENI testbed.It consisted of 35 buses each with a Diesel Brick, connected to three radios. The EU's Seventh Framework N4C project also used data mules and is based on previous work from the SNC project. These projects use opportunistic encounters with data mules in the Swedish Arctic, for the most part helicopters, but also hikers, to transfer bundles between the temporary camps of the Sámi people, and the Internet. There are a number of drawbacks for experimental validation over emulation environments: poor repeatability and lack of experimental control such that the nature of failures cannot be controlled. Other experimental testbeds have been designed to effectively mimic the scale of the Internet. Such mechanisms organise network elements, network and mobility models, under a common control framework. PlanetLab, managed by Princeton University, was the first effort to design a worldwide interconnection of testbeds for networking technologies which virtualises its resources, to provide isolated resource allocation to multiple concurrently active experiments. Gluhak et al. (2011) present a survey of testbeds for the Internet of Things. The ns-3 discrete-event network simulator for Internet systems is particularly relevant to our work as it can be used as a realtime network emulator, interconnected with the *real* world and this allows many existing *real*-world protocol implementations to be reused within ns-3. The ns-3 can be used to allow a host system or virtual machines to interact with a simulation.

The Internet of things (IoT)

There is no standard definition of *Internet of Things* (IoT). The IoT, broadly speaking, allows the association of information with physical objects. The concept of IoT has grown to encompass sensor networks capable of providing intelligence and collaboration of distributed objects via local networks or the Internet (Gluhak et al., 2011). Tan and Wang (2010) define the IoT as "Things have identities and virtual personalities operating in smart spaces using intelligent interfaces to connect and communicate within social, environment, and user contexts". Generally IoT involves information about an object being stored in databases, distributed and maintained by participants in the object's supply chain. Some, usually centralised, Discovery Service (DS) is responsible for collecting all these database URIs and feeding them to clients that query about an object. DSs are designed to effectively aggregate as many information sources as possible, rather than trying to respond optimally to unforeseen user queries. The unit of IoT which provide communication and contextual services is the often called the smart object. IoT objects are envisaged to handle resource constrained, and lossy network contexts. However, IoT is not designed to handle more broadly challenged network environments with long propagation delays, and consequently other protocols are needed. IoT Things are expected to be the main traffic makers where communication forms are expected to be "human - human", "human - IoT thing", and "IoT thing - IoT thing" (Tan and Wang, 2010). In contrast, the work in this thesis must provide not only access to physical network elements, but also to non physical network elements, such as named data, which are decoupled from the underlying communication forms.

2.4.2 Challenged network environments

Intermittent connectivity and/or constrained network contexts are characteristics of challenged network environments. Disruptions or constraints may be planned or unplanned.

Mobile and vehicular ad hoc networks

Mobile Ad hoc Networks (MANETs) are designed to support robust and efficient operation in mobile wireless networks by incorporating routing functionality into mobile nodes. MANET is designed to extend mobility into the realm of autonomous, mobile, wireless domains, in which MANET nodes form the network routing infrastructure in an ad hoc fashion. Communication in MANET is frequently interrupted by, for example, nodes moving out of communication range of peers or obstructions. MANETs are designed to have dynamic, sometimes rapidly-changing, multihop topologies which are likely composed of relatively bandwidth-constrained wireless links. Vehicular Ad hoc Networks (VANETs) might be described as a network of autonomous mobile vehicles communicating asynchronously. While VANETs are similar to MANETs, the key contrasting features are high speed mobility and unpredictable movement characteristics (Chowdhury et al., 2011). Kozat and Tassiulas (2003) state that when no direct link exists between consumers and producers in a MANET, efficient network layer support for service discovery in a dynamic topology takes the form of broadcast, multicast, or anycast. VANETs are highly partitioned networks, in which movement of vehicles causes unpredictable changes in network topology and consequently continuous connectivity may not be assumed. When no relay is available messages are lost using the proposed broadcasting protocols for vehicular networks. Communication paradigms that provide decoupling in time, space and synchronisation between communicating entities are well suited to VANETs. VANETS and MANets exploit other protocols with store-and-forward approaches like those of DTN, to allow long range information dissemination beyond the extension of a single network partition (Mehar et al., 2012).

DTN

DTN grew out of attempts to develop an Interplanetary Internet, started in 1998, and evolved into an active area of networking research, with applications in space networking, military tactical networking, and networking for various challenged communities. The DTN Research Group (DTNRG) (URL, 36) provides an open forum in which DTN researchers and developers can collaborate to further develop this experimental technology. Many important Internet transport protocols fail in environments without contemporaneous end-to-end connectivity. In Section 2.5, we review the DTN approach to dealing with this problem.

As described in Chapter 1, DTN is a generalisation of an architecture designed for an IPN. Anything used in DTN can be named in order to support "radical heterogeneity", a major tenet of DTN (Fall, 2012).

The DTN architecture is said to provide a sender-initiated unicast communication model (Demmer, 2008). However, while the DTN model is sender-initiated, zero or more nodes may be associated with an endpoint. Furthermore, the minimum reception group, which is the minimum subset of the nodes in the endpoint, may refer to one node (unicast), one of a group of nodes (anycast), or all of a group of nodes (multicast and broadcast) (Cerf et al., 2007). While the focus of DTN has not been on forwarding performance, DTN protocols have been demonstrated, such as in rural telemedicine networks using store-and-forward Voice-over-IP (Scholl et al., 2009), to work with contemporaneous low-latency connectivity. The RHNC calls for a sender-initiated communication model that can be used to exploit an opportunistic contact in the presence of disruption and/or constrained network contexts, that do not restrict the flexibility of the network, and facilitate decoupling of information from hosts. This motivates our choice of DTN.

Haggle

A focus of Haggle (URL, 43), was to provide a networking architecture for content-centric opportunistic communication. Haggle was envisaged as a new networking architecture designed to enable communication in the presence of intermittent connectivity. The focus of Haggle is on content dissemination. Haggle focuses on applications that incorporate social aspects, have content-focus, and work on consumer devices such as mobile phones. A reference implementation of Haggle has been developed at Uppsala University (URL, 44).

The creators of Haggle understood the focus of DTN to be more on efficient communication, in very challenged environments, with a aim to create a virtual link to move data between two endpoints, by using mobile data mules (URL, 45). A view of the DTN architecture is presented, which has a traditional host-centric approach to addressing via use of endpoint identifiers (EIDs). In contrast, Haggle is presented as having a data-centric addressing approach, based on metadata in the form of key-value pairs. However, DTN uses associations, and as an EID is used by elements of an endpoint to identify themselves, i.e., not only endpoints, it is not tightly coupled with a host. The authors of Haggle state that, in contrast to DTN, Haggle aims to allow users to disseminate content using their mobile devices even though there is no infrastructure, or it is too costly and slow (URL, 45). However, we disagree as the dissemination of ADUs, that include content ADUs, using heterogeneous autonomous mobile nodes has been demonstrated in DTN using session based communication (Demmer, 2008).

2.4.3 ICN NDOs and DTN endpoints

The ICN approach has been considered by researchers to share some assumptions, objectives and certain structuring architectural properties but differ from DTN with respect to their architecture (URL, 37). While the DTN architecture provides a sender-initiated communication model, Demmer (2008) defined extensions to the DTN Bundle Protocol to support a publish / subscribe session layer that more naturally supports receiver driven applications and multicast communication, as well as mechanisms for applications to convey ordering relationships among transmissions and a modification to the message expiration protocols that supports in-network deletion of obsolete messages. The DTN architecture can leverage a wide variety of existing Internet protocols. Mobility and the opportunistic exchange of information is inherent in the DTN architecture.

While the DTN architecture corresponds to an ICN architecture (URL, 3) it can also work with a device, process or mechanisms as a first-class abstraction. Multiparty communication through replication, distributed cache/storage and publish-subscribe mechanisms are possible in the RHNC using DTN protocols.

2.4.4 Analysis

The idea of working with any element of a network as a first-class abstraction is an important aspect of this thesis. Elements can be constrained in their relationships in order to achieve a desired set of architectural properties. The ICN (Ahlgren et al., 2008) approach has been considered by researchers to share some assumptions, objectives and certain structuring architectural properties but differ from DTN with respect to their architecture (URL, 37). The focus of ICN has been on naming and content in mostly moderate to well performing links. ICN holds latency to be important and consequently forwarding performance but it is not clear how well ICN can handle networks that feature 1-way links, long propagation delay and storage persistence. ICN uses storage primarily as a cache to reduce latency and in order to tolerate modest link disruptions.

DTNs might be seen to be comprised of a few layers based on physical constraints. An application layer dealing with sharing content, a transfer layer that deals with multiple hops between nodes and a convergence layer between one-hop neighbours

The DTN is inherently distributed and bound the the physical world.The role of DTN might be thought of as to abstract those characteristics away for the applications. One could envisage a DTN, where the content was stored in the network, and applications never interact with each other, but only with the

DTN to extract that content. In this case, RFC 5050 would be an adequate format for transferring content between locations and applications registering their desires.

In the following section we review the DTN approach, with an emphasis on the work of the IRTF DTNRG.

2.5 DTN

Much of the material presented in this chapter was co-authored with Stephen Farrell, and previously published as “Delay- and Disruption-Tolerant Networking”, in the IEEE Internet Computing Magazine in November 2009 (McMahon and Farrell, 2009). Here, we review the DTN approach to dealing with this problem, with an emphasis on the work of DTNRG.

2.5.1 Background

In 1973, Vint Cerf and Robert Kahn wrote a pioneering paper on TCP (Cerf and Kahn, 1974), and 30 years after sowing this seed, the Internet had become ubiquitous. In 1997, another seed was sown: Cerf thought that “an interplanetary backbone” was necessary for us to prepare for future needs (URL, 63). He and scientists from NASA’s Jet Propulsion Laboratory (JPL), who had been working since the early 1990s on adapting Internet protocols for space missions, shared a space-networking vision and, in 1998, started collaborating on developing an Interplanetary Internet (IPN). To support the IPN, the IRTF formed an Interplanetary Internet Research Group (IPNRG). Initially, the IPNRG included contributors from Worldcom, JPL, Mitre, SPARTA, and a few universities. Completion of the first phase of the IPN project led to the publication of “Interplanetary Internet (IPN): Architectural Definition” (URL, 62) in May 2001.

In the IPN scenario, transmission is subject to significant propagation delays, for example a minimum of roughly 4 minutes one-way light-trip time between Earth and Mars, and intermittent connectivity due to planetary movement and the occultation of spacecraft as they orbit a planet or as planets rotate. The extremely limited power available to many spacecraft dictates a particular need for efficiency at all protocol layers in the IPN. In addition to propagation delays and intermittent connectivity, low and highly asymmetric bandwidth as well as a relatively high bit-error rate also distinguish IPN communication from most of the terrestrial networking scenarios with which we’re familiar. The overall conclusion was that simply extending the Internet protocol suite to operate end-to-end over interplanetary distances wasn’t feasible and that new techniques would be necessary. On the application of ordinary internet protocols Cerf has stated his opinion that “you can’t tweak the protocols enough to make them useful and still be compatible” (Jackson, 2005). The IPNRG referred to their chosen approach as bundling, which builds a store-and-forward overlay network above the lower layers of underlying networks. Whereas the Earth’s Internet was basically conceived as a “network of connected networks”, the IPN was thought of as a “network of disconnected Internets” connected through a system of gateways forming a stable backbone across interplanetary space.

2.5.2 The birth of DTN

During 2001 and 2002, IPN researchers investigated how they could apply the IPN architecture to other situations in which communications were subject to delays and disruptions. In August 2002, the IPNRG published an updated version of the draft as “Delay-Tolerant Network Architecture: The Evolving Interplanetary Internet” (URL, 64), which described a generalisation of the architecture designed for IPN as an architecture for DTNs, a name coined by Kevin Fall. Among other updates, the IPNRG restructured its documents to distinguish between an architecture for delay-tolerant networking and the application of that architecture to various extreme communications environments, including the IPN. By this time, the IPNRG had realised that the different environments in which its architecture was applicable shared some essential characteristics, including the communication challenges introduced by long delays, intermittent connectivity, data rate asymmetry, packet loss, and errors. The updated drafts provided examples of extreme environments and presented some problems inherent to using existing Internet protocols and applications. The authors also considered extreme terrestrial environments in which communications were subject to intermittent, probabilistic connectivity that would benefit from the architecture - these included military tactical networks, sensor networks deployed in oceanic environments, and communities living in extreme environments, such as the Sámi people of northern Scandinavia. By May 2002, discussions were under way to recharter the IPNRG, and soon after, the DTNRG was formed. By 1 October 2002, a "dimming the lights" of IPNRG and its supporting interest groups had begun. Instead, the DTNRG began to address the architectural and protocol design principles arising from the need to provide interoperable communications with and among extreme and performance-challenged environments, where we can't assume continuous end-to-end connectivity. The DTNRG worked first on further generalising the IPNRG's architecture drafts and published its first draft in 2003 (URL, 62). Over the next few years, the research group refined the DTN architecture and eventually published RFC 4838 (Cerf et al., 2007).

2.5.3 The DTN architecture

The DTN Architecture RFC 4838 points out some fundamental assumptions, described in Table 2.21, built into the Internet architecture that are problematic in DTNs.

No.	Assumption
1	An end-to-end path between the source and destination exists for the duration of a communication session.
2	Retransmission based on timely and stable feedback from data receivers is an effective means for repairing errors (for reliable communication).
3	End-to-end loss is relatively small.
4	All routers and end stations support the TCP/IP protocol suite.
5	Applications need not worry about communication performance.
6	End-point-based security mechanisms are sufficient for meeting most security concerns.
7	Packet switching is the most appropriate abstraction for interoperability and performance.
8	Selecting a single route between sender and receiver is sufficient for achieving acceptable communication performance.

Table 2.21: Fundamental assumptions built into the Internet architecture

The DTN architecture relaxes most of these assumptions; it uses variable-length messages as the communication abstraction and a naming syntax that supports a wide range of naming and addressing conventions to enhance flexibility. It's designed to use storage within the network to support store-and-forward operation over multiple paths and potentially long times scales, and not to require but to support end-to-end reliability. The DTN architecture envisages security mechanisms that protect the infrastructure from unauthorised use by allowing for policy based discarding of traffic as quickly as possible. While the development of a DTN-specific policy language and distribution framework are out of scope of the DTNRG, DTN nodes "must at least implicitly incorporate some element of policy-based routing" (Farrell and Cahill, 2006b). Typical policy-based control types; ingress, egress and forwarding can all be applied in DTNs. The DTN architecture also assumes roughly synchronised clocks.

2.5.4 DTN protocols

As of January 2013, the DTNRG published three more informational and seven more experimental RFCs as outlined in Table 2.22.

No.	Title
RFC 4838 (Cerf et al., 2007)	Delay-Tolerant Networking Architecture
RFC 5050 (Scott and Burleigh, 2007)	Bundle Protocol Specification
RFC 5325 (Burleigh et al., 2008)	Licklider Transmission Protocol - Motivation
RFC 5326 (Ramadas et al., 2008)	Licklider Transmission Protocol - Specification
RFC 5327 (Farrell et al., 2008)	Licklider Transmission Protocol - Security Extensions
RFC 6255 (Blanchet, 2011)	Delay-Tolerant Networking Bundle Protocol IANA Registries
RFC 6256 (Eddy and Davies, 2011)	Using Self-Delimiting Numeric Values in Protocols
RFC 6257 (Symington et al., 2011)	Bundle Security Protocol Specification
RFC 6258 (Symington, 2011a)	Delay-Tolerant Networking Metadata Extension Block
RFC 6259 (Symington, 2011b)	Delay-Tolerant Networking Previous-Hop Insertion Block
RFC 6260 (Burleigh, 2011)	Compressed Bundle Header Encoding (CBHE)
RFC 6693 (Lindgren et al., 2012)	Probabilistic Routing Protocol for Intermittently Connected Networks

Table 2.22: DTNRG RFCs

The first of which, was the Bundle Protocol Specification (BP), that specifies the end-to-end protocol and abstract service description for exchanging messages (bundles) in DTN (Scott and Burleigh, 2007). In 2008, three more RFCs (Burleigh et al., 2008) (Ramadas et al., 2008) (Farrell et al., 2008) followed, describing the Licklider Transmission Protocol (LTP), a delay- and disruption-tolerant point-to-point protocol. LTP provides retransmission based reliability over links characterised by extremely long round-trip times (RTTs). In May of 2011 five RFCs were published. Two are informational RFCs to address Bundle Protocol IANA Registries (Blanchet, 2011) and use of Self-Delimiting Numeric Values in Protocols (Reschke, 2011). The three experimental RFCs published in 2011 specify the Bundle Security Protocol (Symington et al., 2011), Delay-Tolerant Networking Metadata Extension Block (Symington, 2011a) and Compressed Bundle Header Encoding (CBHE) (Burleigh, 2011). CBHE is a method for reducing bundle protocol overhead in bandwidth constrained environments in which the scheme specific part of any URI formed under any CBHE conformant scheme always has the following structure “node_number.service_number”(Clare et al., 2010).

Many routing protocols are specified for DTN. For example Park et al. (2012) describes position-based DTN Routing, Kim and Han (2012) describe contour routing in cellular networks and Aziz and Yamada (2012) describe a hand-off based and limited flooding (HALF) routing. RAPID (Balasubramanian et al., 2007b) treats DTN routing as a resource allocation problem that translates the routing metric into per-packet utilities such as minimising average delay, minimising missed deadlines and minimising maximum delay. A long term, and cyclic delivery metric is proposed in Cyclic MobiSpace (Liu and Wu, 2008), called expected minimum delay (EMD). EMD is the expected delay an optimal single-copy forwarding scheme takes to deliver a message generated at a specific time from a source to a destination assuming that optimal single-copy forwarding decisions are made in all nodes to minimise the delay of the message. However, only one routing protocol has been specified as an RFC: the Probabilistic Routing Protocol for Intermittently Connected Networks, an experimental RFCs published in 2012. PRoPHET describes a probabilistic routing protocol using the history of encounters and transitivity. PRoPHET presents a framework for probabilistic routing in intermittently connected networks, using an assumption of non-random mobility of nodes to improve the delivery rate of messages while keeping buffer usage and communication overhead at a low level. A probabilistic metric called delivery predictability and probabilistic routing protocol using this metric are defined.

The DTNRG is an active research group with roughly ten current Internet drafts on topics related to DTN cipher-suites, routing, and various other BP extensions. Aside from the DTNRG, an active research community is working on DTN-related topics. Researchers, for example, have conceived and implemented Delay-Tolerant Link State Routing (DTLSR) (Demmer and Fall, 2007) and Contact Graph Routing (CGR) (URL, 46; Burleigh, 2007), although, so far, only one routing scheme has been fully documented as an RFC. Further details about the DTN architecture's evolution are available in an architectural review paper (Fall and Farrell, 2008).

Licklider transmission protocol

The Licklider Transmission Protocol (LTP) (Burleigh et al., 2008) (Ramadas et al., 2008) (Farrell et al., 2008) is designed to serve as a reliable convergence layer for BP over single-hop, high-latency links. Long RTTs imply a substantial delay between the transmission of a block and the reception of an acknowledgement from the block's destination signalling its arrival. Unlike TCP, LTP sessions are unidirectional, so LTP peers can only achieve bidirectional data flow using two unidirectional links. To support scheduled communications, we might think of LTP as operating at a separate layer that knows the network state and uses lower-layer cues to tell each node when and how much to transmit. LTP does Automatic Repeat Request (ARQ) of data transmissions by soliciting selective-acknowledgement reception reports. To avoid under utilising expensive links, LTP doesn't postpone transmission until it receives acknowledgement that all prior blocks have arrived, but it allows multiple parallel data block transmission *sessions* to be in progress concurrently. Although LTP is principally aimed at supporting "long-haul" reliable transmission in interplanetary space as a convergence layer for BP, it's also been used in terrestrial environments such as in the Sensor Networking with Delay Tolerance (SeNDT) project (URL, 47).

The bundle protocol

BP was specifically developed conformant to the DTN architecture. It essentially runs at the application layer and generally follows the overlay-network approach. Although BP can run over TCP/IP, it can also run over other, lower-layer protocols (so-called convergence layers); for example, proprietary protocols deployed in sensor networks or, for deep-space deployments, LTP. BP's key capabilities include custody based retransmission; an ability to cope with intermittent connectivity; an ability to take advantage of scheduled, predicted, and opportunistic connectivity (in addition to continuous connectivity); and late binding of overlay network endpoint identifiers (EIDs) to convergence layer-specific addresses, such as IP addresses. Devices implementing BP are called DTN nodes. BP forms an overlay that employs persistent storage to help combat network interruption and for its store and forward function. This overlay includes transfer of reliable delivery responsibility, optional end-to-end acknowledgement, and several diagnostic and management features. For naming, BP uses a flexible scheme based on URIs RFC 3986 that can encapsulate different naming and addressing schemes in the same overall naming syntax. BP is layer-agnostic and focuses on virtual message forwarding rather than packet switching. Relatively little work has been done on how routing will work between different schemes. The *dtn* URI scheme is not fully standardised. When the application domain is limited to the space context, a much simplified naming scheme is preferred, as demonstrated in the DINET deep space flight experiment and the NASA DTN Experimental Network (DEN) (Clare et al., 2010).

As mentioned, bundle sources and destinations are identified via variable-length EIDs, which identify the original sender and bundles' final destinations, respectively. Bundles might also contain a *report-to* EID, used when a DTN node requests special operations to direct diagnostic output to an arbitrary entity. A single EID might refer to one or more DTN nodes, which can be members of groups called DTN endpoints. A DTN endpoint is therefore a set of DTN nodes. EIDs need not be related to routing or topological organisation. An EID is a name, expressed using the general URI syntax that identifies a DTN endpoint. In URI terminology, each URI begins with a scheme name followed by a series of characters constrained by the syntax defined by the scheme, and called the scheme-specific part (SSP). The DTN architecture dictates that the scheme designer is responsible for defining how to interpret an EID's SSP so as to determine whether it refers to the equivalent of a unicast, multicast, or anycast set of nodes. A DTN system can use any URI scheme it chooses, and there are as yet no conventions as to how different schemes could be used and how they might interact (Clare et al., 2010). Each node is required to have at least one EID that uniquely identifies it (Cerf et al., 2007). A bundle is considered to have been successfully delivered to a DTN endpoint when some minimum subset of the nodes in the endpoint has received the bundle without error. This subset is called the "minimum reception group" (MRG) of the endpoint. The MRG of an endpoint may refer to one node (unicast), one of a group of nodes (anycast), or all of a group of nodes (multicast and broadcast). A single node may be in the MRG of multiple endpoints.

Binding means interpreting an EID to select a next hop to which a bundle can be forwarded toward its destination. Because the destination EID is potentially reinterpreted at each hop, binding might occur at the source, during transit, or possibly at the destination. The latter two scenarios are referred to as late binding. This late binding of addresses to particular hosts in the Interplanetary Internet was important partly because it provided a way of accommodating node mobility in communication environments characterized by lengthy end-to-end delivery latency (Clare et al., 2010).

Bundles must wait in place in a queue until a communication opportunity is available. DTN nodes generally use some form of persistent storage, and stored bundles survive system restarts. Persistence requires that storage is available, well-distributed throughout the network, and sufficiently robust to store bundles until forwarding can occur.

In DTNs, information for making scheduling and path selection decisions is based on the requested data transfers' size and performance requirements. To enable such decisions, bundles contain an originating time stamp, useful life indicator, and a class of service designator. Routing schemes developed for DTN use various mechanisms, including packet replication, discovery of the meeting probabilities among nodes, and network coding. A DTN node might make forwarding decisions using measurements based on, for example, the known state of other DTN nodes, information on resource utilisation, and the probability of an encounter. Store-and-forward routing schemes also use information on node contacts, location, and future movement. A DTN node must in general base such decisions on locally held information, and might constantly reassess forwarding decisions as contacts come and go; for example, to determine the best next-hop DTN node, time to forward, and highest delivery probabilities for each bundle, and to remove failed paths. The DTN architecture allows for the use of many different routing schemes, each of which might prove to be advantageous depending on circumstances. However, contact information must be known, or discovered by a DTN node to form the basis for routing.

2.5.5 DTN-enabled applications

The communication primitives provided by the DTN architecture are based on asynchronous, message-oriented communication which differs from *chatty* request/response communications. A DTN-enabled application is modelled around sending and delivering application data units (ADUs), which can be of arbitrary length but which might be subject to implementation limitations; for example, in space applications, ADUs might be limited to being less than 64 Kbytes in size. The network might or might not preserve their relative order. Typically, the DTN sends or delivers ADUs to applications in complete units, although fragmentation can also occur within the network unless the ADUs are marked so as to prevent it. The BP transforms ADUs into one or more protocol data units called bundles that are then forwarded by DTN nodes. Applications send ADUs destined for an EID, and might arrange for delivery of ADUs sent to a particular EID, a so-called registration, which a DTN node can maintain persistently. This allows application registration information to survive application and operating system restarts. Bundles contain a primary block and one or more other blocks of data. The primary block contains basic information, such as the destination EID, which is required for bundle routing and forwarding. Each block can contain either application data or other information used to deliver the containing bundle to its destination. Blocks hold information typically found in the header or payload portion of protocol data units in other protocol architectures. The term *block* is used instead of *header* because blocks might not appear at the beginning of a bundle due to particular processing requirements. Bundles can be fragmented into multiple constituent bundles during transmission, and these fragments can be further fragmented. Two or more fragments might in principle be reassembled anywhere in the network, again forming a new bundle.

2.5.6 Forwarding in DTNs

DTN routing involves forwarding by either a single-copy or multi-copy approach. Single copy schemes are more efficient in terms of reducing traffic overhead. Single-copy approaches try to reduce buffer usage and the number of message transfers, but suffer from longer delays and low delivery ratios. Multi-copy schemes,on the other hand, achieves lower delays and higher delivery ratio at the cost of buffer space and more message transfers (Abdulla and Simon, 2007). DTN routing schemes may rely on node mobility to transfer messages. (Schurgot et al., 2012) describe the evolution of routing protocols for intermittently connected ad hoc networks and discuss the trend toward social based DTN routing protocols.

2.5.7 DTN adoption

Several wireless sensor networks have deployed DTN, and many other DTN deployments are described elsewhere, (Farrell and Cahill, 2006a) including using DTN for underwater acoustic networking, meteorological and animal tracking, and various other sensor networks. Since 2003, DARPA has had a DTN program with the aim to develop and field network services that deliver critical information reliably even when no end-to-end path exists through the network. DARPA based phases one and two of its program on the Spindle project led by BBN Technologies. The third phase of DARPA's DTN program aims to create the first *fieldable* equipment that uses DTN to access military tactical information. The Technology and Infrastructure for Developing Regions (TIER) (URL, 48) project aims to address challenges in bringing the IT revolution to the masses in developing world regions. TIER, a research group at the University of California, Berkeley, has key projects in educational tools, health care, wireless (WiLDNet), distributed storage (TierStore), and speech technologies. DakNet was an early DTN project the MIT Media Lab developed, with, apparently, some commercialisation from First Mile Solutions (URL, 49). It was one of the first instances to use scheduled transport, using data mules, to carry bundles between Wi-Fi equipped "kiosks" in villages on a regular basis. The N4C project also used data mules and is based on previous work from the SNC project. These projects use opportunistic encounters with data mules in the Swedish Arctic, for the most part helicopters to transfer bundles between the temporary camps of the Sámi people, and the Internet. Researchers are also investigating DTN in the context of disaster and emergency network support. Web and email applications deployed during the final two month trial of the 2010 N4C project(McMahon et al., 2011). Merani et al. (2011) describe a design and implementation of an Underwater Convergence Layer (UCL) for the DTN2 Reference Implementation, which provides DTN support for the WHOI Micro-Modem, which was tested in the field during the Acommsnet10 sea trial (URL, 50). The Multimedia and Mobile Communications Laboratory (MMLAB) at Seoul National University has been investigating its Architecture for Intelligent Emergency DTN using extensive temporary wireless communications. And, not forgetting the origins of DTN, in 2008, NASA JPL conducted experiments simulating communications with rovers on the surface of Mars relayed through a DTN bundle agent installed on the Epoxi spacecraft, previously known as Deep Impact. As far as we know, this experiment set a distance record for RFC-compliant protocols with BP and LTP being used over 25 million km hops, for a total round trip of 50 million km. The Consultative Committee on Space Data Systems (CCSDS) (URL, 51) started a DTN working group that's examining the suitability of the BP and LTP experimental RFCs for use as CCSDS standards for future space missions. Both NASA and the European Space Agency (ESA) have several DTN-related activities underway that are feeding into this standardisation process. Several schemes combine DTN and mobile ad-hoc networks (MANETs). A performance evaluation of MANET and DTN routing protocols attempts to identify when to employ each protocol (Del Duca Almeida et al., 2012). Several DTN routing schemes for VANETs have been proposed in the literature (Khan et al., 2012), (Benamar et al., 2012),(Martinez Tornell et al., 2012), (Mekbungwan et al., 2011). At the heart of Delay-Aware Data Delivery (DADD) scheme for Vehicular Intermittently Connected Networks (VICNs) is a novel mechanism that allows a source stationary roadside unit (SRU) to carry out necessary bundle retransmissions to high speed vehicles newly entering its communication range (Khabbaz et al., 2013). DTN are proposed for use in energy constrained Disaster Response Networks (DRN). One approach by Uddin et al. (2012) exploits naturally recurrent mobility and contact patterns in the network, to reduce the number of message copies needed to attain an adequate delivery ratio in the face of disconnection and intermittent connectivity. Azad et al. (2012) consider DTN based Autonomous Underwater Vehicles (AUVs) in a coastal surveillance context.

2.5.8 DTN resources

Various other implementations of the BP and LTP are linked from the DTNRG Web site (dtn, 2013). The DTNRG maintains an open mailing list (URL, 52) for general discussions, and a specific list exists for developers and users of the DTN2 reference implementation (URL, 53). Khabbaz et al. (2012) present a comprehensive survey on recent developments and persisting challenges in DTN. A document that examines the state of the art of DTN as it existed during the early part of the N4C project both from a technical point of view and in terms of the demonstrations and test beds that had been created using this technology, (n4c, 2010a), which also reviews other projects related to rural Internet initiatives both from a technical and business point of view was an outcome of the N4C project. A functional specification of the DTN infrastructure software was needed to support the testbeds developed in the N4C project (Elwyn Davies, 2010). This functional specification also details the network communications mechanisms used by the applications developed, and deployed during the project. A document that describes one of the DTN email applications, the *PyMail* system (n4c, 2010b), used to send email in DTN bundles. This version contains information on the progress and evaluation of the system during the period of the 2009 Summer tests in arctic Sweden. We provide a description of the design for a DTN node used in the N4C project's summer 2009 trials (McMahon et al., 2009). The design uses COTS products and applications; the DTN2 implementation of the Bundle Protocol (BP) and provides basic network access, e-mail and web access via the DTN. In our report on an Arctic Summer DTN Trial in 2009 (McMahon et al., 2011), we validate our design and successfully demonstrated the use of email via helicopter transported data-mules. NASA have a Disruption Tolerant Networking program (URL, 67) aimed at establishing a long-term, readily accessible communications test-bed onboard the International Space Station (ISS). Two Commercial Generic Bio processing Apparatus (CGBA), CGBA-5 and CGBA-4, serve as communications test computers that transmit messages between ISS and ground Mission Control Centers. All data is monitored and controlled at the BioServe remote Payload Operations Control Center (POCC) located on the Engineering Centre premises at the University of Colorado - Boulder.

NASA announced an opportunity for industry to participate in NASA's "Space DTN project" mid 2011 (URL, 54).

2.5.9 DTN implementations

The most prominent DTN implementations are reviewed in the following.

As part of his PhD thesis work (Demmer, 2008), Mike Demmer developed an open source reference implementation of the BP called "DTN2".A comprehensive description of the major components and their function is provided in Demmer's Thesis. Since 2008, and until July 15, 2014, I have maintained and managed the DTNRG code, including DTN2, and the DTNRG WiKi, as part of the N4C and SAIL projects. The goal of DTN2 is to clearly embody the components of the DTN architecture, while also providing a robust and flexible software framework for experimentation, extension, and real-world deployment. DTN2 provides a fairly complete DTN and BP software suite and has been used in many experimental DTN deployments. DTN2 soundly validated the core proposals of the DTN architecture and its mechanisms: that in challenging environments, a store-and-forward message overlay network performs significantly better than existing approaches, in some cases within a small percentage of the theoretical maximum achievable throughout (Demmer et al., 2003). DTN2 The system uses standard POSIX interfaces and UNIX system designs and the C++ Standard Template Library for most internal data structures.

The Institute of Operating Systems and Computer Networks (URL, 55) provided an implementation of RFC5050, IBR-DTN (URL, 56), designed for embedded systems which can be used as framework for DTN applications. The module-based architecture with miscellaneous interfaces, makes it possible to change functionalities like routing or storage of bundle just by inheriting a specific class. The modules comprise a DTN Core, Bundle Router, Persistent Storage and a Convergence Layer Manager. While IBR-DTN is work in progress a practical evaluation in a mobile scenario in which a vehicle mounted node passes a stationary node has been presented (Doering et al., 2008)

The Interplanetary Overlay Network (ION) (Burleigh, 2007) is another open source DTN implementation, developed by JPL and is partially maintained by Ohio University (URL, 57). ION consists of a completely separate BP implementation, contact graph routing, LTP, and NASA's Asynchronous Message Service (AMS), which provides an application-layer framework for using DTN. ION is an implementation of the bundling protocol (BP) as described in RFC 5050, AMS, and the LTP found in RFCs 5325, 5326, and 5327, that is intended to be usable for interplanetary communications. ION includes implementations of RFC 5050, the LTP (RFC 5326), and the CCSDS File Delivery Protocol and Asynchronous Message Service. ION, JPL's alternative implementation of RFC 5050, is aimed at addressing the constraints faced with digital communication between interplanetary spacecraft and space flight control centers on Earth and enabling delay-tolerant network communications in interplanetary mission operations. The space-flight-specific constraints, and architecture of ION is presented in (Burleigh, 2006). ION is designed to enable inexpensive insertion of DTN functionality into embedded systems such as robotic spacecraft. The intent of ION deployment in space flight mission systems is to reduce cost and risk in mission communications by simplifying the construction and operation of automated digital data communication networks spanning space, planetary surface and terrestrial links. ION was announced as available on SourceForge at the end of June 2011 (URL, 58). The ION software is a suite of communication protocol implementations designed to support mission operation communications across an end-to-end interplanetary network, which might include on-board flight subnets, in-situ planetary or lunar networks, proximity links, deep space links, and terrestrial Internets.

JDTN (URL, 59) is a Java-based implementation of RFC 5050 and RFC 5326.

The Bytewalla project (URL, 60) of Telecommunication Systems Laboratory (TSLab) (URL, 61) developed an implementation of RFC 5050 for the Android Platform. The projects aim was to connecting African rural villages using Android phones. The idea is that people act as "data-mules" travelling from villages to cities carrying their Android mobile phones carry data, such as email, with them.

The Postellation project (URL, 65) of Viagenie URL (66) developed "Postellation", an implementation of RFC 5050 with a focus on providing a lean Bundle protocol implementation, smart HTTP proxy and for easy deployment of DTN networks. Amongst other features Postellation supports IPv6 and is intended to be deployed without prior configuration.

2.5.10 Analysis

Overlay networks are used as a promising platform to deploy wide area of applications and services in the Internet. The application level state maintained by the overlay networks should have high degree of availability. This can be compromised when a significant percentage of overlay nodes fail simultaneously leading to loss rate, and end-to-end latency increases (Beitollahi and Deconinck, 2009).

The evolution of transport infrastructures for IP provide an example of how decreasing vertical integration leads to various efficiencies (Beitollahi and Deconinck, 2009). Through provision of a layered, network stack DTN promotes standard based application code reuse and leads to various efficiencies. Convergence is achieved via a "convergence layer". The end state of the convergence argument is the concept of

Everything Over Some Layer (EOSL). (Clare et al., 2010) asserts that a simplified scheme should be the minimal naming mechanism that all space DTN nodes must implement.

In 2008, Wood et al. (2009) asserted that the DTN bundling architecture had a number of open real-world deployment issues that could be addressed. Table 2.23 provides an overview of the problems with the bundle protocol.

No.	Problem	Description
1	Reliability, error detection, checksums and performance	Bundle Protocol specification does not attempt to detect erred bundles
2	Time synchronisation problems	The Bundle Protocol assumes that all communicating bundle nodes share a common, simultaneous, synchronised, conception of UTC time
3	Learning the current time	not possible for a node to learn the correct time using the Bundle Protocol
4	Time standard	The Bundle Protocol uses Coordinated Universal Time (UTC)
5	Convergence layer adaptor roles	Different UDP implementations are currently only compatible when sending bundles that fit within single UDP datagrams without convergence-layer fragmentation.
6	Maximum transmission sizes and fragmentation	no method for advertising or negotiating a bundle MTU
7	Agreement on naming schemes	Different Bundle Protocol implementations are currently supporting multiple different naming schemes for Bundle Protocol Endpoint Identifiers (EIDs)
8	Standardisation of routing methods	Need for common routing protocols and address resolution techniques is related to the issue of common EID schemes for naming of destinations.
9	Network management	DTN nodes currently have no support for remote management, which is common in IP networks.
10	Quality of Service	The bundle protocol has defined some bits to indicate one of three priority levels in a bundle, but the semantics of these levels are as yet undefined
11	Efficiency of protocol overhead	The Bundle Protocol can be efficient in terms of metadata overhead if bundles can be made large, but this is not natural for all applications.
12	Complexity and performance	The use of variable-length, rather than fixed-length, fields with Self-Delimiting Numeric Values (SDNVs) and Endpoint Identifiers, and the referencing back to the dictionary in the primary block from later payload blocks
13	Security	The bundle security mechanisms are not finalised, and some aspects of them remain to be completed.
14	Content identification	The Bundle Protocol does not identify the content it carries to select an application to hand the content off to.

Table 2.23: Problems with the Bundle Protocol as described by Wood et al. (2009)

Since 2008, many of these problems have been addressed. In their comprehensive survey on recent developments in DTNs, Khabbaz et al. (2012) describe several problems outlined in Table 2.24. The authors assert the less network information there is available, the more the requirement for learning procedures increases, and that such procedures are bandwidth consuming.

The bundle security standards have since been defined. Nevertheless, existing DTN problems must be addressed to make a better Internet, and this is a goal of the work in this thesis. Khabbaz et al. (2012) state that an essential challenging problem is “the design of more intelligent, efficient, and chattiness free network learning procedures”. The work in this thesis addresses this problem through provision of a more intelligent, efficient, and chattiness free network learning procedure: TDP. The DTN architecture deals with a certain set of underlying constraints that include size, connectivity, and interconnection

No.	**Defines**
1	Development of a generalised DTN model
2	Procedure of routing protocols for setting up paths between communicating nodes.
3	The design of more intelligent, efficient and chattiness free network learning procedures
4	Procedure to instruct intermediate nodes to discard delivered copies of a bundle
5	No bundle security standards have been defined

Table 2.24: Current DTN problems described by Khabbaz et al. (2012)

of disparate name space structures. DTN is appropriate for broad range of contexts as it provides a standards based, publicly available, networking protocol that can use the Internet. It reduces the need for operator intervention through automated commands, telemetry transmission and receipt. DTN can reduce the infrastructure requirement, and exploit an opportunistic contact by minimising transmissions, and use of telemetry receipt, leading to efficient link utilisation.

2.6 DTN discovery mechanism

Providing delay tolerant discovery guarantees requires specific mechanisms at every layer. The approach to addressing the architectural and protocol design principles in the RHNC characterises work in the DTNRG. A number of drafts in the DTNRG are particularly relevant to the work presented in this thesis. In particular, a number of projects have explored how to capture the DTN routing requirements of discovery mechanisms. In addition to this work, four bodies of work deal with discovery of DTN endpoint elements and their attributes. We review each of these in detail.

2.6.1 Routing requirements of discovery mechanisms

Routing protocols may require prior knowledge, or the ability to learn about various elements of a network. Prior knowledge of node mobility and connectivity, are assumed by several DTN routing schemes to perform message transfers (Abdulla and Simon, 2007). Conventional routing protocols do not consider a path to be viable when there is no next hop route. In contrast, with DTN routing protocols, a prediction of a future viable paths might be informed by calculating the probability of link availability, based on history. Through prioritisation, based on prediction, of both the schedule of packets transmitted to other peers and the schedule of packets to be dropped, MaxProp is demonstrated to outperform protocols that have access to an "oracle" that knows the schedule of meetings between peers (Burgess et al., 2006). Distance vector, link state, and source routing approaches are used to determine an immediately available path from source to destination. The path computation of Delay Tolerant Link State Routing (DTLSR) takes prediction into account when discovering viable paths (Demmer, 2008). The convergence layer informs DTLSR on link availability. DTN routing protocols are generally not responsible for providing neighbour discovery. DTLSR requires a discovery mechanism to provide connectivity information which it distributes throughout a network. The DTLSR implementation in DTN2 uses CLA specific discovery protocol mechanisms for discovery of proximate nodes, which issues calls to the routing layer when connectivity is detected, or lost, between nodes.

Resource Allocation Protocol for Intentional DTN routing (RAPID) discovers network resources via a control plane that helps nodes in acquiring complete information about network state. In RAPID each node exchanges such information as the number and location of replicas, and average size of past transfers (Khabbaz et al., 2012).

Probability prediction requires information. An optimal probabilistic forwarding protocol is demonstrated to maximise the delivery rate of each message when mean inter-meeting times between all pairs of nodes is known (Liu and Wu, 2009). Previous works have proposed a variety of long term metrics, including social pattern similarity (Daly and Haahr, 2007) and encounter patterns (Lindgren et al., 2003) which routing protocols can exploit. One advantage of long-term delivery metrics is that they are relatively stable once generated from historical connectivity information or prior knowledge on the contact pattern of nodes, avoiding the cost associated with frequent updates. On the other hand, "many real objects have cyclic motion patterns, and therefore, it is possible and valuable in practise to increase the accuracy of a delivery metric by allowing it to be time-variant" (Liu et al., 2009). Delegation forwarding (Erramilli et al., 2008) forwards the messages based on different delivery probability metrics. Delegation forwarding maintains a forwarding threshold when contact of node pairs occurred. The forwarding threshold indicates the quality of node such as cost, delivery rate and average delay. Node mobility, contact, and delivery metrics are a subset of network elements that can be structured and stored using data storage models. Such elements can be extracted and made accessible in ageneric interchange format to local routing protocols, or made available to the routing protocols of remote entities, via appropriate DTN mechanisms.

In the context of intermittently connected networks, a flooding approach, in which every node that receives a packet and broadcasts it to all of its neighbours, may not achieve the goal of reaching as many nodes as possible due to network partitions (Abdulla and Simon, 2007). The “Epidemic” routing approach has been shown to outperform other ad hoc routing protocols whic are unable to deliver any messages in the presence of network partition (Vahdat and Becker, 2000). The DTN Publish / Subscribe Protocol (DPSP) (Greifenberg and Kutscher, 2008) is a probabilistic multicast routing protocol for opportunistic networks. DPSP routers do not try to maintain a view of the network topology and select an optimal path. Instead, the routers replicate bundles to their neighbours in order to get the bundle delivered by multiple hops using store-and-forward techniques. Greifenberg and Kutscher (2008) assert that epidemic routing and “spray-and-wait” approaches have the advantage that they can be applied without any knowledge about the network, and do not require any kind of global coordination. The goal of Epidemic routing is “to deliver a message (update) with high probability to a particular host”and a key issue is deciding whether to transmit a message in the event of a contact (Vahdat and Becker, 2000). Such decision might be based on knowledge gained through appropriate discovery mechanisms.

A general destination EID mapping function is required that can be used by a number of routing schemes; endpoint discovery protocols may be a good vehicle for developing such a scheme (Farrell, 2010). Routing protocols could make use of information learned from DTN endpoint discovery protocols. In particular, by supplying the attributes of endpoints to epidemic protocols (Vahdat and Becker, 2000), such as Context-aware Adaptive Routing (CAR) protocol (Musolesi and Mascolo, 2009) which derives delivery probabilities from context information and defines context as the set of attributes that describe the aspects of the system that can be used to drive the process of message delivery.

A node-density based adaptive multicast routing scheme to address the challenges of opportunistic link connectivity in DTNs is described (Yang and Chuah, 2006). The routing scheme and simulation assume DTN neigbour discovery, that is, configurable contact probing or beaconing is assumed. Discovery is required in DTNs to enable a source node use routing protocols to attempt to probabilistically transfer data packets to a destination node (Kim et al., 2008). When a network node moves the network topology changes. A consequence of node mobility is that route discovery packets are re-broadcasted, consuming more control packets, in order to rebuild or repair a broken path; a critical challenge in building routing protocols (Chuang et al., 2012).

In DTNs small activity cycle lengths waste energy without discovery of more contacts (Jun et al., 2009)(Xi et al., 2007) and therefore the cycle lengths of adaptive sleep schedules can be relatively large. The optimal stopping rule of (Liu et al., 2009) extends the notion of expected minimum delay, which is the expected time an optimal single-copy forwarding scheme takes to deliver a message generated at a specific time from a source to a destination, from the single-copy to the multi-copy forwarding case. Three potential utility utility-based spraying functions were proposed by (Spyropoulos et al., 2007): Last-Seen-First Spraying, Most-Mobile-First Spraying and Most-Social-First Spraying. A methodology to derive the optimal spraying policy was proposed by (Jindal, 2006) that used distance utility-based spray strategy to distribute copies amongst the potential relays and utilised dynamic programming to achieve expected delays very close to the optimal. An adaptive distributed spray mechanism (AMR) that determined the depth of spray tree by relay nodes was proposed in (Li et al., 2006).

To exploit a contact a data transfer application or routing protocol should be capable of reordering or prioritisation of messages, for transfer, and preemption of transfer opportunities. While protocol studies generally assume that two nodes will exchange all the information they possibly can while the contact lasts, routing protocols should not just determine the order in which to replicate messages but also when to stop and turn to a potential next peer, without knowing if such a peer is present (Pitkanen et al., 2012).

2.6.2 DTN with new serial convergence layer

Bong and Yeo (2011) defined a discovery mechanism for wireless serial RS-232 communication and networking protocol based on DTN2 reference implementation, called Delay Tolerant Network with New Serial Convergence Layer (DTN-NSCL). Periodic announcement of local DTN endpoint ID are used to inform any DTN peer nodes about itself. Discovery mechanism also registers and tabulates announcement sent by neighbouring DTN peer nodes. The destination address of a peer is obtained through periodic neighbour discovery signalling. We observe that the results highlight the significant improvements over the DTN2 serial layer, that are attributed to discovery. The authors describe discovery operations that involve periodic announcement and a neighbour discovery signal packet. While knowing what content of the beacon is amd how to specify the content is crucial, the authors do not provide a description of how to achieve this. Ultimately, the proposed approach, is specific to the DTN-NSCL CLA and consequently not broadly applicable.

2.6.3 DTN2 RI base discovery

The base discovery mechanism of the DTNRG DTN reference implementation (DTN2)(Demmer, 2008) does not rely on link state announcements to determine next-hop neighbours. Because DTN is an overlay network, the different CLAs may have protocol specific mechanisms for discovering nearby nodes. The DTN2 discovery mechanisms are designed so that the different CLAs are responsible for issuing calls to the routing layer when connectivity is detected, or lost, between nodes. Consequently, the routing layer is responsible only for distributing discovered connectivity information throughout the network, and not necessarily for discovering neighbours. An RFC 4838 opportunistic contact lasts until the link is lost or terminated. Opportunistic links are opened in response to some external event, such as discovery of a nearby node, but are not automatically reopened if the connection fails.

The DTN2 discovery mechanisms generally bind to a UDP socket to listen for neighbour beacons. IP based neighbour discovery mechanisms bind to a UDP port to listen for unicast, broadcast, or multicast beacons from advertising neighbours. Bluetooth has built-in discovery mechanisms, consequently DTN2 Bluetooth-based discovery polls via Inquiry instead of listening on a socket, as beacons use built in inquiry and SDP mechanisms. To advertise a local IP based CLA, a local IPv4 address (and UDP or TCP port) is registered. Outbound beacons are formatted to advertise a CLA instance. For each CLA registered, the discovery mechanism advertises or announces the CLA's presence to neighbours via beacons, and distributes each received event of neighbour discovery to each CLA. DTN2 has a discovery instance that uses a multicast DNS (mDNS) protocol, i.e. Bonjour, to browse, or announce TCP services on the local network. An announce instance, for which there may be several, represents a CLA. Each announce instance records its CLA's IP address and the interval at which to announce or poll for neighbours. The DTN2 discovery mechanisms maintain a list of announce instances, which serve as the basis for its CLA advertisement. An announce instance also serves as a responder. Announce instances respond to inbound advertisements by creating a new contact. For each CLA discovered DTN2 creates a new contact to the neighbour node by placing the appropriate call into its CLA. If no other options are set for destination, the default is to transmit to the IPv4 broadcast address.

DTN2 discovery has a modular design which enables CLA specific discovery instances to use their CLA specific mechanisms, in order to discover proximate CLA instances, of the same type. When connectivity is detected, or lost, between CLA discovery instances, the instances subsequently issue up-calls to the routing layer. As the structural components are CLA specific, consequently, the objects exchanged by discovery are tightly coupled with the different underlying transports. The structure of information, interchange mechanisms, data storage models for discovery information, are CLA specific and therefore not generically applicable. In contrast, the work in this thesis must support the discovery of named network elements, that can be operated on by generically applicable structural and functional components of the naming facility. Furthermore, a node may not discover its CLAs, and consequently each CLA specific discovery and announce instance must be configured using either static, manual configuration prior to conducting discovery. Finally the DTN2 mechanisms described for CLA discovery were not formally defined, not independent of, and are applicable only to the subset of DTN2 CLAs, which are IP based. In contrast, the work in this thesis must support formally defined discovery, and exchange of all CLA attributes.

2.6.4 Automated bundle agent discovery

The Automated Bundle Agent Discovery for Delay/Disruption-Tolerant Networking (ABAD) (J. Wyllie, 2007) draft describes a mechanism that RFC 5050 Bundle Agents (BAs) can use to discover when they are in contact with one another and optionally provide information on the additional properties of current or future contacts, such as duration and capabilities, in order to provide a DTN node with information that can be used to trigger bundle forwarding or make future bundle scheduling decisions.

ABAD is specified to inter-operate between differing BA implementations. Automated means for BAs to discover neighbouring nodes, some of their neighbours' capabilities, and contact times as well as the mechanism's relationship to potential DTN routing protocols are described. Two nodes are said to be "neighbours" over a convergence layer if and only if the nodes can communicate symmetrically directly over that convergence layer.

When they lack bidirectional connectivity, the node capable of sending is called a "pitcher", and the node capable of receiving is called a "catcher". DTN ABAD borrows the terms "pitcher" and "catcher" from JPL's implementation of the Asynchronous Message Service (AMS) (for Space Data Systems (CCSDS), 2011). Discovery is defined for use within a DTN by allowing BAs to automatically discover other nodes to which they can directly forward bundles. A means that will inter-operate between differing BA implementations is outlined that allows them to exchange contact information over any shared convergence layers. All auto-discovery data is encoded in the payload block of the RFC5050 bundle. Auto-discovery bundles are addressed to the administrative endpoints of BAs themselves. A BA that receives a request to initiate auto-discovery initiates a specific procedure based on whether the data is domain dependent or domain independent. Depending on the outcome of this negotiation contact information will be updated on the BA. Discovery information is classed as either domain dependent or domain independent. Domain dependent information might be seen as deployment context specific, and domain independent is information that is consistently useful across contexts. While the domain independent portion of the auto-discovery bundle is always included, it is optionally followed by the domain-specific portion. However, domain dependent information may be useful across conexts. Both domain dependent and domain independent information describe CLA capabilities; the structuring of other information is not described. The dual-level design is said to provide flexibility in determining information about other BAs.

Amongst the other approaches to discovery, the ABAD approach is particularly relevant, as it takes into account the importance of CLA agnostic discovery in contexts with relatively "long" propogation delay, "High" bit-error rates, and "high" link asymmetry. All auto-discovery data is encoded in the payload block of the bundle in order to work regardless of the specific CLAs available, and could be used to indicate predicted future contacts, and contact durations. The semantics and application-development support offered, however, are constrained. In contrast, our work must provide a systematic process to develop applications composed of autonomous network elements. The lack of structure in the protocol is presented as being a necessary flexibility in deployment. However, it seems inflexible in terms of the discovery data and data structures. We assert ABAD is limited by a structure that is CLA centric. In order to provide a set of CLAs a BA can communicate using an SNDV based interchange format. SDNVs provide a mechanism to avoid sending more bytes than needed, as well as avoiding inappropriate fixed sizes. However SDNV is specific to DTN, and consequently does not provide a generic interchange mechanism. Finally, ABAD is not generally applicable as a single bundle-layer discovery protocol that maps to underlying discovery protocols, such as Boujour for exmple. In contrast, the work in this thesis must specify a discovery mechanism that is generally applicable as a single bundle-layer discovery protocol that maps to underlying ones

2.6.5 DTN Internet protocol neighbor discovery

The Disruption Tolerant Networking (DTN) Internet Protocol Neighbor Discovery (IPND)(D. Ellard, 2012) is particularly relevant to our work. DTN IPND describes a protocol for nodes to learn of the existence, availability, and IP addresses of other DTN nodes. IPND sends and listens for small IP UDP announcement "beacons" that are small UDP messages in the IP underlay, to advertise presence. IPND is designed to be simple, efficient, and general and use of IPND is recommended in DTNs only.

Beacon messages are addressed to an IP unicast, multicast, or broadcast destination to discover specified remote neighbours, or unspecified local neighbours in the topology. IPND beacons advertise neighbour availability by including the DTN node's canonical endpoint identifier. IPND beacons optionally include service availability and parameters. In this way, neighbour discovery and service discovery may be coupled or decoupled as required. Once discovered, new neighbor pairs use advertised availabilities to connect and exchange routing information. Neighbor Discovery (ND) and the ability to dynamically discover other DTN nodes is described as an "important primitive" of DTNs. IPND specifies beacon messages as small UDP messages in the IP underlay with content that is agnostic to the underlying transport mode may be sent either as IP unicast, multicast, or broadcast UDP packets. Unicast beacons are transmitted to known and enumerated neighbours. Broadcast beacons are used within the local network broadcast domain. Multicast RFC 1112 beacons potentially include multiple networks across routed boundaries.

In IEEE LANs beacons are transmitted as link-layer broadcast messages.

Upon discovering a neighbour and its services, an IPND IP based CLA can negotiate the connection per its individual specification and installs the appropriate next-hop routing information in the local node. In contrast to link or physical layer discovery, IPND is said to enable a general form of neighbor discovery across a heterogeneous range of IP based links, as are often found in DTN networks. IPND is expected to be useful in mobile, ad-hoc DTN environments where meeting opportunities are not known a priori and connections may appear or disappear without warning.

To discover the bi-directionality of links, an IPND Neighborhood Bloom Filter (NBF) (URL, 107) facility in which each node advertises a Bloom filter representation of the set of neighbours from whom it has received enough recent beacons to be considered “up”. When a node receives a beacon from an “up” neighbour that advertises an NBF which represents a set containing the receiving node’s ID, then the link is considered bi-directional.

Beacon messages optionally include a block of service availability information, the structure of information is intended to be sufficiently general to accommodate implementation-specific services provided by the advertising node. To maintain the current neighbour set, IPND removes stale neighbours after the defined neighbour receive timeout period elapses without receiving any beacon messages from a particular neighbour.

Raytheon BBN Technologies (BBN) has implemented and deployed an earlier version of IPND as part of the Spindle project. BBN have developed an implementation of DTN IPND which has been added to the bundle protocol reference reference implementation DTN2. This implementation is a feature separate from the IP Discovery functionality that is included by default in DTN2. The IPND reference implementation was designed to be highly extensible. Implementing custom types and services requires little more than mirroring of definitions in code. The existing IPND Service Block tag-length-value (TLV) encoding infrastructure handles almost all reading and writing of raw byte streams.

IPND attempts to maximise the utility of each beacon message without requiring multiple round-trip transmissions in order to perform complex protocol negotiation.

While the IPND draft describes an IP based neighbour discovery protocol, the authors assert the principles and basic mechanisms used may also be expressed in terms of other datagram protocols. However the principles and basic mechanisms used are described not only in terms of IP datagrams but also in terms of CLAs that are tightly coupled with IP, for example the TCP or UDP CLA. Beaconing methods are expressed in terms of IP unicast, braodcast and RFC 1112 multicast. That is an IPND node must support RFC 1112 (Deering, 1989) multicast IP destination addresses and multicast RFC 3376 (Cain et al., 2002) IGMP group membership and the service definitions for CLA-TCP-v4 and CLA-UDP-v4. In contrast, the work in this thesis must use beacon messages that are not UDP messages in the IP underlay

The main difference between IPND and traditional IP discovery architectures is that complex protocol negotiation is bounded in round-trip transmissions. A publisher defines a UDP beacon message, that optionally includes a block of service availability information and transmits it to an IP unicast, multicast, or broadcast destination. The proposed approach, however, seems inappropriate for discovery in challenged contexts, as cooperating nodes first have to establish communications using UDP, which is unreliable and was not designed to be tolerant of disruption and constrained networks in order to establish communications using DTN protocols. Furthermore, the characteristics of the environment might mean that in some contexts DTN nodes using IP addressing cannot communicate with DTN nodes that are not, which can lead to missed opportunity. Finally, the scalability of the mechanism has not been studied, and no guidance is provided for the design of other datagram protocols.

.

2.6.6 The DTN endpoint discovery protocol

Communication may be unidirectional in DTNs, that is a DTN node may be able to send to another DTN node, but not vice versa. Pull mechanisms are not appropriate in such contexts. The DTN Endpoint Discovery Protocol (EDP) (McMahon and Fall, 2010) draft describes Version 1 of EDP (EDPv1). EDP is a DTN discovery application protocol designed to be used with RFC5050 bundles within the context of the DTN architecture (RFC 4838. EDP can be used by a DTN node to discover the presence of DTN nodes wishing to receive EDP Application Data Units (ADUs) on their EDP registrations, and to discover specifically which CLAs and registrations are of interest to those proximate DTN nodes. A DTN node with an EDP registration is an EDP node. EDP nodes report a description of their active EID registrations, CLAs EIDs, storage, routing, security capabilities and whether they are willing to be custodian to proximate EDP nodes.

2.6.7 Analysis

The proposed approach of DTN-NSCL, is specific to the DTN-NSCL CLA and consequently not broadly applicable.

DTN2 RI discovery has a modular design which enables CLA specific discovery instances to use their CLA specific mechanisms, in order to discover proximate CLA instances of the same type. When connectivity is detected, or lost, between CLA discovery instances, the instances subsequently issue up-calls to the routing layer. As the structural components are CLA specific, consequently, the objects exchanged by discovery are tightly coupled with the different underlying transports. The structure of information, interchange mechanisms, data storage models for discovery information, are CLA specific and therefore not generically applicable. In contrast, the work in this thesis must support the discovery of named network elements, that can be operated on by generically applicable structural and functional components of the naming facility. Furthermore, a node may not discover its CLAs, and consequently each CLA specific discovery and announce instance must be configured using either static, manual configuration prior to conducting discovery. Finally the DTN2 mechanisms described for CLA discovery have not been provided as a specification. Furthermore they are not independent of, and are applicable only to the subset of DTN2 CLAs.In contrast, the work in this thesis must support and be capable of exchange of all CLA attributes.

Amongst the other approaches to discovery, the ABAD approach is particularly relevant, as it takes into account the importance of CLA agnostic discovery in contexts with relatively “long” propagation delay, “High” bit-error rates, and “high” link asymmetry. All auto-discovery data is encoded in the payload block of the bundle in order to work regardless of the specific CLAs available, and could be used to indicate predicted future contacts, and contact durations. The semantics and application-development support offered, however, are constrained. In contrast, our work must outline a systematic process to develop applications that translate network elements into things. The lack of structure in the protocol is presented as being a necessary flexibility in deployment. However, it seems inflexible in terms of the discovery data and data structures. We assert that ABAD is limited by a structure that is CLA centric, to provide a set of CLAs a BA can communicate with, and an interchange mechanism that is an SDNV. The capabilities SDNV comprise the ABAD interchange mechanisms. SDNVs provide a mechanism to avoid sending more bytes than needed, as well as avoiding inappropriate fixed sizes. However SDNV is specific to DTN, and consequently does not provide a generic interchange mechanism. Finally, ABAD is not generally applicable as a single bundle-layer discovery protocol that maps to underlying discovery protocols.In contrast, the work in this thesis must specify a discovery mechanism that is generally applicable as a single bundle-layer discovery protocol that maps to underlying ones

While the IPND draft describes an IP based neighbour discovery protocol, the authors assert the principles and basic mechanisms used may also be expressed in terms of other datagram protocols. However the principles and basic mechanisms used are described not only in terms of IP datagrams but also in terms of CLAs that are tightly coupled with IP, for example the TCP or UDP CLA. Beaconing methods are expressed in terms of IP unicast, braodcast and RFC 1112 multicast. That is an IPND node must support RFC 1112 (Deering, 1989) multicast IP destination addresses and multicast RFC 3376 (Cain et al., 2002) IGMP group membership and the service definitions for CLA-TCP-v4 and CLA-UDP-v4. In contrast, the work in this thesis must use beacon messages that are not UDP messages in the IP underlay The main difference between IPND and traditional IP discovery architectures is that complex protocol negotiation is bounded in round-trip transmissions: a publisher defines a UDP beacon message, that optionally includes a block of service availability information and transmits it to an IP unicast, multicast, or broadcast destination. The proposed approach, however, seems inappropriate for discovery in challenged context, as cooperating nodes first have to establish communications using UDP, which is unreliable and was not designed to be tolerant of disruption, delay and constrained networks in order to establish communications using DTN protocols. Furthermore, the characteristics of the environment might mean that in some contexts DTN nodes using IP addressing cannot communicate with DTN nodes that are not, which can lead to missed opportunity. Finally, the scalability of the mechanism has not been studied, and no guidance is provided for the design of other datagram protocols.

2.7 Comparison

Previous sections have reviewed related work on the discovery of DTN endpoint elements. This section outlines the requirements that a solution to this challenge must meet. It then compares the mechanisms previously presented using these criteria, and analyses their approach to discovery. Finally, we recall the concepts reviewed in the previous sections that have particularly influenced our work.

2.7.1 Requirements

In this section, we outline the requirements that a mechanism needs to fulfil to ensure discovery of DTN endpoint elements. Three types of requirements are distinguished: the problem characteristics, the information means supported, i.e., how elements can obtain information about themselves, and the solution characteristics.

Problem characteristics

A network element can either be an abstraction, i.e., a service, mechanism, state or session for example, or be physical, i.e., a device, CLA, data representation for example. Elements can be associated with, or uniquely identified by, a name. A solution needs to define the structural and functional components that compose a naming scheme. Additionally, it may be less meaningful to name real-time elements which are not cachable than elements which are cachable. Work on the Future Internet, typically offers fairly elaborate means for the binding of ADUs, or services with names and typically does not decouple named objects, attributes and values from underlying communications mechanisms.

Elements can relate to attributes with values, a bundle has a lifetime and creation timestamp for example. In addition, elements, attributes and values can relate to other elements, attributes and values, an ADU on a storage device for example. The physical elements are not sufficient and higher layer element discovery is needed to correlate common elements. Knowledge of the relationships of elements facilitates informed decision making. Existing mechanisms that provide a structure of information are either specific to an application domain, a frame capable of representing a specific element only for example, or comprise frameworks, that are not appropriate for our requirements, such as frameworks for the Web that are capable of representing any element in the universe. Thus, a solution is required to provide a data model, appropriate for structuring element, attribute and value relationships.

Representations of information that are tightly coupled with underlying communications mechanisms, are not appropriate for heterogeneous networks. Protocols cannot safely assume the network provides a directory like structure of information that maps names to addresses of devices hosting services. Universal deployment of a new DHCP option takes a considerable amount of time and often, networking equipment needs to be updated in order to support the new option. Furthermore DHCP and DNS are coupled with their transport mechanisms and due to this constraint, information they provide is not available in the RHNC. Objects in the MIB have highly complicated encoding rules and large data handles, that when transferred, consume substantial parts of each SNMP message. MIB is part of an architecture designed to manage TCP/IP-based Internets. SNMP is not a particularly efficient protocol. Use of UDP may not be possible in some contexts. The benefits of using XML as a generic interchange format, are questionable. RDF is designed as language for representing information about resources in the World Wide Web, in which low latency contemporaneous end-to-end connectivity can be assumed. Additionally the formal semantics along with a rigorously defined notion of entailment that provide a basis for well founded deductions in RDF data are to a lesser extent appropriate to our needs. The structured information is for generic interchange with local and remote sets of consumers and producers. Therefore a solution is required to provide a generic portable representation of the structured data, that decouples the representation from the processing program, and caters for many to many relationships. A mechanism to provide persistence for disruption tolerance of this structured sparse information, and additionally to store temporally evolving information needs to be provided.

Contexts can be static, in which context attributes do not change temporally, or dynamic, in which context attributes might be subject to change temporally, objects may collide, or occlude one another for example. Elements, attributes and values are subject to change in static and dynamic contexts. Furthermore, a static context might have constraints and dynamic context might evolve radically, extreme and performance-challenged environments where continuous end-to-end connectivity cannot be assumed for example. This thesis caters for the discovery of physical and abstract network elements operating in a dynamic context. Interactions can be deterministic, with elements of known type and capabilities interacting, in space flight operations for example, or conversely, stochastic. Networking designs that exclude network elements provide communications capabilities in a limited range of contexts.

A significant proportion of existing standards, protocols and models assume the characteristics of communication provided by *IEEE 802* networks to provide continuous end-to-end connectivity and negligible latency. A significant proportion specify discovery of representations of information that are tightly coupled with underlying communications mechanisms. Means to exchange structured information between any two network endpoints, are often context specific and wont work well in contexts they were not intended. For example, a close to real time constraint or a CLA constrained solution is not appropriate to our requirements. Discovery protocols that are unable to discover services if the producer moves to another network are inappropriate.

Discovery mechanism can use push or pull methods via symmetric or asymmetric protocols. Symmetric protocols and the pull mode are not appropriate in a wide range of contexts. Additionally IP broadcast, unicast, and multicast mechanisms to efficiently distribute information are specific to IP contexts. Finally, a solution might be specific to a type of element, or to a specific discovery problem, or generic, i.e., applicable to numerous discovery problems. Our work aims to be generic, which requires that a solution assumes as little as possible about the network elements or the discovery problem.

Information means

Network elements can obtain information about themselves by push and pull mechanisms. Push and pull mechanisms are used to exploit functions, features, protocols at various *IEEE 802* layers. Such mechanisms can be generally applicable across all *IEEE 802* networks or applicable to a specific *IEEE 802* network. Additionally, generic and specific mechanisms can be layer specific, or provide for certain aspects of a layer. Information may be obtained via local or remote sources. As described in Chapter 1, and explained further in Chapter 3, sources that are not local, are inherently unreliable. Solutions for sources that are not local need to support the push method via heterogeneous, partitioned and constrained wireless communication. The unreliability of communication links can be alleviated by the use of DTN protocols.

Generic IEEE mechanisms can provide information on services of the LLC sublayer, the physical topology from adjacent stations via LLDP / set of MIBs, and information that influences selection during handovers via media independent information service and IEs. IEEE mechanisms such as WLAN radio measurement, WNM protocols, the mesh facilities and TDLS are specific to *IEEE 802.11*. A pull mechanism, with an asymmetrical inquiry procedure is specific to *IEEE 802.15*, and a pull mechanism for neighbourhood discovery/measurement is specific to *IEEE 802.16*. Link layer mechanisms of the IETF, IRTF and elsewhere, for the most part, exploit the IEEE mechanisms. The generally applicable IETF mechanisms for Internet are the NETCONF and ISMF frameworks and their methods. NETCONF is oriented around manipulating the configuration of network devices, and ISMF oriented around managing TCP/IP-based Internets. Specific IETF mechanisms provide discovery in a wide range of application domains. A significant proportion of the work in this area from other sources than IEEE, IETF and IRTF deals with vendor specific aspects of discovery. A solution should be capable of exploiting the information provided by generic and specific mechanisms. A solution for local sources might support exploitation of the methods of existing mechanisms to obtain information. To provide delay tolerant discovery of endpoint elements, a solution needs to support all of these information means, and cater for their unreliability.

Solution characteristics

Physical and abstract network elements provide interoperable communications, with, and among, extreme and performance challenged environments, where continuous end-to-end connectivity cannot be assumed. Consequently, solutions should support a push mode and asymmetric protocol that can use broadcast, unicast, and multicast like mechanisms. A solution should provide a sender-initiated communication model that can be used to exploit an opportunistic contact in the presence of disruption and/or constrained network contexts. A solution should not restrict the flexibility of the network, and facilitates decoupling of information from hosts. Information that is not tightly coupled with its producer facilitates interchange between different applications, consumers or producers as such information is not bound to underlying communications mechanisms. Appropriate discovery frameworks are designed to exchange structured information, that is independent of producer or consumer, between any two heterogeneous network endpoints, and over any convergence layer.

Solutions can associate physical and abstract elements. An essential characteristic of a solution to the problem tackled in this thesis is the degree of abstraction supported. Exclusion of network elements implies constrained range of context information. A solution should be capable of working with any thing as a first-class abstraction. Identifiers can have properties and a solution needs to provide an appropriate scheme to structure use of identifier properties: the namespace, namespace partitioning, syntactic structure and non-syntactic structure rules of identifiers. Associations can have properties. Therefore, a solution needs to provide an appropriate scheme to structure use of association properties: the semantics, uniformity, scope, persistence, extensible, ease of transcription, and uniqueness of associations. This thesis calls for association properties which describe extensible, non persistent, unique, difficult to transcribe associations that do not structure a semantic free namespace and consequently decouple location from name. URIs, are extensible, have a global scope, facilitate parsing of identifier components without knowing the scheme-specific requirements and enable uniform identification of resources via a scheme part.

A solution should provide an appropriate structure of information to represent the data, or fragments of a data, and how the data relates to its environment. Furthermore, the data model is appropriate for structuring element, attribute and value relationships. To support communications in the RHNC a solution must support networks which are extremely heterogeneous. Consequently, a solution should cater for *IEEE 802* networks that use TCP, UDP or IP and for the exchange of structured information and *IEEE 802* networks that do not use TCP, UDP or IP. The structure should be capable of representing many to many relationships. A generic interchange format is required that can decouple the structured data from the processing program by providing a simple portable representation of the structured data that does not require complex parsing rules. The lean set of formatting rules of JSON fulfil our requirement for generic interchange format completely. An abstract syntax that reflects a simple graph-based data model approach, to provide a basis for well founded deductions, is appropriate to our requirements. Additionally, a storage model that can provide persistence for disruption tolerance of this structured sparse information, and store temporally evolving information is required. A solution should provide an extensible, flexible storage model of data representation for space-efficient storage of this sparse data. This model should map naturally to the appropriate interchange format. Operations are required to register (or deregister) intent to receive discovery ADUs, and to modify and update storage and / or cache. A solution should provide a set of such operations.

Solutions support the discovery of physical and abstract network elements operating in a dynamic context, as well as handle a dynamic context. A solution can support interaction between elements, without prior knowledge of their quantity, types and capabilities. A solution needs to provide appropriate functional components composed of mechanisms for resolution, assignment, encoding, authenticity, ownership, validation, and scalability of associations. A consumer should be able to ascertain the validity of interchange information. A solution for authenticity, validation and ownership should ensure validity of interchange information. A solution calls for flat association identifiers that can avail of network resolution, and that do not restrict the flexibility of the network. Such identifiers should facilitate decoupling of information from administrative domains. This motivates our choice of flat names,along with AV trees to facilitate multiple-attribute searching. We require scalability, at least on the order of trillions. A solution requires association mechanisms that are designed for very high generation, via some hashing algorithm, and assignment rates of identifiers, unique across both UUID space and time, that do not require centralised administration of each single identifier. Finally, a solution should assume as little as possible about the network elements, or the discovery problem. Our solution should offer explicit support for the discovery of network elements, i.e., offer laws that network elements can use to describe themselves for the purpose of discovery. The final requirement on a solution is the support it offers for application development. While some solutions offer only formal semantics, rigorously defined notions, or a model, others focus on providing a protocol to support discovery. Ideally, the whole process from specifying an overview of the requirements to implementation should be supported.

Summary

To summarise, Table 2.25 lists the problem characteristics, Table 2.26 lists the information means and Table 2.27 lists the solution characteristics that comprise the requirements on a system to ensure the discovery of mobile DTN endpoint elements.

Criteria	Values
Elements:	**abstraction / physical**, **named** / not named, **temporally evolving** / not temporally evolving, **spatially evolving** / not spatially evolving, **have attributes** / no attributes, **have values**, no values, **have relationships** / no relationships
Elements relationships:	**correlated** / not correlated, **represented** / not represented
Context:	static constrained / static not constrained, **dynamic with radical evolution** / dynamic with moderate evolution
Dynamic context:	continuous end-to-end connectivity / **no continuous end-to-end connectivity**, negligible latency / **no negligible latency**
Interactions:	deterministic / **stochastic**
Name scheme characteristics:	complex / **simple**, coupled location and ID / **not coupled location and ID**, context specific / **context generic**, exclusive / **not exclusive**
Data model characteristics:	coupled / **not coupled**, assumption / **no assumption**, complex encoding / **not complex encoding**, **formal semantics** / not formal semantics, rigorously defined notions, **no rigorously defined notions**, **sparse** / not sparse, context specific / **context generic**, **efficient** / inefficient, **provides a basis for well founded deductions** / does not
Interchange format characteristics:	**generic interchange** / not generic interchange, **local sets of consumers and producers** / no local sets of consumers and producers, **remote sets of consumers and producers** / no remote sets of consumers and producers
Storage model characteristics:	**disruption** / no disruption, **temporally evolving information** / static information, **hierarchical** / flat
Network assumptions:	Assume mechanisms are available / **do not assume mechanisms are available**

Table 2.25: Problem characteristics. Criteria and values are presented with required values in bold

Criteria	Values
Method:	push local reliable / push local unreliable, push remote reliable / **push remote unreliable**, **pull local reliable** / pull local unreliable, pull remote reliable / pull remote unreliable
Local source:	**reliable** / unreliable and alleviated with DTN, **exploit existing mechanisms** / do not exploit existing mechanisms
Remote source:	reliable / **unreliable and alleviated with DTN**, exploit existing mechanisms / **do not exploit existing mechanisms**

Table 2.26: Information means. Criteria and values are presented with required values in bold

2.7.2 Mechanism comparison

In this section, we review the mechanisms identified in the related work using the criteria defined above.

Problem characteristics

A number of the projects reviewed, in particular in the IEEE, IETF and IRTF communities, cater mostly for discovery mechanisms focused on near real-time, low overhead, discovery of network elements, i.e., the provision of appropriate standards, protocols and models that assume the characteristics of communication provided by *IEEE 802* networks to provide continuous end-to-end connectivity and negligible latency. This work cannot be applied to the discovery of network elements, as interaction with and among extreme and performance-challenged environments where continuous end-to-end connectivity cannot be assumed implies extra requirements to provide interoperable communications. Other work in these communities ((Demmer, 2008), (J. Wyllie, 2007), (D. Ellard, 2012)), (McMahon and Fall, 2010), (Bong and Yeo, 2011)) and work in military, vehicular and disaster response communities, however, deals with delay-

Criteria	**Values**
Exchange:	**structured information** / unstructured information, **generic interchange format** / non generic format, **heterogeneous network endpoints** / homogeneous network endpoints, **CLA agnostic** / CLA specific
Abstraction scope:	partial / **any**, **global** / not global, **element, attribute and value as a first-class abstraction** / no element, attribute and value as a first-class abstraction
Discovery communication method:	push symmetric / **push asymmetric**, pull symmetric / pull asymmetric
Discovery push method:	**beacon** / no beacon, **broadcast** / no broadcast, **unicast** / no unicast, **multicast** / no multicast, coupled with underlying communications mechanisms / **not coupled with underlying communications mechanisms**
Discovery operations:	**register (or deregister)**, **modify and update storage**, **modify and update cache**
Structural components:	**comprised of association and identifier properties** / comprised of approaches that describe the relationship between the hosts and the named objects, classified as centralised, distributed and hybrid
Functional components:	**comprised of association mechanisms** / comprised of mechanisms that relate to communication, database management and name management
Identifier properties:	hierarchical / **flat**, restrict the flexibility of the network / **do not restrict the flexibility of the network**, **facilitate decouple of location from name** /does not facilitate decoupling of location from name
Association properties:	**non structured and semantic free namespace** / namespace has structure and semantics, **enable uniform identification** /does not enable uniform identification, **extensible**, not extensible, **scalable > trillions** / not scalable, persistent / **non persistent**, **difficult to transcribe** / easy to transcribe, **globally unique**/ unique / not unique, facilitate parsing of identifier components without knowing specific requirements / does not facilitate parsing of identifier components without knowing specific requirements
Data model:	application domain specific elements / any element in the universe formally, **element, attribute and value relationships**, couples data with producer or consumer / **does not couple data with producer or consumer**, **facilitates generic interchange** / does not facilitate generic interchange
Interchange format:	complex encoding / **lean and simple encoding**, context specific / **context generic**, **appropriate for data model** / not appropriate for data model, **efficient** / inefficient, coupled with processing program / **not coupled with processing program**, **supports many to many** / no support many to many
Storage model:	**persistent** / non persistent, **structured relationships** / unstructured relationships, **sparse** / not sparse, **temporally evolving** / not temporally evolving, **appropriate for data model** / not appropriate for data model, **facilitate multiple-attribute searching with AV trees** / does not facilitate multiple-attribute searching with AV trees
authenticity, validation and ownership:	**verification of the signature of interchange information** / host security
Scalability:	**> trillions**, < trillions
Resolution:	**avail of network resolution** / no network resolution
Assignment:	**very high generation and assignment rates of identifiers** / low generation and assignment rates of identifiers, require an RA / **do not require an RA**
Encoding:	**encoding**/ no encoding
Assumptions:	**No prior knowledge**/ prior knowledge
Support:	**model** / no model, **protocol** / no protocol, **application** / no application, **model generated code** / no model generated code

Table 2.27: Solution characteristics. Criteria and values are presented (required values are in bold)

tolerant discovery of endpoint elements.

Some of these working groups, in particular the earlier work in the IEEE and IETF, solved the discovery problem only in a static context with few constraints. This assumption is restrictive and inappropriate for the application domain of our work in a dynamic context with radical evolution. Most of the more recent work, however, deals with a dynamic context. Some work, in particular in the routing community (e.g., (Liu and Wu, 2009), (Lindgren et al., 2003)), only caters for discovery which assumes prior knowledge of the meeting between pairs. However, this constrains the application and prevents evolution and addition of new network elements. Other work caters only for the discovery between elements, with prior knowledge of their quantity, types and capabilities. Once more this determinism is restrictive. A solution for the problem tackled in this thesis facilitates the addition of new elements, and the discovery of an unspecified quantity of elements and attributes through supporting stochastic interaction between elements, without prior knowledge of their quantity, attributes and capabilities.

Earlier work in the IETF, solved the naming problem by classifying the naming facility structural components as approaches that describe the relationship between the hosts and the named object, and functional components as those that relate to communication, database management and name management. The tight coupling and constraints of such a naming facility are inappropriate for the application domain of our work. A significant body of work is inspired by work on the Future Internet, and typically offers complex means for the binding of ADUs, or services with names that typically do not decouple named elements from underlying communications mechanisms. This assumption is restrictive and not directly applicable to our work. Amongst the other approaches, the ICN approaches to naming are particularly relevant, as they name data and take into account metadata associated with named data objects. The application development support and semantics offered for such metadata, however, are limited and consequently also inappropriate for the application domain of our work.

Several contributions reviewed, solve the generic interchange problem through specification of an XML schema. These are inappropriate for decoupling the representation from the processing program or catering for many to many relationships. Significant contributions from the W3C communities provide an abstract syntax that reflects a simple graph-based data model, along with formal semantics with a rigorously defined notion of entailment are not appropriate to our application domain.

Finally, work in the IETF communities, often caters for specific discovery applications designed to exchange information of a specific aspect of a producer or consumer, between homogeneous network endpoints, and over a specific convergence layer: The objectives for each router of the 1-hop and symmetric 2-hop MANET neighbourhood discovery protocol are specified, the procedure to discover the ID of a remote SNMP engine is described, an overview of the MSDP process is provided and the process for FCIP Service Discovery is described for example. Other work in these communities, and work in the DTNRG, however, offers generic solutions.

Information means

The approach of using of existing Internet protocols where possible is a design goal of this thesis. However, work must not only cater for *IEEE 802* networks that use TCP, UDP or IP but also for the exchange of structured information in *IEEE 802* networks that do not use TCP, UDP or IP. A number of the projects reviewed rely only on DHCP and DNS based solutions that specify representations of information that are tightly coupled with underlying communications mechanisms. This assumption is inappropriate for heterogeneous networks, and additionally such mechanisms do not offer an appropriate means to express how a data representation, or fragments of a data representation, relate to the environment.

A number of the projects reviewed rely only on SNMP and the MIB, which are part of an architecture designed to structure network management information in order to manage TCP/IP-based Internets, not networks which are radically heterogeneous. Several describe XML based solutions that inherently do not cater for for many to many relationships and introduce a tight coupling between the representation and the processing program and parsing implementations that are complex in nature.

As explained in Chapter 1, an abstract syntax, that reflects a simple graph-based data model, is appropriate for our structure of information, and due to the resource constraints and extreme heterogeneity of the networking contexts considered, a generic interchange format, with a lean set of formatting rules used to record a mapping from keys to values, is appropriate for our structured discovery information. Amongst the reviewed work, these characteristics are only taken into account by two bodies. JSON and RDF are significant contributions from the IETF and W3C. The lean set of formatting rules of JSON provide a generic format that is suitable for the portable representation of structured data we require. JSON is therefore adopted by TDP for this purpose. While the graph-based data model approach of RDF is appropriate to our requirements the formal semantics with a rigorously defined notion of entailment providing a basis for well founded deductions in RDF data are to a lesser extent appropriate to our needs which we shall see in Section 3.

Solution characteristics

The solutions that we have reviewed have published specifications. Most of these, use the consumer initiated pull method with a symmetric protocol. The pull method and symmetric protocols are not always appropriate for the discovery of DTN endpoint elements that cannot easily communicate. The exceptions that adopt a push method are discovery without DTN solutions and DTN discovery in DTNRG.

Most of the discovery work cited offers physical element discovery and consequently limited ability to relate information, as no protocol is defined to enable the higher layer abstract element discovery needed to correlate common elements, with the exception of Future Internet supporting protocol based discovery, and various application-specific solutions. Note however, that most solutions assume prior knowledge or that underlying mechanisms will provide elements with resolution and consistent information, but these protocols do not define how elements should use the information to discover themselves.

Some mechanisms do not take high-latency connectivity that is not orientated around hosts and addresses into account at all, for example, most work in the IEEE, IETF, W3C and other communities, as well as most routing and naming schemes. Only five bodies of work cater for the provision of DTN endpoint element discovery requirements ((Demmer, 2008), (J. Wyllie, 2007), (D. Ellard, 2012)), (McMahon and Fall, 2010), (Bong and Yeo, 2011)).

Summary

In this section, we classified the mechanisms reviewed according to the criteria listed in the previous section. A summary of this classification is shown in Table 2.28, in which the **Mechanism** column indicates the mechanism, and the **Specification** columns indicates whether the mechanism is specified, and if so, the type of specification. The **CLA** column indicates whether the mechanism is CLA bound, the **Exchange** column indicates the exchange format, the **Data** column indicates whether the mechanisms data is generic or application specific and the **Persist** column indicates whether the data is persisted to storage by the mechanism.

Mechanism	Specification	CLA	Exchange	Data	Persist
Endpoint Discovery Protocol (EDP)(McMahon and Fall, 2010)	ID	agnostic	Bundle	application specific	No
Automated Bundle Agent Discovery (ABAD) (J. Wyllie, 2007)	ID	agnostic	Bundle	application specific	No
Internet Protocol Neighbor Discovery (IPND)(D. Ellard, 2012)	ID	specific	Datagram	application specific	No
Base discovery of DTN RI (DTN2)(Demmer, 2008)	No	specific	Datagram	application specific	No
New Serial Convergence Layer (DTN-NSCL)(Bong and Yeo, 2011)	No	specific	Datagram	application specific	No

Table 2.28: The most significant discovery mechanisms in our application domain

These systems are ranked in accordance with their applicability and ability to exploit existing mechanisms. While this does not capture all described criteria, this subset is sufficient to show that none of the existing work addresses the discovery of DTN endpoint elements as specified in Chapter 1. A mechanism appropriate to the problem needs to have generic applicability as a single bundle-layer discovery protocol that maps to underlying ones, in which the structure of information, carried in the payload of beacons composed of RFC 5050 bundles, is sufficient to correlate common elements. Table 2.28 shows that none of the mechanisms are appropriate.

2.7.3 Analysis

As outlined above, the most significant work reviewed adopts a DTN approach, characterised by an application specific data model, and beacons that are transferred by a push method via asymmetric protocols. This section first assesses how appropriate this approach is for solving contexts in which communications may be over asymmetric links, be subject to long propagation delays, disruption, bandwidth asymmetry and/or packet loss, such as those catered for by this thesis.

DTN approaches

The DTN2 RI discovery mechanism does not have a published specification that we know of, is not independent of, and is applicable only to a subset of DTN2 CLAs. DTN-NSCL, is specific to the DTN-NSCL CLA and consequently not broadly applicable. While ABAD beacons are not UDP messages in the IP underlay, it is ultimately not generally applicable as a single bundle-layer discovery protocol that maps to underlying ones. Although the authors assert the principles and basic mechanisms of IPND may also be expressed in terms of other datagram protocols, the scalability of IPND has not been studied, and no guidance is provided for the design of other datagram protocols. While EDP nodes use bundles to report a description of their active EID registrations, CLA EIDs, storage, routing, security capabilities and whether they are willing to be custodian to proximate EDP nodes, the structure of information provided by EDP is application specific and not sufficient to correlate common elements.

Conclusion

The DTN approach, characterised by an application specific data model, and beacons that are transferred by a push method via asymmetric protocols does not have generic applicability as a single bundle-layer discovery protocol that maps to underlying ones, in which the structure of information, carried in the payload of beacons composed of RFC 5050 bundles, is sufficient to correlate common elements. In contrast we require an alternative approach to discovery, which allows knowledge of the relationships of physical, and abstraction elements to facilitate informed decision making. An approach might build on the EDP, but in addition be used by DTN nodes to report their association and association attribute identifiers, Furthermore an approach must provide structural and functional components along with a structure of information as part of a scheme to organise network element meta-data, and its representation which can build on the EAV model. We require an approach to provide a (delay-tolerant) network of associations in which the verification of the association between the association identifier and an associated is possible. These models, along with AV trees to facilitate multiple-attribute searching, offer the architectural constraints that, when applied as a whole, emphasise scalability of loosely coupled systems in order to exploit an opportunistic contact in the RHNC.

In the following section we summarise this chapter.

2.8 Summary

This chapter reviewed work related to discovery protocols, the focus of which is to equip a node with facts to exploit an opportunistic contact in order to transfer distinct ADUs in the RHNC. We noted that all mechanisms specify a structure of information, and further summarised the characteristics of reviewed discovery mechanisms as operations, structured information, mechanisms, persistence and an exchange format. We have reviewed work related to the ability to structure the relationships between network elements, the discovery of named network information elements and the support of heterogeneous networks as mobile nodes interacting with the physical environment must do so in the RHNC. A number of criteria have been defined to evaluate and classify the work reviewed. We have identified a requirement for naming, data and storage models. We have demonstrated that existing work is not appropriate to address the problems tackled in this thesis. Finally, we have shown how the design of TDP was influenced by some of the work presented, and introduced its main characteristics. The following chapter presents TDP. Italics are used as previously specified and also to indicate a class, method, module, timestamp, application state/event, or a set. We use graphs to represent the protocols, models, modules and classes of our design.

Chapter 3

Design

In this chapter, we describe the hypotheses on which TDP rests. These hypotheses are captured in models as graphed in Figure 3.1.

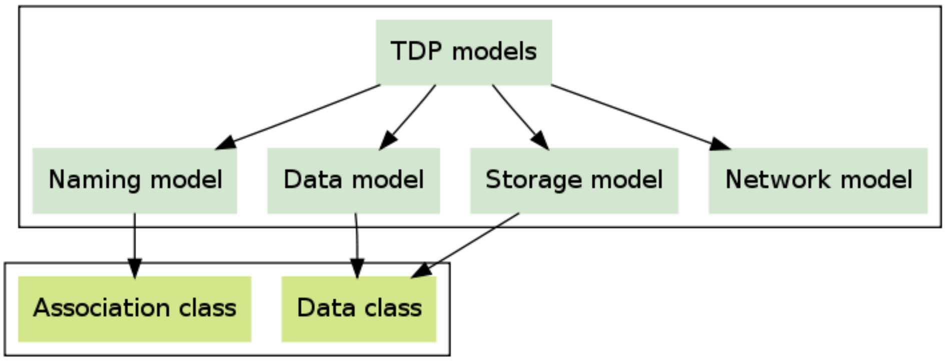


Figure 3.1: The naming, data, storage and network models with their classes

In Section 3.1 we provide a systems description that explains how TDP can be used and describe it's functions at the system level. Then we provide a description of our design methodology in Section 3.2. We describe a naming model in Section 3.3, a data model in Section 3.4, a storage model in Section 3.5, and a network model in Section 3.6.

Next, the system architecture is captured in the analysis and design of a set of modules as shown in Figure 3.2. We describe the module, *TDP-module* in Section 3.7, the module *TAV-maker* in Section 3.8, and the module which makes decisions based on TDM data, *TDP-decider* in Section 3.9.

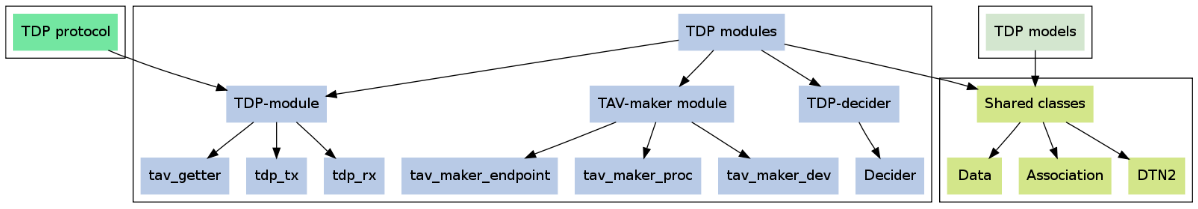


Figure 3.2: The modules *TDP module*, *TAV-maker* and *TDP-decider* with their classes

The design of the modules that build and use ns-3 as a real-time network emulator, which corresponds to the network model, and that can be interconnected with the *real* world is specified in Section 3.10 and shown in Figure 3.3).

In Section 3.11 we specify the design of the *Analysis* module used for analysis shown in Figure 3.4. This module analyses cases, where in the event of a contact, “real” world data transfer applications that use TDP are enabled to transfer more application data units than applications that use information provided by other sources. As part of this process the object classes of each module are identified and their methods specified. The final section of this chapter presents the fault model for our work.

In Section 3.4 we provide a framework for understanding associations via architectural abstractions, revealing how abstraction can be used to guide the architectural design of network-based application of a name and an associated thing, including a consistent set of terminology for describing the scheme. It then shows how the thing model can be used to design things for discovery mechanisms. In *Section*

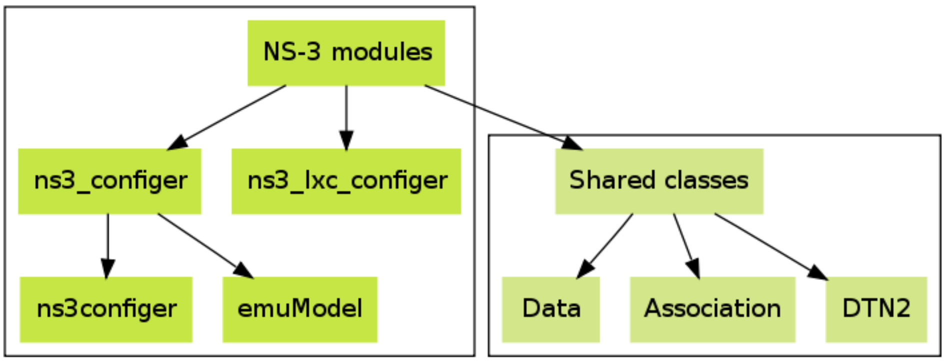


Figure 3.3: The NS-3 modules and their classes

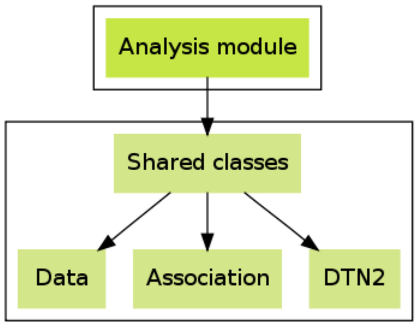


Figure 3.4: The *Analysis* module and related classes

3.7 we present an overview of how some things of a DTN network of associations can be discovered to exploit opportunistic contact networking scenarios that will be examined in detail and in the scope of this thesis. Then we introduce and elaborate on the Thing Discovery Protocol (TDP) for delay-tolerant networks. The relationship of the DTN model and therefore TDP models to classical fault models for distributed systems is explained in Section 3.12.

3.1 Systems description

TDP is specified to provide a consistent way to send and receive discovery data, i.e. DTN information that can be used to exploit an opportunistic contact. We assume that a TDP node, i.e. a TDP "speaker", has no knowledge about other TDP nodes.

TDP is used to exploit an opportunistic contact to transfer distinct application data in radically heterogeneous networking contexts. To accomplish this a TDP node extracts information, obtained by self-learning, into TDP specific data structures and saves this data to a store. In order to communicate with other TDP nodes, a TDP node must be able to obtain, transmit and receive TDP discovery data. A TDP node retrieves structured discovery data from storage. Once the ownership of the data is established, the data are encoded as a compact representation. This representation is transferred in the payload of RFC 5050 bundles. In order to work with bundles a TDP node communicates with a RFC 5050 BPA, i.e. a BP "speaker".

To present an overview of how TDP can be used, in the following we describe its functions at the system level. To aid our description we focus on an example. Our example describes how knowledge about the TDP nodes might be used to reduce network congestion and forward efficiently. We describe how network congestion might be reduced by correctly identifying and removing duplicate bundles that are queued for transfer. Then we describe how efficiencies in forwarding bundles might be attained through optimisations in the bundle transmission schedule.

3.1.1 Learning of unstructured data

A TDP node provides the means to translate information about DTN endpoint elements into structured discovery data. A TDP node translates discovery information into TDP specific data structures, composed of an unordered set of unique “key/value” pairs and saves these dictionary data structures to a EAV/CR schema based store.

A TDP node extracts specific information from an DTN endpoint which can be used to make a decision and translates this information into structured discovery data. In this example the information is metadata of the bundles themselves and BPA mechanisms for scheduling transmission of these bundles. A “virtual” class of this data takes the form of application responses of the BPA and a “physical” class that describes the actual bundle payload representation on a file system. The resulting data structure for both “virtual” and “physical” classes is the same. A TDP node extracts some of those data that can be used to formulate a decision.

3.1.2 Knowledge transfer

A TDP EID registration is used by a TDP node for receiving TDP messages. TDP messages are report bundles generated periodically or on demand and transmitted to the TDP EID. The TDP node creates a TDP specific DTN EID registration, generates a bundle with TDP data in the payload and transmits it to the TDP EID. A TDP node receives locally and remotely generated TDP messages, decodes the compact data representation and saves the discovery data to a store.

Prior to each transmission of a TDP bundle the TDP node obtains the list of available communication interfaces from the discovery data. Then the TDP node communicates with the local BPA in order to add an interface for each CLA, add a link for each interface, configure link discovery for each link and delete stale routes. If a route does not already exist for a link, a “copy” route is added to the TDP EID. For example, if a UDP CLA is discovered, the TDP node will obtain the associated IP broadcast addresses, create a UDP link for each and add a “copy” route to the TDP EID via the link, which it will use to transmit a TDP message.

3.1.3 Decisions

The TDP node makes decisions based on the discovery data. The TDP node defines the set of actions that are taken based on the learnt structured information. In the following we provide an overview of an approach to use the TDP node to reduce congestion through identification of duplicate bundles, and forward efficiently by scheduling the transmission and re-transmission of bundles.

The TDP node extracts discovery data from a store and translates them into variables used to take logical decisions. Variable values, such as the globally unique identifier of a bundle, are used along with other variables to evaluate whether certain conditions have been met, for example to ascertain whether a bundle has already been processed or a duplicate is queued for processing by the BPA. The actions that are to be taken by a TDP node are formulated as BPA commands, represented as objects suitable for exchange and placed in weighted queues pending execution by the BPA. In order to action decisions, the TDP node communicates with the BPA via a set of BPA APIs.

On initialisation the TDP node establishes it's unique DTN EID and ownership identifier (thing identifier) in order to ascertain those discovery data for which it has ownership. The TDP node extracts the discovery data thing identifiers for which it has ownership. Then the TDP node extracts the specific "things" of interest from the discovery data and once complete, flags them as processed. For our example we are interested in the metadata of the local BPA endpoint, BPA bundle and BPA bundle payload representation. We can use this information to ascertain queued bundle characteristics such as the bundle's unique identifier, source EID, destination EID, reply-to EID, creation timestamp and expiration timestamp. These bundle characteristics can be used to identify queued bundles which are duplicates prior to an opportunistic contact. We are also interested in information on the BPA bundle payload representation itself, such as its physical location, size and type, which can be used to generate a replacement bundle.

If the "thing" has not already been processed by the TDP node, and the bundle creation timestamp indicates that the bundle ID and payload was already properly processed by the BPA, a command is put on the queue to expire the duplicate. This method is used to reduce congestion through identification of duplicate bundles. Otherwise the bundle's creation timestamp is added to a processed list. In order to realise optimisation in scheduling the transmission of bundles, a command is placed on a queue which expires that bundle ID, whilst retaining the same payload, and generates a new bundle with specific transmit (and re-transmit) timers.

Based on the retrieved priority of the bundle, the TDP node places a command object on a transmit queue. While the queues are not empty the TDP node retrieves the objects, decodes them and passes the BPA commands to the local BPA for execution. In a following opportunistic contact, the BPA immediately attempts to transmit the optimised set of bundles. These distinct bundles are transmitted in their optimised queue sequence and failing that, based on their optimised re-transmission timers.

3.2 Design methodology

An object oriented design (OOD) approach (Booch, 1994) is taken that encompasses modularity, abstraction, and encapsulation that decomposes the system into modules. We define classes that describe the objects our applications use. Conceptual models are provided to represent the concepts and relationships between them. Guarantees on what each model shall provide to other models are given. Models are the basis of a module specification for a suite of modules: *TDP-model*, *TAV-getter* and *TDP-decider*, the ns-3 *ns3_configer* and *ns3_lxc_configer* modules along with their generated modules, and the *Analysis* module. The implementation of these modules is described in Chapter 4.

An overview of the design principles that guide the design process of TDP and its related models is presented, followed by an overview of the key design choices and the novel mechanisms that are at the basis of its implementation.

Three high-level design principles guide the design process and OOD:

1. Thing discovery should support the use of the Internet. It should be defined as a standard protocol so that independently developed applications of TDP can interoperate across the Internet.
2. Favour reference implementations, open protocols, standards and tools.
3. Do not build everything into the discovery mechanism, but assume that complementary mechanisms will arise to fill in the necessary functionality.

As part of the OOD analysis and design, for each model the rationale for the model is first detailed, then the model specifications are described. Next the guarantees on the capabilities of the model are provided. Assumptions of the model and implementation are then discussed. and finally a conclusion is provided.

3.3 Naming model

3.3.1 Rationale

As described in Section 2.3 of Chapter 2, the work in this thesis caters for the notion of working with any network element and attribute as a first-class abstraction of information. Handling abstractions necessitates the generation of names, that when associated provide appropriate identifiers. In addition, various mechanisms are needed in order to operate on the associations. Thus, we define a model of the functional and structural aspects of associations, as described in Chapter 1.

3.3.2 Specifications

In Chapter 1, we derived an *identifier* as "a sequence of characters associated with a set of zero or more elements.". The structure of information, generic interchange mechanisms, data storage models, identifier properties and the association properties comprise the structural components of the naming facility. In our approach the functional components of the naming facility are the association mechanisms for resolution, scalability, assignment, encoding, ascertaining authenticity and ownership, and validation.

As part of our review of the name scheme components in Chapter 2 we defined the identifier properties as described in Table 3.1, the association properties as described in Table 3.2, and the association mechanisms as described in Table 3.3. Table 2.27 listed the solution characteristics, which included those of the naming facility, that comprise the requirements on a system to ensure the discovery of mobile DTN endpoint elements.

The name scheme components, include the identifier properties, the association properties, and the association mechanisms. The identifier properties, described in Table 3.1, are the properties of identifiers. These properties include the logical grouping of a set of identifiers and the partitioning of such a namespace into identifier subsets. Other properties include the structural features of an identifier in a namespace, for example whether an identifier is flat or hierarchical. Finally they include identifier rules, such as rules that can be used to establish whether identifiers in a namespace are equivalent.

Property	**Description**
Namespace	The set of identifiers.
Namespace partitioning	The partitioning of a namespace into identifier subsets.
Syntactic structure	The structural features of an identifier in a namespace.
Non syntactic structure rules	Identifier rules e.g., rules for lexical equivalence

Table 3.1: Identifier Properties

The association properties described in Table 3.2, are the properties of associations.Association properties relate to the semantics, interpretation and scope of interpretation of associations. Whether an association is globally unique or universally unique and the lifetime of that association are also association properties. How an identifier can be transcribed from one medium to another and how an association scheme can be extended are association properties.

Our work must provide processes for the functional components of the naming facility to use structural components in order to name, and when required, authenticate associations. We require mechanisms to verify, encode, decode and compare associations. Our association mechanisms must be capable of

Property	Description
Semantics	Allows the semantic interpretation of the identifier syntactic conventions.
Uniformity	Uniformity allows uniform semantic interpretation of common syntactic conventions across different types of identifiers. It allows the identifiers to be reused in many different contexts
Scope	The scope of use has the same semantic interpretation in a namespace
Persistence	The lifetime of associations is an important factor to consider as a persistent association may be used as a reference to a thing beyond the lifetime of the thing. If an association expires then it is no longer possible to use it a as a reference to a thing.
Uniqueness	When an association is unique that association is used to identify a specific thing only.
Transcription	How an identifier can be transcribed from one medium to another.
Extensible	For an association scheme to be extensible it must allow extensions.

Table 3.2: Association Properties

naming autonomous mobile network elements. The association mechanisms described in Table 3.3 are the mechanisms that work with associations.

Mechanism	Description
Authenticity	Mechanisms are provided to ensure authenticity of the association, that it remains valid as long as the associated thing is available.
Validation	Mechanisms verify an association, sometimes called name-data integrity.
Encoders	Mechanisms to encode, decode and compare associations
Scalability	Mechanisms to search and exploit vast numbers of associations.
Resolution	Mechanisms to resolve an identifier to an associated thing.
Assignment	Mechanisms to determine the naming authority and the conditions under which an identifier is associated with a thing.
Ownership	Mechanisms to determine an owner of a generated association

Table 3.3: Association Mechanisms

3.3.3 Guarantees

In this model, guarantees on the ability to name endpoint elements are provided to thing producers, and guarantees on the ability to operate on named associations are provided to both thing producer and consumers. Table 3.4 provides the guarantees to TDM producers and consumers.

No.	Guarantee to	Guarantee
1	Producers	Enable generation of names that when associated provide appropriate identifers, so as to fulfil the the naming facility solution characteristics listed in Table 2.27.
2	Consumers	Enable operations on associations, so as to fulfil the the naming facility solution characteristics listed in Table 2.27.

Table 3.4: Guarantees

3.3.4 Assumptions

While this model assumes no prior knowledge, it assumes the availability of mechanisms for the verification of a signature of interchange information, and for attribute authentication, such as the GSS-API

Naming Extensions. Additionally this model assumes that UUIDs, in the order of trillions, unique across both UUID space and time, can be generated via an algorithm for hashing of a thing at very high generation and assignment rates. The model also assumes the ability to use appropriate resolution mechanisms if required.

3.3.5 Implementation

The work in this thesis adopts content-oriented security models as they allow validity of the content to be ascertained by the consumer.The association properties describe extensible, non persistent, unique, non transcribable associations that do not structure a semantic free namespace and consequently decouple location from name. The identifier properties cater for a syntactic structure, that is appropriate for fixed size universal identifiers, that facilitate partitioning of the namespace into identifier subsets, and support limited identifier rules. Flat association identifiers that can avail of network resolution, that do not restrict the flexibility of the network, and that facilitate decoupling of information from administrative domains are required. This motivates our choice of flat names which are more suitable for DHT based look-up services, along with AV trees to facilitate multiple-attribute searching. Scalability is at least on the order of trillions and possibly beyond to accommodate future growth. We implements the RFC 4122 UUID, which can be generated hashing of a thing. Association mechanisms generate UUIDs in the order of trillions, unique across both UUID space and time, via hashing of a thing, at very high generation and assignment rate, and which do not require an RA for administration of each single identifier.

URIs, are extensible, have a global scope, facilitate parsing of identifier components without knowing the scheme-specific requirements and enable uniform identification of resources via a scheme part. UUIDs can be used as URIs and RFC 4122 (Leach et al., 2005) describes a scheme for rendering a UUID. To use a UUID as a URI a scheme is supplied, such as “urn:uuid:” in addition to the URI content that is the UUID. The functional and structural components comprised of association mechanisms, identifier and association properties can be fulfilled through use of the UUID and URI, hashing, content-oriented security models and attribute authentication.

3.3.6 Conclusions

This model can be used by applications to:

- Support direct verification of the association.
- Generate identifiers with a certain set of properties and in a specific way.
- Use associations with a certain set of properties and in a specific way.
- Reason about the identity of network elements, attributes, values and their relationships
- Perform functions on associations

3.4 Data model

3.4.1 Rationale

The thing data model (TDM) exploits the rationale observed by Wittgenstein (1922). The translation of which is “the world is the totality of facts, not of things” (Sullivan, 2000). This thesis deals with protocols used to discover facts. Thus, a common, simple model for expressing information so that it can be exchanged without loss of meaning is required. TDM models facts about things.

3.4.2 Specifications

“The time has come”, the Walrus said,“to talk of many things, of shoes and ships and sealing wax, Of Cabbages and Kings”(Carroll, Carroll). As decribed in Chapter 1, the term thing is used instead of resource, entity, object or element so that the simple principle is not obscured by language. A named thing is an association represented as a set of zero or more elements that is identified by at least one unique identifier. TDM uses thing identifiers to determine the facts about a thing such as “what” or “who” a thing is associated with (see figure 1.1).

TDM and Association Lists

TDM is based on the “association lists” (URL, 1) of LISP which are used to record a mapping from keys to values. We adopt the EAV storage model, which was first used in the form of association lists (Winston, 1984) to base our definition of the TDM which we formulate as:

“a set of tuples $\{(T),(A),(V)\}$ where T and A are identifiable by at least one unique identifier, and T is the set of zero or more elements, A is the set of zero or more element attributes and V is the set of zero or more attribute values.”

The facts of a relationship between elements, attributes and values may be represented as an TDM triple. Any element, attribute or value can be associated with one or more names. The value associated with an attribute (see figure 1.2) is denoted a thing attribute value (TAV).

TDM Endpoints and Edges

A multidigraph can be used to reflect TDM. Each endpoint T represents a physical or abstract network element, parametrised as described in Table 3.5. Each Thing has three identifers. An “ID” that uniquely identifies it, a “Class ID” that identifies the sub class of Thing it is, and and “Owner ID” used to determine the owner of the Thing. A “Creation timestamp” records the time the thing was associated and “Last change timestamp” records the last time it was modified. A “Name” contains a name that is unique for an owner and “Description” provides the class of the thing.

Each endpoint A represents an attribute or characteristic of a network element, parametrised as described in Table 3.6. Each attribute has two identifers. An “ID” uniquely identifies the attribute, and a “Class ID” that identifies the sub class of the attribute. A “Type” provides the class of thing the attribute belongs to and “Class of attribute” provides the class of the attribute. A “Name” contains a name that is

No.	Description	Parameter
1	ID (a thing ID)	$thingID_{v,i}$
2	Name	$thingName_{v,i}$
3	Description	$thingDescription_{v,i}$
4	Creation timestamp	$ThingCreated_{v,i}$
5	Last change timestamp	$thingLastChanged_{v,i}$
6	Owner ID (a thing ID)	$thingOwner_{v,i}$
7	Class ID (a thing ID)	$classID_{v,i}$

Table 3.5: Each endpoint T represents a physical or abstraction network element

unique for a class of attribute. "An indication of wheteher it is required" is used to indicate an attribute is required or optional. "An indication of whether it is searchable" is used to provide or mask the attribute, and "An indication of whether there are more instances of it" is used to indicate multiple instances of the attribute. Finally a "sequence number" indicates the sequence of the attribute in a set of attributes.

No.	Description	Parameter
1	ID (a thing ID)	$attributeID_{v,i}$
2	Name	$attributeName_{v,i}$
3	Type	$dataType_{v,i}$
4	Class of attribute	$attributeClass_{v,i}$
5	An indication of wheteher it is required	$required_{v,i}$
6	An indication of whether it is searchable	$searchable_{v,i}$
7	The formula used to compute it	$computedFormula_{v,i}$
8	An indication of whether it is virtual attribute	$virtualAttribute_{v,i}$
9	A sequence number	$sequenceNumber_{v,i}$
10	An indication of whether there are more instances of it	$multiInstance_{v,i}$
11	Class ID (a thing ID)	$classID_{v,i}$

Table 3.6: Each endpoint A represents an attribute or characteristic of a network element

Each endpoint V represents an attribute value, parametrised as described in Table 3.7. Each value has two identifers. The "ID of a thing" is the identifier associated with the thing the value belongs to. The "ID of an attribute" is the identifier associated with the attribute the value belongs to. The "Value" can be anything.

No.	Description	Parameter
1	Value	$value_{v,i}$
2	ID of a thing	$thingID_{v,i}$
3	ID of an attribute	$attributeID_{v,i}$

Table 3.7: Each endpoint V represents an attribute value

Endpoints may be connected by multiple edges, representing different relationships. A relationship between endpoints is parametrised by its source i and destination d endpoint. The reason for this is there may be one or more different relationships. The resulting graph is a multidigraph in which an edge can connect an endpoint to itself such that a multidigraph G is an ordered five-tuple $G := (T, A, V, i, d)$.

For an endpoint v_n the functions listed in Table 3.8 are used to provide functional components described in Table 3.3.

For an endpoint v_n the functions listed in Table 3.9 are used to provide structural component described in Table 3.1.

No.	Function
1	$(authenticity(ID))$
2	$(validation(ID))$
3	$(encoders(ID))$
4	$(scalability(ID))$
5	$(resolution(ID))$
6	$(assignment(ID))$
7	$(provenance(ID))$

Table 3.8: Functions of the association functional components for an endpoint v_n

No.	Function
1	$(namespace(ID))$
2	$(structure(ID))$
3	$(rules(ID))$

Table 3.9: Identifer properties functions for an endpoint v_n

For an endpoint v_n the functions listed in Table 3.10 are used to provide structural component described in Table 3.2.

No.	Function
1	$(semantics(ID))$
2	$(uniformity(ID))$
3	$(scope(ID))$
4	$(persistence(ID))$
5	$(uniqueness(ID))$
6	$(transcription(ID))$
7	$(extensible(ID))$

Table 3.10: Association properties functions for an endpoint v_n

A name associated with an element or attribute is denoted a thing identifier. A thing identifier has properties, specified by a naming model, outlined in Chapter 2 and described in Section 3.3. Any thing, physical or abstract, can have a thing identifier. A node can be represented as a set of thing identifiers, that can be identified by at least one unique thing identifier. (see figure 3.11).

3.4.3 Guarantees

In this model, guarantees on the ability to generate things are provided to thing producers, and guarantees on the ability to extract and verify things are provided to consumers. Table 3.11 provides the guarantees to TDM producers and consumers.

3.4.4 Assumptions

This data model assumes the storage model, identifier properties and the association properties that comprise the structural components of the naming model are specified. It also assumes the association mechanisms for resolution, scalability, assignment, encoding, ascertaining authenticity and provenance, and validation that comprise the functional components are specified.

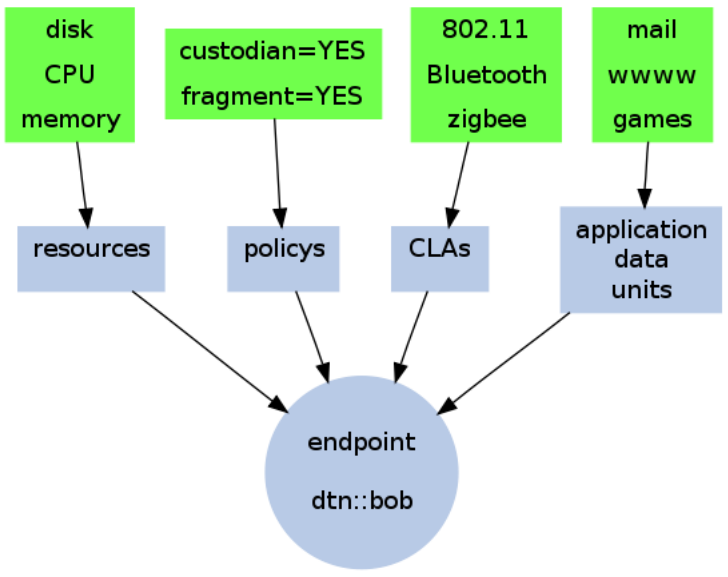


Figure 3.5: An endpoint containing a node identifiable by at least one unique identifier (*dtn* :: *bob*). A node is a set of zero or more identifiers

.

No.	Guarantee to	Guarantee
1	Producers	To generate associations with network elements, attributes, values and their relationships using an appropriate naming model
2	Producers	To be able to structure the its sets of elements, attributes and values, in a way that maps naturally to appropriate storage and interchange models
3	Producers	To be able to structure the relationships of its sets of elements, attributes and values, in a way that maps naturally to appropriate storage and interchange models
4	Producer	To support generation of a “self certifying” name using an appropriate naming model
5	Consumers	Enable the extraction the structured relationships, elements, attributes and values from interchange formatted discovery information, in a way that maps naturally to appropriate storage models
6	Consumers	Support verification of associations using an appropriate naming model

Table 3.11: Guarantees

3.4.5 Implementation

The TDM implements an associative array. TDM is based on an associative array abstract data type. The feasibility of the flexible EAV model to provide space-efficient storage of this sparse data has been demonstrated in real-world settings (URL, 68). We provide an implementation of the extensible EAV model, in the form of an on-demand, scheduled and event-based module called *TAV-maker*. *TAV-maker* generates TDM structured information, and persists the structured TDP information to the application's storage model layer,

Facts that indicate a relationship between a value, attribute and thing may be represented as an TDM triple. A representation of a fact might be as a row in a table in a relational database. The table has three columns, corresponding to a value, and the element and attribute identifiers of the TDM triple.

Thing Attribute Value (TAV)

TDM can be used to represent any physical or abstract network element as a first-class abstraction of information. A node can be denoted as a set of thing identifiers, identified by at least one unique thing identifier (see figure 3.6).

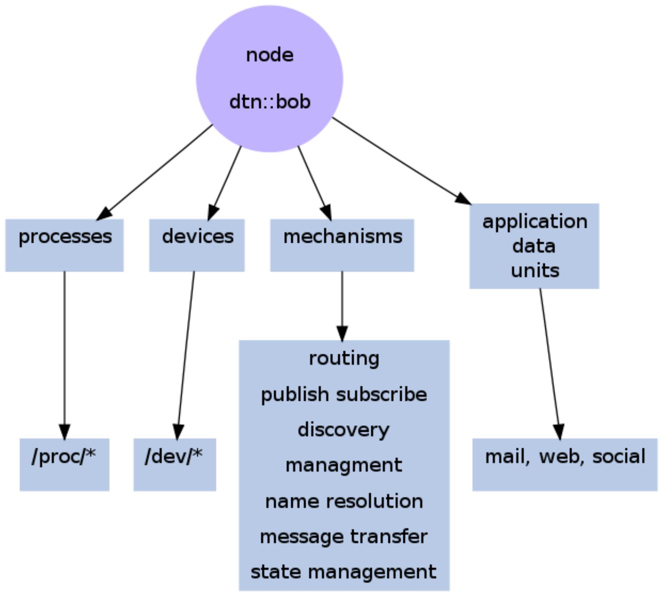


Figure 3.6: A node is a set of thing identifiers

A user, service, resource, process or mechanism of the network can be described as a node. A named thing attribute can be associated with one or more attribute identifiers, which can be associated with one or more thing identifiers.

DTN endpoint, a thing

A DTN endpoint, hereon denoted endpoint, can be represented as a bounded type of thing; a set of zero or more engines for sending and receiving RFC 5050 bundles. An endpoint can be represented as a type of thing with a set of zero or more thing identifiers, that is identified by at least one unique thing identifier denoted by an DTN endpoint identifier. A DTN endpoint identifier, hereon denoted an endpoint identifier, is a bounded type of thing identifier; a URI (Berners-Lee et al., 1998) conveyed in any bundle block used by DTN endpoint elements to identify themselves for BP purposes (see figure 3.7).

Any thing can have relationships with any thing. One or more things can be related to an endpoint. One or more endpoints can be related to a thing (see figure 3.8).

A thing can be signed, via the association functional components listed in table 3.8, in order to allow direct verification of the association between the thing and name; a “self certifying” identifier. An endpoint identifier can be signed in order to allow direct verification of the binding between the endpoint and an associated thing. An attribute identifier can be associated with one or more values. An endpoint thing

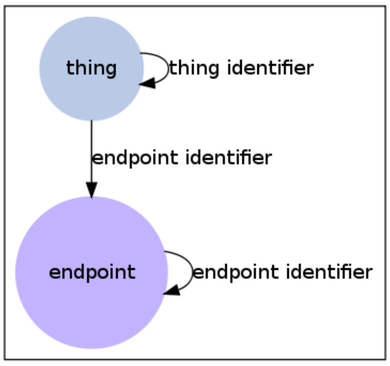


Figure 3.7: A DTN endpoint is a type of thing

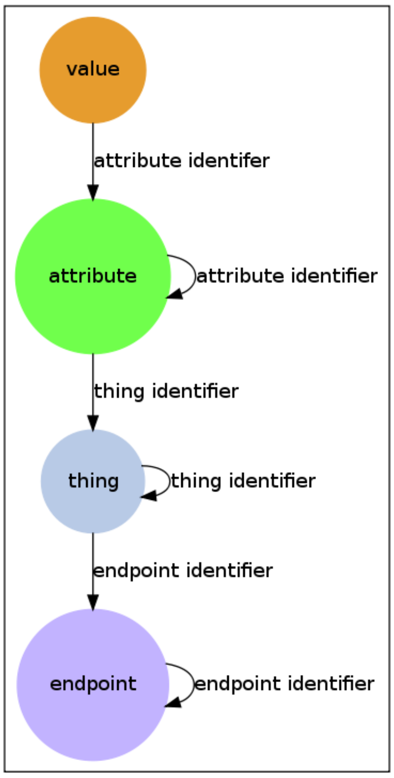


Figure 3.8: Value of an attribute of a thing associated with an endpoint

attribute value (ETAV) is the value of an attribute of a thing, associated with an endpoint (see figure 3.9)

The naming of things

A thing identifier can be assigned to any element or attribute which can, for example be signed using ElGamal keys. DTN thing identifiers are EIDs.

This model allows applications to reason about available network elements, attributes, values, and their relationships.

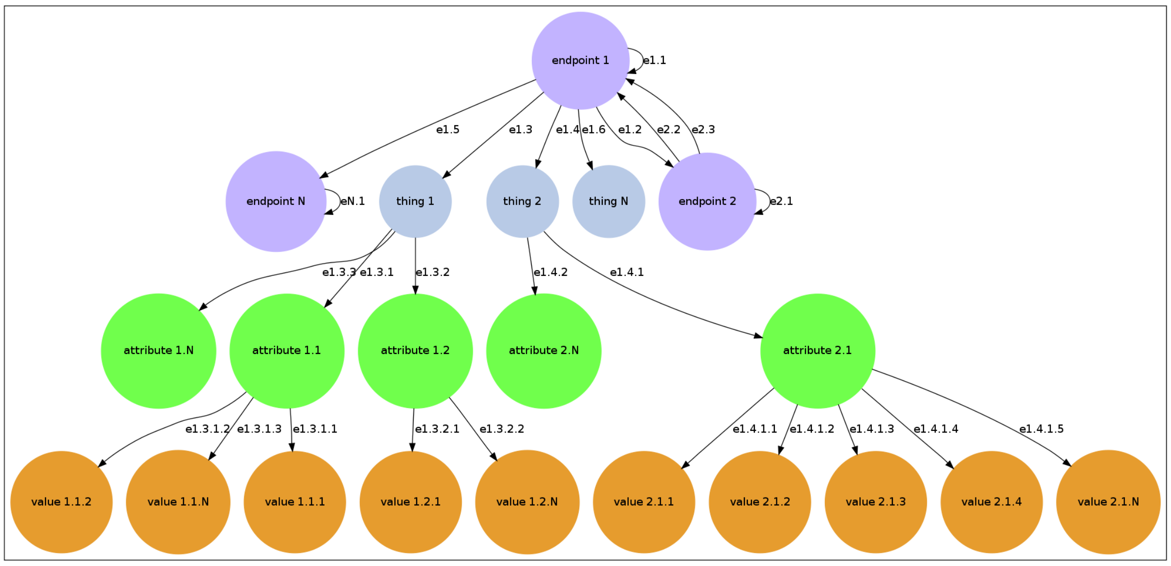


Figure 3.9: DTN endpoint (a type of thing), thing, attribute, and value

The thing data model can be leveraged by applications to:

- determine the set of things for which the node is responsible.
- determine the set of attributes of a thing.
- determine the set of values of a thing attribute.
- organise knowledge into main element classes
- determine relationships of network elements in the overlay topology when the overlay is structured.
- determine relationships of network elements in the underlay topology when the underlay is structured.
- support direct verification of associations using an appropriate naming model.

3.5 Storage and interchange models

3.5.1 Rationale

Our definition of a TDM is based on an associative array abstract data type. TDM implements an associative array hash table to provide a solution to this dictionary problem. The structuring of TDM data in association lists (URL, 1) is inherent in the TDM design of knowledge organisation. TDM information is sparse which means that it might involve empty information slots, i.e. involve fields that are null. In addition the sparse TDM information might be temporally evolving.

The TDM information is for generic interchange with local and remote sets of consumers and producers. Consequently, we need to provide a generic portable representation of the TDM data, that decouples the TDM associative array data representation from the processing program, and caters for many to many relationships. Additionally, a flexible storage model that can provide persistence for disruption tolerance of this structured sparse TDM information, and store temporally evolving information is required.

Thus, we require an appropriate generic interchange format that maps naturally to an extensible, flexible storage model of data representation, for space-efficient storage of this sparse data. The feasibility of the entity-attribute-value (EAV) storage model implementation in production systems, when data are sparse, and when numerous classes of data need to be represented have been demonstrated in the literature (Dinuab and Nadkarnia, 2007). The feasibility of JSON to provide a generic format appropriate for the portable representation of data has also been demonstrated (URL, 69).

3.5.2 Specifications

The EAV storage model was first used in the form of association lists (Winston, 1984). The EAV model can be used for efficient storage of sparse TDM data. A representation of a fact in a relational database might be as a row in a table in which the table has three columns, corresponding to a value, the element and the attribute identifiers of the TDM triple. For an EAV architecture, system metadata have no explicit semmantic structure, since the semantics of the system are in the data representations rather than the table structure.

EAV is an inappropriate model for relatively static or simple data. EAV is inappropriate for simultaneous bulk retrieval of several objects and is not well suited to complex attribute-centric queries, based on values of attributes.Nevertheless, strong typing can facilitate ensuring the correctness of attribute-centric queries. An enhancement to EAV representation called EAV with classes and relationships (EAV/CR), was designed to model heterogeneous data and has been demonstrated URL (70) with sufficiently rich metadata (Nadkarni et al., 1999). A formal object-oriented framework is overlayed, though definition of classes and the permissible attributes that each class may contain, on the basic EAV model by the EAV/CR schema. Physical schema describe how to persist data and the EAV/CR representation simulates a complex logical schema by using a simple physical schema.

The EAV/CR model (URL, 71) of data representation can be used to overlay a network of associations on top of the EAV physical structure. In robust production systems, Dinuab and Nadkarnia (2007) show that EAV-modeled databases trade a modest data sub-schema for a complex metadata sub-schema. An object dictionary (OD) approach is taken by the EAV/CR schema, in which information on all objects across all classes is stored in a single table and class-specific information is stored elsewhere. The OD approach is orthogonal to EAV/CR and information in these EAV/CR tables is linked to things through their unique identifiers.

The suitability of a EAV/CR model is related to the number of classes and the inter-class or intra-class relationships. As the application domain of this thesis involves many classes and the inter-class or intra-class relationships are evolving, a EAV/CR model is appropriate. While EAV/CR design is not intended to support temporal logic, it is suitable for integration of diverse data into a single database. The EAV/CR representation uses strong data typing, i.e., when an attribute is defined, its data type is defined as well.

In contrast to the EAV/CR schema of Nadkarni et al. (1999), in which the classes and attributes tables contain the logical schema description and information essential to a Web-based user interface, the schema is altered in several ways to “better fit” our application domain. The EAV/CR schema of the TDP storage model is depicted in Figure 3.10.

Interchange format

The lean set of formatting rules of JSON provides a generic format that is suitable for the portable representation of TDM data we require. JSON is therefore adopted by TDP for this purpose. The TDM maps naturally to EAV and interchange formats like JSON (Crockford, 2006).

3.5.3 Guarantees

In this model, guarantees on the ability to store TDM data are provided to thing producers, and guarantees on the ability to retrieve and alter TDM data are provided to consumers. Table 3.12 provides the guarantees of the storage model to TDM producers and consumers.

No.	Guarantee to	Guarantee
1	Producers	Enable a producer to store TDM data.
2	Consumers	Enable a consumer to retrieve and change stored TDM data

Table 3.12: Guarantees

Guarantees on the ability to represent the TDM data appropriate for interchange are provided to thing producers, and guarantees on the ability to retrieve and alter TDM data are provided to consumers. Table 3.13 provides the guarantees of the interchange model to TDM producers and consumers.

No.	Guarantee to	Guarantee
1	Producers	Enable provision of a generic portable representation of the TDM data, that decouples the TDM associative array data representation from the processing program, and caters for many to many relationships.
2	Consumers	Enable extraction of the structured relationships, elements, attributes and values from interchange formatted discovery information, in a way that maps naturally to TDM

Table 3.13: Guarantees

3.5.4 Assumptions

EAV design is potentially more challenging than conventional design due to the need to design the metadata effectively (Dinuab and Nadkarnia, 2007). TDM is designed to structure metadata effectively. The *TAV-maker* module presents the structured TDP information to the application's EAV storage model layer to persist it to the database.

3.5.5 Implementation

EAV requires complex, multi-table joins to populate an object with the relevant attributes. The expensive processing required to do so can be mitigated through careful caching and the selective use of denormalisation. A denormalised model can be used as a means of addressing performance or scalability.Triggers may be implemented to re-index and update the *flat* denormalised tables when thing data changes. The TDM maps naturally to JSON (Crockford, 2006).

3.5.6 Conclusions

Existing solutions to the storage and interchange model are appropriate. The contexts where EAV modeling is a useful alternative to conventional relational methods of database modeling that is provided by Dinuab and Nadkarnia (2007), map to our requirments. While the EAV model is extensible, changes in data type of a value can be difficult i.e., may require migration of all the records for an attribute to the corresponding table that matches the new datatype. The TDM TAV maps naturally to JSON which is selected as JSON fulfils our requirments of an interchange format.

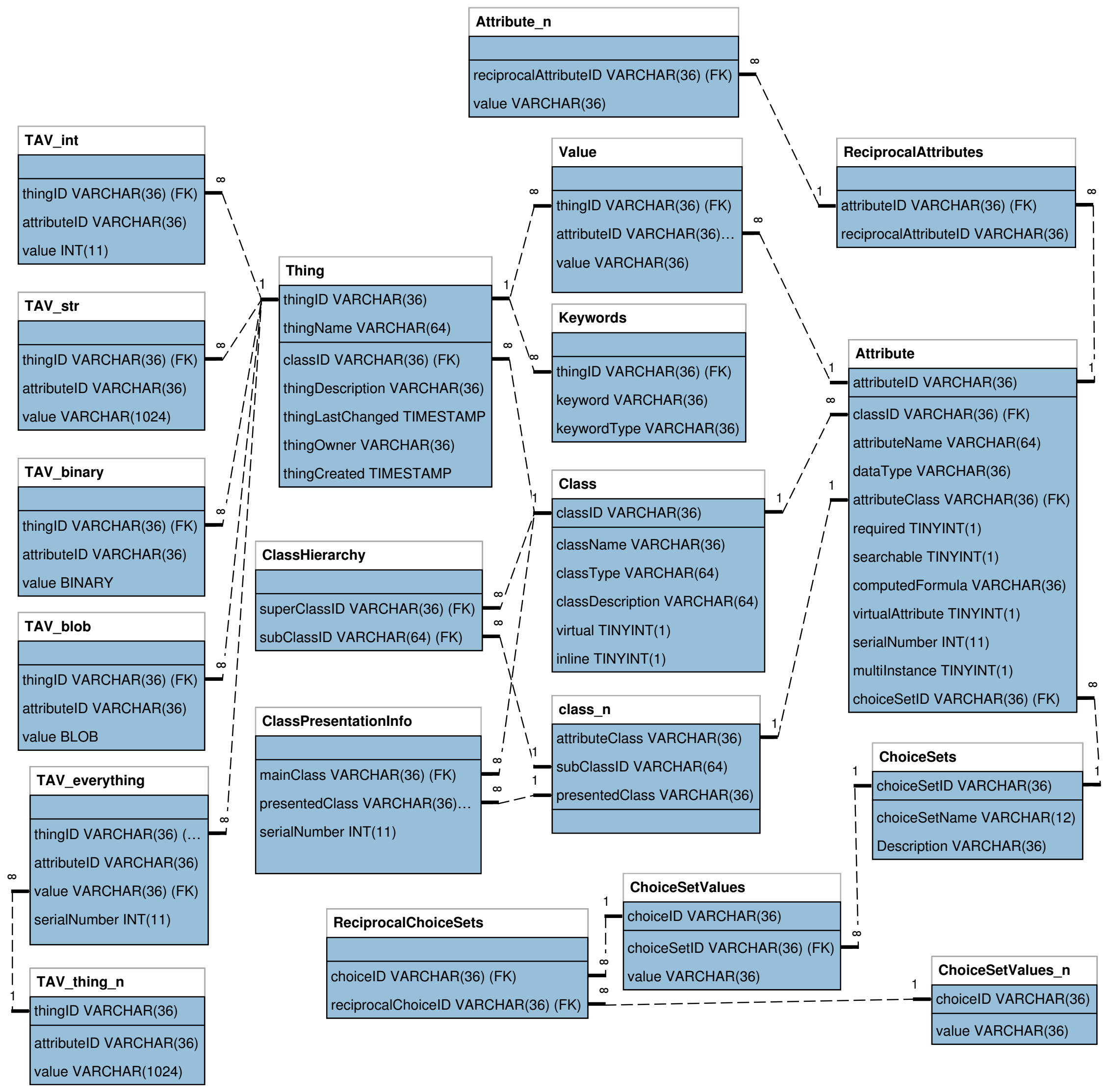


Figure 3.10: Storage model (EAV/CR), a table name with the suffix "_n" indicates use of the same table more than once. The schema is altered in several ways to "better fit" our application domain

3.6 Network model

3.6.1 Rationale

No prior knowledge is assumed in the radically heterogeneous DTN contexts we have described that are composed of any number of potential mobile DTN nodes. Additionally, potential nodes move in an arbitrary fashion.

The feasibility of graph based models that use abstractions to determine the optimal routing for minimising average delay in a DTN have been demonstrated in the literature (Jain et al., 2004). When a DTN is partitioned, the topological graph is also partitioned. A multigraph is used because it may be possible to select and combine many distinct physical CLAs to move data between the same pair of nodes. The set of edges in the graph should capture time-varying node resource capacity, edge capacity and propagation delay as well as cater for multiple edges. Node resource capacities, propagation delay and edge capacities are time dependent i.e, edge capacity is zero at times when the link is unavailable.

3.6.2 Specifications

Let any element of an endpoint be associated with, or uniquely identified by a DTN endpoint identifier, hereon denoted endpoint identifier. Let a DTN node, hereon denoted node be a set of zero or more endpoint identifiers that is identifiable by at least one unique endpoint identifier (see figure 3.11).

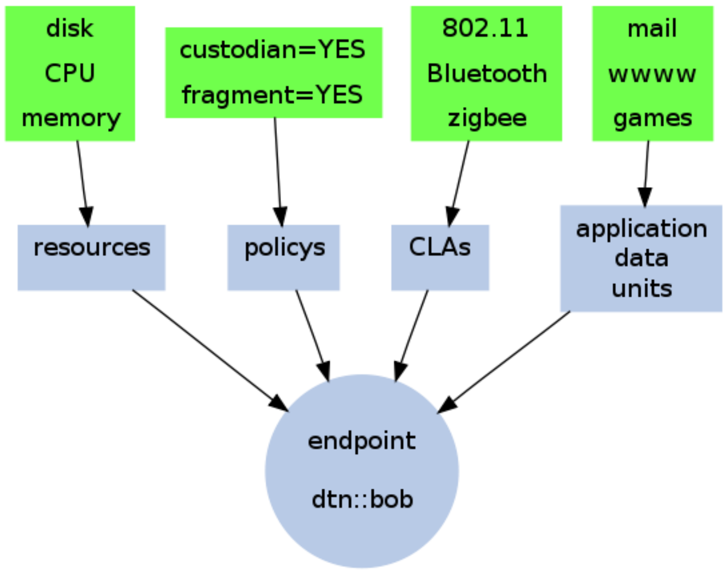


Figure 3.11: A DTN endpoint containing a DTN node identifiable by at least one unique endpoint identifier ($dtn :: bob$). Node is represented as a collection of a set of zero or more DTN endpoint identifiers

3.6.3 Network

Consider a delay-tolerant network composed of V potentially mobile DTN nodes populating a rectangle s. Movement of potential nodes is restricted within s, i.e., closed population. The opportunity to communicate is denoted a contact as in Fall (2003), which is parametrised by a source i, destination d, direction, time t, capacity, and a propagation delay . Propagation delay is assumed to be constant during the duration of the contact. In addition resource constraints are also considered (as in Jain et al. (2004)). The resulting topological graph of the network is a multidigraph with edges having their own identifier. A multidigraph is used because it may be possible to select (and combine) from many distinct physical CLAs to move data between the same pair of nodes. The multidigraph is permitted to have an edge connect a node to itself, such that a multidigraph G is an ordered 4-tuple $G := (V, E, i, d)$.

When the network is partitioned, the topological graph is also partitioned. The set of edges in the graph must capture time-varying node resource capacity, edge capacity and propagation delay as well as cater for multiple edges. Node resource capacities, propagation delay and edge capacities are time dependent, i.e., edge capacity is zero at times when the link is unavailable.

Each node j performs store-and-forward routing, and is parametrised by its finite storage capacity ($b_{v,t}$), memory capacity ($m_{v,t}$) and processor capacity ($p_{v,t}$). An edge is parametrised by its source i node and destination d node, plus a capacity $c_{e,t}$ and propagation delay $d_{e,t}$ and communication range. Nodes may not have the same communication range r and can communicate only if their range is below r. r_{e_t}. For an edge e_n the functions $(c(t))$, $(d(t))$ and $(r(t))$ are used to retrieve capacity, delay and range respectively (see Figure 3.12).

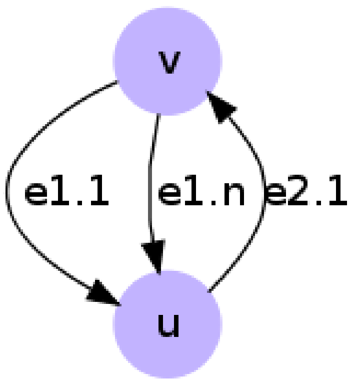


Figure 3.12: It may be possible to select from many distinct CLAs to move data between the same pair of nodes. In this graph two distinct CLAs are represented: e_1 and e_2. In the graph multiple contacts of type e_1 are represented as $e_{1.1}$,..$e_{1.n}$. The edge direction represents the source node (tail) and destination node (head). For an edge e_n the functions $(c(t))$, $(d(t))$, and $(r(t))$ are used to retrieve the capacity, delay, and range at time t respectively. For an edge e_n the functions $(b(t))$, $(m(t))$, and $(p(t))$ are used to retrieve the storage, memory and CPU capacity of the node v at time t respectively. Therefore, an edge e_n can be represented as $e_n = ((v, e, i, d)_n, c(t), d(t), r(t), b(t), m(t), p(t))$

.

3.6.4 Assumptions

It is assumed that both the capacity and delay of an edge are constant over a contact.

3.6.5 Implementation

The definitions of Table 3.14 describe the directed multigraph construction

Definition	Description
t	time_t epoch relative time
V	a set of vertices
E	a set of edges
$i : E \to V$	assigning to each edge its source node
$d : E \to V$	assigning to each edge its target node
b_v	storage capacity of the *lxc* TDP node v (1GB)
$b_{v,t}$	storage capacity of the *lxc* TDP node v at time t ($b_{v,t} = b_v$ - storage used at time t)
m_v	memory capacity of the *lxc* TDP node v (1GB)
p_v	CPU capacity of the *lxc* TDP node (the number of CPUs assigned to a *lxc* node (1)
c_e	estimated capacity of the edge
d_e	contact duration of the edge e in seconds
f_e	size sum of unique messages of the edge e in bytes
g_e	size sum of messages of the edge e in bytes
$c_{e,t}$	capacity of the edge e at time t
$d_{e,t}$	propagation delay of the edge e at time t
$r_{e,t}$	range of the edge e at time t (assumed to be constant)
I^v	is the set of edges whose destination node is v (incoming RX edges)
O^v	is the set of edges whose source node is v (outgoing TX edges)

Table 3.14: The definitions that describe the directed multigraph construction

Traffic demand Traffic demand is the set of all messages and is denoted by K. A message is a tuple (u,v,t,m) where (u,v) is the source-destination node pair, t is the time at which the message is injected and m is the message size. For a message $k \in K$, the functions $s(k), d(k), \omega(k), m(k)$ are used to retrieve the source node, the destination node, the start time and the size of the message respectively. The delay of a message is the time between when it is completely transmitted and when it is completely received.

Lower bound

The realised contact volume, i.e. the amount of data transferred during the contact, can be calculated to provide a lower bound. This lower bound signifies the actual usage of the contact.

The realised contact volume (i.e., the realised capacity of the edge e) is calculated using the contact duration in seconds and sum of bytes transferred (RX +TX). The realised capacity of the edge, a lower bound, is estimated using the lower bound realised capacity function.

Lower bound capacity function: The objective of the lower bound realised capacity function is to calculate the realised capacity of a contact, a lower bound, which can be realised by dividing the sum of the size of messages transferred during a contact by the contact duration (see Figure 3.13).

$$c_e = \frac{g_e}{d_e}$$

Figure 3.13: Lower bound realised capacity function

The realised contact volume can include duplicate bundles, i.e. include received or transmitted copies of the bundle with the same ID. Therefore, another lower bound, the lower bound unique capacity function is used to signify the realised volume of unique bundles transferred during the contact.

Lower bound unique capacity function The objective of the lower bound unique capacity function is to calculate the capacity of a contact realised for transfer of unique bundles, a lower bound, which can be realised by dividing the sum of the size of unique messages transferred during a contact by the contact duration (see Figure 3.14).

$$\frac{f_e}{d_e}$$

Figure 3.14: Lower bound unique capacity function

Higher bound

There are two functions used to estimate the the potential capacity of a contact; the higher bound capacity function, and the higher bound unique capacity function.

Higher bound capacity function: The objective of the higher bound capacity function is to estimate the potential capacity of a contact, an upper bound, which is calculated from a set of contacts selected to meet a criteria. The higher bound capacity function can be realised to estimate the potential capacity of a contact by multiplying its duration by the number acquired through dividing the size of the messages of the contact in the set that transferred the maximum size of RX and TX messages by its duration (see Figure 3.16).

$$(max\frac{g_e}{d_e}) \cdot d_e$$

Figure 3.15: Higher bound capacity function

The higher bound capacity function does not distinguish duplicate bundles, i.e. include recieved or transmitted copies of the bundle with the same ID. Therefore, another higher bound, the “higher bound unique capacity function” is used to signify the estimated potential volume of unique bundles transferred during the contact.

Higher bound unique capacity function: The objective of the higher bound unique capacity function is to estimate the potential capacity of a contact to transfer unique messages, an upper bound, which is calculated from a set of contacts selected to meet a criteria. The higher bound capacity function can be realised to estimate the potential capacity of a contact by multiplying its duration by the number acquired through dividing the size of the messages of the contact in the set that transferred the maximum size of RX and TX unique messages by its duration.

$$(max\frac{f_e}{d_e})\cdot d_e$$

Figure 3.16: Higher bound unique capacity function

3.6.6 Conclusions

Graph based model are appropriate for modelling DTNs.

In scenarios where the duration of transfer events is not known, but the contact duration, bytes transferred, and total number of messages transferred, along with the number of those messages that are unique is known, lower and higher bound metrics can be established. Using these bounds the exploitation of a contact can be evaluated. The objective of such lower bound functions is to provide a performance limiting function to be applied to links between pairwise nodes in an emulation. The functions to calculate the upper and lower bounds are described in Table 3.15

No.	**Name**	**Bound**	**Description**
1	lower bound capacity function	lower	volume of a contact realised
2	lower bound unique capacity function	lower	volume of a contact realised for transfer of unique bundles
3	higher bound capacity function	higher	estimated potential volume of a contact
4	higher bound unique capacity function	higher	estimated potential volume of a contact for transfer of unique bundles

Table 3.15: The functions of the directed multigraph

3.7 TDP and TDP module

This section describes the design of the Thing Discovery Protocol (TDP), a discovery protocol to support the use of the TDP models, and *TDP-module*, a module based on the TDP specification. The protocol is specified so that DTN nodes may leverage it to discover the presence of proximate endpoints, their endpoint elements, attributes and values. The scheme is intended to provide a consistent way to send or receive discovery data. The bundle-layer discovery protocol described is designed to be capable of mapping to underlying discovery mechanisms.

3.7.1 Rationale

We cannot assume centralised routing table construction and a static network in DTNs. Using a set of abstract knowledge oracles, such as Contacts Summary, Contacts, Queueing and Traffic Demand, Jain et al. (2004) have shown that with more knowledge, efficient routing algorithms can be constructed for routing in DTNs; that more knowledge is required to attain better performance. Routing protocols are expected to make use of information learned from TDP. Through supplying routing protocols with complete or partial knowledge, TDP enables protocols and applications to exploit an opportunistic communication opportunity in order to transfer more ADUs than protocols and applications that utilise information provided by other discrete sources. In particular supplying the endpoint elements and their attributes to epidemic protocols (Vahdat and Becker, 2000), such as Context-aware Adaptive Routing (CAR) protocol (Musolesi and Mascolo, 2009), could benefit from TDP. Knowledge of the presence of endpoints and elements could be used to distribute storage, forward efficiently, reduce congestion, find the most efficient CLA, multiplex CLAs, discover topology, avail of publish/subscribe mechanisms and select a peer node to forward which has the required enabled extension blocks.

As the first step, we introduce the assumptions underlying the design of the protocol. We assume that a node has no a priori knowledge about other nodes. In particular, we assume that a node is not aware of its absolute geographical location or of the location of nodes to whom it might deliver the message.

3.7.2 Protocol Design Assumptions

1. Communications may be over asymmetric links, be subject to long propagation delays, disruption, bandwidth asymmetry, packet loss and jitter.
2. Routing and application protocols will make use of information learned via discovery in DTNs
   (a) Using a routing protocol, a node exchanges TDM information with other nodes.
   (b) To route packets through the Internet towards their receivers, routers need to populate their routing tables with information how to reach different networks.
   (c) Operator intervention is not possible
   (d) Reliance on manual configuration is infeasible
   (e) Nodes experience transient failures, etc.

3. There is a common, simple model TDM, which models facts about things, capable of expressing information so that it can be interchanged without loss of meaning.
4. The TDM information is for generic interchange with local and remote sets of consumers and producers.
5. There is a naming model that describes the functional and structural aspects of associations.
6. JSON formatted TDM maps naturally to EAV.
7. Various BPA APIs are accessible by TDP in order to query and set parameters. BPA APIs are required to facilitate manipulation of the data, query CLA names and parameters, query active EID registration URIs and registration parameters.

3.7.3 Thing Discovery Protocol Specification

The Thing Discovery Protocol (TDP) can be used by DTN nodes to report their elements and element attributes to neighbouring DTN nodes. The TDP specification is provided in the following.

The TDP registrations

Within a DTN node, there is a registration which is the state machine characterising a given node's membership in a given endpoint. Any number of registrations may be concurrently associated with a given endpoint, and any number of registrations may be concurrently associated with a given node. An EID is a name, expressed using the general syntax of URIs RFC 3986, that identifies a DTN endpoint. TDP uses the "dtn:" scheme. The TDP EID registration, used by an application for receiving TDP ADUs, has the non-expiring registration "dtn://TDPv1".

An TDP registration receives TDP ADUs, which are report bundles generated periodically or on demand and transmitted to the endpoint "dtn://TDPv1". Registrations are strings, that might indicate a physical entity or named data/service, used by upper-layer protocols or DTN applications to ask the CLAs to enable and disable reception of ADUs sent to specific EIDs.

Each DTN node is required to have at least one EID that uniquely identifies it. The source EID of an TDP report bundle is any of those registrations.

The Naming models

TDP works with any DTN endpoint element and attribute as a first-class abstraction of information. Handling abstractions necessitates the generation of names, that when associated provide appropriate identifiers. In addition, various mechanism are needed in order to operate on the associations. Thus, we define a model of the functional and structural aspects of associations as per the naming model of this thesis.

The identifier properties cater for a syntactic structure, that is appropriate for fixed size universal identifiers, that facilitate partitioning of the namespace into identifier subsets, and support limited identifier rules. Flat association identifiers that can avail of network resolution, that do not restrict the flexibility of the network, and that facilitate decoupling of information from administrative domains are required. This motivates our choice of flat names, along with AV trees to facilitate multiple- attribute searching. Scalability is at least on the order of trillions and possibly beyond to accommodate future growth. The RFC 4122 UUID, which can be generated via MD5 or SHA-1 hashing of a thing, is implemented.

Association mechanisms generate UUIDs in the order of trillions, unique across both UUID space and time, via MD5 or SHA-1 hashing of a thing, at very high generation and assignment rate, and which do not require centralised administration of each identifier.

The association properties describe extensible, non persistent, unique, non transcribable associations that do not structure a semantic free namespace and consequently decouple location from name.

URIs, are extensible, have a global scope, facilitate parsing of identifier components without knowing the scheme-specific requirements and enable uniform identification of resources via a scheme part. UUIDs can be used as URIs and RFC 4122 [RFC4122] describes the correct scheme for rendering a UUID. To use a UUID as a URI a scheme is supplied, such as urn:uuid in addition to the URI content that is the UUID. The functional and structural components comprised of association mechanisms, identifier and association properties can be fulfilled through use of the UUID and URI, MD5 and SHA-1 hashing, content-oriented security models and attribute authentication.

Elements or attribute can be associated with a Thing ID as defined by this thesis. The EID of a CLA is a CLA EID as defined by this thesis. CLA EIDs are primarily used for the identification of CLAs and are used by TDP as potential "next hops".

Data, storage and interchange models

The data model (DM) is formulated as defined by this thesis. As are the storage and interchange models as defined by this thesis. JSON provides a generic interchange format.

TDP Flags for the Primary Block

The value encoded in the Bundle Processing Control Flags of the primary block is a string of bits, expressed as a SDNV RFC 6256 and is used to invoke selected bundle processing control features. By default, a bundle with a payload that is a TDP ADU has the individual bits of this string of bits set to zero to indicate false for the following flags:

1. Custody transfer requested
2. Status report request
3. Application data unit is an administrative record
4. Bundle must not be fragmented
5. Destination endpoint is a singleton
6. Acknowledgement by application is requested

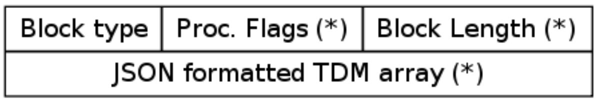
The TDP Bundle

The TDP payload block uses the Canonical Bundle Block Format as defined in section 4.5.2. of RFC 5050. That is, each TDP

1. Block type code
2. Block processing control flags
3. Block data length
4. Block JSON RFC 4627 formatted TDM array

A bundle with an TDP payload is uniquely identifiable and all bundle protocol features that rely on bundle identity, such as the Bundle Security Protocol RFC 6257, can be enabled. A bundle with a TDP payload may be fragmented.

TDP ADUs contained in bundles are called TDP reports. In order to take full advantage of the capabilities of TDPv1, a TDP node MUST be capable of generating a bundle payload block, a record with the format shown in Figure 3.17, and of transmitting that bundle to the TDP EID.



Block type	Proc. Flags (*)	Block Length (*)
JSON formatted TDM array (*)		

Figure 3.17: The TDP Report Bundle Format. A RFC 5050 Canonical Bundle Block Format

- TDP is a DTN application protocol that uses the canonical RFC 5050 bundle block format
- (*) Indicates field contains a Self Delimiting Numerical Value (SDNV)
- Routing protocols and algorithms are identified by EIDs in this thesis and have EID registrations. Active registrations (ARs) have registration references in the TDP payload block.
- ”Block Type” (see section 3.1 of RFC 5050) is a 1-byte mandatory field that indicates the type of the block. This field contains the value 1 to indicate it is the bundle payload block.
- ”Block Processing Control Flags” (see section 4.2 RFC 5050) is a SDNV that contains the BP block processing control flags, an unsigned integer expressed as a SDNV. The individual bits of this integer are set to ’1’ for the Last block flag and Discard block if it can’t be processed flag. A one octet SDNV is shown here for convenience in representation.
- ”Block Length” (see section 3.1 of of RFC 5050) is a mandatory SDNV that contains the aggregate length of all remaining fields of the block - which is to say, the length of the bundle’s payload. A one octet SDNV is shown here for convenience in representation.

TDP application agent

The TDP application agent is responsible for constructing TDP report ADUs to be transmitted in bundles. By default, bundles with TDP payloads are generated and transmitted periodically every 30 seconds. Bundles with TDP payloads can also be generated on demand.

The TDP agent is responsible for receiving an TDP ADU on the TDP registration and processing it.

3.7.4 Design

In this section, we give an overview of the *TDP-module* module design, that supports the discovery in a DTN composed of autonomous mobile TDP nodes. We present the key design choices and the novel mechanisms that are at the basis of its implementation. First, we describe the general requirements of the module. Second, discuss the design assumptions. Finally, we analyse the discovery theory and its foundation algorithms. The *TDP-module* module design is based on the TDP specification.

In order to transfer TDM data, stored TDM data must be retrieved, their ownership must be established, and they must be encoded as a compact JSON, UTF-8 encoded dictionary object. In order to work with bundles *TDP-module* must communicate with the DTN2 BPA. TDP must obtain, transmit and receive TDP reports.

Therefore, a class is designed to work with TDM, which we call the *Data* class. TDM assumes specification of the identifier properties and the association properties that comprise the structural components of the naming model. TDM also assumes specification of the association mechanisms for resolution, scalability, assignment, encoding, ascertaining authenticity and provenance, and validation that comprise the functional components of the naming model. Therefore a class is designed to work with the Naming model which we call the *Association* class. Another class, which we call the *DTN2* class is designed in order to communicate with the DTN2 BPA.

The methods and functions of the *Data*, *Association* and *DTN2* classes are not specific to the *TDP-module* module. Other applications are expected to use these shared classes. However *TDP-module* specific classes are required for *TDP-module* specific methods. A *TDP-module* specific class is required to extract TDM data. Another *TDP-module* specific class is designed to receive TDP bundles. Finally, a *TDP-module* specific class is designed to transmit TDP bundles. Figure 3.18 show TDP, the *TDP module* module, and related classes.

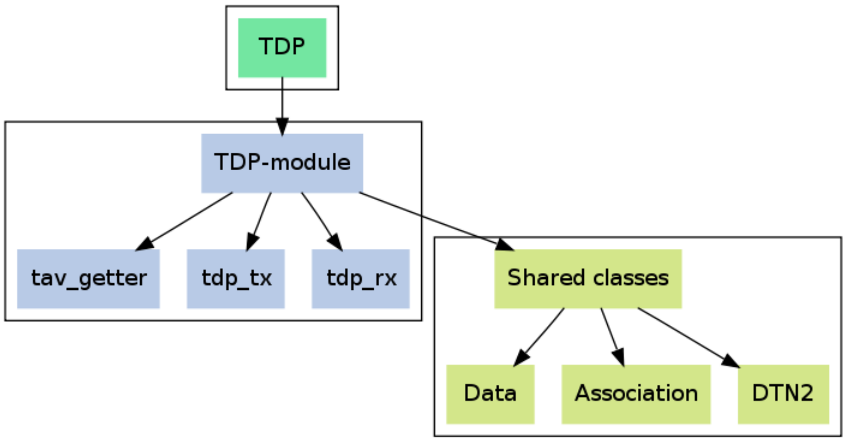


Figure 3.18: TDP, the *TDP module* module, and related classes

The class and method design of *TDP-module* are listed in Table 3.16.

Class	Type	Description
tav_getter	specific	the retrieval process of stored TDM data
tdp_tx	specific	the transmit process of queued TDM data
tdp_rx	specific	the receive process of bundle TDM data
Data	shared	to work with TDM
Association	shared	the methods of the naming model to work with the TDM
DTN2	shared	communication with the DTN2 BPA in order to work with bundles

Table 3.16: Class and method design requirements. *tav_getter*, *tdp_tx* and *tdp_rx* classes are specific to *TDP-module*. The *Data*, *Association* and *DTN2* classes are shared

Methods of tav_getter class

The *TDP-module* specific class that is required to extract TDM data is called the *tav_getter* class. The *tav_getter* class must use the shared *Association* class to create public and private keys and sign identifiers. The *tav_getter* class must use the shared *Data* class to distinguish the ownership of things and extract TDM data. Methods of the *tav_getter* class are used to create list objects, associative arrays, create JSON objects and place those object on a transmit queue. The method design of the *tav_getter* class are presented in Table 3.17.

Class	Method
Association	create keys (e.g., ElGamal)
Association	sign thingID or things
Data	obtain an *ownership* identity establish ownership of the data
Data	extract TDM data from data stores
tav_getter	Create list object of data store table selections
tav_getter	Create associative arrays list objects
tav_getter	Create a compact JSON UTF-8 encoded dictionary object
tav_getter	Place the JSON object on the *transmit* queue

Table 3.17: Class and method design requirements. *tav_getter* class is specific to *TDP-module* which uses methods from *Data* class

Methods of tdp_tx class

A *TDP-module* specific *tdp_tx* class is designed to transmit TDP bundles. The *tdp_tx* class must use the shared *DTN2* class to use the DTN2 API calls to construct TDP bundles, discover, remove and add links, discover, remove and delete routes and transmit TDP bundles. TDP bundles are constructed, as described in Table 3.18, using the *dtn_send* API call.

Methods of the *tdp_tx* class must obtain, delete and queue TDP JSON object and discover viable CLAs, The method design requirements of the *tdp_tx* class are presented in Table 3.19.

Methods of tdp_rx class

The *TDP-module* specific *tdp_rx* class is designed to receive TDP bundles, decode them, and persist the TDM data to storage. The *tdp_rx* class must use the shared *Association* class to check integrity of an association between the unique thing name and thing identifier and validate a signed thing identifier or signed thing. The *tdp_rx* class must use the shared *Data* class to persist the TDM data to storage. Methods of the *tdp_rx* class are used to create the TDP DTN registration, poll the receive queues

Variable	**Reason**
regid	Value for an unspecified registration cookie set to indicate that the daemon should allocate a new unique id.
source	The bundle source EID which is the non signleton TDP DTN EID *dtn://tdp_v1.0*.
dest	The bundle destination EID which is the non singleton TDP DTN EID *dtn://tdp_v1.0*
replyto	The bundle reply to EID which is the non singleton TDP DTN EID *dtn://tdp_v1.0*
priority	The priority with which the bundle is sent. A regular (normal) priority is set in the bundle priority specifier
dopts	Bundle delivery option is set to flag that no options are selected for a given bundle i.e. no custody, etc
expiration	Bundle expiry is set by default to 24 hours (60 * 60 * 24) to expire bundles not consumed 24 hours after they have been generated. Expiry can be altered as required
location	The payload of a bundle can be sent in a file, in which case the payload structure contains the filename, Indicates bundle payload contents are in a file. TDP Payload ADU location
filename	TDP Payload ADU name

Table 3.18: TDP report bundle format

Class	**Method**
DTN2	Open a connection to the BPA
DTN2	Close a connection to the BPA
DTN2	Add link for each CLA interface to the TDP DTN EID *dtn://tdp_v1.0*
DTN2	Delete routes to the TDP DTN EID *dtn://tdp_v1.0*
DTN2	Obtain the set of DTN2 links
DTN2	Add copy routes to links so that any TDP bundles are queued for transmission on every CLA
DTN2	Construct a TDP bundle as described in Table 3.18 using the *dtn_send* API call
DTN2	Transmit TDP bundles
tdp_tx	Discover resources, such as CLA interfaces, on the system
tdp_tx	Queue TDM JSON data objects
tdp_tx	Obtain the TDM JSON data objects FROM the TX queue and persist them to a temporary file
tdp_tx	delete the TDP payload file after successful transfer

Table 3.19: Class and method design requirements. *tdp_tx* class is specific to *TDP-module* and uses methods from a DTN2 class

for incoming TDM JSON objects, and receive and decode a TDM JSON object. The method design requirements of the *tdp_rx* class are presented in Table 3.20.

3.7.5 Module Design Assumptions

This design assumes the specification of the following:

1. DTN RI interface API

2. DTN RI API

3. An accessible TDM data store

Class	Description
tdp_rx	Bind to an existing TDP application registration or if not present create the non expiring TDP application registration for TDP DTN EID *dtn://tdp_v1.0*
tdp_rx	*poll()* TDP file descriptors for I/O events
tdp_rx	Monitor the TDP file descriptors with *select()*
tdp_rx	receive the bundle and process the primary block and payload block. Supplementary DTN protocol specifications (including, BSP) may require that additional measures be taken at this point of the procedures. The TDP payload ADU is placed at a specified location
tdp_rx	open the TDP payload file
tdp_rx	extract the JSON object
tdp_rx	iterate through JSON formatted TDM associative array structure to extract the unordered set of *key: value* pairs
tdp_rx	close the payload file
tdp_rx	delete the payload file
Association	verify the integrity of a things using the version 5 UUID i.e. check integrity of a named association of the unique thing name and thingID)
Association	validate signed thingID or signed thing i.e. a given a thing or thingID by its signature
Data	persist the TDM data to storage

Table 3.20: Class and method design requirements.*tdp_rx* class is specific to *TDP-module* which uses methods from *Data* and *dtnapi* classes

3.7.6 TDP Specification

Methods of the *Data* and *Association* class to work with TDM data, along with methods of the *DTN2* class to work with the DTN2 BPA using its tcl API are specified. The application domain of these classes is not specific to *TDP-module* module. Three *TDP-module* module specific classes *tav_getter*, *tdp_tx* and *tdp_rx* are specified for *TDP-module* specific methods.

3.8 TAV-maker module

This section describes the design of the *TAV-maker* module, that supports the development of applications to translate endpoint elements into TAVs using the TDM. The *TAV-maker* module enables endpoints to discover their own TDM data.

3.8.1 Rationale

To structure information on abstraction and physical elements using TDM, a module is required with classes to translate endpoint elements into TDM data, and persist them to a data store as EAV/CR. Furthermore, the classes are designed to name, sign, verify integrity of, validate signed data. The *TAV-maker* module uses the association, data and storage models.

3.8.2 Design

In this section, we give an overview of *TAV-maker* module design, presenting the key design choices and the novel mechanisms that are at the basis of its implementation. First, we describe the general requirments of the module. Second, discuss the design assumptions. Finally, we analyse the logical decision theory and its foundation algorithms.

In order to map information from other discovery protocols and applications or extract information from other sources a class required to work with TDM to extract, parse and structure endpoint element information. This class is called the *Data* class. TDM methods are designed to translate extracted data into TDM data structures composed of an unordered set of *key: value* pairs, with the requirement that the keys are unique. Furthermore methods are designed to persist TDM data to a EAV/CR schema based data store. Methods must be capable of handling duplicate definitions of classes.

A set of classes are designed for the generation of TDM structured data for various classes of TDM data. While TDM should be capable of working with everything, as we described in Chapter 1, that is not to say everthing should be given a name; *"good abstration is ignoring the right detail at the right times"* (Shaw, 1990). A set of classes is required that collectively provides a set of endpoint elements which can be used to exploit a contact. Figure 3.19 shows the *TAV-maker* module and related classes.

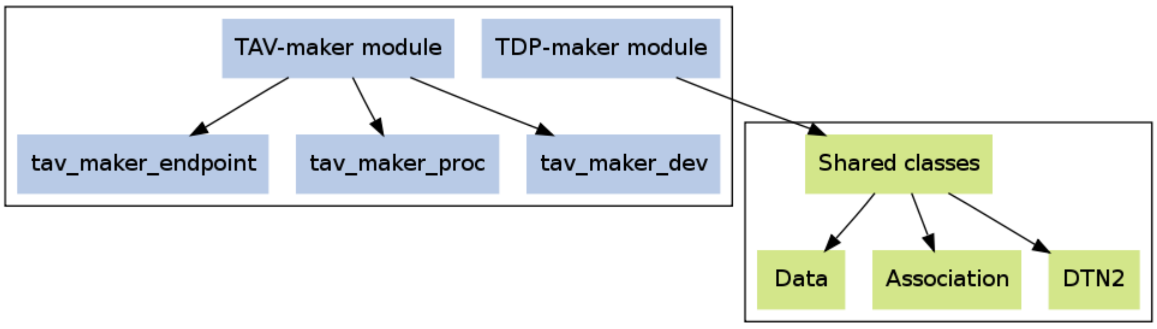


Figure 3.19: The *TAV-maker* module and related classes

Additionly the set is designed to demonstrate the steps to translate physical and abstraction endpoint elements into TDM data and provide template structures to develop new translators. Five classes are derived for this purpose are presented in Table 3.21.

Class	Type	Description
Data	shared	to work with TDM
Association	shared	to provide functions and structures of the association model
tav_maker_endpoint	specific	for BPA specific elements
tav_maker_proc	specific	for */proc* elements
tav_maker_dev	specific	for */dev* elements

Table 3.21: Class design requirements

Methods of Data class

Methods of a *Data* class are required to process abstraction data, such as application responses and physical data such as files. Once processed, in order to translate extracted data into TDM, list data type objects can be used to create associative arrays of things; dictionary data structures. A generic method is required to persist TDM data to the data store. Methods are required to demonstrate extraction, parsing, and translation of data from various physical and virtual sources to TDM data.

Persist TDM

Once discovery protocol, applications and information from other sources has been translated into TDM data, A method is required to persist TDM data to the data store as specified in Table 3.22. This method is used generically to persist TDM data whether from phyical or virtual sources.

Class	Description
Data	persist the thing list to the data store Thing table
Data	iterate through associative array to extract attributes
Data	check type of object
Data	translate objects into association list using type specific parser
Data	enumerate objects
Data	check if attribute exist
Data	persist the attribute to the data store attribute table
Data	check if reciprocal attribute ID exists
Data	persist the reciprocal attribute ID to the data store ReciprocalAttributesID table
Data	check if reciprocal attribute exists
Data	persist the reciprocal attribute to the data store ReciprocalAttributes table
Data	check if TAV exists
Data	persist the TAV to the data store specific TAV type (e.g., TAV_str) table
Data	check if TAV thing exists
Data	persist the TAV thing to the data store TAV_thing table

Table 3.22: Sequential steps of the *persist_thing* method to persist TDM to the data store

List processing

A method is required to process abstraction list type data. Such data includes generated application data retrieved in response to a query, and consequently provides little other information by direct analysis.

The method to do this is called *get_thing_list()* and is specified in Table 3.23. As part of this process a method is required to translate extracted lists data into TDM data.

Class	Description
Data	iterate through list and for each device
Data	invoke *e_thing()* method to translate extracted list data into TDM data
Data	iterate through associative array to extract attributes
Data	invoke *persist_thing()* method to persist TDM to the data store

Table 3.23: Sequential steps of the *get_thing_list()* method to to process abstraction list type data object

Representation processing

A method is required to process a physical file and directory type representations. Some file information can be obtained from direct analysis of the representation. Once processed a method is required to translate extracted the data into TDM data.

The method to do this is called *get_thing_file()* and is specified in Table 3.24. As part of this process a method is required to translate extracted lists data into TDM data.

Class	Description
Data	open the file
Data	read entire file
Data	format data as list object
Data	iterate through list and for each device
Data	invoke *h_thing()* method to translate file / directory representation into TDM data
Data	iterate through associative array to extract attributes
Data	invoke *persist_thing()* method to persist TDM to the data store

Table 3.24: Sequential steps of the *get_thing_file()* method to to process abstraction list type data

.

List type translation to TDM

Methods are required to translate processed data into TDM data. The method required to translate virtual type input to TDM are specified in Table 3.25. The method to do this is called *e_thing()*.

Class	Description
Association	generate a version 5 UUID from the unique association name to assign as a thingID
Data	establish associative array type (i.e., list, string or tuple)
Data	establish whether class exists. If class exists use classID, otherwise create a new classID and populate class
Data	create thing creation timestamp
Data	return a thing list

Table 3.25: Sequential steps of the *e_thing()* method to translate virtual type input to TDM

Representation type translation to TDM

The method required to translate physical representation of file and directory type input to TDM are specified in Table 3.26. The method to do this is called *h_thing()*.

Class	Description
Association	generate a version 5 UUID from the unique association name to assign as a thingID
Data	establish file MIME type (i.e., “application/x-directory”)
Data	establish whether class exists. If class exists use classID, otherwise create a new classID and populate class
Data	create thing creation timestamp
Data	return a thing list
Data	establish associative array type (i.e., list, string or tuple)

Table 3.26: Sequential steps of the *h_thing()* method to translate phyical representation type input to TDM

TDM Classes

Three classes are required for the generation of TDM structured data for three classes of TDM: *tav_maker_endpoint*, *tav_maker_proc*, and *tav_maker_dev*.

TAV class: tav_maker_endpoint

The *tav_maker_endpoint()* method requires specification of a *DTN2* method to enable query of the DTN2 BPA using its Tcl interface and the *Data* class to translate and persist information, the BPA is capable of providing, as TDM things. The information retrieved by query of the BPA is detailed in Table 3.27. The *tav_maker_endpoint()* method uses the *get_thing_list()* method to translate and persist TDM things to data store

Class	Description
DTN2	Get own unique EID
DTN2	Get discovery
DTN2	Get bundles
DTN2	Get bundle statistics
DTN2	Get bundle daemon statistics
DTN2	Get registrations
DTN2	Get interface
DTN2	Get link
DTN2	Get link statistics
DTN2	Get routes
DTN2	Get DTLSR route graph

Table 3.27: The *DTN2* methods to enable query of the DTN2 BPA using its Tcl interface

TAV class: tav_maker_proc

The *tav_maker_proc()* method requires specification of *Data* class methods to obtain a comprehensive list of */proc* files and directories through recursive iteration and uses *get_thing_file()* method to translate and persist information as TDM things. The methods to retrieve information are detailed in Table 3.28.

Class	Description
Data	Obtain */proc* location
Data	Create list of */proc* file system files (no recursion)
Data	Filter unwanted items from proc file list
Data	Check file list type
Data	Create list of directories of */proc* file system (no recursion)
Data	Create list of PID directories from directory
Data	Filter unwanted *proc* PID directory list items
Data	iterate through */proc* objects and populate tables with */proc* file contents

Table 3.28: Sequential steps of the *tav_maker_proc()* methods to iteratively extract file and directory data of */proc*

TAV class: tav_maker_dev

A class is required to obtain, translate and persist virtual */dev* information as TDM things. *tav_maker_dev* is a class to translate and persist abstraction list type data, like the *get_thing_list()* method, but also includes methods to obtain data directly. This class is required to demonstrate the methods to query using operating system tools, create list object, translate and persist virtual TDM things. The methods of *tav_maker_dev* are detailed in Table 3.29.

Class	Description
tav_maker_dev	set environment paths for udevadm application
tav_maker_dev	create a list of the set of all device names using udevadm
tav_maker_dev	format and sort the device name list using udevadm
Data	iterate through list and for each device
Data	invoke *h_thing()* method to translate file / directory representation into TDM data
Data	iterate through associative array to extract attributes
Data	invoke *persist_thing()* method to persist TDM to the data store

Table 3.29: Sequential steps of the methods to to process */dev* abstraction list type data

3.8.3 Design Assumptions

This design assumes the specification of the following:

1. While the design is application agnostic, specific methods are required to translate application specific information into list objects.
2. privilages to use applications to extract data
3. The Association class maps to the association model
4. The Data class maps to the storage, interchange model and TDM
5. A data store is accessible to persist TDM

3.8.4 TAV-maker Specification

We specifiy methods of the *Data* and *Association* class to work with TDM data, along methods of the *DTN2* class to work with the DTN2 BPA using its tcl API (URL, 81). The application domain of these classes is not specific to *TDP-maker* module. Three *TDP-maker* module specific classes are specified for the generation of TDM structured data for three classes of TDM: *tav_maker_endpoint*, *tav_maker_proc*, and *tav_maker_dev*.

3.9 TDP-decider module

This section describes the design of the module which makes descisions based on TDM data, *TDP-decider*, that is designed to assist the developers of autonomous mobile TDP nodes to specify the application domain of *TDP-module* and its constraints, by providing an application that defines the set of actions that are taken based on the learnt structured information.

3.9.1 Rationale

As described in Chapter 1 to evaluate TDP, we implement it encapsulated in RFC 5050 bundles, in order to understand whether mechanisms of a node that utilise information provided by TDP can exploit a contact, as exemplified by contacts described by the 2009, 2010 and 2011 N4C trial data, in order to transfer ADUs.

We must establish whether there are cases where in the event of a contact, data transfer applications that utilise information provided by TDP are enabled to transfer more application data units than applications that utilise information provided by other sources. Thus a data transfer application to utilise TDP is required. An appropriate data transfer application to utilise TDP is exemplified by *TDP-decider*

3.9.2 Design

In this section, we give an overview of *TDP-decider* module design, presenting the key design choices and the novel mechanisms that are at the basis of its implementation. First, we describe the general requirements of the module. Second, discuss the design assumptions. Finally, we analyse the logical decision theory and its foundation algorithms.

In order to take logical decisions, TDM data must be extracted and translated into variables with which logical decisions on more than one choice of actions, that depend on the variables' values, are to be taken. In order to action decisions, *TDP-decider* must communicate with the DTN2 BPA. Therefore *TDP-decider* specific classes are required for *TDP-decider* specific methods. Additionally a class to work with TDM and a class to work with DTN2 are required. Figure 3.20 show TDP, the *TDP module* module, and related classes.

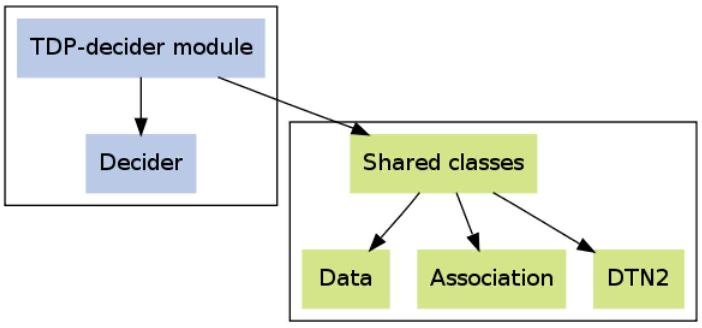


Figure 3.20: The *TDP decider* module, and related classes

The class and method design of textitTDP-decider are presented in Table 3.30.

Class	**Method**
Data	to extract TDM data from the data store
Data	to parse and translate extracted TDM data into logical decision variables for TDM classes
DTN2	to effect or action a logical decision
Decider	to work with the TDM data to demonstrate how TDM data can be extracted
Decider	to obtain the TDM data via different axis to demonstrate the flexibility of the TDM data
Decider	to place action events on a queue
Decider	to test conditions on which decisions, based on logical decision variables, are actioned

Table 3.30: Class and method design requirements. Decider class is specific to *TDP-decider* and uses methods from Data and DTN2 classes

3.9.3 Design Assumptions

This design assumes the specification of the following:

1. TDP application
2. TAV-maker application
3. DTN RI API
4. An accessible TDM data store

3.9.4 TDP-decider Specification

A class *Data* with methods to work with TDM data and class *DTN2* with methods to work with the DTN2 BPA using its tcl API are specified. The application domain of these classes is not specific to *TDP-decider* module. The methods of the *decider* class are used to obtain the values, attributes and relationships of a certain set of things with certain ownership.

A simple example of use of the *TDP-decider* module follows. We could use the *TDP-decider* module to obtain the *proc* class thing for "net/wireless", extract a list of wireless interfaces for an owner, their status, link quality and their layer 2 metrics, such as missed beacons, and select a wireless interface and configure it.

Using the lower and higher bound metrics that we described in Table 3.15, we can evaluate the exploitation of a contact. The objective of the lower bound functions is to provide a performance limiting function to be applied to links between pairwise nodes in an emulation. The *TDP-decider* module can use TDM data in and emulation in order to select whether to compress payloads and detect, and remove duplicate bundles.

3.10 Ns-3 modules

This section describes the design of the modules that build and use ns-3 as a real-time network emulator, which corresponds to the network model, and that can be interconnected with the *real* world.

3.10.1 Rationale

As described in Chapter 1 to evaluate TDP, we implement it encapsulated in RFC 5050 bundles, in order to understand whether mechanisms of a node that utilise information provided by TDP can exploit a contact, as exemplified by contacts described by the 2009, 2010 and 2011 N4C trial data, in order to transfer ADUs.

We must evaluate TDP using the *TDP-module*, *TDP-decider* and *TAV-maker* modules to establish whether there are cases where in the event of a contact, *real*-world data transfer applications that utilise information provided by TDP are enabled to transfer more application data units than applications that utilise information provided by other sources. An appropriate real-time network emulator, that allows *real*-world protocol implementations, such as the DTN RI *DTN2* and TDP using the *TDP-module*, *TDP-decider* and *TAV-maker* modules, to be re-used within it, and which emulates a network that corresponds to the network model, is required.

Khabbaz et al. (2012) describe evaluation of the performance of DTNs through simulation studies as built around many unrealistic assumptions. Poor repeatability and a lack of experimental control are some of the problems of experimental validation which can be overcome by emulation environments. Our survey of potential emulation environments for experimental testbeds in Section 2.4 found ns-3 to be appropriate for evaluation using our data from real DTN testbeds to provide a realistic evaluation of performance.

Therefore, modules are required to enable use of ns-3 as a real-time network emulator of contacts described by the 2009, 2010 and 2011 N4C trial data.

3.10.2 Design

In this section, we give an overview of the *ns3_configer* and *ns3_lxc_configer* module design, presenting the key design choices and the novel mechanisms that are at the basis of its implementation. First, we describe the general requirments of the module. Second, discuss the design assumptions. Finally, we analyse the logical decision theory and its foundation algorithms.

In order to evaluate TDP using the *TDP-module*, *TDP-decider* and *TAV-maker* modules, and ns-3 as a real-time network emulator, modules are required to interface with the ns-3 tap bridge module *TapBridge*, designed to integrate hosts that support Tun/Tap devices into ns-3 simulations. Modules are required to interface with the LINUX containers *lxc* of the mainstream linux kernel, that provide resource management through process containers and resource isolation through namespaces. Modules are required to configure the network as per specification of the network model and the TDP nodes themselves.

Therefore specific classes are required for *ns-3*, *lxc* specific methods. A *ns3_configer* module class, called *ns3configer*, is required to build ns-3 as a real-time network emulator, create *lxc*, install *TDP-module*, *TDP-decider* and *TAV-maker* modules and related software, and configure the *lxc* node. A class, called *emuModel*, is required to use ns-3 as a real-time network emulator to model a DTN contact, and the events that occur within its duration, that corresponds to the network model using real DTN2 contact data.

A module *ns3_lxc_configer* must be specified to generate ns-3 native configuration modules, for example *dtn-thing-lxc-cmsa-wifi-1.01*, that uses the *emuModel*, along with the start up module *startup* and shut down module *shutdown* to initialise and teardown Tun/Tap devices, the *lxc* components and the emulation itself. The method design requirements of the *ns3_configer* module are described in Table 3.31. The class and function design requirements of the ns-3 native configuration module generated by the *ns3_lxc_configer* module are detailed in Table 3.32. The method design requirements of the network model emulation *emuModel* class to be generated by the *ns3_lxc_configer* module are specified in Table 3.33.

Class	**Method**
ns3configer	configure, build and make ns-3 as an emulator
ns3configer	create an *lxc* node from a template
ns3configer	install packages on the ns-3 host
ns3configer	chroot to lxc image and install packages
ns3configer	output dtn configuration (DTN2 dtn.conf)
ns3configer	output sudo configuration (sudo sudoers)
ns3configer	output odbcini configuration
ns3configer	output odbcinst configuration
ns3configer	output dtn daemon init.d startup/shutdown scripts (dtnrc)
ns3configer	create users
ns3configer	authorise users
ns3configer	create directory structure for dtn, tdp, logs etc
ns3configer	configure, build and make DTN2
ns3configer	configure, build and make *TDP-module*, *TDP-decider* and *TAV-maker* modules
ns3configer	create a DTN2 data store

Table 3.31: Method design requirements of the ns3configer class specific to *ns3_configer* module for generation of ns-3 environment and *lxc* based TDP nodes

Type	**Description**
Class	for network model emulation (*emuModel*)
Function	start and stop simulation of network model
Function	pre-simulation actions to take

Table 3.32: Class and function design requirements of the ns-3 native configuration module generated by the *ns3_lxc_configer* module

3.10.3 Design Assumptions

This design assumes the specification of the following:

1. The *TDP-module* module
2. The *TDP-decider* module
3. The *TAV-maker* module

Class	**Method**
emuModel	execute an operating system process and wait to complete
emuModel	execute an operating system process and do not wait to complete
emuModel	emulate contact by schedule stop and start of a CLA
emuModel	emulate contact up event (DTN2 *CONTUP*) on an interface
emuModel	emulate contact down event (DTN2 *CONTDOWN*) on an interface
emuModel	create a payload test file of a certain size
emuModel	send a bundle
emuModel	register intent to receive a bundle
emuModel	start dtn daemon (DTN2 dtnd)
emuModel	stop dtn daemon (DTN2 dtnd)
emuModel	start DB
emuModel	stop DB
emuModel	start *TDP-module*, *TDP-decider* and *TAV-maker* modules
emuModel	stop *TDP-module*, *TDP-decider* and *TAV-maker* modules

Table 3.33: Method design requirements of the network model emulation emuModel class to be generated by the *ns3_lxc_configer* module

4. The DTN RI

5. An accessible TDM and DTN2 data store

6. At least one host that fulfils the host requirements to use ns-3 as a real-time network emulator

7. Python

3.10.4 Ns3_configer and ns3_lxc_configer Module Specification

Two ns-3 module are required: *ns3_configer* and the *ns3_lxc_configer* module. The *ns3_lxc_configer* module is specified to generate four modules: a ns-3 config, a start up, a shut down, and a "run emulation" module. A class *ns3configer* is specified with methods to work to build ns-3 as a real-time network emulator, create *lxc*, install *TDP-module*, *TDP-decider* and *TAV-maker* modules and related software, and configure the *lxc* node is specified. A class *emuModel* with methods to use ns-3 as a real-time network emulator to model a DTN contact, and the events that occur within its duration, that corresponds to the network model using real DTN2 contact data is specified. The two ns-3 module are shown in Figure 3.2.

3.11 Analysis module

This section describes the design of the module used to analyse cases where in the event of a contact, *real*-world data transfer applications that utilise information provided by TDP are enabled to transfer more application data units than applications that utilise information provided by other sources.

3.11.1 Rationale

As described in Chapter 1 to evaluate TDP, we implement it encapsulated in RFC 5050 bundles, in order to understand whether mechanisms of a node that utilise information provided by TDP can exploit a contact, as exemplified by contacts described by the 2009, 2010 and 2011 N4C trial data, in order to transfer ADUs.

We must evaluate TDP using the *TDP-module*, *TDP-decider* and *TAV-maker* modules to establish whether there are cases where in the event of a contact, *real*-world data transfer applications that utilise information provided by TDP are enabled to transfer more application data units than applications that utilise information provided by other sources.

Therefore, a module is required examine DTN logs of an arbitrary number of DTN nodes in order to provide a comprehensive statistical analysis of pairwise contacts, extract a set of those contacts, profile, and parametrise each contact so that is corresponds to a contact of our network model. To evaluate the exploitation of a contact, a contact volume and contact volume upper bound must be estimated. To estimate a contact volume and contact volume upper bound, each contact must include information on the size and uniqueness of the ADUs transferred, the contact duration, the contact duration no used, the duration of transmit and receive events.

3.11.2 Design

In this section, we give an overview of the *Analysis* module design, presenting the key design choices and the novel mechanisms that are at the basis of its implementation. First, we describe the general requirements of the module. Second, discuss the design assumptions. Finally, we analyse the logical decision theory and its foundation algorithms.

In order to evaluate TDP using the *TDP-module*, *TDP-decider* and *TAV-maker* modules, and ns-3 as a real-time network emulator, modules are required to analyse contacts described by the 2009, 2010 and 2011 N4C trial data. Trial data logs are in DTN2 log format. The format of DTN2 logs is subject to change and the format of the DTN2 logs for 2009, 2010 and 2011 N4C trial data differ. In order to allow generic processing the log data should be presented uniformly. A tool is require to process DTN2 logs of an arbitrary number of nodes and providing a comprehensive statistical analysis of pairwise contacts as an output, which should be persisted to a data store to enable querying. The module should facilitate the automated treatment of data, such as indexing, enumeration, filtering and resolution, so that treatments are accountable. The module should provide a mechanism to randomly select an appropriate set of contacts from the data to be used as an input to the real-time network emulator in order to evaluate the amount of ADUs transferred during those contacts.

Therefore specific classes are required for *Analysis* module specific methods. Classes are required to interact with the data store *DB*, work with data *Data*, resolve EIDs, hostnames, IPs and ports *Resolve*, filter and analyse contacts events *Analysis*, and provide statistical analysis of contacts *Stats*.

In order to analyse the opportunistic contact data of DTN nodes, the processing of the DTN2 log data should be automated to mitigate human error and provide transparent treatment of the data. This process requires the creation of a data store table to persist the full data set described in Table 3.34.

Field	**Description**
rfc3339	RFC 3339 date
timetINusec	time_t in usec
node	logging node
event	event
size	size
src	source EID
ID	bundle ID
dest	destination EID
expiry	bundle Expiry
flags	bundle flags
peer	peer id
obINsrc	other bundle source
obINID	other bundle ID
otimeINousec	other time
conttype	contact type
duration	duration

Table 3.34: Schema for presentation of DTN2 S10 log fields

Once persisted, a table is required to persist a subset of the data and additional treatment related information as described in Table 3.35.

Field	**Description**
rfc3339	RFC 3339 date
timetINusec	time_t.usec
node	Logging-node
event	event
size	size
src	Src-EID
ID	Bundle-ID
dest	Dest-EID
expiry	Bundle Expiry
flags	Bundle flags
peer	peer
obINsrc	Other bundle source
obINID	other bundle ID
otimeINousec	Other time
conttype	Contact type
duration	Duration
peername	peer host name
peerport	TCP /UDP port
peerid	Enumerated peer
nodename	logging node hostname
nodeid	enumerated logging node

Table 3.35: Schema of log subset table with additional fields *peername, peerport, peerid, nodename*, and *nodeid*

The number of subset tables created is arbitrary. For each subset table, a certain set of transformations are required to be carried out and three summary tables are required to provide a summary for links, a summary for each contact extracted from those links and a statistical summary of the bytes transferred for each contact. Contacts are required to be enumerated and a table view, with fields as described in Table 3.36, is required to represent each unique contact, which corresponds to a contact of our network model described in Section 3.6. In order to extract pairwise contacts, methods and algorithms are required to extract unique peer tuples into table views.

No.	**Field**	**Description**
1	con_id	contact identifier (a generated uuid)
2	tablename	the table name of the link in which the contact occurred
3	event	event (TX / RX / CONTUP / CONTDOWN)
4	rfc3339	RFC 3339 date time
5	timetINusec	time_t epoch time
6	bytes	bytes involved
7	tx_dur	TX duration
8	tx_tp	TX throughput
9	tx_cum_total_dur	TX cumulative total of TX events for this contact
10	rx_dur	RX duration
11	rx_tp	RX throughput
12	rx_cum_total_dur	RX cumulative total of RX events for this contact
13	c_dur	contact duration
14	txrx_cum_total_dur	combined TX/RX contact duration
15	c_dur_u	unused contact duration
16	d_dur	delivery duration
17	eventid	enumerated contact event
18	ID	RFC 5050 bundle ID
19	peername	peer node name
20	peerid	enumerated peer node identifier
21	nodename	logging node name
22	nodeid	enumerated logging node identifier

Table 3.36: Table view of the *test contacts* set that provide a representation of a contact that corresponds to the network model

Three summary tables are required. A table to summarise the contact statistics for each link is described in Table 3.37.

A *contact_detail* table is required to provide detailed information for each contact extracted as described in Table 3.38.

A *contact_uniq* table is required to provide statistics, using the combined TX/RX event duration and bytes received or transmitted, for the subset of contacts extracted as described in Table 3.39. Figure 3.4 shows the *Analysis* module and related classes.

An overview of the class and method design requirements of the *Analysis* module to extract pairwise contact from DTN2 log files into data store table views are detailed in Table 3.40.

3.11.3 Design Assumptions

This design assumes the specification of the following:

1. The DTN RI S10 log events
2. An appropriate data store

Field	Description
tablename	table name
cnt_up	count of contact up events
cnt_down	count of contact down events
cnt_tx	count of TX events
cnt_rx	count of RX events
cnt_uid	count of unique bundles involved in TX and RX
cnt_id	count of all bundles involved in TX and RX
size_tx	size of TX in kb
size_rx	size of RX in kb
size_txrx	size of combined TX and RX in kb
tx_throughput	throughput of TX in bits per second
rx_throughput	throughput of RX in bits per second
combined_throughput	combinded throughput of TX and RX in bits per second
combined_dur	combined duration for all contacts
table_dur	table duration for all events
node	logging node
peer	peer
count	count (of all values)
count_distinct	count (of distinct values)
sum	sum
min	min
max	max
mean	mean
stddev_pop	population standard deviation
stddev_samp	sample standard deviation
dof	for a set of n values, this is standard deviation based on n degrees of freedom
n_1_dof	for a set of n values, this is standard deviation based on n-1 degrees of freedom
dev_from_mean	> 3 standard deviations from mean
sumofsquares	represents sum of squares
var_pop	population variance
var_samp	sample variance
modal	the 1st most frequent (the modal) value
modal_t	the amount of times the the 1st most frequent (the modal) value occurs
modal_p	percentage the 1st most frequent (the modal) value occurs
regression_slope	the regression slope
intercept	intercept
ls_regression	the regression equation (least-squares regression)
c_coefficient	correlation coefficient

Table 3.37: DTN2 log analysis summary for each link table that provides link statistics and contact summary for the combined TX/RX event duration and bytes received of transmitted. For a subset table called *2010s10un* this is called *2010s10un_contacts_summary*

3.11.4 Analysis Module Specification

A module *Analysis* is required for contact analysis and evaluation. The *Analysis* module classes are specified for *Analysis* module specific methods. A *DB* class is specified to interact with the data store, a *Data* class is specified to work with data, and a class named *Resolve* is specified to resolve EIDs, hostnames, IPs and ports. A class named *Analysis* is specified to filter and analyse contacts events. A class named *Stats* is specified to provide statistical analysis of contacts.

No.	Field	Description
1	con_id	contact identifier (uuid)
2	tablename	table name
3	event	event TX/RX/CONTUP/CONTDOWN
4	rfc3339	rfc3339 date time
5	timetINusec	time_t epoch time
6	bytes	Bytes involved
7	tx_dur	TX duration
8	tx_tp	throughput
9	tx_cum_total_dur	cumulative total of tx events for this contact
10	rx_dur	TX duration
11	rx_tp	throughput
12	rx_cum_total_dur	cumulative total of tx events for this contact
13	c_dur	contact duration
14	txrx_cum_total_dur	combinded TX/RX contact duration
15	c_dur_u	unused contact duration
16	d_dur	delivery duration
17	eventid	enumerated contact event
18	ID	RFC5050 Bundle ID
19	peername	peer node name
20	peerid	peer node identifier (EID)
21	nodename	logging node name
22	nodeid	logging node identifier (EID)

Table 3.38: DTN2 log analysis summary for each contact is provided in the contacts detail table. For a subset table called *2010s10un* this is called *2010s10un_contacts_detail*

Field	Description
con_id	contact identifier (uuid)
count	count (of all values)
count_distinct	count (of distinct values)
sum	sum
min	min
max	max
mean	mean
stddev_pop	population standard deviation
stddev_samp	sample standard deviation
dof	for a set of n values, this is standard deviation based on n degrees of freedom
n_1_dof	for a set of n values, this is standard deviation based on n-1 degrees of freedom
dev_from_mean	>3 standard deviations from mean
sumofsquares	represents the sum of squares
var_pop	population variance
var_samp	sample variance
modal	the 1st most frequent (the modal) value
modal_t	the amount of times the the 1st most frequent (the modal) value occurs
modal_p	percentage the 1st most frequent (the modal) value occurs
regression_slope	the regression slope
intercept	the intercept
ls_regression	the regression equation (least-squares regression)
c_coefficient	correlation coefficient

Table 3.39: DTN2 log analysis unique contacts statistics table for the combined TX/RX event duration and bytes received of transmitted. For a subset table called *2010s10un* this is called *2010s10un_contacts_uniq*

Class	**Method**
DB	Drop and create database
DB	Drop and create table for set of S10 logs, as described in Table 3.34
Data	Populate table for set of logs with DTN2 logs
DB	Drop and create table for subset of logs as described in Table 3.35
Data	Populate subset table for subset of logs with data for a duration from table with set of logs. This is the working set
Analysis	Update peer column for *DELIVERED* events with correct peer tuple
Data	Create columns for peer name *peername*, peer port *peerport*, peer ID *peerid*, nodename *nodename*, and node ID *nodeid*
Data	Create list of node unique peer contacts
Analysis	Filter out all events but *CONTUP*, *CONTDOWN*, *TX*, *RX*, and *DELIVERED*
Resolve	Resolve peer *IP:PORT* tuple to name and port with logging node id no
Data	Get the set of unique node EIDs and set of unique peer names from the *peername* column
Statistics	Generate statistics and insert contact summary data into *contacts_summary* table (Table 3.37) and detailed statistics into *contacts_detail* table (Table 3.38) with statistics data for unique contacs *contact_uniq* table (Table 3.39)
Contact	Create per unique contact pair views using the methods of the *Contact* class and return detailed statistics using *Statistics() method*. This process is described in Section 4.7.1

Table 3.40: Classes and methods of the *Analysis* module to extract pairwise contacts

3.12 Fault model

As described in Chapter 1, the limitations of an opportunistic contact in the RHNC mean that the information that nodes have about their peers varies significantly over time and space. Consequently, nodes might not have a complete knowledge of their context and might not be able to reach a consensus on how to behave, or even to request information. Thus DTN nodes need to take decisions independently of each other, using only limited information. Charron-Bost and Schiper (2007) claim that inappropriate modelling choices have led to over-complicated approaches to solving consensus, thus yielding too restrictive solutions for real systems.

The models, described in Chapter 3, use the DTN model, described in Section 3.6, for communications. The following describes the relationship between the DTN model and the classical fault model for distributed systems. We are particularly concerned with best effort systems where timing guarantees cannot be assured at all times in a given duration. The degree of synchrony and the failure model are the two independent parameters that determine a particular type of system which we deal with in the following.

Several different assumptions can be made about the degree of synchrony in a system. Synchronism divides distributed systems into two main streams: asynchronous and synchronous. Distributed fault-tolerant real-time system models have been observed to polarise themselves in extreme positions (Veríssimo and Almeida, 1995). Using the synchronous model, components are assumed to take steps simultaneously, using the asynchronous timing model, separate components are assumed to take steps in an arbitrary order, at arbitrary relative speeds, and using the partially synchronous (timing-based) model some restrictions on the relative timing of events (Lynch, 1996).

Veríssimo and Almeida (1995) define synchronism as characterised by bounds and prior knowledge as provided in Table 3.41. The implementation of DTN relies on these assumptions. Our work makes the assumption that both the capacity and delay of an edge are constant over a time interval. When this is not the case, the time interval can be subdivided so that the variable capacity, or delay, can be satisfactorily approximated by a constant in each interval. Our model assumes the availability of global and relative time differences. Therefore the partially synchronous (timing-based) model maps to a DTN. While the timing of events can be efficient, they will not run correctly if the timing assumptions are violated. Hence, TDP algorithms although they are general and portable, the are not guaranteed to run correctly in networks with arbitrary timing guarantees. While partially synchronous systems, in which drastic changes of behaviour of timing-based algorithms can result from small changes in timing assumptions, have less uncertainty than asynchronous systems they are probably more realistic than either completely synchronous or completely asynchronous models (Lynch, 1996). The method of invariant assertions and the method of simulation relations are two important proof techniques adapted by Lynch (1996) for timing-based algorithms.

Bounded and known
Processing speed
Message delivery delays
Local clock rate drift
Load patterns
Difference among local clocks

Table 3.41: Characteristics of synchronism (Veríssimo and Almeida, 1995)

The DTN model is *Clock-driven* as it relies on the availability of global time in which processes have access to clocks whose differences are bounded, such as provided in (Mills, 1992) for example. The DTN model also relies on the availability of relative time differences described by *Timer-driven* protocols (Veríssimo and Almeida, 1995). DTN nodes are assumed to have a basic time synchronisation capability. The DTN model relies on the availability of global time in which processes have access to clocks whose differences are bounded, such as provided in Mills (1992) for example. The DTN architecture depends on time synchronisation among DTN nodes, supported by external, non-DTN protocols, for four primary purposes provided in Table 3.42.

No.	**Purpose**	**Description**
1	Bundle and fragment identification	Bundle identification is supported by a creation timestamp and an explicit expiration field in each bundle. The origination time-stamps on arriving bundles are made available to consumers. Each ADU is required to contain a time-stamp unique to the sender's EID. The EID, time-stamp, and data offset/length information together uniquely identify a bundle.
2	Routing with scheduled or predicted contacts	Along with the originating time-stamp and useful life indicator, a class of service designator, and a length is provided in each bundle. This information provides bundle-layer routing with knowledge of the size and performance requirements of requested data transfers. When there is a significant amount of queueing knowing this information may be significant for making scheduling and path selection decisions
3	Bundle expiration time computations	Bundle expiration is supported by a creation timestamp and an explicit expiration field in each bundle.
4	Application registration expiration	When an application expresses its desire to receive ADUs destined for a particular EID, this registration is only maintained for a finite period of time, and may be specified by the application. For multicast registrations, an application may also specify a time range or *interest interval* for its registration. In this case, traffic sent to the specified EID any time during the specified interval will eventually be delivered to the application (unless such traffic has expired due to the expiration time provided by the application at the source or some other reason prevents such delivery).

Table 3.42: The DTN architecture depends on time synchronisation for these four primary purposes

Various distributed clock synchronisation mechanisms for DTN are proposed in the literature (Ye and Cheng, 2008)(Choi and Shen, 2010). The protocol of Choi and Shen (2010) achieves global clock synchronisation under asynchronous, long delayed, and intermittent network dynamics of DTN by compensating clock errors with contacted nodes using propagated relative clock information and updating stored information according the compensated logical clock information.

Two primary bundle fields *Lifespan* and *Creation Timestamp* relate to time. The *Lifespan* field denotes the time-of-day at which the message is no longer useful. If a bundle is stored in the network when its lifespan is reached, it may be discarded. The lifespan of a bundle is expressed as an offset relative to its creation time. The *Creation Timestamp*, a concatenation of the bundle's creation time and a monotonically increasing sequence number such that the creation timestamp is guaranteed to be unique for each ADU originating from the same source. The creation timestamp is based on the time-of-day an application requested an ADU to be sent.

Failures occur when a producer does not behave as it is specified to and manifest as omission, crash, timimg or response failures (Cristian, 1991). An *omission failure* occurs when a producer omits any output that it must deliver and any state transition that it must undergo. A *crash failure*, in which a producer omits to provide output to all subsequent input post first omision until it restarts, and is the most benign type of process failure in the classical failure classification (Hadzilacos and Toueg, 1993). A *timing failure* occurs when a functionally correct output is received outside of the specified interval. A *response failure* occurs when a producer provides incorrect output to an input

As described in Section 3.6, in order to model link behaviour over time to reflect high variance loss rates and capture the state and the transitions in the network, variables are defined in our DTN model which are used in the linear Programming formulation.

Communication failures can include long propagation delays *timing failure*, packet loss (omission failures) or significantly reordered packets and acknowledgements (response failures). The TDP model assumes all failures. It masks these failures by its push model and TDP ADU lifetime which is the same as the bundle lifetime. Our work assumes the DTN model, but does not tolerate any failures of the DTN model, or any process failures.

3.13 Summary

This chapter presented models to describe a delay tolerant network of associations that are first class abstractions of any-thing, i.e., including any data, process, device or mechanism. We presented Thing identifiers, thing attributes and attribute values as part of a scheme to orgainise DTN endpoint element metadata and to enable the verification of the binding between the thing identifier and an associated thing. A protocol that describes the opportunistic exchange of information was given. The delay tolerant network of associations is scalable when faced with yottabyte of things. The philosophical guidelines "The Simplicity Principle", described by (Bush and Meyer, 2002), to which architects and designers of Internet backbone networks should adhere are considered to provide the simple knowledge construct described.

This chapter described the design of the TDP, a discovery protocol to support the use of the TDP models, and *TDP-module*, a module based on the TDP specification, along with the *TAV-maker* module, enables endpoints to discover themselves that supports the development of applications to translate endpoint elements into TAVs using the TDM. The design of the module, *TDP-decider*, that enables actions to be taken based on the learnt structured information of the TDM was detailed. The design of the modules that build and use ns-3 as a real-time network emulator, which corresponds to the network model and the design of the module used to analyse cases where in the event of a contact, *real*-world data transfer applications that utilise information provided by TDP are enabled to transfer more application data units than applications that utilise information provided by other sources were specified.

A loosely coupled system, applicable in a broad range of contexts which has more flexibility in time constraints, sequencing, and environmental assumptions than a tightly coupled system was presented. The overlay discovery protocol proposed is capable of mapping to underlying discovery mechanisms. Figure 3.21 shows the protocol, models, other modules and their classes.

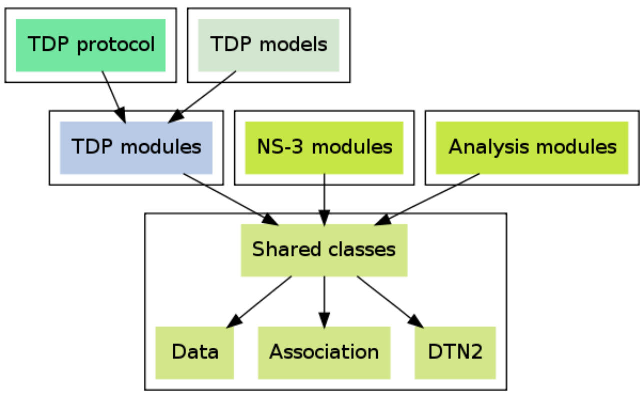


Figure 3.21: The TDP protocol, models, modules and their classes

How endpoint elements of DTN can be exploited in opportunistic contact networking scenarios will be examined in detail in Chapter 4 and within the scope of this thesis.

Chapter 4

Implementation

This chapter describes an implementation of TDP, the TDP models and the TDP modules that we described in Chapter 3. TDP is implemented as the *TDP-module* module that uses the DTN2 API to communicate with the bundle protocol agent. The *TAV-maker* module is an on-demand, scheduled or event-based module to support the use of the TDP and its models. *TAV-maker* includes a generator module that generates TDM structured information, and a module that presents the structured TDM information to the application's storage model layer to persist it to a database. The *TDP-decider* module assists the developers of TDP nodes to specify the application domain of *TDP-module* and its constraints, by providing an application that defines the set of actions that are taken based on the structured information.

This chapter first describes appropriate languages to express building of the system in Section 4.1. Next it details the modified EAV/CR schema used by TDP in Section 4.2. It then describes the architecture of *TDP-module* in Section 4.3. Section 4.4 presents the *TAV-maker* module, the steps to develop classes of thing, as well as the key algorithms used. Section 4.5 presents *TDP-decider* and the key algorithms used. Next, *TDP-module*, *TAV-maker* and *TDP-decider* are implemented in conjunction with the DTN RI and a database to generate, persist, transfer and act upon *TDP-module* reports in Section 4.8. Finally we highlight possible future work for the extension of the tools and provide a summary of the chapter.

4.1 Languages for building information systems

It has been proposed that in comparison with third-generation languages, "Agile languages", such as TCL, Perl, Python, Java Script, Ruby and Visual Basic, provide the appropriate flexibility to express building modern information systems (Wright and Moore, 2006). In an Agile language comparison by Wright and Moore (2006) only TCL and python fulfilled all criteria attributed to it's speed of use, breath of functionality and enterprise usage. While Tcl is a candidate for building our system, it lacks some of the new features present in other languages. In contrast, the fast adoption of new features of python might be attributed to its widespread use. The mature, stable and complete intuitive object model supported by Python makes it a candidate for system building. Python constructs support rapid prototyping and can map easily into modelling languages and design patterns. A large variety of applications written in Python have been successfully demonstrated and widely adopted (Vinoski, 2004). Furthermore, Python is open source, extensible in C and C++, is modular, portable and has clean and readable syntax. While other languages might provide comparatively better performance, it is for the reasons cited above that Python is sufficient for our implementation.

4.1.1 Concurrency

The Python *multiprocessing* API (URL, 72) is used extensively throughout to provide local concurrency using sub-processes in order to fully leverage multiple processors on a given machine. The *Thread* (URL, 73) Python module provides low-level primitives for working with multiple light-weight processes or tasks. The *Thread* module provides synchronisation using mutexes. A higher-level threading API called *Threading* (URL, 74) is used by TDP.

4.1.2 Locking semantics

The Queue class of the Python *Queue* module (URL, 75), used in the threaded programming of TDP, implements the required locking semantics to enable information to be exchanged safely between multiple threads using multi-producer or multi-consumer queues.

4.1.3 Waiting for I/O completion

The Python *Select* module (URL, 76) provides access for the TDP receive class (*tdp_rx* described later) to the Unix operating system *select()* and *poll()* functions.

4.1.4 Process creation

The Python *subprocess* module allows new processes to be spawned, connection to their input/output/error pipes, and provides a way to obtain their return codes. Process creation and management of *subprocess* is handled by the *Popen* class used extensively by *TAV-maker* described in Section 4.4.

4.1.5 Naming model

The Python *uuid* module URL (80) provides the uuid functions and immutable UUID objects (the UUID class) for generating version 1, 3, 4, and 5 UUIDs as specified in RFC 4122 (Leach et al., 2005) and used by the associations model described in Section 3.3.

4.1.6 Data store

The Python *MySQLdb* module (URL, 77) is a interface to MySQL which implements the MySQL C API used by *TAV-maker* to persist TDM data to the MySQL data store and by *TDP-module* described in Section 4.3 to consume and alter TDM data.

4.1.7 Operating system interfaces

The Python *os* (URL, 78) module provides a portable way of using operating system dependent functionality that is used by *TAV-maker*.

4.1.8 String matching

The Python string classes URL (79) support the string-specific method *find()* used by the *TDP-decider* module to return the lowest index in the string where a substring match is found.

4.2 EAV/CR storage model

The EAV/CR of Nadkarni et al. (1999) implementation operates at two levels: the Web server and the Web browser. As the tables in this EAV/CR schema contain the logical schema description and information essential to a Web-based user interface, we alter the schema in several ways to provide a "better fit" to our application domain. The EAV/CR schema of the TDP storage model is depicted in Figure 3.10

4.2.1 Tables

The EAV/CR model contains information orientated around a Web-based user interface. In contrast our application domain has no Web-based user interface and consequently several fields have been removed and the function of several field has been altered. The following describes how the EAV/CR model of Nadkarni et al. (1999) has been altered for our application domain.

Class

In the *Class* table, which is also called *Class* in our schema, the *EAV_Flag* field, used to indicate whether the class is stored in EAV or another form has been removed as all TDM data is stored in EAV form. The EAV/CR *Class Type* which usually indicates whether a class is an "Entity" or an "Association", has been altered to indicate the unique association name i.e, the Thing name denoted *thingName*, thus allowing classes to be searched by Thing name. The *Inline* field, a Boolean usually used to signify whether a class is contained in another class is not used to trigger generation of hyperlink.

A description of the fields of the Class table is provided in Table 4.1

Field	Description	Example
classID	a unique ThingID. This is the 36 character hexadecimal ASCII string representation of a UUID	bb626847-8273-43a7-98a8-a0d7f27c6425
className	The name of the class. Either dev, proc or endpoint	dev
classType	A thing class. An association between class instances is itself treated as a class.	vcsa1
classDescription	A MIME type	application/x-character-device
virtual	If set indcates a class without any actual instances	0

Table 4.1: Class table fields. This class shows a class type *vcsa1* of a */dev* class of thing is provided as an example of a populated table row

ClassHierarchy

The *Class_Hierarchy* table, called *ClassHierarchy* in our schema, also has two fields, a parent class *superClassID* and a child class *subClassID*, both linked to the Classes table. This table is consulted when a query is based on a superclass that might encompass subclasses. A description of the fields of the ClassHierarchy table is provided in Table 4.2

Field	Description
superClassID	Parent class records parent-child relationships between class definitions. It is consulted when a query is based on a superclass
subClassID	Child class records parent-child relationships between class definitions. It is consulted when a query is based on a superclass that encompasses subclasses as well

Table 4.2: ClassHierarchy table fields

Thing

The EAV/CR *Objects* table, called *Thing* table in our schema, records the following data for each object. Although the fields retain the same purpose the field names in our schema differ as provided in Table 4.3.

Field	Description	Example
thingID	A unique ThingID. This is the 36 character hexadecimal ASCII string representation of a UUID	0119bc75-d95d-40b7-9400-36161a708b33
thingName	A unique association name	vcsa1
classID	A unique ThingID. The class to which the thing belongs. This references a Classes metadata table	bb626847-8273-43a7-98a8-a0d7f27c6425
thingDescription	An association type description	dev
thingLastChanged	The time of last change of thing	2013-03-07 13:27:28
thingOwner	A unique ThingID. The creator / owner of the thing	34cc99e4-2b3b-4698-9919-4874257a3725
thingCreated	The time the Thing was created	2013-03-07 13:27:28

Table 4.3: Thing table fields. A class of */dev* thing is provided as an example of a populated table row

Attribute

The most detail of all the metadata tables in EAV/CR is represented by the *Attributes* table, called the *Attribute* table in our implementation. Web-based user interface related fields have been removed and the Web oriented function of several field has been altered to our requirments. The *Width and Height*, *Format*, *URL Template*, *Inline Image*, and *Default File Extension* Web related fields have been removed. The *Defaulted Value*, and consequently the related *Upper Bound* and *Lower Bound* fields, are removed as we do not require default values to be presented to users. The *Attribute_description* field was removed as not required.

A description of the fields of the *Attribute* table is provided in Table 4.4.

4.3 TDP

The *TDP-module* module is a Python implementation of TDP described in Section 3.7 that supports the discovery and dissemination of reports in a DTN composed of autonomous mobile nodes.

TDP is composed of three classes, *tdp_tx*, *tdp_rx* and *tav_getter*, that work as multiple threads of control sharing their global data space. All TDP threads run concurrently.

Field	Description	Example
attributeID	A unique ThingID. This is the 36 character hexadecimal ASCII string representation of a UUID	9b27e5c0-b636-4c7e-925b-ada72c561bb3
classID	points to the class to which the attribute belongs	bb626847-8273-43a7-98a8-a0d7f27c6425
attributeName	Internal Name	path
dataType	indicates which of several TAV table types is used to store the data	dev
attributeClass	Indicates the class of the attribute itself	TAV_str
required	If true, this value must be supplied for a new record	0
searchable	If true, it indicates that a field for this attribute should be included in the search to let the user search for objects within a class on complex Boolean criteria	1
computedFormula	Certain attributes may be computed on the basis of the value of other attributes (if they are non-null). This field holds an expression with placeholders that are replaced by the values of the appropriate attributes during run time	udevadm info -q all -n <name>
virtualAttribute	This is applicable only to attributes with computed formulas. If true, this implies that the computed value is not permanently stored in the database	0
serialNumber	order of presentation	1
multiInstance	This allows an array of objects, if needed, instead of a single object alone. Each object in an array is distinguished by a serial number in the TAV Objects table. Serial numbers are used for ordering the elements during presentation	1
choiceSetID		cd170fe-8803-11e2-a27f-0026b9d41bed

Table 4.4: Attribute table fields containing example data

4.3.1 TDP class: tav_getter

The tav_getter class is organised in four steps corresponding to the retrieval process of stored TDM data. The *tav_getter* class thread executes this sequence periodically (every 30 seconds by default). These steps and their interaction with the data and storage models are detailed below. While the steps must be completed in sequence, the sequence can be altered at any time by going back to the previous step.

In addition, a back-end automates the systematic steps of the process as described in the following sequence :

1. obtain the *ownership* identifier; a thing ID used to establish ownership of the data
2. extract TDM structured information from the storage model into list data type objects for each thing which it has ownership
   (a) list object of things for which it has ownership
   (b) list object of attributes for those things
   (c) list object of values of those thing attributes

   (d) list object of creation sequence of those things (TAV_thing)

   (e) list object of classes of those things

   (f) list object of ClassPresentationInfo of those classes

   (g) list object of ClassHierarchy of those classes

3. Using list data type objects create an associative array of things, a dictionary data structure composed of an unordered set of *key: value* pairs, with the requirement that the keys are unique.

4. Create a compact JSON UTF-8 encoded dictionary object.

5. Place the JSON object on the *transmit* queue called *tx_q*.

4.3.2 TDP class: tdp_tx

The main loop of the tdp_tx class is organised in eight steps corresponding to the transmit process of queued TDM data. The *tdp_tx* class thread executes this sequence when an object is placed on the *transmit* queue called *tx_q*. These steps and their interaction with the data and DTN models are detailed below. The tdp_tx class interacts with the DTN2 BPA called *dtnd* via the DTN2 API (URL, 81). DTN applications interface to DTN dtnd through the application API, which is a simple Sun RPC mechanism. In addition to the basic C language binding for the API (URL, 81), there is also a Python binding that is used by *tdp_tx*. While the steps must be completed in sequence, a TDP node attempts to discover the evolution of a system, such as addition of a new CLA for example, each time a JSON object is retrieved from the transmit queue.

In addition, a back-end automates the systematic steps of the process as described in the following sequence :

1. Open a new connection to the DTN router BPA with the DTN2 API call *dtn_open()* (URL, 82). On success, *dtn_open* initialises the handle parameter as a new handle to the daemon and returns DTN_SUCCESS. On failure, sets handle to NULL and returns a dtn_errno error code.

2. If *dtn_open* returned DTN_SUCCESS start the following transmit queue loop sequence

   (a) Discover the list of CLA interfaces on the system by means of a system call

   (b) Add a DTN2 UDP link of type *ALWAYSON* with port *4556* for each IP broadcast interface to the TDP DTN EID *dtn://tdp_v1.0* using the DTN2 BPA Tcl interface.

   (c) Delete existing routes to the TDP DTN EID using the DTN2 BPA Tcl interface.

   (d) Obtain the set of DTN2 links using the DTN2 BPA Tcl interface

   (e) For each DTN2 link add a copy route, in contrast to a forwarding route as described in Section 2.5.6, to the TDP DTN EID using the DTN2 BPA Tcl interface. These routes mean that any TDP bundles are queued for transmission on every CLA

   (f) Obtain the TDM JSON data objects from the TX queue and persist them to a temporary file: the TDP payload file

(g) Construct a TDP bundle as described in Table 3.18 using the *dtn_send()* (URL, 83) API call

(h) Attempt to transmit the TDP bundle and on success delete the TDP payload file.

4.3.3 TDP class: tdp_rx

The main loop of the tdp_rx class is organised in four steps corresponding to the receive process of bundle TDM data. The *tdp_rx* class thread executes this sequence when the TDP module registers its intent to receive TDP bundles on the TDP DTN EID *dtn://tdp_v1.0.* A subsequent sequence of five steps executes when a properly formatted TDM JSON object that is the ADU of a TDP bundle is receieved on the TDP registration *dtn://tdp_v1.0* to persist the TDM things to storage. These steps and their interaction with the data, storage and DTN models are detailed below. The tdp_rx class interacts with the DTN2 BPA via the DTN2 API described earlier. The steps must be completed in sequence.

In addition, a back-end automates the systematic steps of the process as described in the following sequence :

1. Bind to an existing TDP module registration or if not present create the non expiring TDP module registration for TDP DTN EID *dtn://tdp_v1.0.*

2. Start the TDP receive loop

   (a) *poll()* TDP file descriptors for I/O events using the DTN2 API. The *poll()* begins with a timeout of 5000ms which is adequate for TDP.

   (b) Monitor the TDP file descriptors with *select()*, waiting 10ms until one or more of the file descriptors become *ready* for some class of I/O operation

   (c) When file descriptors become *ready* use DTN API call *dtn_recv* (URL, 84) to receive the bundle and process the primary block and payload block. While 10ms is adequate for TDP bundles, supplementary DTN protocol specifications (including, BSP) may require that additional measures be taken at this point of the procedures. The TDP payload ADU is placed at a specified location.

   (d) process the TDP JSON formatted payload and persists the TDM data to storage.

      i. open the TDP payload file

      ii. extract the JSON object

      iii. close the payload file

      iv. delete the payload file

      v. iterate through JSON formatted TDM associative array structure to extract the unordered set of *key: value* pairs and persist them to the data store.

4.4 TAV-maker

TAV-maker (for thing naming, data and storage modelling) is a Python tool supporting the development of modules to translate endpoint elements into TAVs using the TDM. *TAV-maker* includes a generator module that generates TAV structured information, and a module that presents the structured TDP information to the module's storage model layer, that is described in Section 3.5, to persist it to a database. TAV-maker has a file based user interface, that enables specification by developers through the process of defining an class of Thing for use by TDP.

In addition, a back-end automates the systematic steps of the process as described in Table 4.5, and uses the models derived in Chapter 3 to automatically translate endpoint element class specifications into requirements on discovery. As a one off preliminary step of this automated process, by default the required TAV tables of the storage model, described in Section 3.5, are dropped and then created by the back-end on initial execution. This is because *TAV-maker* is designed to run continuously.

No.	**Step**	**Description**
1	Class information extraction and parsing	Getting thing class information required for data model of Section 3.4
2	Class information structuring	Structuring and naming thing class as per data model of Section 3.4 and association model of Section 3.3
3	Populate, modify or delete class information	Persisting structured thing class as per storage model of Section 3.5
4	Class information relationship tables update	Persisting structured thing class relationships as per storage model of Section 3.5

Table 4.5: TAV-maker steps generically applicable to Thing Classes taken subsequent to creation of required TAV tables of the storage model described in Section 3.5 by a DB class

4.4.1 Extraction and parsing

In order to map information from other discovery protocols and applications or extract information from other sources methods have been defined in a shared Python class *Data* to extract, parse and structure endpoint element information.

Information is extracted into associative arrays, which are then persisted the tables of MySQL database that contain the EAV/CR schema. Duplicate definitions of classes are automatically removed.

4.4.2 TAV persistence

Once information has been translated into TDM data, a method of the *Data* Python Class is used to persist it to the MySQL data store. A method *get_thing_list()*, which takes a list as input, is used to process list data. A method *get_thing_file()*, which takes a ASCII file as input, is used to process file data. Alternatively specific methods can be specified to extract, parse, translate and persist data to the data store. Such a method is exemplified by *get_devices()* of the *Data* class, which has been implemented to use an application *udevadm* to extract, parse, translate and persist */dev* virtual data to the data store. The *udevadm* application controls the run-time behaviour of udev, requests kernel events, manages the event queue, and provides simple debugging mechanisms

4.4.3 Association class

The implemented methods of the *Association* class used by the *TDP-module* and *TAV-maker* modules are detailed in Table 4.6.

Method	Description
create_ElGamal_keys()	create keys (ElGamal)
sign_thingID()	sign a thing or thingID
verify_signed_thingID()	verify a given a thing or thingID by its signature
create_thingID()	create a unique version 5 UUID thingID from a unique association name
integrity_check_thingID()	check integrity of a named association (i.e. of the unique thing name and thingID)

Table 4.6: Methods of the *Association* class

Data class: *get_thing_list()*

As described in in Section 2.7.1, a network element can either be an abstraction or be physical. The *get_thing_list()* method is used to process abstraction list data. The *get_thing_list()* method takes a class (e.g., the *endpoint* class for DTN2 things), an associative array (e.g., a Python list), a class type (the type of class data e.g., endpoint *bundle stats*), the TAV type (e.g., *TAV_str* to indicate a string), a class type name (e.g., *bundle* to indicate a bundle), a class description (e.g., *bundle statistics* for a succinct description) and an owner ID to establish ownership (e.g., a thing ID) as input.

1. translates input to TDM using *e_thing()* method
    (a) generate a version 4 UUID to assign as a thingID
    (b) establish associative array type (i.e., list, string or tuple)
    (c) establish whether class exists. If class exists use classID, otherwise create a new classID and populate class.
    (d) create thing creation timestamp
    (e) return a thing list
2. persist the thing list to the data store Thing table using the *insert_Thing()* method
3. iterate through associative array to extract attributes
    (a) check type of object
    (b) iterate through sub objects and translate those into association list using type specific parser
    (c) enumerate sub objects
    (d) check if attribute exists
    (e) persist the attribute to the data store attribute table
    (f) check if TAV exists
    (g) persist the TAV to the data store specific TAV type (e.g., TAV_str) table

(h) check if TAV thing exists

(i) persist the TAV thing to the data store TAV_thing table

Data class: get_thing_file()

The *get_thing_file()* method is used to process a physical type representation; a file. The *get_thing_file()* method takes a class (e.g., *proc* to indicate /proc), a filename string, the TAV type (e.g., *TAV_str* to indicate a string) and an owner ID to establish ownership (e.g., a thing ID) as input. This method requires less input than the *get_thing_list()* method as required information is obtained directly from the representation.

1. translates input to TDM using the *h_thing()* method
    (a) open the file
    (b) read entire file
    (c) format data as list
    (d) generate a version 4 UUID to assign as a thingID
    (e) establish file MIME type (e.g., "application/x-directory")
    (f) establish whether class exists. If class exists use classID, otherwise create a new classID and populate class.
    (g) create thing creation timestamp
    (h) return a thing list
2. persist the thing list to the data store Thing table using the *insert_Thing()* method
3. iterate through associative array to extract attributes
    (a) check type of object
    (b) iterate through sub objects and translate those into association list using type specific parser
    (c) enumerate sub objects
    (d) check if attribute exist
    (e) persist the attribute to the data store attribute table
    (f) check if TAV exists
    (g) persist the TAV to the data store specific TAV type (e.g., TAV_str) table
    (h) check if TAV thing exists
    (i) persist the TAV thing to the data store TAV_thing table

Data class: get_devices

The *get_devices* class is used to process abstraction type operating system query data. The *get_thing_dev()* method takes only a class (e.g., *dev* to indicate /dev), and an owner ID to establish ownership (e.g., a thing ID) as input. This method requires less input than the *get_thing_list()* method as required information is obtained directly from the detailed, generic query response.

1. set environment paths for udevadm application
2. create a list of the set of all device names
3. format and sort the device name list
4. iterate through list and for each device
    (a) translate input to TDM using the *h_thing()* method
        i. generate a version 4 UUID to assign as a thingID
        ii. establish associative array type (i.e., list, string or tuple)
        iii. establish whether class exists. If class exists use classID, otherwise create a new classID and populate class.
        iv. create thing creation timestamp
        v. return a thing list
    (b) persist the thing list to the data store Thing table using the *insert_Thing()* method
    (c) create associative array for each device by query of all information of that device name
    (d) iterate through associative array to extract attributes
        i. check name of association object
        ii. iterate through sub objects and translate those into association list using type specific parser
        iii. enumerate sub objects
        iv. check if attribute exist
        v. persist the attribute to the data store attribute table
        vi. check if reciprocal attribute ID exists
        vii. persist the reciprocal attribute ID to the data store ReciprocalAttributesID table
        viii. check if reciprocal attribute exists
        ix. persist the reciprocal attribute to the data store ReciprocalAttributes table
        x. check if TAV exists
        xi. persist the TAV to the data store specific TAV type (e.g., TAV_str) table

xii. check if TAV thing exists

xiii. persist the TAV thing to the data store TAV_thing table

4.4.4 TAV Classes

Three Python classes are implemented for the generation of TDM structured data for three classes of TAV: *tav_maker_endpoint* for BPA specific elements, *tav_maker_proc* for */proc* elements and *tav_maker_dev* for */dev* elements. The reason for implementing these specific classes is three fold. First they collectively provide a set of endpoint elements which can be used to exploit a contact. Second they demonstrate the steps to translate physical and abstraction endpoint elements into TDM data. Last they provide template structures to develop new translators.

On execution each *TAV-maker* class spawns a process using the python *multiprocessing* API that it executed periodically as specified in the file based user interface. All three *TAV-maker* class processes run concurrently and are detailed in the following.

TAV class: tav_maker_endpoint

The *tav_maker_endpoint* class uses the *DTN2* class to query the DTN2 BPA using its Tcl interface. The *tav_maker_endpoint* class's main method is structured in three steps corresponding to the development process. These steps and their interaction with the naming, data and storage models are detailed, and automated by a back-end that spawns a processes, as described in the following sequence below.

1. Extract own unique EID using *uniq_EID()* method
2. Use methods of the *DTN2* class to query the DTN2 BPA, and the *Data* class *get_thing_list()* method described in Section 4.4.3 to obtain, translate and persist information, the BPA is capable of providing, as TDM things for the following types:
   (a) Get a list using the *get_discovery_list()* method and *get_thing_list()* method to obtain, translate and persist to data store.

   (b) Get bundles using the *get_parsed_bundle_dump_tcl* method and *get_thing_list()* method to obtain, translate and persist to data store.

   (c) Get bundle statistics using the *get_bundle_stats()* method and *get_thing_list()* method to obtain, translate and persist to data store.

   (d) Get bundle daemon statistics using the *get_bundle_daemon_stats()* method and *get_thing_list()* method to obtain, translate and persist to data store.

   (e) Get registrations using the *get_registration_list()* method and *get_thing_list()* method to obtain, translate and persist to data store.

   (f) Get interface using the *get_interface_list()* method and *get_thing_list()* method to obtain, translate and persist to data store.

   (g) Get link using the *get_link_list()* method and *get_thing_list()* method to obtain, translate and persist to data store.

(h) Get link statistics using the *get_link_stat()* method and *get_thing_list()* method to obtain, translate and persist to data store.

(i) Get routes using the *get_route_dump_tcl()* method and *get_thing_list()* method to obtain, translate and persist to data store.

(j) Get DTLSR route graph using the *get_DTLSR_routing_graph()* method and *get_thing_list()* method to obtain, translate and persist to data store.

TAV class: tav_maker_proc

The *tav_maker_proc* method uses the *Data* class *get_thing_file()* method described in Section 4.4.3 to obtain, translate and persist */proc* file information as TDM things. Methods of the *Data* class are used to extract and classify information, such as the method *get_recursive_file_list()* used to obtain a list of files through recursive iteration. The *tav_maker_proc* class's main method is structured in three steps corresponding to the development process. These steps and their interaction with the naming, data and storage models are detailed and automated by a back-end that spawns a process as described in the following sequence below.

1. Obtain */proc* location
2. Use methods of the *DTN2()* class to query the DTN2 BPA, and the *Data()* class *get_thing_file()* method to obtain, translate and persist */proc* file information as TDM things in the following steps

   (a) Create list of */proc* file system files (no recursion) using the *get_recursive_file_list_limited()* method

   (b) If there is a set of *proc* elements we are not creating things for, we filter these unwanted items from a *proc* file list

   (c) Check file list contains ASCII files

   (d) Start loop to iterate through */proc* objects

      i. populate tables with */proc* file contents using the *get_thing_file()* method to obtain, translate and persist to data store.

   (e) Create list of directories of */proc* file system (no recursion) using *get_recursive_dir_list_limited()*

   (f) Create list of PID directories from directory list using *is_number()* method

   (g) Filter unwanted *proc* PID directory list item from list

   (h) Start loop to populate tables with files of */proc* directories (non recursive)

      i. populate tables with */proc* file contents using the *get_thing_file()* method to obtain, translate and persist to data store.

TAV class: tav_maker_dev

The *tav_maker_dev* class uses the *Data* class *get_dev()* method described in Section 4.4.3 to obtain, translate and persist virtual */dev* information as TDM things.

4.5 TDP-decider

The *decider* class of the Python *tdp_decider* module uses *Data* class methods to work with TDM data and *DTN2* class methods to query and alter the DTN2 BPA using its Tcl interface. The methods of the *tdp_decider* class are used to obtain the values, attributes and relationships of a set of things that are associated with an ownership identifier.

4.5.1 Logical decision variables

In order to take logical decisions, TDM data is extracted and translated into variables. Depending on the variables' values, logical decisions are taken on a set of possible actions. In our implementation the *proc* and *dev* TDM data is extracted and processed in a different way than the *endpoint* TDM data. While *endpoint* TDM data could be extracted and processed in the same way as *proc* and *dev* TDM data the reason for this difference is two-fold. First two different methods of working with the TDM data are described, i.e., selection of everything of a TDM class as demonstrated with *endpoint* TDM and selection of anything of a TDM class as demonstrated with *proc* and *dev*, to demonstrate how TDM data can be extracted. Second the flexibility of the TDM data is demonstrated by obtaining the data via different axes i.e., by leveraging the unique association name and the association class type description to obtain TDM data. These methods are outlined in the next two sections, we then detail the methods used to effect a decision, and finally provide a description of how logical decision variables are used to make decisions.

Variables for proc and dev TAVs

The *tdp_decider* class's main method for working with *proc* and *dev* TDM data is organised in two steps. These steps and their interaction with the data and storage models are described in the following sequence:

1. The *get_dtnowner()* method of the *tdp_decider* class is used by the node to establish the ID of itself which it uses to establish ownership of the data
2. Obtain a certain set of thing IDs by their unique association name (contained in the *thingName* field of *Thing* table described in Section 4.2.1) using the *get_valuelist_thingName()* method, or the set of thing IDs by their values using the *get_thinglist_tav_value()* method of the *tdp_decider* class
   (a) Using *Data* class parsing methods, such as the *get_stat()*, *get_cpuinfo()* and *get_loadavg()* methods for example, translate the TDM TAV data for each thing ID to variables with which logical decisions on more than one choice of actions to be taken depending on the variables' values are made.

Variables for endpoint TAVs

The *tdp_decider* class's main method for working with *endpoint* TDM data is structured in three steps. These steps and their interaction with the data and storage models are described in the following sequence:

1. The *get_dtnowner()* method of the *tdp_decider* class is used by the node to establish the ID of itself which it uses to establish ownership of the data

2. Obtain a certain set of distinct class names (contained in the *classNames* field of the *Class* table described in Section 4.2.1) for which things exist (i.e. have a valid identifier) by their association class type description (i.e. *endpoint*, *dev* or *proc* as contained in the *thingDescription* field of *Thing* table described in Section 4.2.1) using the *get_distinct_classNames_by_thingDescription()* method of the *tdp_decider* class

3. For each distinct class name of the set:

   (a) Obtain the set of thing IDs for the distinct class name using the *get_thinglist_className()* method

   (b) Obtain *Thing* TDM TAV data for each thing ID of the set of each thing ID from the previous step using the *get_tav_thingID()* method

   (c) Translate the TDM TAV data for each thing ID to variables with which logical decisions on more than one choice of actions to be taken depending on the variables' values are made.

4.5.2 Actioning of logical decision

The *tdp_decider* methods for working with TDP, that are used to effect decisions based on decision variables extracted from TDM, use *DTN2* class methods. These *DTN2* class methods are used to query and alter the DTN2 BPA via its Tcl interface. Commands are dispatched to the DTN2 BPA Tcl interface via a queue. The queue is implemented as a Python queue mechanism described in Section 4.1.2. Several *tdp_decider* methods can be effected, that utilise the *DTN2* class methods to alter the DTN BPA, for example to add a link, delete a route etc.., are defined. When effected the *tdp_decider* methods described put a specific DTN2 BPA command on a non blocking queue that is ultimately executed by the DTN2 BPA.

4.5.3 Logical decision

The *tdp_decider* methods to action logical decisions, i.e. put a specific DTN2 BPA command on a non blocking queue that is ultimately executed by the DTN2 BPA, can be effected when a condition is met. Conditions and actions can be based on any TDM data. Conditions are based on the values of logical decision variables extracted from TDM. As a simple example, in the event of receiving a TDP report from a TDP node that informs of two CLAs with different capabilities, a receiving node can add a route via a suitable common CLA.

4.6 Ns-3 modules

Two Python ns-3 module are implemented: *ns3_configer* and the *ns3_lxc_configer* module.

4.6.1 ns3_lxc_configer module

A module *ns3_lxc_configer* is implemented to generate ns-3 native configuration modules: a ns-3 config, a start up, and a shut down module.

ns-3 config module

The implemented class and functions of the ns-3 native configuration module generated by the *ns3_lxc_configer* module are detailed in Table 3.32. An example ns-3 config module created is *dtn-thing-lxc-cmsa-wifi-1.01*, that uses the *emuModel*. The implemented methods of the network model emulation *emuModel* class to be generated by the *ns3_lxc_configer* module are specified in Table 3.33.

Start and shtdown modules

The start up and shut down modules, *startup* and *shutdown* respectively, are implemented to initialise and teardown Tun/Tap devices, the *lxc* components and the emulation itself.

4.6.2 ns3_configer module

The methods of the *ns3_configer* module implemented are described in Table 3.31. A class *ns3configer* with methods to work to build ns-3 as a real-time network emulator, create *lxc*, install *TDP-module*, *TDP-decider* and *TAV-maker* modules and related software, and configure the *lxc* node is specified. A class *emuModel* with methods to use ns-3 as a real-time network emulator to model a DTN contact, and the events that occur within its duration, that corresponds to the network model using real DTN2 contact data is specified.

4.7 Analysis module

The Python *Analysis* module uses the *DB* class to interact with the data store, the *Data* class to work with data, the *Resolve* class to resolve EIDs, hostnames, addresses and ports, the *Analysis* class to filter and analyse contacts events, and *Stats* class to provide statistical analysis of contacts.

The *Analysis* module automates the processing of the DTN2 log data in order to provide transparent treatment of the data. This process involves the creation of a data store table to persist the full data set described in Table 3.34. Once persisted, a table is created for a subset of the data as described in Table 3.35. For each subset table, a certain set of transformations are carried out and three summary tables are created to provide a summary for links, a summary for each contact extracted from those links and a statistical summary of the bytes transferred for each contact. Contacts are enumerated and a table view, with fields as described in Table 3.36, is created to represent each unique contact, which corresponds to a contact of our network model described in Section 3.6. How each table view corresponds to a contact of our network model is described (and represented as SQL statements) in Table 4.7. In order to extract pairwise contacts, methods and algorithms are described to extract unique peer tuples into table views in Section 4.7.1

Definition	**Description**	**Representation**
t	*timetINusec* as time_t epoch relative time.	SET @csum := 0; SELECT (@csum := (@csum - (SELECT timetINusec FROM ‘c2a31a07-1206-4846-9e0e-730bf7705390‘ ORDER BY timetINusec LIMIT 1)) + timetINusec) AS cumulative_sum FROM ‘c2a31a07-1206-4846-9e0e-730bf7705390‘ WHERE event = ’CONTUP’ OR event = ’TX’ OR event = ’RX’;
V	a set of vertices	SELECT DISTINCT con_id as V FROM ‘c2a31a07-1206-4846-9e0e-730bf7705390‘;
E	a set of edges	SELECT eventid AS E FROM ‘c2a31a07-1206-4846-9e0e-730bf7705390‘ WHERE event = ’CONTUP’ OR event = ’TX’ OR event = ’RX’ OR event = ’CONTDOWN’;
$i : E \rightarrow V$	assigning to each edge its source node	SELECT nodeid AS i, eventid AS E, con_id AS V FROM ‘c2a31a07-1206-4846-9e0e-730bf7705390‘ WHERE event = ’TX’;
$d : E \rightarrow V$	assigning to each edge its target node	SELECT nodeid AS d, eventid AS E, con_id AS V FROM ‘c2a31a07-1206-4846-9e0e-730bf7705390‘ WHERE event = ’RX’;
b_v	is the storage capacity of the *lxc* TDP node v	the size in MB of *lxc* filesystem storage $b\_v = 1024$
b_v,t	is the storage capacity of the *lxc* TDP node v at time t.	SET @csum := 0; SET @vsum :=0; SELECT rfc3339, (@csum := @csum + bytes) AS cumulative_sum, (@vsum := 1073741824 - @csum) AS ”c_e,t” FROM ‘c2a31a07-1206-4846-9e0e-730bf7705390‘ WHERE event = ’TX’ OR event = ’RX’;
m_v	is the memory capacity of the *lxc* TDP node v	the size in Mega Bytes of RAM available $m\_v = 1024$
p_v	is the CPU capacity of the *lxc* TDP node	this is the number of CPUs assigned to a *lxc* node (output of *lxc-cgroup -n <NAME> cpuset.cpus*) v $p\_v = All$
d_e,t	is the propagation delay of the edge e at time t	
r_e,t	is the range of the edge e at time t	assumed to be constant
I^v	is the set of edges whose destination node is v (incoming edges)	SELECT * FROM ‘c2a31a07-1206-4846-9e0e-730bf7705390‘ WHERE event = ’RX’;
O^v	is the set of edges whose source node is v (outgoing edges)	SELECT * FROM ‘c2a31a07-1206-4846-9e0e-730bf7705390‘ WHERE event = ’TX’;

Table 4.7: Contact parameters of contactID c2a31a07-1206-4846-9e0e-730bf7705390

Three summary tables are created. A table to summarise the contact statistics for each link is described in Table 3.37. A *contact_detail* table is created to provide detailed information for each contact extracted as described in Table 3.38. A *contact_uniq* table is created to provide statistics, using the combined TX/RX event duration and bytes received or transmitted, for the subset of contacts extracted as described in Table 3.39.

A phased approach is taken in order to extract data. The treatment of that data is detailed in Section 4.7.1, an output of which is a set of enumerated contacts, each profiled with parameters that corresponds to a contact of our network model. The processes used to extract contact representation tables are detailed in Section 4.7.1. This phased approach is applied to the data of the N4C project in Section 5.5, to provide an overview analysis and set of enumerated contacts and table views, called the *test contacts*, which are subsequently analysed to baseline performance and gauge the exploitation of that set of pairwise contacts. The baseline performance is parametrised by the contact duration, unused duration, bundles transferred, unique bundles transferred, total bytes transferred and mean bytes transferred.

Each table view of the *test contacts* set, is then used to emulate each contact by the Python ns-3 modules *ns3_configer* and *ns3_lxc_configer*, whose design is detailed in Section 3.10 and which were developed in this thesis to use of ns-3 as a real-time network emulator.

The phased approach taken in order to extract, filter, create suitable objects processing, analyse, and provide an appropriate contact selection for evaluation, is described in the following:

4.7.1 Phase 1: extraction

The high level analysis steps used to extract pairwise contact from DTN2 log files into data store table views are detailed in Table 4.8.

No.	Class	Step description
1	DB	Drop and create database
2	DB	Drop and create table for set of logs, as described in Table 3.34
3	Data	Populate table for set of logs with DTN2 logs
4	DB	Drop and create table for subset of logs as described in Table 3.35
5	Data	Populate subset table for subset of logs with data for a duration from table with set of logs. This is the working set
6	Analysis	Update peer column for *DELIVERED* events with correct peer tuple
7	Data	Create columns for peer name *peername*, peer port *peerport*, peer ID *peerid*, nodename *nodename*, and node ID *nodeid*
8	Data	Create list of node unique peer contacts
9	Analysis	Filter out all events but *CONTUP*, *CONTDOWN*, *TX*, *RX*, and *DELIVERED*
10	Resolver	Resolve peer *IP:PORT* tuple to name and port with logging node id no
11	Data	Get the set of unique node EIDs and set of unique peer names from the *peername* column
12	Statistics	Generate statistics with Statistics class and insert contact summary data into *contacts_summary* table and detailed statistics into *contacts_detail* table with statistics data for unique contacs *contact_uniq* table
13	Contact	Create per unique contact pair views using the methods of the *Contact* class and return detailed statistics using *Statistics* class. This process is described in Section 4.7.1

Table 4.8: DTN2 log analysis steps to extract pairwise contacts. The class for each step is also specified.

Contact Views

Extraction of pairwise contacts is the final step of the DTN2 log analysis steps outlined in Table 4.8. A detailed description of the methods and algorithms involved is provided in the following. A subset table of the data is extracted as a step of the process listed in Table 4.8, along with a list of unique peer host names (*peerlist*), and list of unique logging node EIDs *nodelist.* The *nodelist* is the set of unique EIDs for associated with *CONTUP*, *CONTDOWN*, *TX*, *RX* and *DELIVERED* events. Using this data, the *extract_peers* method the class *Contact* class is used to extract unique peer tuples into MySQL table views.

The processes of the *extract_peers* method involved are detailed in the following:

1. For each unique peer host name of *peerlist*
   (a) For each unique EID of *nodelist*
      i. Get the set of unique peer IP:PORT tuples
      ii. For each unique IP:PORT tuple, i.e. for each unique link
         A. Use the *create_peers_view()* method to create a time ordered table view of the link with name *node_peer* of a (udp or tcp cla) contact pair based on detination IP:PORT tuple
         B. Create a count of *CONTUP*, *CONTDOWN*, *TX*, *RX* and *DELIVERED* events
         C. Create a count of unique bundles and non-unique bundles
         D. Create a list object of the table view
         E. Use the *throughput()* method, described in Table 4.9, to extract data and populate the *contacts_detail* table described in Table 3.38
         F. Calculate descriptive statistics using the *calcstats()* method of the *Statistics* class.
         G. Append the summary values (all values) to an array *all_summary*
2. Return the an array *all_summary*

No.	Step description
1	Calculate the start of the table *t_sot* as the time stamp *timetINusec* of the first record, the end of the table *t_eot* as the time stamp *timetINusec* of the last record (if it exists) and the table duration as $t\_eot - t\_sot$
2	initialise a counter *i* as the event ID, *eventid*, and iterate through lines of the pair list (i.e. the records of the pair table)
3	Set the contact ID *con_id* value to "NEW" to indicate this is a new contact
4	Based on the value of *event* field, extract and generate fields that correspond to the fields of a record of the *contacts_detail* table, described in Table 3.38), and append this record to an array *contact_summary*
5	On completion of iteration insert records into the *contact_summary* table and return summary for transferred bundle size, throughput, and the table duration

Table 4.9: The *throughput()* method to extract data and populate the *contacts_detail* table

4.7.2 Phase 2: filtering and processing

The steps used filter pairwise contacts extracted in *Phase 1*, and N-dimensional array objects are created for processing as detailed in Table 4.10.

4.7.3 Phase 3: analysis

In this phase the N-dimensional array objects are analysed to provide a contacts overview and profile of specific links. The steps used to profile the selected contacts of *Phase 2* are outlined in the following.

No.	Step description
1	get the set *selected_links* of link tables from table *2010s10un_contacts_summary*, that fulfil the selection criterion. The selection criterion is fulfiled in cases where there are the same number of CONTUP and CONTDOWN events, there is 1 or more CONTUP events, and there is 1 or more TX/RX events.
2	get the set *selected_contactids* of distinct contact IDs their duration and unused duration from the set of link tables that fulfil a selection criterion using table *2010s10un_contacts_detail*. The selection criterion is fulfiled in cases where the contact duration is greater than the unused contact duration.
3	get the set of specific interesting links. For example the set *max_id* of statistics from *2010s10un_contacts_summary* table for the specific the link on which most bundles were transferred
4	get the set *con_ids* of distinct contact IDs *selected_contactids*, their duration and unused duration from a specific link (e.g. link which most bundles were transferred), that fulfil a selection criterion using table *2010s10un_contacts_detail*. The selection criterion is fulfiled in cases where the contact duration is greater than zero
5	get the set *con_det* of *contacts_detail* fields for contacts of a specific link (e.g. link which most bundles were transferred)
6	get the set *distinct_con_ids* of distinct contact IDs from *contact_detail* that fulfil a selection criterion. The selection criterion is fulfiled in cases where the contact duration is greater than zero
7	get the set *all_cdetail* of *contacts_detail* records
8	create *numpy* (URL, 85) N-dimensional array objects for *max_id*, *con_ids*, *con_det*, *distinct_con_ids*, *all_cdetail*, *selected_links* and *selected_contactids*

Table 4.10: Phase 2 steps to create N-dimensional array objects for analysis of sets of contact IDs that fulfil a selection criterion

Contacts overview analysis

The steps to provide a contacts overview are detailed in Table 4.11.

No.	Step description
1	get total of unique contacts that fulfil a selection criterion. The selection criterion is fulfiled in cases where the contact duration is greater than zero
2	get number of events logged

Table 4.11: Phase 3 steps to provide a contacts overview

Profile specific links

The steps to profile selected links with interesting characteristics are detailed in Table 4.12. An example of a link with interesting characteristics is the link which had contacts that transferred the most bundles.

4.7.4 Phase 4: selection for evaluation

A criteria that contacts must fulfil is derived in order to enable selection of a set of *valid* contacts. In order to describe and evaluate the set of contacts, a subset of the contact population which meet a criteria is established. Next a sample size is calculated to enable creation of a set of contacts through random selection from the set of contacts. This is the set of contacts used to evaluate TDP using the

No.	Step description
3	get the maximum number of bundles transmitted or received over a link in the period
4	get the number of unique bundles involved
5	get the number of RX and TX events
6	get the number of RX bundles delivered to valid registration
7	get link initiator source EID, CLA, next hop port of peer and peer hostname
8	get number of TX/RX events that occurred within number of contact up and contact down events
9	get link duration elements: start date, end date and duration in seconds and hours

Table 4.12: Phase 3 steps to profile specific links in order to provide a summary analysis

TDP-module, *TDP-decider* and *TAV-maker* modules, by way of a discrete-event network simulator that is used as a real-time network emulator, interconnected with the *real* world in order to allow use of appropriate existing *real*-world protocol implementations.

4.8 TDP and related module integration

The *TDP-module*, *TAV-maker* and *TDP-decider* modules work independently of one another. The purpose of *TAV-maker* is to provide a means for endpoints to discover themselves by structuring data corresponding to TDM that is named by an association model, and persist that data structure to a data store corresponding to EAV/CR. The purpose of *TDP-module* is to provide a means to exchange generically formatted interchange TDM data in the RHNC, process structured data corresponding to TDM that is named by an association model, and persist that data structure to a data store corresponding to EAV/CR. The purpose of *TAV-decider* is to demonstrate the need for discovery mechanisms, such as *TDP-module* and *TAV-maker*, that can be used by nodes to exploit an opportunistic contact in the RHNC to deliver messages despite the limitations of such an opportunistic contact.

4.9 Achievements and future work

TDP and its related models support the discovery of radically heterogeneous network elements. An ultimate goal of this investigation is to make a better Internet through eventual incorporation of TDP and related models features and capabilities into the standard suite of Internet protocols. TDP and its models support decision making on on how to exploit an indeterminate communications opportunity within the RHNC and a mechanism to determine how those decisions relate to other protocol aspects. The development process allows individual elements of networks to be mapped to the TDM. If these requirements are met, exchange of information by the protocol will be ensured. TDP and its related models have been designed to support the radical heterogeneity of the RHNC, not just networks with moderate to well performing links. That is, the constructs defined facilitate reasoning about opportunistic contacts of physical nodes within heterogeneous networks and would not be efficient for generic networks with moderate to well performing links, and consequently for underlay discovery mechanisms. Ultimately, TDP and its related models support exploitation of an opportunistic contact in the RHNC to deliver messages.

The TDP models can be used in other non DTN contexts than described and demonstrate architectural constraints that, when applied as a whole, emphasise scalability of loosely coupled systems. For example, if applied to phases of the "e-discovery" (URL, 86) to enable organisations comply with rules concerning how to identify electronically stored information *ESI*. While the thing model can be used to design things for many contexts, interaction with the things are assumed within the described context and their interactions with other context are not covered by this work.

An area that could be improved, however, is the selection of generic tools used for persisting and querying data. A data structure server, such as "Redis" (URL, 87), could be used to build a complete system for TDM to store things as scalars, lists, hashes and sets. Association list are inherent in RedisâĂŹ five data structures which each have at least a key and a value where values represent the actual data associated with the key, which can be anything. Furthermore an indexing engine such as Lucene (URL, 88) could be used to index TDM data. Another area that could be improved is the representation of relationships. For example, algorithms could be specified to mine TDM data and establish relationships automatically, which could provide decision support for appropriate endpoint selection. Additionally, the authenticity, validation and encoders association mechanisms of the association model, described in Section 2.3.2 and Table 3.3, could be extended using existing full Python bindings for GSSAPI and the Python wrapper for OpenSSL, *M2Crypto* (URL, 89).

As a possible extension to TDM, a class could be specified to transform information of the contact analysis module into TDM data in order to use historic information to base decisions. Finally, formal definitions of TDP and its related models and formal proofs of their correctness are desirable in order to provide the maximal freedom for implementations while preserving a globally coherent notion of meaning.

4.10 Summary

This chapter first presented Python as an appropriate language to express building of the systems. It decribed the modified EAV/CR schema used by TDP and then described the architecture of *TDP-module* in Section 4.3. It then presented *TDP-module*,*TAV-maker*, and *TDP-decider*, described how they map to the protocols and models of Chapter 3, their development steps, as well as the key algorithms used. We then described how *TDP-module*, *TAV-maker* and *TDP-decider* are used in conjunction with the DTN RI and a database to generate, persist, transfer and act upon *TDP-module* reports. Next we described the integration of *TDP-module*,*TAV-maker*, and *TDP-decider*. Finally, this chapter detailed the achievements of TDP and its related models and implemented modules and highlighted possible future work for the extension of the tools.

Chapter 5

Evaluation and results

This chapter describes the evaluation of TDP using the modules detailed in Chapter 3 and subsequently in Chapter 4. The hypothesis on which TDP builds is evaluated using the the *TDP-module*, *TAV-maker*, and *TDP-decider* modules, described respectively in Sections 4.3, 4.4, and 4.5, by way of the ns-3 modules, *ns3_configer* and *ns3_lxc_configer*, described in Section 4.6, and the automated *Analysis* module for analysis and evaluation, described in Section 4.7. The *ns3_configer* and *ns3_lxc_configer* modules use the ns-3 discrete-event network simulator as a real-time network emulator, to enable ns-3 to be interconnected with the real-world, allowing use of appropriate existing real-world protocol implementations. The *Analysis* module takes the DTN2 logs of an arbitrary number of nodes, from which it extracts the pairwise contacts, provides a comprehensive statistical analysis to predict the contact volume, and persists same to a data store. The *Analysis* module includes mechanisms to randomly select, extract, analyse, generate an analysis summary of contacts and present a summary of the overall analysis. Additionally, the *Analysis* module provides methods to select an appropriate set of contacts from the data to be used as an input to the real-time network emulator in order to evaluate the number of ADUs transferred during those contacts.

A phased approach taken in order to extract data. The treatment of that data is detailed in Section 4.7.1, an output of which is a set of enumerated contacts, each profiled with parameters that corresponds to a contact of our network model.

This chapter first presents the characteristics of the work that are evaluated as well as the evaluation strategy. The 2009, 2010 and 2011 N4C trial data is described in Section 5.2. Next, the treatment of the data is detailed in Section 5.3. In Section 5.4 we describe how each contact is modelled and used to derive an estimation of the contact capacity volume lower and higher bounds. Along with the contact model data these bounds are used emulate each contact in real-time by using the *ns3_configer* and *ns3_lxc_configer* modules to implement ns-3 as a real-time network emulator. The experimental configuration is detailed in Section 5.6. The phased approach is applied to the data of the N4C project in Section 5.5, to provide a comprehensive analysis, set of enumerated contacts and table views. The set of enumerated contacts are called the *test contacts*, which are subsequently analysed to baseline performance and gauge the exploitation of that set of pairwise contacts. The baseline performance is parametrised by the contact duration, unused duration, bundles transferred, unique bundles transferred, total bytes transferred and the mean of the bytes transferred. In Section 5.7 we evaluate TDP using the *TDP-module*, *TAV-maker*, and *TDP-decider* modules. The results are presented in 5.8. Finally a summary is provided.

5.1 Evaluation perspective

Our set of protocol requirements that we derived in Chapter 1 can be used to examine how existing discovery protocols fulfil the discovery protocol requirements listed in Table 1.2. TDP must be capable of providing information to exploit an opportunistic contact in the RHNC described in section 1.4.1. Sections 1.4.2 to 1.4.6 provides the set of time varying information, that comprises the set of protocol requirements, TDP is required to transfer within the RHNC. In this chapter we evaluate whether TDP addresses the problems outlined in sections 1.4.2 through 1.4.7 to ascertain if TDP, realised as the *TDP-module* module, along with *TAV-maker* and *TDP-decider* modules exploit an opportunistic contact to transfer ADUs within the bounds of the RHNC. The results are discussed in Section 5.8 and finally a summary is provided.

5.2 The N4C data

As part of the N4C project, TCD and Intel, together with help from other N4C partners constructed and operated a DTN in remote areas of the Padjelanta national park URL (90) in northern Sweden, see Figure 5.1, in the summer of 2009 and 2010. As part of the SAIL project (funded by the EU under the FP7 programme), TCD and the Swedish Institute of Computer Science (SICS), together with help from Tannak constructed and operated a week-long DTN trial in the summer of 2011.

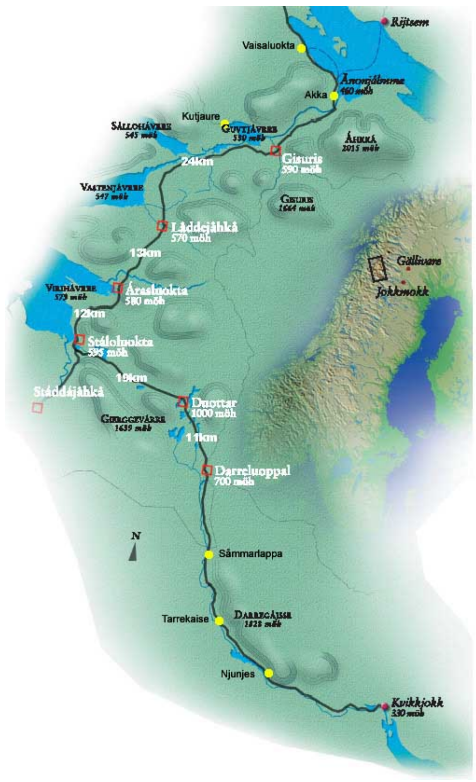


Figure 5.1: The Padjelanta national park in northern Sweden

5.2.1 2009 trial data

In our paper (McMahon et al., 2011), we detailed a week long DTN trial we carried out during the summer of 2009 in the Laponia area of Northern Sweden that involved the provision of basic email and web services to users who were 57km distant from any power or networking infrastructure. The trial validated our design and successfully demonstrated the use of email via helicopter-transported data-mules. With the aim of making it easier for others to replicate this kind of trial, the hardware and software used are described in the paper with references to full specifications, as are the results of the 2009 trial. We described the DTN router node design (McMahon et al., 2009) and 2009 results (McMahon et al., 2010),

Background to the 2009 trial

The following has been taken directly from our paper (McMahon et al., 2011). It is important to note that the implementation described here has mainly been designed for the extended trials in summer 2010, and the 2009 trial was intended to, and did, demonstrate that the basic equipement design, software and networking could work in a short trial, prior to expending the resources required for the longer trial.

The area for the 2009 and 2010 trials (shown in Figure 5.2) is part of the Padjelanta national park (67°22'N,16°48'E) and therefore has almost no road, power or networking infrastructure. As a national park, there are many restrictions on the development of such infrastructure, even temporarily.

As there are no roads, the area is serviced by a small number of helicopter companies, that have semi-regular flights, between for example Ritsem and Staloluotka, for which the flight duration is about 20 minutes. Since our remote site for the 2009 trial was in Staloluotka, the Ritsem-Staloluotka helicopter link was our only data-mule route.

The helicopter companies also charter their aircraft for ad-hoc flights, and even with their regular service may make detours to pick up, or drop off, people and goods, mainly on behalf of the local population. The helicopters thus provide a roughly scheduled service.

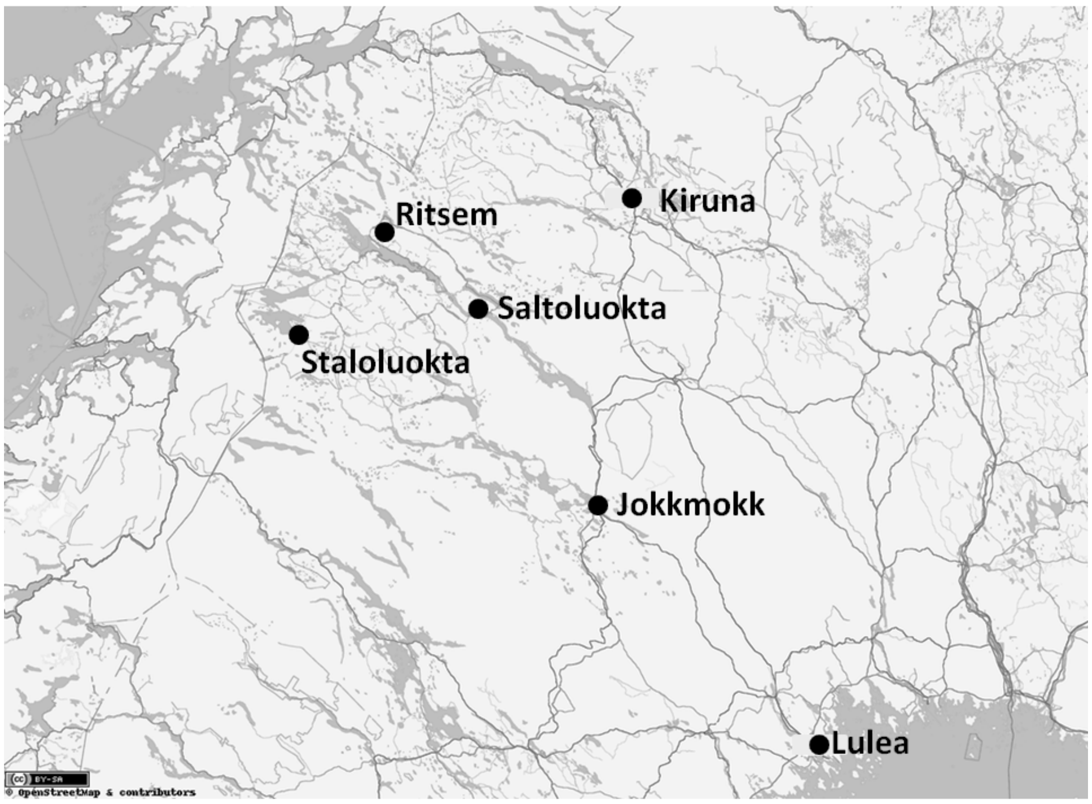


Figure 5.2: Map of Test Area (c) OpenStreetMap and Contributors. The Internet gateway with GSM connectivity was in Ritsem, the village DTN router was installed in Staloluotka, approx. 50km distant.

Aside from the more (and less) obvious logistical and organisational challenges inherent in any field trial, at this stage in the development of DTN, the main challenge in carrying out a DTN field trial in an area such as this is transtioning technology designed for laboratory use to use in the real world. For example, while the DTN stack used has been in laboratory use for a number of years and has seen some limited field use, there were a number of issues with logging that required significant work before, during and after the field trial.

In addition, trials such as this validate the DTN architecture, for example, the ability to really use a data mule is demonstrated here, which involves more than just deploying the technology, but also requires that the technology is suitable for operation by the untrained personnel e.g., helicopter pilots.

This trial also required an innovative integration of the DTN stack with off-the-shelf hardware products and open-source software to produce a sufficiently robust DTN node.

Lastly, this trial involved integration of the DTN stack with standard web and mail services, so that the end-user devices did not have to have any DTN stack installed. While that represents a fairly obvious variant, had not, to our knowledge, been done before in this way.

Experimental setup of the 2009 trial

The system described here has been designed to allow end-users to use familiar applications and interfaces and to hide the DTN. End-users do not have to install any special software and are able to use a standard laptop with wireless access, or indeed any device that can support IEEE 802.11 (b or g, at 2.4GHz) such as various types of mobile phones, or PDAs.

The logic behind these design constraints is that the user populations concerned should only need the most basic conception of DTN, essentially being able to view it as sending their email via helicopter with no DTN specific knowledge being required to use the service and with no requirement to install DTN software on the end-users' devices.

Our overall system for the 2009 trial is shown in Figure 5.3 and consisted of two computers in Ritsem, one of which had basic Internet connectivity via a 2G GSM modem and the other of which acted as a BP relay; a unit which we call a "Village DTN Router" in Staloluotka and data-mules carried by the helicopters. In addition, users could make use of their own (or the experimenter's) laptop or other wireless device. In addition to laptops, both Nokia 810 PDAs and iPhones were used during the trial by other participants in the ExtremeCom 2009 URL (96) event, and from the N4C project.

The village DTN router essentially acts as a traditional WiFi hotspot, but instead of connecting directly to the Internet uses the BP to send and receive bundles encapsulating application traffic.

The unit in Ritsem was connected, via a Virtual Private Network (VPN), back to servers in TCD that de-capsulated and processed the relevant application traffic.

Data-Mules

Data-mules are nodes that physically move and carry bundles so as to connect otherwise disconnected parts of the DTN. In our trials, data-mules do not run any application layer code, but only act as BP nodes.

Two types of data-mule were deployed for the 2009 trials. One was a single board computer (SBC) mounted in an enclosure in the helicopters, the other, initially intended as a backup in case the helicopter

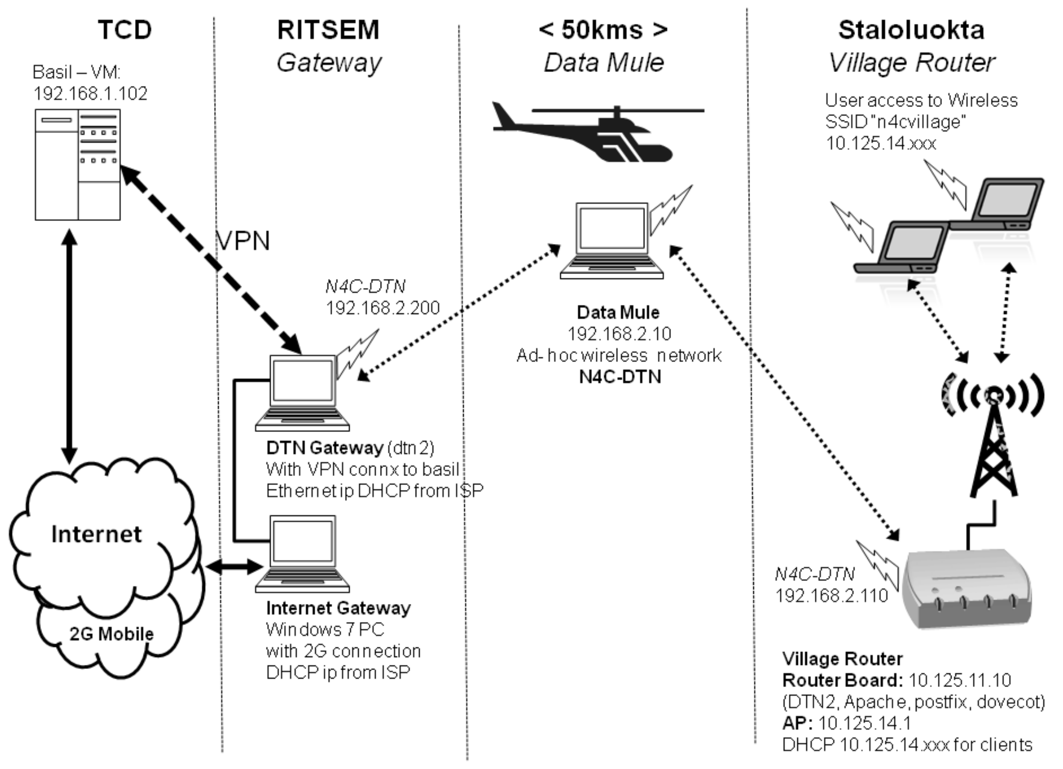


Figure 5.3: Network design. From left to right: TCD is connected to the gateway at Ritsem via a VPN running over a GSM 2G link; the gateway connects via ad-hoc WiFi to the data-mule carried by the helicopter; the data-mule connects via ad-hoc WiFi to the village DTN router when the helicopter is present; end-user devices connect via infrastructure-mode WiFi to the village DTN router; DTN protocols are not used on this last link.

mounted unit failed, (which occurred), was an *Asus EEE PC 901* that was handed to the pilot. Suring the actual trial, none of the data reported on here was transported via the WRAPs but all used our backup data mule solution.

Our backup data-mule solution was an Asus EEE PC 901 netbook, running the Ubuntu 8.04 operating system, using ad-hoc mode WiFi, with the DTN2 implementation of the BP (as had the WRAPs).

The modus-operandi for these mules was to keep them powered up whilst at the helicopter base in Ritsem, and in WiFi contact with the DTN gateway node described below. A pilot making a trip to Staloluokta simply carried the mule (still powered up) in the helicopter and handed it to our personnel on arrival in Staloluokta. Our personnel in turn brought the data-mule into range of the village DTN router (and checked it was operating correctly) so that the bundles were transferred, and then handed the data-mule back to the pilot. On arrival back in Ritsem, the pilot simply plugged the data-mule back into its power supply which was in range of the DTN gateway device described below. All of the traffic reported on below was carried by the EEE PC data-mule.

Internet/DTN Gateway

The Internet/DTN gateway was composed of two devices - a tablet PC with a 2G/3G USB dongle to provide basic Internet connectivity and DHCP address service, and another Asus EEE PC 901 which acted as a DTN gateway.

Since the tablet PC ran Microsoft Windows, and the DTN2 stack has not been ported to that operating system we needed the EEE PC to receive bundles from the data-mule and to forward them via the Internet to a server (*basil*) back in Dublin which was associated with the source or destination unique endpoint for DTN services.

Interestingly, the 2G connection from the tablet PC often experienced outages, and so the EEE PC relay was in fact quite useful since those outages often co-incided with the arrival or departure of helicopters.

We secured the connection back to *basil* using a virtual private network (VPN), which was installed on the DTN gateway EEE PC using OpenVPN (URL, 97) which connected to the our server, *basil*, in Dublin. *Basil* then routed email and executed web requests. The VPN was necessary as we were using static routing for DTN and such routes need fixed IP addresses or DNS entries. The VPN eliminated any issues that might have arisen since the tablet PC could get different IP addresses when it received its own Internet-facing IP address from the 2G/3G network - essentially it meant we controlled all the IP addresses of all of the DTN nodes.

Radio Links

Our initial design was for all clients, including data-mules, to operate in WiFi infrastructure mode. The main antenna for the village DTN router was mounted on the top of the aluminium stand over the solar panels. We chose a weather-proof, 12dBi high-gain, vertical polarisation, omni directional antenna (URL, 98) and an n-type connector to a large diameter co-axial cable. Large diameter co-axial cables are less flexible but result in far less signal attenuation (of the order of 0.5dB per meter).

However, the WRAP data-mules installed on the helicopters (by another partner in the N4C project) had been configured to operate in ad-hoc WiFi mode in order to support other experiments by other partners in the N4C project. This is an example of the kind of “late-surprise” that is very likely to occur in multi-partner field trials. Luckily, the additional capabilities of the Proteus board allowed us to to install an additional PCIe wireless card on the Proteus board, which we configured to operate in ad-hoc WiFi mode, thus allowing us to interoperate with the deployed WRAPs (in principle, if not in practice). The second radio also of course required a second antenna, which was an EnGenius 7dbi vertical polarisation omni-directional antenna.

In the field, these ad-hoc WiFi connections proved very troublesome, with many associations not being established. The main pattern was that two identical devices would work reliably but that mixed devices would not.

Routing

We designed our routing scheme, based on static routes, well in advance of the actual field trial. This scheme allowed us to replace one village DTN router with another without re-configuring in case a node failed.

Test Timing

The data was gathered between our arrival in Ritsem, (approx. 2009-08-07T17:00:00 UTC) and the time when we turned off the systems in Saltoluotka (approx. 2009-08-14T00:30:00 UTC) which encompasses three different setups, detailed in Table 5.1.

No.	Location	Start	End	Duration (hours)
1	Ritsem	2009-08-07 17:00	2009-08-09 07:00	38h
2	Staloluotka	2009-08-09 16:00	2009-08-12 09:00	65h
3	Saltoluotka	2009-08-12 14:00	2009-08-14 00:30	34.5h
	Overall	2009-08-07 17:00	2009-08-14 00:30	151.5h

Table 5.1: Timing of 2009 Trial (UTC)

In the first phase, equipment was assembled and tested in Ritsem, which involved setting up the full test topology and end-to-end testing, but with low latencies. In the second phase, the village DTN router was moved to Staloluotka, set up and re-tested by experimenters. Two days later the ExtremeCom hikers arrived in Staloluotka and made use of the system. In the third phase, the same village DTN router was moved to Saltoluokta, where the workshop (presentations) portion of the ExtremeCom event was held, and the system was restarted, mainly in order to “finish” in-progress transactions.

An overview of the roles of the DTN nodes of the 2009 trial in Table 5.2.

Short name	Role	Comment
dtnrouter-11-10	router	The main router used
dtnrouter-11-20	router	A backup/test router
dtnmule-2-10	data-mule	The Asus EEE PC mule
dtngateway-2-200	gateway	The gateway at Ritsem
dtngateway-1-102	basil	The well-connected endpoint

Table 5.2: Nodes and Roles in 2009 Logs

Data Collection and Preparation

All DTN2 daemons used recorded logs in *info* mode. While these logs are less verbose than those produced using *debug* mode they do still contain a lot of information that is unnecessary for results-analysis, so scripts were developed to filter and translate these logs into a more useful format. These scripts used some simple “C” code, but mostly grep and awk to parse and analyse DTN2 logs into a more easily processed comma-separated value (CSV) format.

5.2.2 2010 trial data

The validated implementation of the 2009 trial was designed and used for the N4C extended summer 2010 trial conducted during July and August 2010, which validated enhancements to our design and successfully demonstrated the use of existing helicopter routes to carry e-mail and web traffic to and from four different locations about 20 - 50km distant from any power or networking infrastructure. We deployed “village DTN routers”, in the four locations to serve that traffic to browsers and email clients using the bundle protocol. The email and web applications provided over DTN during the 2010 trial were detailed in our report (McMahon et al., 2011). The hardware and software used was described with references to full specifications (URL, 91), and all of the source code and the non-private data and analysis scripts underlying these results were made public in an informal report (URL, 92).

Background to the 2010 trial

The following is taken directly from a a work-in-progress draft report. Like the 2009 trial the 2010 system described here has been designed to allow end-users to use familiar applications and interfaces and to hide the DTN. End-users do not have to install any special software and are able to use a standard laptop with wireless access, or indeed any device that can support *IEEE 802.11* (b or g, at 2.4GHz) such as various types of mobile phones, or PDAs.

The village DTN router essentially acts as a traditional WiFi hotspot, but instead of connecting directly to the Internet uses the BP to send and receive bundles encapsulating application traffic. The gateways in Ritsem and Sjöfallet were connected, via a Virtual Private Network (VPN), back to servers in TCD that de-capsulated and processed the relevant application traffic as described in our 2009 report.

Experimental setup of the 2010 trial

Our overall network topology for the 2010 trial is shown in Figure 5.4.

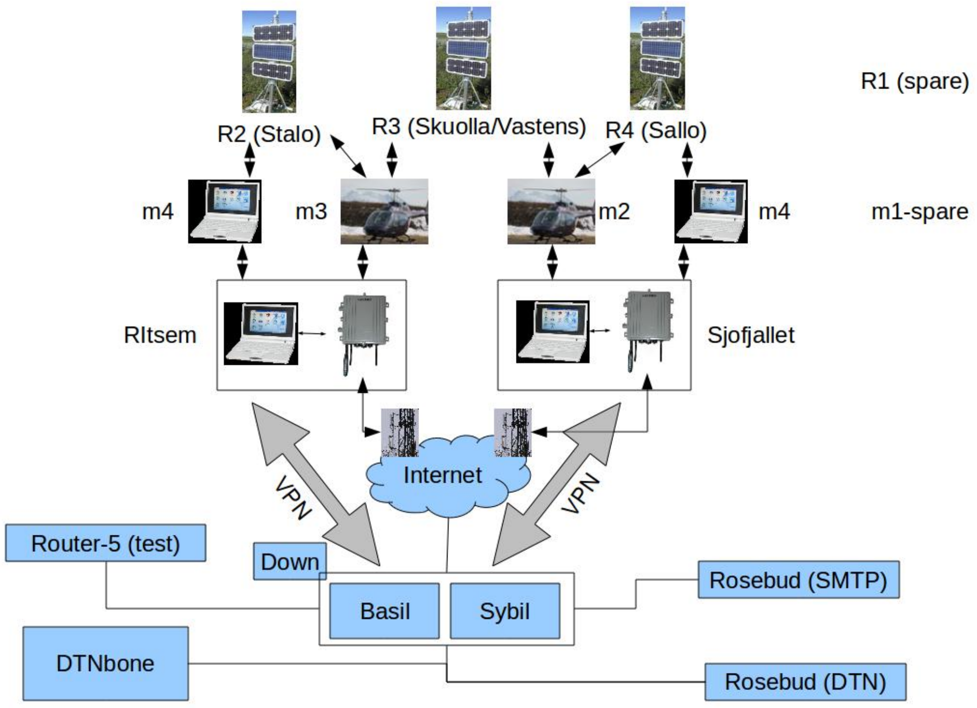


Figure 5.4: Summer 2010 N4C Trial DTN Topology

The network topology shown in Figure 5.4 consisted of, from top to bottom:

- The village DTN router (shown in Figure 5.5) is an application server and WiFi hotspot.

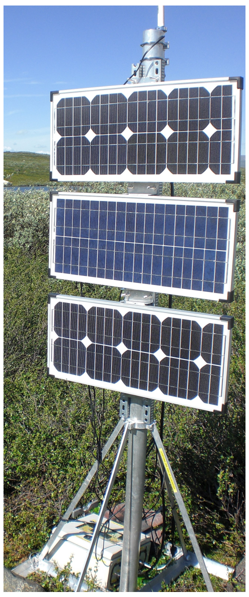


Figure 5.5: Deployed Village DTN Router. This is the router shown at Sallohaure during the trial in summer 2010. One can see the stand, 3 x 20W solar panels, infrastructure-mode WiFi antenna (cropped at the top of the image) and the enclosure (at bottom) containing the electronics.

  – We distributed some surplus handheld WiFi devices (thanks to Intel) and used laptops and phones as client application devices.

  – Three routers were deployed with two spares - one had been planned for deployment, but we moved router-3 from Vastenjaure to Skuolla since there was no-one in Vastenjaure during the calf-marking week. Router-1 was subsequently deployed as a second router in Staloluotka.

  – End-user devices connect via infrastructure-mode WiFi to the village DTN router

- There were two types of so-called *data mule*, single-board computer (*WRAP*) helicopter mounted, and handheld netbooks.

  – Two helicopters had WRAPs installed (mules 2 and 3).

  – Two EEE PC 901 netbooks (mules 4 and 5) were occasionally used when experimenters were present.

  – Mule 1 was spare. Mule-5 had actually been our spare gateway netbook, but we decided to re-purpose it as a mule when we encountered some issues with mule-contacts.

- There were two Internet gateways, one in Ritsem (gateway-2), and one at Sjöfallet (gateway-1). In both cases, these consisted of a netbook and OSbridge (which is a WiFi access point with a GSM backhaul).

- The gateway netbooks connected back to our main host in Dublin ("basil") via a VPN connection. The gateway links proved very slow (about 3KB/s was frequently seen) and liable to disruption.

- The **basil** host was actually running two DTN2 daemons, one (called *sybil*) that used flood routing to send bundles from Dublin to the two gateways, and one (the original "basil") that received bundles from applications, other DTNbone nodes, and the gateways.
- Since the basil host was somewhat unreliable (it occasionally crashed), we had another host ("down") that pinged it each hour and alerted staff (via email) when it received no response.
- We had a test router (router-5) running on an IBM thinkpad locally connected to basil, so that we could see the content that should be present on the remote routers.
- basil was then connected to the DTNbone, and to other application services (e.g. "rosebud" which is the inbound and outbound mail MTA for our DTN mail service).

The 2010 router design differed slightly from 2009. We brought the wireless capability on board the Proteus card using a PCIe card. The 2009 data mule was an EEE PC which was manually carried by the helicopter pilots. In 2010 we mounted data mules inside the helicopters. We ported DTN2 and created a custom firmware for the single board computer (SBC) helicopter-mounted data-mules, "Wireless Router Application Platforms", referred to as "WRAPs" URL (99). The 2009 WRAPs relied on the power from the helicopter, which meant they were off most of the time that the helicopter was on the ground in range of a router or gateway. In order to give the WRAPs the maximum time to transfer data we included a battery in the enclosure which would was designed to give us about an hour of operation after the pilot turned off the ignition. We used a solar charger to manage the charging of this battery from the helicopter's power.

The only network connectivity available to use at both Ritsem and Sjöfallet was a GPRS/EDGE connection. We used an OSBridge 3GN WiFi router (URL, 100) to provide this network connection. For 2010 year we added router functionality to each gateway so that it was possible for users to connect to the gateway node just like a village router.

Data-Mules

Two types of data-mule were deployed for the 2010 trials. One was the WRAP SBC mounted in an enclosure in the helicopters, the other, in 2009 intended as a backup in case the helicopter mounted unit failed, was an Asus EEE PC 901 that was handed to the pilot. For 2010, we planned from the start to use two EEE PC data mules to handle cases where only un-instrumented helicopters were available.

The modus-operandi for the EEE PC mules was to keep them powered up whilst at the helicopter bases, and in WiFi contact with the DTN gateway nodes described below. A pilot making a trip from the base simply carried the EEE PC mule (still powered up) in the helicopter and handed it to our personnel on arrival in a village. Our personnel in turn brought the data-mule into range of the village DTN router (and checked it was operating correctly) so that the bundles were transferred, and then handed the data-mule back to the pilot. On arrival back at the helicopter base, the pilot simply plugged the data-mule back into its power supply which was in range of the DTN gateway device described below.

The WRAPs were powered directly from the helicopter 12V power supply and were mounted below the front passenger seat. These devices had a battery backup since there were a number of potential communications opportunities lost in 2009 due to the pilot turning off the helicopter ignition switch immediately on landing, before the WRAP had succeeded in establishing contact with another node. However the battery backup did not work well as we shall see.

The 2009 trial also showed up a number of problems with ad-hoc WiFi (reported on in(McMahon et al., 2011) so for 2010 all data-mules used infrastructure mode WiFi only, with much better results.

Our 2010 battery backup solution for the WRAPs only partially worked. While the system did work in testing, in practice, we only got a couple of minutes additional time powered on, after the pilot had turned off the helicopter ignition. While we never fully got to the bottom of the reason for this (as these unexpected hard power-offs caused us to loose some data-mule local log information), we suspect that the helicopter flights were not long enough for the battery to be re-charged, once it initially depleted. However, while the helicopter was powered up, the WRAPs did function well in 2010, unlike in 2009. We attribute this to the change from ad-hoc to infrastructure WiFi and to a later and better build of the OpenWRT platform and the use of a different filesystem (ext4) for the data-mule's store.

Internet/DTN Gateway

As in 2009 we secured the connection back to *basil* using a virtual private network (VPN), which was installed on the DTN gateway EEE PCs using OpenVPN (URL, 97) which connected to the our server, basil, in Dublin. Basil then routed email, executed web requests and pushed web content. The VPN was necessary as we were using static routing for DTN and such routes need fixed IP addresses or DNS entries. The VPN eliminated any issues that might have arisen since the tablet PC could get different IP addresses when it received its own Internet-facing IP address from the 2G/3G network - essentially it meant we controlled all the IP addresses of all of the DTN nodes.

Radio Links

Our design was for all clients (including data-mules) to operate in WiFi infrastructure mode. The main antenna for the village DTN router was mounted on the top of the aluminium stand over the solar panels. We chose a weather-proof, 12dBi high-gain, vertical polarisation, omni directional antenna (URL, 98) ,and an n-type connector to a large diameter co-axial cable. Large diameter co-axial cables are less flexible but result in far less signal attenuation of the order of 0.5dB per meter.

Routing

Flood and static routing protocols were used. Flood routing was used on *sybil.dsg.cs.tcd.ie* and all mules whilst all other nodes used static routing. Flood routing was implemented for multcast or anycast like functionality. In order to transmit a bundle from Dublin to all village routers it was likely that the bundle would have to traverse all gateways. The capability to transmit a single bundle to all villages via all gateways was desirable as the 2G up-link on the gateways at the helicopter bases were expensive and had very little bandwidth. The main idea was that an application in Dublin could generate a single bundle with a destination endpoint common to all village routers. This bundle would be transmitted to each gateway. Each gateway would then transmit the bundle to the first arriving mule. The mule would transmit the bundle to any other mule or village router that it came in contact with for the lifetime of the bundle and that was statically defined in its routing tables. As the helicopters often travelled from village to village once leaving a base this configuration allowed inter village communications such as email account synchronisation from village to village.

Congestion control in terms of handling duplicate bundles, was intended to occur on each receiving node which was to realise that it had already received a bundle and discard it as each bundle could be uniquely identified. While this worked as planned in many cases, there was an issue with the DTN2 implementation of duplicate detection in that it is essentially implemented as part of the run-time state of the dtnd process - in cases where the daemon was prematurely shut down and then restarted that state was lost and some duplicates could not be detected.

Initially we had planned to use a single dtnd daemon in Dublin configured with static routes however the way static routing is configured in DTN2.7 is that a bundle is only forwarded to the first route on the routing table. Given that we didn't know which route (via Ritsem or Sjöfallet) would work for any given bundle this didn't work for our setup. As only one routing protocol can be configured per daemon two DTN2 daemons running on node basil.dsg.cs.tcd.ie were configured. This configuration meant we had one DTN daemon (basil) communicating with the Internet, and with local applications and then forwarding traffic to another DTN2 daemon (sybil) which routed to the villages via the gateways. Return traffic from the gateways didn't have the same problem and was routed from the gateway to the basil DTN2 instance. Essentially, all of this was really only needed due to the lack of a more configurable flood routing implementation in DTN2.

The Skuolla site is used for one week for reindeer calf-marking. Once the calf-marking week at Skuolla was over, one router ("n4crouter-1,") was transported to Staloluokta to act as a backup/test router. This second, backup, router ("n4crouter-1") was not required, other than for testing. However, in this section, we describe the full routing setup, including the backup router. Bundles from the field test, with email or web request payloads were received by basil via sybil, which then routed email and executed web requests. Response bundles were routed back to the revelant field nodes. Bundles transmitted between router nodes and gateway or other router nodes always traversed a mule node.

Five routers, two Internet and DTN gateway and four mules were deployed. The DTN gateways in Ritsem and Sjöfallet relayed bundles to basil in Dublin and vice versa. The four data mules consisted of two WRAP units, each mounted in a helicopter serving the helicopter base/village links, and two EEE PC mules that were handed to helicopter pilots as described previously. Each node was configured with static routes to the EIDs of all the available applications and the CLAs of every other node in the topology (Please see Figure 5.6).

Static routes were configured on:

- Each router for every EID not contained on that router via the EID of each CLA of each mule in the topology.
- The gateways in Ritsem and Sjöfallet for every EID registered on a router via the EID of each CLA of each mule in the topology and directly to the EID of the CLAs of basil for EIDs that did not contain village.n4c.eu.
- Basil, for every EID that contained village.n4c.eu via the EID of each CLA of the Ritsem gateway.
- Each mule to every EID of each CLA of each router and gateway in the topology.

Initially all of the static routes were cofigured as TCP "ONDEMAND" links. When such links are dropped

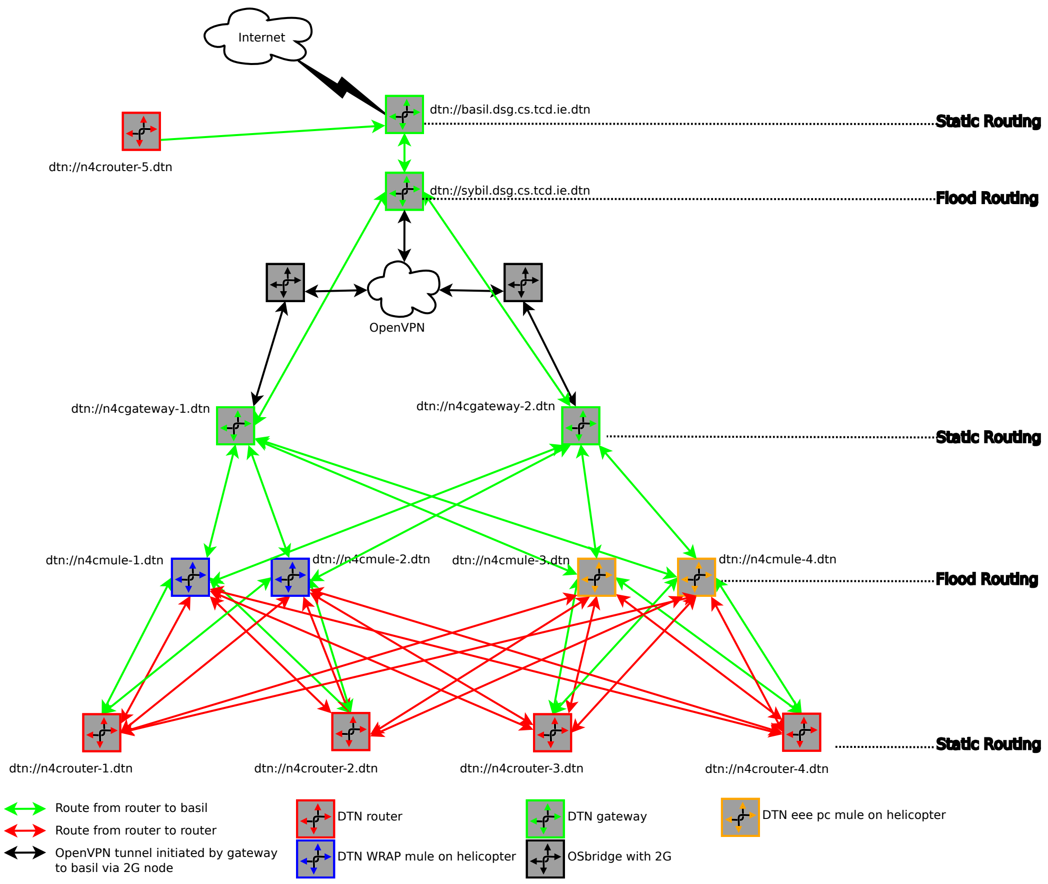


Figure 5.6: Static and flood routing. Redundant routes were configured to allow for inter village communication. See Figure 5.4 for the physical topology. Bundles from basil were always routed to all five routers.

(e.g. after a data-mule travels out of the coverage area of a village DTN router), then the DTN2 daemon will periodically attempt to re-establish the TCP connection. In our configuration, this was attempted after a random delay of between 1 and 10 seconds.

In addition to static routes, we also used the link discovery features of the DTN2 stack on the routers. The main reason to turn on link discovery on village routers was in order to support other N4C experimeters devices to be discovered and to route over the infrastructure.

Link discovery worked by having each DTN2 daemon emit a UDP broadcast message periodically (every 10s) containing contact information for that node. A node receiving such a message (i.e., a data-mule coming into range of a village DTN router) would establish a TCP connection to the relevant address and port. That TCP connection was then treated as a potential route for any bundle.

However this mix of "ONDEMAND" and discovered routes can lead to cases where a bundle will be transmitted both over the just-reestablished pre-configured static link and over the discovered link. One could argue that DTN2 should not do this where the destination CLA, host and port are the same, but that is the current behaviour. Nonetheless, we felt this was worthwhile since our overriding concern was to get bundles back from the field and optimising usage of link capacity is a secondary issue.

Each node accepted custody for each bundle it received and requested custody for each bundle it transmitted. That meant that each node was capable of re-transmitting any bundles as required.

Test Timing

The trial began in early July and ran until late August. (Tear-down started with the routers in Stalo on August 19th.) The results presented here reflect data gathered between equipment commissioning, (approx. Jun 29 10:19:26 UTC 2010) and the time when we turned off the systems (approx. Wed Sep 15 14:42:00 UTC 2010) which encompasses different phases, detailed in Table 5.3.

No.	Location	Start	End	Duration (days)
1	Startup	2010-06-30 00:00:00	2010-07-11 23:59:59	12
2	Skuolla	2010-07-12 00:00:00	2010-07-16 23:59:59	5
3	Attended	2010-07-16 23:59:59	2010-07-25 23:59:59	10
4	Unattended	2010-07-26 00:00:00	2010-08-17 23:59:59	23
5	Partners	2010-08-18 00:00:00	2010-08-19 23:59:59	2
6	Teardown	2010-08-20 00:00:00	2010-08-27 23:59:59	8
	Overall	2010-06-30 00:00:00	2010-08-27 23:59:59	60

Table 5.3: Timing of 2010 Trial (UTC). The longest duration was the 23 days of the Unattended phase

An overview of the Links, i.e., abstractions for a DTN2 one way communication channel to a next hop node in the DTN overlay, during the trial are depicted in Figure 5.7.

We held an Internet cafe event on one of those days. (More were planned, but bad weather intervened.) The Staloluokta site has a large tourist cabin for hikers and is visited each day at least once by a helicopter from late June until early September. The Sallohaure and Vastenjaure sites are Sámi villages where reindeer herders stay for the summer.

Data Collection and Preparation

All DTN2 daemons used recorded logs in *info* mode along with *S10* logging. In 2009 *grep* and *awk* were used to parse and analyse DTN2 logs into a more easily processed comma-separated value. An output of this process was a log schema, called *S10*, that we implemented and used to prepare DTN2 daemon log events in a more useful format. This schema was implemented prior to 2010 trial commencement as *S10* logging annotated events in DTN2 as described by change-set *3504* of DTN2 (URL, 95).

5.2.3 2011 trial data

A third trial, during the summer of 2011 was conducted for a period of two weeks in July.

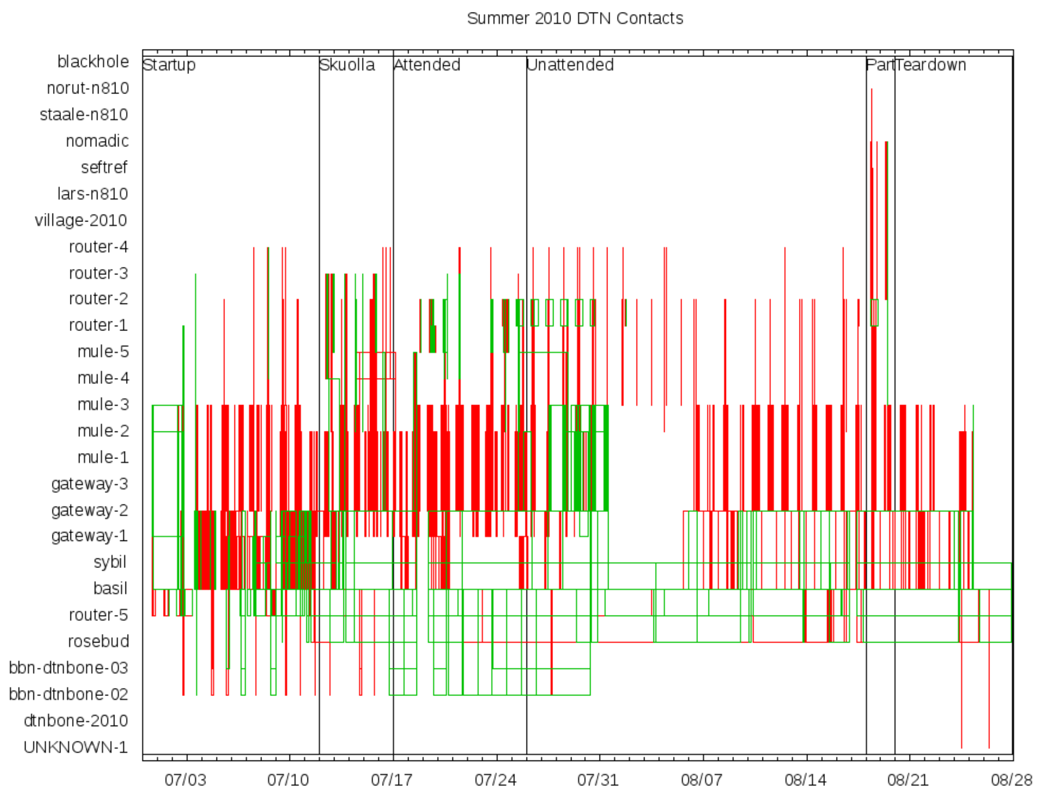


Figure 5.7: Links during the 2010 Trial. The figure plots each link as a rectangle with the horizontal line matching the start and end of the link and the vertial line matching the two peers involved. Contact peers are named on the vertical axis. 9575 link were seen seen during the trial of which the ends points for 469 were resolved, e.g. due to hard-poweroffs, crashes. These were assumed to end at the next time at which the DTN daemon exits or the node powers off, and are shown in green. At the top, there were five different "DTN user" devices that contacted one of the village routers. At the bottom, there were links between basil and rosebud, for the "nomadic" mail experiment, and there were regular DTNbone contacts with BBN and a few with UNKNOWN (Braunschweig). Most contacts are too short to be individually visible at this scale. The trial phases are shown separated by vertical lines in this and many subsequent figures.

Background to the 2011 trial

Much of the following has been taken directly from our paper (McMahon et al., 2011). As part of the SAIL project (funded by the EU under the FP7 programme, TCD and the Swedish Institute of Computer Science (SICS), together with help from Tannak constructed and operated a week-long trial Delay-Tolerant Network (DTN) in the Padjelanta national park in northern Sweden in July 2011 during the Sirges village calf-marking at Skuolla. An outline of this trial, which includes a description of the Bundle Protocol Query (BPQ) extension block implemented and subsequently tested during the trial, has been provided (McMahon et al., 2011). As in previous years the dates of the trial were selected to coincide with the week that the Sáami gather their reindeer for calf-marking. Similar services to 2010 were provided with some additions: was trialled alongside the existing web applications.

Experimental setup of the 2011 trial

Helicopters flying between Ritsem and Skuolla, and this year, along with a 30km GSM link from Skuolla, to carry e-mail and web traffic to village DTN routers. The trial ran from July 4th to 12th. Village DTN routers were setup in Ritsem and Skuolla. Figure 5.8 describes the topology and Figure 5.9 describes the data flow for the 2011 DTN trial, which unlike previous years, used a local GSM link on a hill in the area of the calf marking pens.

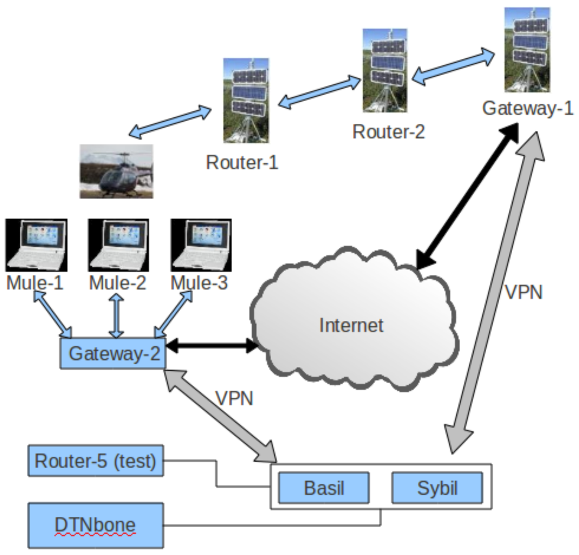


Figure 5.8: Summer 2011 SAIL Trial DTN Topology

Data-Mules

For the 2011 tests, we only made use of three handheld data-mules, the same EEE PCs as used in 2010. This is due to their overall better performance, but also because they require much less effort at deployment time - the helicopter company do not like to have aircraft waiting whilst experimenter's tweak on-board equipment so the handheld approach is far simpler and more robust in this environment. Note however, that for the "unattended" portion of the 2010 trial, only the WRAP data-mules could have worked (there being no-one present to hand an EEE PC to the pilot).

Internet/DTN Gateway

There were two Internet gateways, one in Ritsem *gateway-2*, and one at Skuolla on the hill *gateway-1* over looking the vally in Skuolla.

Using the hardware updates described in (McMahon et al., 2011) one of our village routers was converted into a DTN gateway. This gateway opened a VPN tunnel back to *Basil* to provide a second connection between Basil and the router. A bridge node (router-2) was provisioned for the case where a connection between router-1 and gateway-1 was not possible. In addition to the GSM link on the hill, the previous routes via the data mules travelling in the helicopters to gateway-2 were maintained.

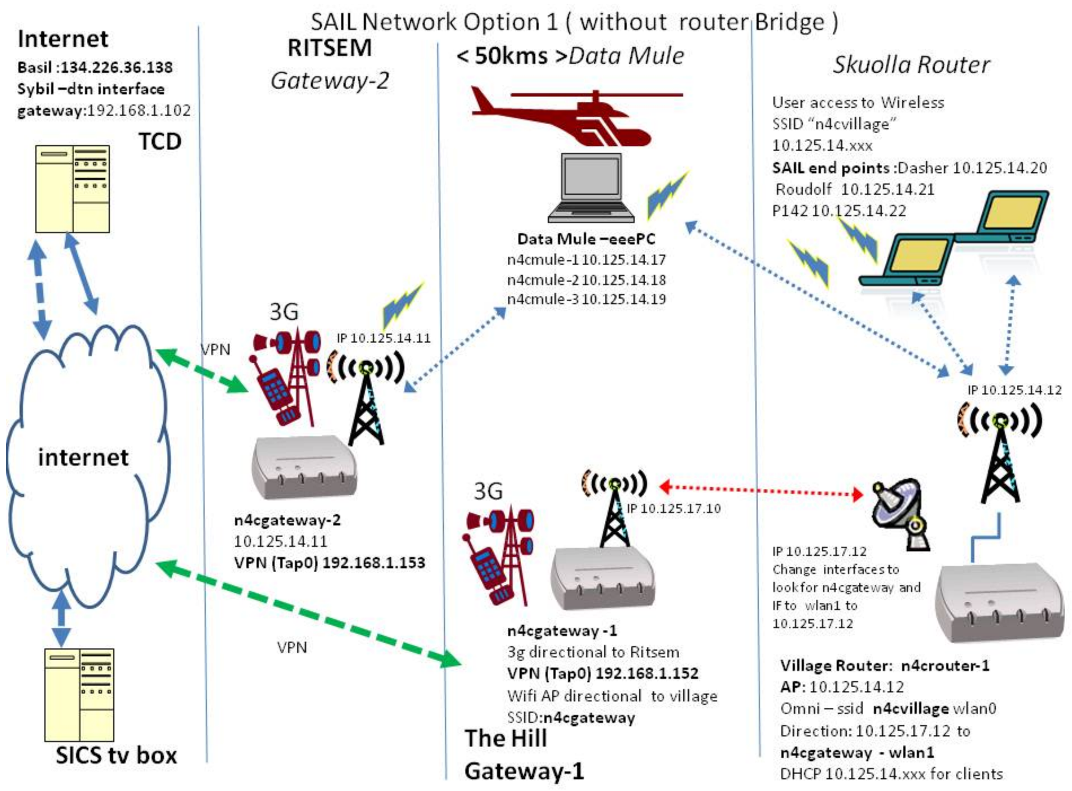


Figure 5.9: Summer 2011 SAIL Trial DTN flow

The gateways connected back to our main host in Dublin *basil* via a VPN connection. Basil was actually running two DTN2 daemons, one with a unique EID *dtn://sybil.dsg.cs.tcd.ie.dtn* that used flood routing to send bundles from Dublin to the two gateways, and one with unique EID *dtn://basil.dsg.cs.tcd.ie.dtn* that received bundles from applications, other DTNbone nodes, and the gateways.

As in 2009 and 2010, the Ritsem base of the helicopter company (FiskFlyg) was usas the location for an Internet gateway to provide Internet connectivity and carried eee-PC data mules back and forth between Skuolla and Ritsem.

Radio Links

There was a 30km GSM link, with a directional antenna from Skuolla back to Ritsemk. This provided a line to the village DTN router via WiFi. Both Internet gateways provided the functions of a wifi access point with a GSM backhaul.

Our design was for all clients (including data-mules) to operate in WiFi infrastructure mode. Like 2010 the main antenna for the village DTN router was mounted on the top of the aluminium stand over the solar panels. We chose a weather-proof, 12dBi high-gain, vertical polarisation, omni directional antenna (URL, 98) and an n-type connector to a large diameter co-axial cable. A directional 3G GSM antenna was used

Test Timing

The data was gathered between July 4th to 12th detailed in Table 5.4.

No.	Location	Start	End	Duration (days)
1	Startup	2011-07-07 00:00:00	2011-07-09 23:59:59	2
2	Skuolla	2011-07-10 00:00:00	2011-07-13 23:59:59	3
3	Teardown	2011-07-14 00:00:00	2011-07-15 23:59:59	1
	Overall	2011-07-07 00:00:00	2011-07-15 23:59:59	6

Table 5.4: Timing of 2011 Trial (UTC)

Data Collection and Preparation

All DTN2 daemons used recorded logs in *info* mode along with S10 logging.

5.2.4 DTN2 links, contacts and events

The N4C project data is composed of DTN2 logs. Using these logs we describe how contacts and contact events are represented in DTN2. We provide an overview of the DTN2 Link and it's relationship to a contact. Next we provide an overview of the DTN2 class that manages all communication to and from the main bundle daemon thread for CLAs that maintain a connection. We then describe the main steps of the thread that manages established connections with peer daemons. Next we describe how the TCP CLA interacts with the BPA to transmit and receive bundles. Finally we describe the set of DTN2 events used in evaluation.

Link overview

A Link is an abstraction for a DTN2 of a one way communication channel to a next hop node in the DTN overlay. When a link is closed and not able to be opened it's state is *UNAVAILABLE*. When the link is closed but can be opened it's state is *AVAILABLE*. When a convergence layer session has been established a link state is *OPEN* if the link has capacity for a bundle to be sent on it. A link state of *CLOSED* is a state that is used for signalling the daemon thread with a link state change request.

All links in the *OPEN* state have an associated contact that represents an actual connection. *ALWAYSON* links are immediately opened upon creation and remain open for their duration. *ONDEMAND* links have to be opened every time one wants to use them and then close after a configurable idle period. An *OPPORTUNISTIC* link has to be opened every time one wants to use it. The difference between *OPPORTUNISTIC* link and an *ONDEMAND* link is that the *ONDEMAND* link does not have a dedicated bundle queue. *SCHEDULED* links have a list of future contacts.

Contact overview

A contact is a communication opportunity (Fall, 2003), characterised by a duration of time (relative to the source), capacity, latency, endpoints and direction. For this work propagation delay is assumed to remain constant during the contact duration. A DTN2 contact is represented as a subset of events that occur on a DTN2 Link.

CLAs that maintain a connection

All DTN2 convergence layers that maintain a connection (e.g. a TCP socket) to next hop peers derive from a class that manages all communication to / from the main bundle daemon thread. This handles the main thread of control for each connection, and dispatches to the specific CLA implementation to handle the actual wire protocol. The design is as follows:

OPEN links contain a *Connection* class. The lifetime of this object is the same as the duration of the Contact object. The *Connection* class is a thread that has a queue for commands to be sent from the bundle daemon. The commands are *SEND_BUNDLE*, *CANCEL_BUNDLE*, and *BREAK_CONTACT*. When in an *IDLE* state, the thread blocks on this queue as well as the socket or other connection object so it can be notified of events coming from either the daemon or the peer node.

To enable backpressure i.e., the use of congestion gradients, each connection has a maximum queue depth for bundles that have been pushed onto the queue but have not yet been sent or registered as in transit by the CLA. The state of the link is set to *BUSY* when this limit is reached, but is re-set to *AVAILABLE* when below this limit. By default, there is no hard limit on the number of bundles that can be in transit, instead the limit is determined by the capacity of the underlying link. A link must be *AVAILABLE* and *OPEN* for a contact to exist.

There is a DTN2 code specific race condition between the underlying connection breaking and the higher layers determining that the link should be closed. If the underlying link breaks due to a timeout or goes *IDLE* for an *ONDEMAND* link, a *ContactDownEvent* is posted and the thread terminates. In response to this event, the daemon will close the contact by calling the *close_contact* method. In this case, the connection thread has already terminated so it is cleaned up when the *Contact* object goes away.

If the link is closed by the daemon thread due to user configuration or a scheduled link's open time elapsing, then *close_contact* will be called while the connection is still open. The connection thread is informed by sending it a *BREAK_CONTACT* command. Reception of this command closes the connection and terminates, setting the *is_stopped()* bit when it is done. All this logic is handled by the *break_contact* method in the *Connection* class. Finally, for bidirectional protocols, such as TCP, *OPPORTUNISTIC* links are created in response to new connections arriving from a peer node.

A DTN2 thread manages established connections with a peer daemon. This thread initiates a connection or accepts one, whilst it is the responsibility of the underlying CLA to make sure that a contact structure is found / created. Along with checking whether the socket is broken the thread carries out the following steps:

1. poll() to wait for data arriving from the remote side, or write-readiness on the socket indicating that we can send more data.

2. consume and transmit any chunks of data until all chunks have been transmitted

TCP CLA

The TCP CLA uses a convergence layer class *StreamConvergenceLayer* for use with reliable, in-order delivery protocols (i.e. TCP, SCTP, and Bluetooth RFCOMM) the goal of which is to share as much functionality as possible between protocols that have in-order, reliable delivery semantics.

Several Link parameters are shared among all stream based convergence layers such as, indicators to enable per-segment acks (default is true) or enable negative acks (default is true), and parameters to set the seconds between keepalive packets (default is 10) and maximum size of transmitted segments (default is 4096). Link parameters are shared for a local IP address, remote next hop IP address and remote next hop port, a send buffer length, receive buffer length and data timeout.

The TCP CLA sets up the base class *CLConnection* as non blocking. When an address is in use when a new server socket is created for an interface, the TCP CLA waits for 10 seconds by default and tries again. The TCP CLA attempts to accept a new connection or connect to the remote host without blocking.

Bundles are broken up into configurable-sized segments that are sent sequentially. Only a single bundle is in transit on the wire at one time (segments from different bundles are not interleaved). When segment acknowledgements are enabled (the default behavior), the receiving node sends an acknowledgement for each segment of the bundle that was received.

Keepalive messages are sent back and forth to ensure that the connection remains open. In the case of *ONDEMAND* links, a configurable idle timer is maintained to close the link when no bundle traffic has been sent or received. Links that are expected to be open but have broken due to underlying network conditions (i.e. *ALWAYSON* and *ONDEMAND* links) are reopened by a timer that is managed by the contact manager.

Flow control is managed through the poll callbacks given by the base class *CLConnection. send_pending_data()* checks if there are any acks that need to be sent, then checks if there are bundle segments to be sent (i.e. acks are given priority). The only exception to this is that the connection might be write blocked in the middle of sending a data segment. In that case, the current segment must first finish transmitting before any other acks (or the shutdown message) are sent, otherwise those messages will be consumed as part of the payload.

Any data that is ready on the channel is drained to ensure the other side is not deadlocked. All incoming messages mark the state in the appropriate data structures (i.e. *InFlightList* and *IncomingList*), then rely on *send_pending_data* to send the appropriate responses.

The *InflightBundle* class is used to record state about bundle transmissions. To record the segments that have been sent, the *sent_data_* sparse bitmap is filled with the range of bytes as segments are sent out. As acks arrive, the *ack_data_* field is extended to match. Once the whole bundle is acked, the entry is removed from the *InFlightList.*

The *IncomingBundle* class is used to record state about bundle reception. The *rcvd_data_* bitmap is extended contiguously with the amount of data that has been received, including partially received segments. To track received segments that haven't yet been acked, a single bit for the offset of the end of the segment is set in the *ack_data_* bitmap. The total range of acks that have been previously sent is recorded in *acked_length_*. As acks are sent out, the bits in *ack_data_* are cleared.

The contact header, the unique EID length and data are copied to the send buffer and the data is sent using *send_data()*. This is a best effort as the bundle may not really be out on the wire yet and there is no way of knowing when it gets into the kernel (nor or that matter actually onto the wire).

S10 CONTUP events

When the convergence layer is told to establish a new session, it sets the link state to *OPEN* and posts a *ContactUpEvent* when it has done so.

S10 CONTDOWN events

When the convergence layer is told to stop an existing session, it sets the link state to *CLOSED* and posts a *ContactDOWNEvent* when it has done so.

S10 RX events

For all cases the S10 RX event is logged post BPA reception. When the BPA's *handle_bundle_received()* method, usually called by a router to receive a bundle, receives an RX bundle event that indicates a remote source i.e., a peer, the BPA's statistics and are updated and an S10 RX event is logged.

S10 TX events

For all cases the S10 TX event is logged post CLA transmission.

If not configured for reliable transmission, when the BPA *handle_bundle_transmitted()* method is called, usually by a router to transmit a bundle, an attempt is made to establish the status of the link and if the link is ready locate transmit blocks. The link is ready to handle the bundle if the block vector exists. If transmit blocks are found, the size of the blocks is established and the bundle and link statistics are updated. The bundle is removed from the link *inflight* queue and verified not to be on the link's *to-be-sent* queue. Finally an dtnd *info* level TX event followed by a S10 TX event is logged.

When the BPA *handle_bundle_transmitted()* method is called by a router the router performs tasks such as attempting to ascertain link capabilities post the S10 TX event. For example, using static routing, if the bundle has a deferred single-copy transmission for forwarding on any links, the link is checked to see if the transmission means that another bundle can be transmitted on the link (i.e. either when the contact is brought up or when a bundle is completed and the link is no longer busy). This is accomplished by looping through the bundle list and making calls to the *fwd_to_matching()* method on all bundles.

During the loop if the link isn't open, or link queue doesn't have space then nothing is done. In another loop, if the link queue is full the loop is stopped, the deferred bundle list is checked for bundles. If the bundle was either already transmitted or is in transit it is left on the deferred list, relying on the transmitted handlers to clean up the state. An attempt is then made to open the available link (if it is closed), remove the bundle from the deferred list and forward bundle to the available open link (queue the bundle)

When a bundle is placed in the queue, an attempt is made to create transmit blocks for the bundle, and the bundle queued on a link if not already queued on the link or in in transit.Finally, the CLA is notified using the accessor to the contact's convergence layer. If configured to wait for reliable transmission, then a check is carried out to verify the special case where some or all a bundle was transmitted but nothing was acknowledged. Finally the forwarding log is updated to indicate that the bundle is no longer in flight.

5.3 The treatment of the data

In order to analyse opportunistic contact data of DTN nodes, the processing of the DTN2 log data is automated to mitigate against human error and provide transparent treatment of the data. This process requires the creation of a data store table to persist the full data set. Once persisted, a table is created for a subset of the data. The number of subset tables created is arbitrary. For each subset table, a certain set of transformations are carried out, as described by the steps outlined in Table 3.40. These steps produce three summary tables to provide a summary for links, a summary for each contact extracted from those links and a statistical summary of the bytes transferred for each contact. A table to summarise the contact statistics for each link as detailed in Table 3.37. A table to provide detailed information for each contact extracted as described in Table 3.38. A table to provide statistics, using the combined TX/RX event duration and bytes received or transmitted, for the subset of contacts extracted as detailed in Table 3.39.

Next, contacts are named using UUIDs and a table view, with fields as described in Table 3.36, is created for each unique contact. Each named table view corresponds to a contact of our network model described in Section 3.6. How each table view corresponds to a contact of our network model is described (and represented as SQL statements) in Table 4.7.

5.4 Contact analysis and modelling

The *Analysis* module takes the DTN2 logs of an arbitrary number of nodes. We take a phased approach, described in Section 4.7.1, in order to extract, filter, create suitable objects for processing, analyse, and to provide an appropriate random contact selection for evaluation. The high level analysis module steps used to extract pairwise contacts from DTN2 log files into data store table views are detailed in Table 3.40. The *Analysis* module is used to select the *test contacts*; an appropriate set of contacts from the data to be used as an input to the real-time network emulator in order to evaluate the number of ADUs transferred during those contacts. Table 5.5 outlines this process.

No.	**Description**	**Module**
1	Create the working data set using the phased approach provided in Section 4.7	Analysis
2	Obtain the IDs of the contacts and persist as a Python numpy object “full_contact_selection.npy”	Analysis
3	Obtain the IDs of the *test contacts* and persist Python numpy object “random_contact_selection.npy”	Analysis

No.	Description	Module
4	Analyse the characteristics of *test contacts*	Analysis

Table 5.5: The four steps to obtain and analyse the contacts using the *Analysis* module

5.4.1 Opportunistic contact modelling

For each contact of *test contacts* the contact ID, contact duration in seconds, contact duration in which there were no events (in seconds), number of bundles for RX, TX and DELIVERED events, number of unique bundles for RX, TX and DELIVERED events, sum of bytes transferred and mean of bytes transferred is presented. For example, a single contact view, with contact ID *6251ba51-d808-4de4-a128-cb4764e9bb70*, is provided in Table 5.6.

ID	duration (sec)	unused duration (sec)	bundles	uniq bundles	bytes (sum)	bytes ($\bar{x}$)
6251ba51-d808-4de4-a128-cb4764e9bb70	724.94	335.63	328	43	55818226.00	170177.52

Table 5.6: A total of 170177.52 bytes in 328 bundles of which 43 were unique were transmitted or received over a total contact duration 724.94 seconds of which 335.63 seconds were not utilised.

5.4.2 Contact parameters

Known variables include the set of bytes transferred, the set of bundles transferred, the bundle identifiers and the number of those that are unique, the contact duration, bundle delivery duration, the order of events, the inter event duration, the contact placement within the *parent* link, the event timings and the node identifiers.

5.4.3 Contact volume estimation

The definitions of Table 3.14 describe the modifications to the directed multigraph construction described in Section 3.6. In scenarios where the duration of transfer events is not known, as exemplified by the N4C data, but the contact duration, bytes transferred, and total number of messages transferred, along with the number of those messages that are unique is known, lower and higher bound metrics can be estimated. Using these bounds the exploitation of a contact can be evaluated. The functions to calculate the upper and lower bounds are described in Table 3.15.

The objective of the lower bound functions is to provide a limiting function to be applied to links between pairwise nodes in the ns-3 emulation. The lower bound functions provide a limiting function for each contact of the *test contacts* set.

The higher bound functions are used along with the *test contacts* set, to estimate the potential volume of a given contact for the transfer of bundles and unique bundles.

A contact can be observed to have been exploited when an equal or greater number of bundles have been transferred and:

1. The volume of a contact realised (as calculated by lower bound capacity function) is reduced to closely match, or equal, the volume of a contact realised for transfer of unique bundles (calculated by the lower bound unique capacity function).
2. The volume of a contact realised for transfer of unique bundles (calculated by the lower bound unique capacity function) is lower than the associated emulation.
3. The estimated potential volume of a contact (as calculated by the higher bound capacity function) is a closer match, or equals the estimated potential volume of a contact for transfer of unique bundles (as calculated by the higher bound unique capacity function).

5.4.4 Contact emulation

The experimental environment is implemented on four physical nodes composed as detailed in Table 5.7. The environment is scalable and not limited to deployment on this hardware. The limit of four is selected due to the limited available physical nodes. Furthermore, while the the physical nodes of experimental environment are adequate, they have been selected on their availability rather than performance.

Type	**Description**
cpu number	4
cpu family	6
cpu model	30
cpu model name	Intel(R) Xeon(R) CPU X3430 @ 2.40GHz
cpu stepping	5
cpu microcode	0x4
cpu cpu MHz	2394.061
cpu cache size	8192 KB
cpu bogomips	4788.12
Memory	2038588 kB
Disk	453GB

Table 5.7: host hardware for experimental environment

Each physical node has the software installed detailed in Table 5.8. All software dependencies are also installed.

Type	**Description**
Linux version	3.2.0-41-generic (buildd@allspice)
Distribution description	Ubuntu 12.04.2 LTS
GCC/G++ compiler	(Ubuntu/Linaro 4.6.3-1ubuntu5) 4.6.3
Compiler target	x86_64-linux-gnu
MySQL	5.5.32-standard
Mercurial Distributed SCM	version 2.0.2
Python	version 2.7.3
mysqldb	Python interface to MySQL
matplotlib	Python based plotting system
LXC version	0.7.5
NS-3 version	3-dev changeset 9758
OpenSSH	5.9p1
OpenSSL	1.0.1

Table 5.8: host software for experimental environment

Each ns-3 lxc node used in in the emulations has the software detailed in Table 5.8

Type	**Description**
Linux version	3.2.0-41-generic (buildd@allspice)
Distribution description	Ubuntu 12.04.2 LTS
GCC/G++ compiler	(Ubuntu/Linaro 4.6.3-1ubuntu5) 4.6.3
Compiler target	x86_64-linux-gnu
MySQL	5.5.32-standard
Mercurial Distributed SCM	version 2.0.2
Python	version 2.7.3
mysqldb	Python interface to MySQL
OpenSSH	5.9p1
OpenSSL	1.0.1
DTN2	version 2.9 changeset 3580
oasys	version 1.5 changeset 2303
tdp_decider	version 1.0
tav_dtn	version 1.0
tdp_dtn	version 1.0

Table 5.9: lxc test nodes for experimental environment

The *ns3_configer* and *ns3_lxc_configer* modules use the ns-3 discrete-event network simulator as a real-time network emulator, to enable ns-3 to be interconnected with the real-world, in order to allow use of appropriate existing real-world protocol implementations. Lxc test nodes are configured with two interfaces (although only one is used in these tests) which are placed under the control of ns-3. Control of these interfaces is accomplished via the Ethernet bridge and interface configuration in the linux kernel (see figure 5.10)

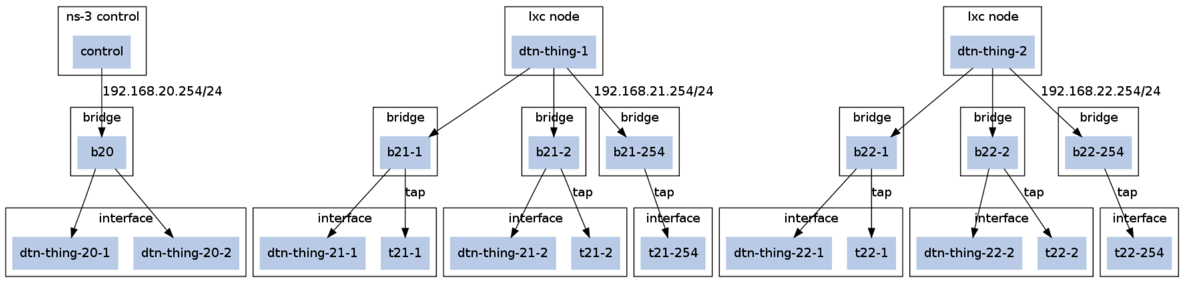


Figure 5.10: The host bridge and interfaces configuration to support Ns-3. The interfaces for experiment control and communication of two lxc nodes are shown. Each lxc node is assigned three bridges and control is assigned one.

The steps to create our experimental environment are listed in Table 5.10. These steps implement ns-3 as a real-time network emulator, and create of a number of lxc test nodes and test scripts.

No.	**Description**	**Class**	**Command**	**Table**
1	Create ns-3 emulation environment and template node	*ns3configer*	ns3_configer.py	5.11
2	Create two test nodes, the test configuration files and test initialisation scripts.	*emuModel*	ns3_lxc_configer.py 2	5.12

Table 5.10: Steps to create our experimental environment

For each node we create an ns-3 “channel”, which is a logical path over which information flows for emulation of WiFi. An ns-3 class “WifiHelper” is used to create a set of “WifiNetDevice” objects and to configure a set of their attributes during creation. All objects are configured for 802.11a to allow transmission and reception of data at rates of 1.5 to 54Mbit/s. The ‘WifiHelper” class is also used to set the remote station manager of each node to use constant rates for data and control transmissions using the ns-3 “ConstantRateWifiManager“ class which always uses the same transmission rate for every packet sent. The transmission mode to use for every data packet transmission is set to 54Mbit/s ($54x10^6$ bits per second). The ns-3 “NqosWifiMacHelper” class is used to create Ad-hoc WiFi MAC (MACs of type “AdhocWifiMac”). The ns-3 “YansWifiPhyHelper” class is used to manage and create wifi channel objects which implements the “yans” channel model (URL, 104). Although we do not alter the channel over time, this channel model can be used to alter attributes, such as TX power over time. As we are not emulating movement of the nodes the ns-3 “MobilityHelper” class is used to allocate a static position for each node. We use the “MobilityHelper” to set the ns-3 constant position mobility model.

No.	**Description**	**Class**
1	Install ns-3 requirement debian packages	ns3configer.installpackages()
2	Build ns-3 environment, in the current working directory by default, in “ns-3-allinone”	ns3configer.buildns3()
3	Create lxc template node from an existing template	ns3configer.createlxcnode()
4	Chroot lxc template node and install debian packages	ns3configer.chrootinstallpackages()
5	Create odbc.ini file	ns3configer.output_odbcini()
6	Create odbcinst file	ns3configer.output_odbcinst()
7	Create dtn.conf file	ns3configer.output_dtn_conf()
8	Create sudoers file	ns3configer.output_sudoers()
9	Create dtn rc file	ns3configer.output_dtnrc()
10	Add test user “dtnuser”	ns3configer.makeuser()
11	Create directory structure for DTN2	ns3configer.makedtndirs()
12	Assign appropriate permissions to test user	ns3configer.authuser()
13	Configure, build, and install oasys and DTN2	ns3configer.builddtn2()
14	Configure, build, and install	ns3configer.buildtdp()

Table 5.11: Emulator environment creation using the *ns3_configer* module

Main components of an lxc test node are shown (see figure 5.11. Each node has two interfaces for communication with other lxc test nodes (eth21 and eth22) which is controlled by ns-3 and a control interface.

No.	**Description**	**Function**
1	Create control script “dtn-thing.init.py”, emulator initialisation “startup.py”, emulator stop “shutdown.py”, and test emulation execute “runemu.py” scripts	createinitfiles() function of *ns3_lxc_configer*
2	Create ns3 config file “dtn-thing-lxc-cmsa-wifi-1.01.py”	makens3conf() function of *ns3_lxc_configer*
3	For each test node required, a node is cloned from the template node, reconfigured and we populate the test and config files	other functions of *ns3_lxc_configer*

Table 5.12: Test node and and script creation using functions of the *ns3_lxc_configer* module

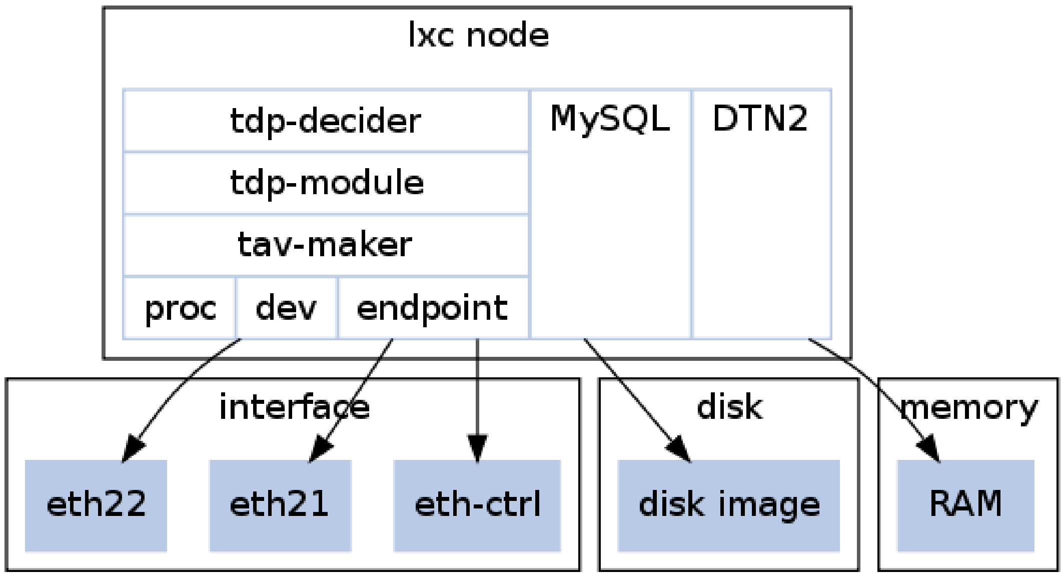


Figure 5.11: Main components of an lxc test node

Once our experimental environment has been created, an emulation is initialised, stopped and a test type is executed, either with TDP or without TDP via commands listed in Table 5.13. A script "phase3.py" automates the procedure for the contacts.

No.	Description	Command
1	Start environment	"sudo ./dtn-thing.init.py –start"
2	Stop environment	"sudo ./dtn-thing.init.py –stop"
3	Run an emulation without TDP	"sudo ./dtn-thing.init.py –run –con_id <CONTACT ID> –test <withTDP or withoutTDP>"

Table 5.13: Test node and and script creation using functions of the *ns3_lxc_configer* module

For each emulation and the contacts we generate an image. For each contact of the contacts we generate three associated images: one for the original contact, one for the emulated contact without TDP and one for the emulated contact with TDP.

Figure 5.12 shows bytes transferred between logging node and a peer node against time (in seconds) for a original contact view, with contact ID *a921c6c9-6229-4579-add1-12f3d782611f*.

Bundles transmitted by the logging node are denoted by "tx" in the legend, and bundles received by the logging node are denoted by "rx". Duplicate bundles transmitted by the logging node are denoted by "tx dup" and duplicate bundles received by the logging node are denoted by "rx dup". Red lines indicate the lower bounds and black lines indicate the higher bounds. A lower bound capacity is denoted "lb" and lower bound unique capacity is denoted "lbu". A higher bound capacity is denoted "hb" and higher bound unique capacity is denoted "hbu"

Figure 5.13 shows an ns-3 emulation of contact ID *a921c6c9-6229-4579-add1-12f3d782611f* which results in a new contact ID *cb41f1dd-d104-411c-a9b8-e08c6e179adb*

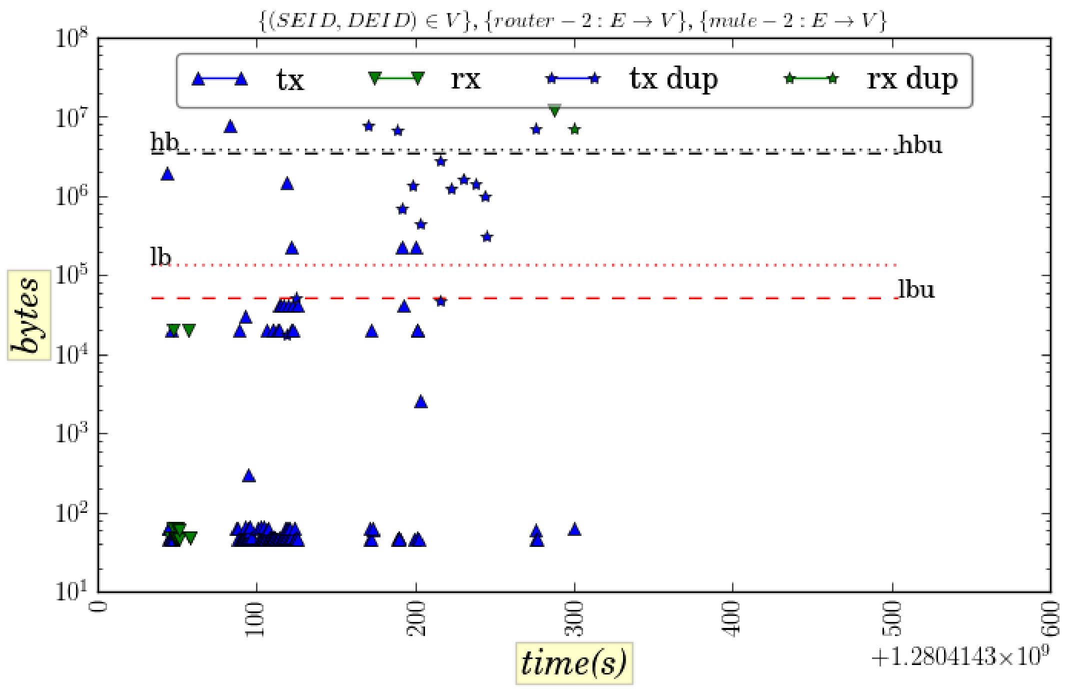


Figure 5.12: Bytes transferred for “original” contact with ID a921c6c9-6229-4579-add1-12f3d782611f. We can see the lower bound unique capacity is denoted “lbu” is lower than the lower bound capacity is denoted “lb”. Triangles indicate RX and TX events while stars indicate duplicates

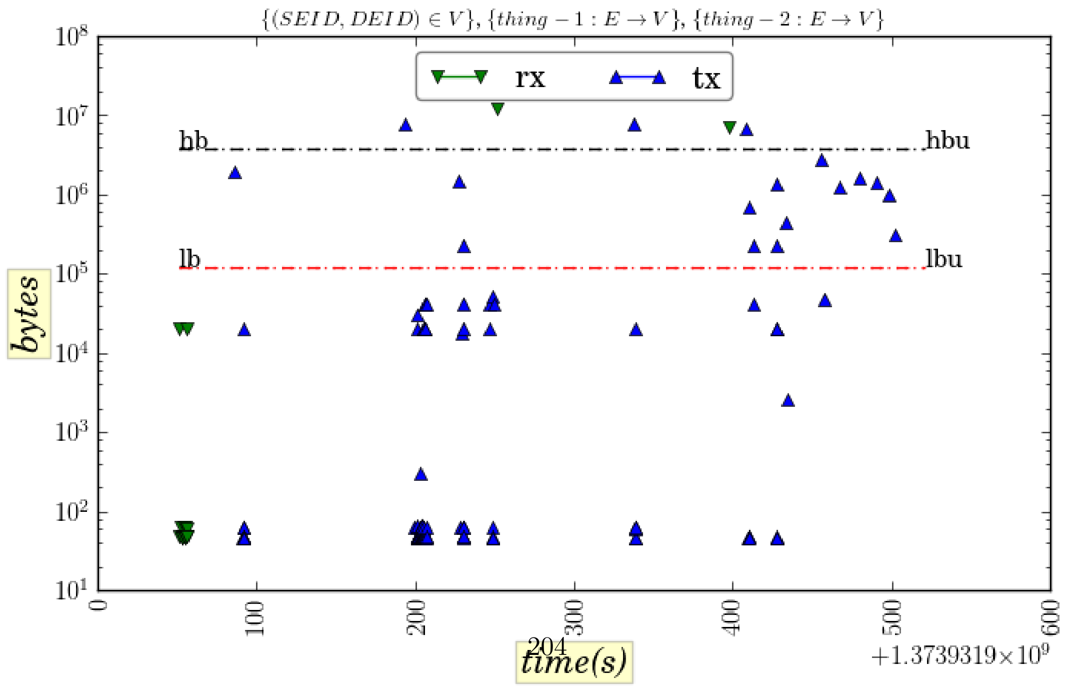


Figure 5.13: Bytes transferred for "emulated" contact ID cb41f1dd-d104-411c-a9b8-e08c6e179adb. We can see transfer of bundles is sparse. We can also see there are no duplicates and therefore the lower bound unique capacity is denoted "lbu" approximates the lower bound capacity is denoted "lb".

Figure 5.14 shows an ns-3 emulation of contact ID *a921c6c9-6229-4579-add1-12f3d782611f* with TDP which results in a new contact ID *4d41b891-b963-42f1-a419-628e1dee6643*

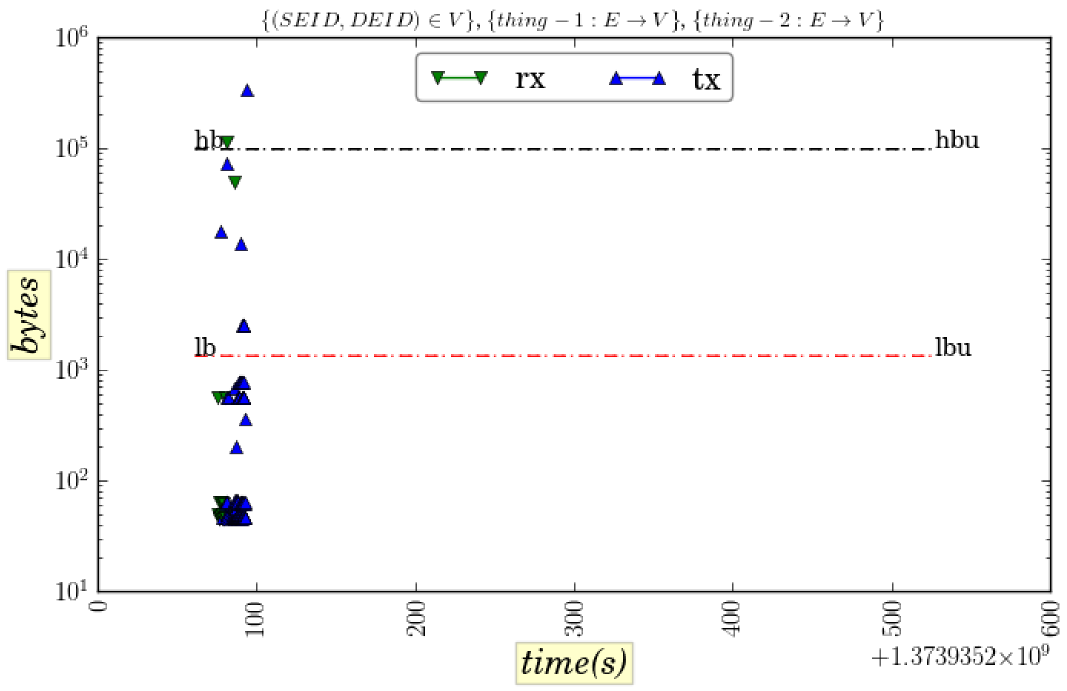


Figure 5.14: Bytes transferred for 'with TDP" emulated contact with ID 4d41b891-b963-42f1-a419-628e1dee6643. We can see transfer of bundles is dense and early in the contact. We can see there are no duplicates and therefore the lower bound unique capacity is denoted "lbu" approximates the lower bound capacity is denoted "lb". We can also see the bytes are generally lower than the "'original" contacts

5.5 Contact analysis of the N4C data

The 2010 trial included a 23 day "Unattended" phase in which it can be ascertained there was no interference from experimenters. We use the "Unattended" phase for our our evaluation.

5.5.1 Contacts Overview

A total of 942 unique contacts with a duration greater than zero were recorded between 2010-07-26 00:02:54 and 2010-08-17 16:55:35.

Enumeration of Contacts

The content of the DTN reference implementation log files which are useful to our evaluation are denoted *S10* log entries. The *S10* entries are the result of logging changes made to the RI code (URL, 95) for the N4C summer 2010 trials. Notably *S10* log entries facilitate tracing of a bundle across multiple nodes without debug logging by providing bundle creation time and makes generation of plots and doing statistics easier. Five events out of 14 events logged are of particular interest in describing the contacts; contact up *CONTUP*, contact down *CONTDOWN*, transmit *TX*, receive *RX* and delivered to application *DELIVERED.* Each distinct contact of each distinct link is enumerated by evaluating each line of a link table ordered by time and assigning a unique UUID to events that lie between and include a *CONTUP* and *CONTDOWN* event. In the case that an *RX*, *TX* or *DELIVERED* event occurs on a point to point link outside of pair of *CONTUP* and *CONTDOWN* events the UUID of the previous contact is attributed to it. Such a scenario is attributed to different events being logged by different threads

Enumeration of Links

For identification of source, destination and intermediary the *S10* logs contain a logging node EID, bundle source and destination EID and next hop peer IPv4 and TCP tuple. EIDs and IPv4 addresses are resolved to hostnames and enumerated. Distinct point to point links are extracted from the logs into distinct enumerated table "views"of the dataset. To do so a list of distinct peer hostnames and IPv4/TCP tuples is constructed for each of a set of distinct EIDs by which a node is uniquely identifiable and each of a set of distinct hostnames by which a a node is uniquely identifiable.

Transformation of Contact Names and Addresses

For each of contact event the identifier of the link from which it was extracted, a time stamp in nano seconds, source and destination EIDs by which source and destination are uniquely identifiable and the peer IPv4 address and TCP port number of the contact are known.

Transformation of Contacts

Each TX / RX / DELIVERED event is associated with an RFC 5050 bundle that is uniquely identifiable by a combination of its creation timestamp and sequence number. We extract the creation timestamp and sequence number of each bundle and enumerate the events. The size of bundles is known.

5.5.2 Subpopulation

In order to describe and evaluate the contacts we derive a subset of the contact population. We select contacts from the set of links established within the sample period, that fulfil criteria. Our selection criteria requires that a set of distinct links are extracted from DTN node logs. The distinct links list is a set limited to point to point links with the same number of contact up (CONTUP) events as contact down (CONTDOWN) events, at least one contact up (CONTUP) event and transmitted 'TX' or received 'RX' events. There are 291 point to point links that met this criterion. A list of distinct contacts are then extracted from the 291 distinct links. The distinct contacts list is bounded to those that have a duration greater than zero, a duration less than 20 minutes, and those that have transmitted or received bytes. There are 307 contacts which met the criteria

5.5.3 Population Sample Size

A confidence interval of 4 is used with the worst case percentage (50) to determine a general level of accuracy of the sample. This means that if 50 percent of the sample provides an answer we can be "sure" that the answer is representative of the entire relevant population between 46 and 54 percent. A 95 percent confidence level is used as it means that we can be 95 percent certain of how often the true percentage of the population that provides an answer lies within the confidence interval of 4. The sample size is calculated as 600.25 using the formula $(\frac{Z^2*(p)*(1-p)}{c^2})$. A correction for Finite Population of 203 is derived using the formula $(\frac{samplesize}{(1+((samplesize-1)/population))})$

5.5.4 Random Selection

203 contacts are randomly selected from the 307 contacts which met the criteria. These 203 contacts constitute the random sample taken from the population; a set of entities concerning which statistical inferences on contact exploitation are to be drawn

5.6 Experimental configuration

Section 5.4.4 describes how each table view of the *test contacts* set is then used to derive an estimation of the contact capacity volume lower and higher bounds, which along with the table view data, are used emulate each contact in real-time by using the *ns3_configer* and *ns3_lxc_configer* modules to implement ns-3 as a real-time network emulator .

The causes of a node's difficulty to handle an opportunistic contact in order to transfer data in the RHNC are summarised in Table 1.1.

5.6.1 Time varying information

Sections 1.4.2 to 1.4.6 provide the set of time varying information, that comprises the set of protocol requirements, TDP is required to transfer within the RHNC. The *TDP-maker* module provides this set of time varying information using the TDM classes, that comprises the set of protocol requirements, which the *TDP-module* can transfer within the RHNC. While this information is available, we use a limited subset of this information, from the “endpoint” TDP class, in our evaluation.

5.7 The emulations

We configure the *TDP-decider* module to act on TDP information from the data store. We require DTN2 bundle information, such as the bundles that are placed on active queues for transmission. Using this information we can determine the order of queued bundles. We require information on the bundle attributes, such as their DTN2 daemon identifier, payload, source EID, destination EID, reply EID, expiry and creation timestamps. Using the bundle creation timestamps we can determine which bundles in our queues are unique. Using a bundle’s DTN2 daemon identifier and creation timestamp we can instruct a BPA to expire a bundle.

We also require other DTN2 information such as our active registration EIDs and our unique EID. Using this information we are able to determine the ownership of things in our data store. We are able to distinguish our things.

As we have selected the subset of required things from the *endpoint* class, we can configure the *TDP-module* to exchange only this subset of endpoint class things. Furthermore, we can configure the TAV-maker module to obtain only this subset of endpoint class things.

5.7.1 Event scheduling

Each emulation has three sequentially executed stages. During the first stage, called “pre emulation”, we configure and then start MySQL, DTN2 (dtnd), DTN2 receive application (dtnrecv) daemon, *TAV-maker* module (tav-dtn) daemon. We then create the set of test payload files that are transmitted or received. Next we transmit bundles using the DTN2 application “dtnsend”. Two lxc nodes are involved in every test, called dtn-thing-1 and dtn-thing-2. The logging node of the contacts is always dtn-thing-1. As we are operating from the perspective of the logging node, an ’RX’ event bundle is transmitted, and subsequently queued on dtn-thing-2 and a ’TX’ event is transmitted, and subsequently queued on dtn-thing-1. Each contact is initiated from dtn-thing-1. If the emulation does not involve TDP the final ’TX’ bundle of dtn-thing-1 is scheduled to transmit at commencement of emulation. The reason for scheduling this final ’TX’ bundle is that doing so makes a call to the DTN2 BPA which brings the contact up (assuming there is an active link) on commencement of the emulation. This also makes the node cycle through its bundle queues, sending bundles in the order they are queued. Finally we schedule the events that will occur during emulation. These include the ’CONTUP’ and ’CONTDOWN’, the reconfiguration of link default timeouts for links which are established for longer than 30 seconds, and the “post emulation“ phase events.

During the second stage, called “emulation” the emulation is executed by ns-3. If the emulation uses TDP then the *TDP-module* and *TDP-decider* modules are started on commencement.

During the final “post emulation” phase the log files are extracted from each node and archived and the emulation is stopped.

Payload

We generate the payload for each bundle transferred on each lxc node during the “pre emulation” phase prior to test commencing. To do so we generate files of the exact ’RX’ and ’TX’ payload size of the contacts using randomly selected “dummy” text of the printing and typesetting industry “Lorem Ipsum”.

5.7.2 Emulation

The contacts are emulated firstly without TDP and secondly with TDP. In the following, we refer to the contacts as “original”, the emulations without TDP as “emulated” and the emulations with TDP as “with TDP”.

5.7.3 Correlation of contact IDs

Execution of an emulation creates a new contact set. We correlate tests using a table of identifiers. Each row has three columns. A column denotes the original contact and a column denotes emulations that have been executed without TDP. A column denotes emulations executed with TDP. Additionally for each row, three images are generated.

5.8 Results

The results of our emulations are presented in summary tables as per our original analysis of the contacts. We do this is to aid comparison of each contact. All of the generated logfiles required to reproduce the results derived from running the emulations described is located in the ns-3 “scratch” directory. Similarly all contact data are stored in files and the generated data is available as a MySQL database. Firstly we present the result without TDP and secondly with TDP.

5.8.1 Emulation results

Our ns-3 emulations of the contacts result in a set of 203 contacts called “emulated” and a set of 203 contacts called “with TDP” in order for statistical inferences on contact exploitation.

5.9 Summary

We provide graphs to aid our evaluation of the performance of TDP against that of the contacts, and against the set of emulations without TDP. In each of the following four summary graphs a vertical symbol indicates a contact. An important factor to note is that in DTNs a "creation timestamp is guaranteed to be unique for each ADU originating from the same source" (Cerf et al., 2007). As a consequence of this the generation of duplicate bundles at the source in the emulations is not possible. For the emulations with TDP in order to effectively generate duplicate bundles pre simulation we generate the creation timestamp for duplicates as the current time plus the sequence number, allowing identification of same. While in the emulations the creation time stamp of each bundle is unique, including duplicates.

For the "original" set we observe that a total of 5.32G of ADUs in 27718 bundles, of which 25591 were unique, were transmitted or received over a total contact duration of 51134.87 seconds, of which 9873.33 seconds were not utilised. For the "with TDP" set we observed that a total of 143.07M of ADUs in 26669 bundles, of which 26201 were unique, were transmitted or received over a total contact duration of 50146.33 seconds, of which 28941.37 seconds were not utilised. For the "emulated" set we observed a that total of 2.29G of ADUs in 16739 bundles, of which 16701 were unique, were transmitted or received over a total contact duration of 41819.21 seconds, of which 28317.34 seconds were not utilised. The 27718 bundles in the "original" set, 26669 in the "with TDP" set, and 16739 in the "emulated" set reflect the total count of bundles transferred. The contacts of Figure 5.15 are ordered by amount of bundles transferred. Consequently a vertical correlation between enumerated contacts doesn't necessarily indicate a common contact. Using TDP each contact incurred 406 extra TDP bundles which resulted in 8890 more unique bundles being transferred in the "with TDP" set then in the "original" set. The optimisation is a direct result of these extra TDP exchanges. As 25795 unique bundles were delivered along with the 406 TDP bundles, this optimisation due to removal of duplicate is demonstrated by TDP enabling transfer of 3.8% less bundles than original and over 37% more bundles than the "emulated" set. Contacts without duplicates are not optimised. The optimisation is more evident in contacts with greater than approximately 400 bundles as the number of duplicates can bee seen to increase. The "emulated" set transferred the lowest number of bundles, which are 10979 less bundles than the original set as can be observed in Figure 5.15. There are a number of reasons for this, firstly, as described above, all ns-3 wirless objects are configured for 802.11a, while the trial used a combination of mainly 802.11b and some 802.11a. Secondly, if the first event of a contact is an 'RX' event, the 'ALWAYSON' state of a link of the DTN2 TCP CLA does not set the DTN2 link to 'AVAILABLE'. When the TCP CLA is used by two nodes, the link of one node, the initiator, has a type of 'ALWAYSON' and the other has a state of 'OPPORTUNISTIC'. When a link is in state 'AVAILABLE', the link is closed but can be opened by the BPA. A DTN2 'CONTUP' event, i.e., which denotes the start of a contact, cannot occur unless a link is in an 'AVAILABLE' state. Finally, the DTN2 BPA is designed around a single event queue, in which bundles are queued in sequence pending timing events and processing. This final reason is the biggest contributor and numerous examples of processing delays can be observed.

The contacts of Figure 5.16 are ordered by all bytes transmitted and received. Consequently a vertical correlation between enumerated contacts doesn't necessarily indicate a common contact. The optimisation is a reduction in bytes transferred due to selective compression decisions. This graph reflects selective compression of 26669 bundles, of which 98% were unique, over a total reduced contact duration of 50146.33 seconds. From the graph we can see that TDP-decider, on average outperforms dtnsend for transfer of unique bundles and bandwidth usage. While all distinct bundles were transferred by the

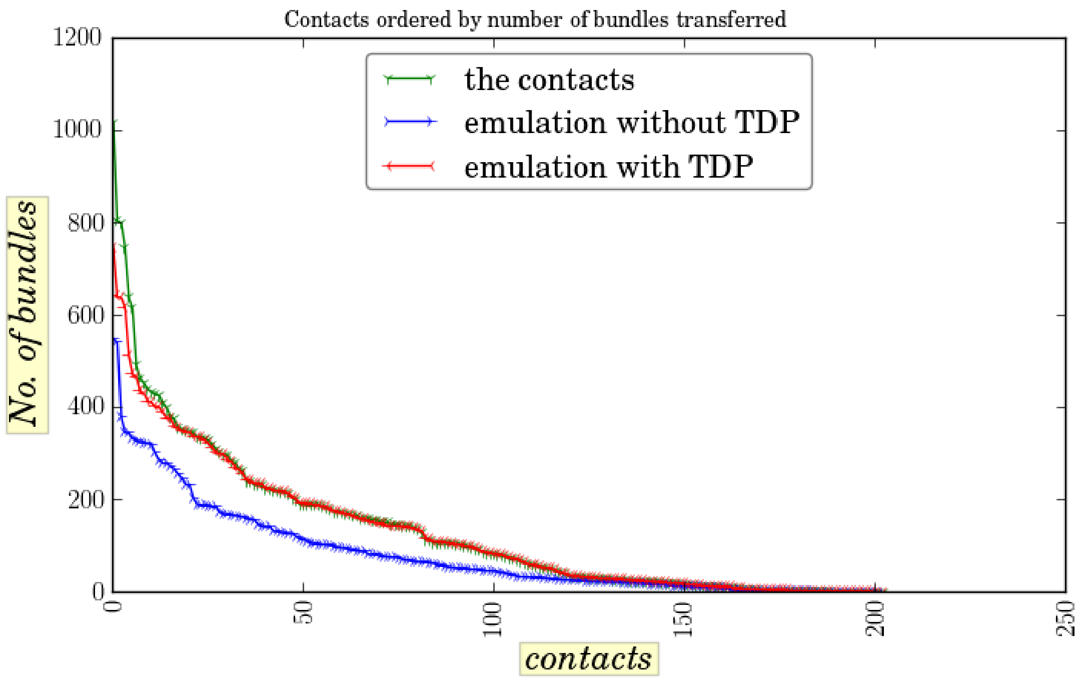


Figure 5.15: The figure plots each "original", "emulated" and "with TDP" contacts with a unique symbol matching the contact involved. Contact are enumerated on the horizontal axis and ordered by amount of all bundles transmitted and received.

"with TDP" set optimisation due to the compression can be seen to be effective from contacts transferring approximately $10^{4.5}$ bytes and above. The horizontal line which can be observed in the "original" and "emulated" plots, between enumerated contacts 160 to 180, is caused by a transfer of same size email application synchronisation messages. Also, while the contour of "emulated" plot closely maps to the "original" it appears out of phase as a consequence of all bundles of the "original" set not being successfully transferred by the "emulated" set as described above. Figure 5.16 shows that although the number of bundles is much lower, the bytes transferred of which there was 5.32G in the "original" set and 2.29G in the "emulated" set, a difference of 57.05 percent. These issues can be also observed in the contacts of the "emulated" set, which were generally under utilised with 28317 seconds "unused".

The graph in Figure 5.16 reflects selective compression of 26669 bundles of which 26201 were unique over a total reduced contact duration of 50146.33 seconds. From this graph we can see that TDP-decider, on average outperforms dtnsend for transfer of unique bundles and bandwidth usage.

The contacts of Figure 5.17 are ordered by the amount of unique bundles transmitted and received. Consequently a vertical correlation between enumerated contacts doesn't necessarily indicate a common contact. In contrast to the "emulated" plot, the "with TDP" plot of Figure 5.17 can be observed to map closely to the "original". This is due to the "with TDP" set transfer of all the distinct bundles

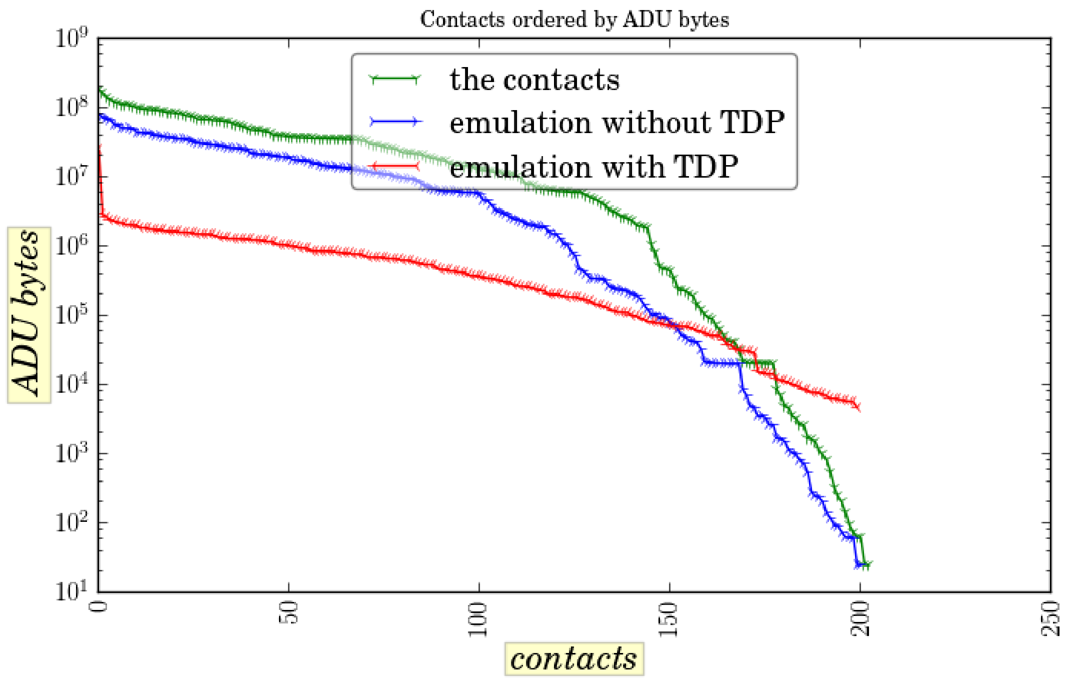


Figure 5.16: The figure plots each “original”, “emulated” and “with TDP” contacts with a unique symbol matching the contact involved. Contact are enumerated on the horizontal axis.

of the “original” set. The “with TDP” plot is slightly raised in comparison to the “original”. This is a consequence of the 406 extra TDP bundles incurred by the contacts of the “with TDP” which resulted in an optimisation of 8890 more unique bundles being transferred then in the “original” set.

We can observe in Figure 5.17 that due to the extra TDP bundles incurred for each contact, overall there were 8890 more unique bundles transferred in the “with TDP” set then in the “original” set. However, we can observe in Figure 5.15 as a direct result of these extra exchanges using TDP 25795 unique bundles were delivered along with 406 bundles. Overall using TDP transferred 3.8% less bundles than original and over 37% more bundles than in emulation due to removal of duplicates. The optimisation related to the number of bundles and those contacts with greater than 400 bundles have more duplicates.

This optimisation can also be observed as the unique bundles for each contact of the “with TDP” set are generally equal to the total amount of the total bundles. This is because TDP has discovered duplicates and removes them from the transmission queues.

We can see in Figure 5.18 that the “unused” contact duration of 28941 seconds (8 hours, 2 minutes, 21 seconds) achieved in the “with TDP” set is vastly increased as compared with the 9873 seconds “unused” of the “original” set. We attribute this to TDP clearing out all queues, and regenerating the bundles in new queue during the contacts. The contacts of Figure 5.18 are ordered by % of “unused” contact duration.

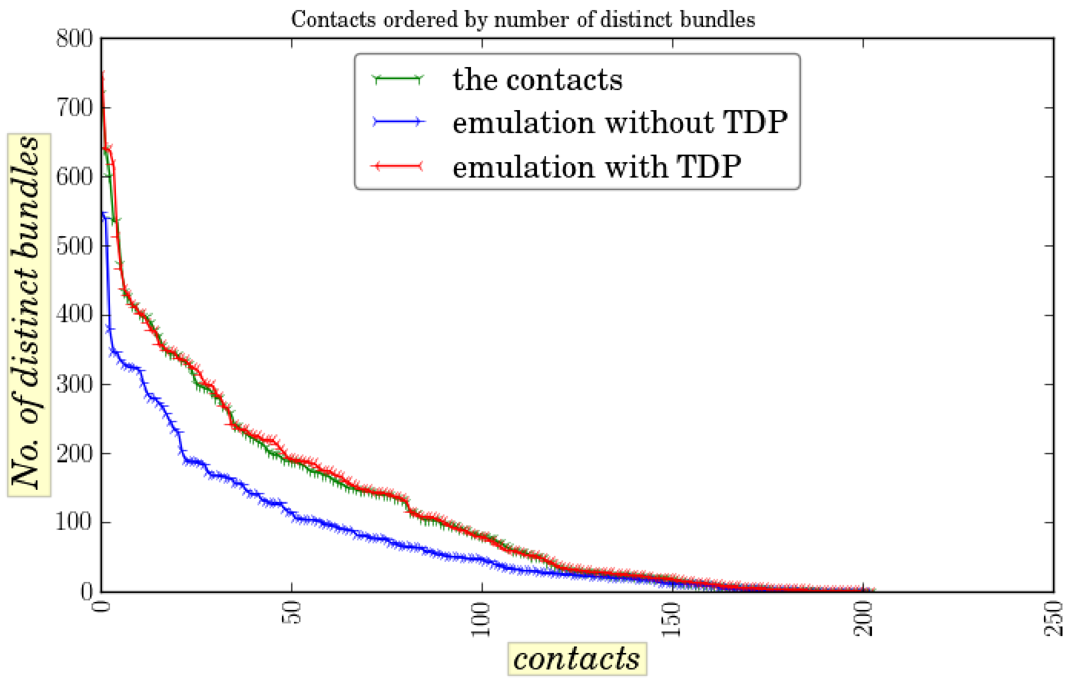


Figure 5.17: The figure plots each “original”, “emulated” and “with TDP” contacts with a unique symbol matching the contact involved. Contact are enumerated on the horizontal axis.

Consequently a vertical correlation between enumerated contacts doesn’t necessarily indicate a common contact. The optimisation is related to the net effect of TDP-decider’s choices of what to transmit and compress and in what order to place bundles on the queue for transmission and the synchronisation of their transmit timers. The optimisation can bee seen as an “unused” contact duration of 28941 seconds in the “with TDP” as compared with the 9873 seconds of the “original” set. The optimisation is a direct result of TDP clearing out all queues, and regenerating the bundles in new queue during the contacts. It is also important to note that there appears to be an optimisation for the “emulated” contacts. However, this is not an optimisation as the “unused” duration is a consequence of a reduction in the amount of bundles transmitted

In summary we have shown cases where nodes can use information provided by TDP to exploit contacts of the 2009, 2010 and 2011 N4C trial data, in order to transfer ADUs. We have shown cases where in the event of a contact, data transfer applications can transfer more distinct ADUs than applications that use existing discovery protocols. The over all number of bundles transferred can be reduced due to effective removal of duplicates using TDP. Compression choices using TDP can result in reduced traffic. Our approach can be used to equip a node with facts to exploit an opportunistic contact in radically heterogeneous networking contexts.

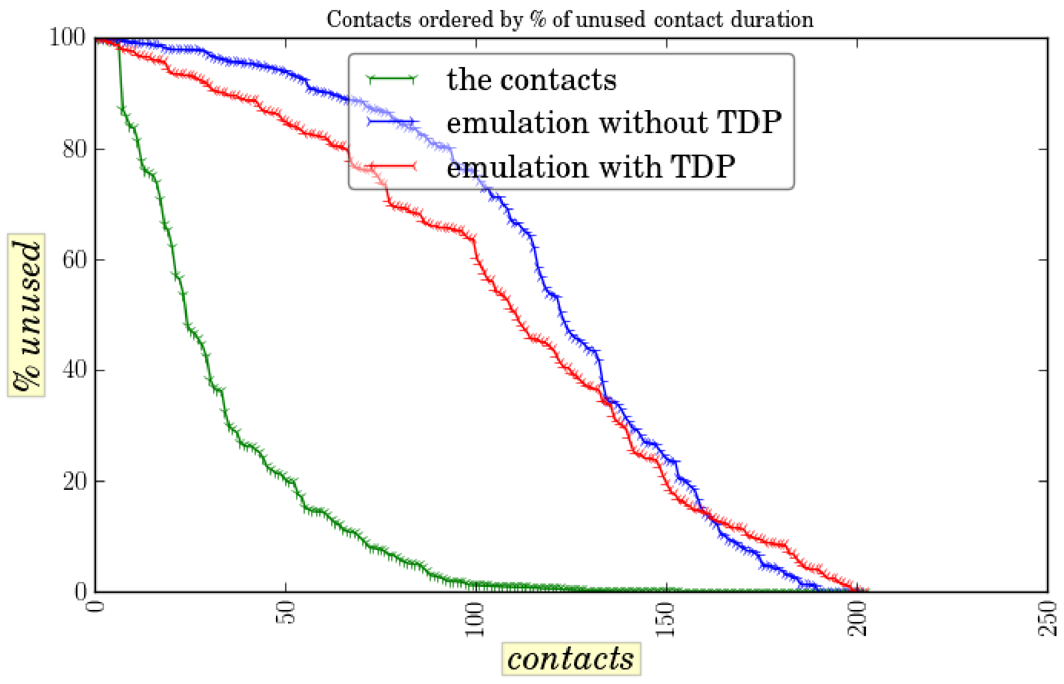


Figure 5.18: The figure plots the "original", "emulated" and "with TDP" contacts with a unique symbol matching the contact involved. Contact are enumerated on the horizontal axis. The % of "unused" contact duration is on the vertical axis.

The main things to note from the above graphs is that even though the *TDP-module* generates extra bundles i.e, at least two TDP bundles are transmitted per contact, due to effective removal of duplicates, the over all number of bundles is reduced. Although duplicates are detected and removed, the distinct bundle are successfully transmitted. The *TDP-decider* module examines the bundle, payload and queue information. The *TDP-decider* decides whether or not to delete or compress the payload of the existing bundle. Then *TDP-decider* creates the replacement bundle and places it on a weighted queue. While all of the bundles of the contacts were transferred, compression choices have resulted in reduced traffic. The net effect of *TDP-decider*'s choices of what to transmit and compress and in what order is that the percentage of unused contact duration is increased.

5.10 Meeting the context specific requirements

Receiving ADUs in certain contexts can be impossible if the communications protocols used by producer and consumer are unsuitable to those contexts. An inability to exploit a contact to transfer ADUs in the RHNC is a consequence of a lack of support of the RHNC detailed in Section 1.4.1. Communications protocols that support the RHNC can be used to exploit a contact in the RHNC. DTN has been demonstrated to provide appropriate communications to fulfil this set of requirements. DTN is the only Internet specification for enablement of a node to exploit an opportunistic contact in the RHNC, and consequently the ultimate choice. As our protocol uses DTN our context specific requirements are fulfilled.

The enablement of a node to exploit an opportunistic contact to transfer ADUs within the bounds of the RHNC (see section 1.4.2 through 1.4.6) is evaluated in Section 5.6.1.

5.11 Meeting the time varying information requirements

The TDM classes provide information the time varying information requirements using the *proc*, *dev*, and *endpoint* classes. We have observed the performance of TDP and are now ready to consider how each of the time varying information requirements of Chapter 1 are fulfilled by TDP.

5.11.1 R.2 resource consumption

The TDM classes provide information on node resources using the *proc* class to provide disk statistics (ThingName.diskstats), memory (ThingName.meminfo), disk partitions (ThingName.partitions), cpu (ThingName.loadavg, ThingName.cpuinfo, and ThingName.stat), and power (ThingName.acpi). Application resource consumption information, such as the state of a BPAs queues or bundle expiry, can be obtained using the *endpoint* class. We demonstrate *TDP-decider*'s ability to use TDM to examine bundle payload sizes, compress bundles using an appropriate scheme, and evaluate whether there is an improvement in doing so.

5.11.2 R.3 peer capabilities

We have demonstrated use of the *endpoint* class to provide detailed higher layer information, supplied by the BPA, on queued bundles. *TDP-decider* uses this information to determine if a peer node has a bundle in common, and if so select whether or not to transmit. The *TDP-decider* module also determines the source EID of the bundle in order to establish whether or not the peer node is the origin (Attribute.dataType.bundle). More generally, requirement R.3 is fulfilled by a combination of *proc*, *dev*, and *endpoint*. Examples include provision of BPA, bundle and link usages statistics (Attribute.dataType.daemon statistics, Attribute.dataType.bundle statistics, Attribute.dataType.link statistics), available protocols (ThingName.net/protocols) and wireless capabilities (ThingName.net/wireless). It should be noted that as ns-3 emulates wireless connectivity using ns-3 "channels" which correspond to a local Ethernet interface on each lxc node, the interface data supplied by *proc*, *dev* is for Ethernet devices and not wireless devices. This means that in emulation, and unlike a "real" world entity, layer 2 wireless information is not available.

5.11.3 R.4 network element relationships

This is demonstrated by the *TDP-decider* modules ability to use TDM to destroy and recreate bundles. Creating bundles requires facts such as the source and destination EID, payload location on a file system, and bundle ID which will be obsoleted by the bundle. The *TDP-decider* module discovers the relationships between networks elements, in this case the relationships between a data representation persisted to storage and a distinct or duplicate bundle. Furthermore, the *TDP-decider* module rates queued data awaiting transfer by determining whether data is duplicate, or whether it is of higher priority, such as a TDP bundle which is queues as the first transmitted bundle. he *TDP-decider* determines the nature of the DTN in order to expire bundles using the BPA ID of the bundle which is required to do so.

5.11.4 R.5 routing information

We have demonstrated the *TDP-decider* modules use of measurements based on the known state of queues of other DTN nodes. We have also demonstrated how *TDP-decider* can structure message information for reordering and prioritisation of bundles for transfer and persist structured information using facts about the messages. Furthermore, TDP does not restrict the lifetime of things: it is the choice of an implementer to make available long or short term metrics. We do not specifically address preemption of transfer opportunities. However, one could envisage using the historic information for calculating path computation, the probability of link availability, or prioritisation based on prediction. The *TDP-module* broadcasts discovery messages to the TDP registration. Prior to each transmission the *TDP module* examines the general connectivity information in order to distribute TDP messages throughout a network. If a new path (link) has been added or removed the module will construct a new route or remove failed paths. As our emulations reproduce contacts of the N4C trials we do not specifically address use of other specific discovery protocol mechanisms for discovery of proximate nodes. However, using the *endpoint* class as a template one could envisage use of information from any discovery protocol. Similarly, as our emulations involve single contacts, contact pattern information is not used. However, one could envisage a mechanism to create things for each of the contact tables generated by the automated *Analysis* modules.

5.11.5 R.6 distributed resources

We have demonstrated the *TDP-module* module's ability to distribute copies of structured information amongst the potential relays. We forced bundles to not fragment during the emulations in order to aid comparison with the original data. Nevertheless TDP is capable of fragmentation as TDP is a DTN application, and DTN is capable of proactive and reactive fragmentation. Furthermore by demonstrating TDP's ability to work with bundles and their metadata, we have demonstrated TDP's ability to work with fragment bundles as a fragment bundle is itself a bundle. We have demonstrated TDP's ability to broadcast to an EID and consequently to an indirection point, which in a DTN is an EID. We have demonstrated the *TDP-decider* module's ability to distinguish things which it is the "owner" from thing which it is not. Using the same mechanisms *TDP-decider* can distinguish the things for previous contacts and resources of a set of nodes.

5.12 Conclusions

In this chapter we documented our evaluation of TDP, which used the *TDP-module*, *TAV-maker*, and *TDP-decider* modules, the proof-of-concept ns-3 discrete-event network simulator as a real-time network emulator, and specific comparative tests of TDP against DTN data transfer applications. We saw TDP meets almost all of the requirements for a DTN discovery protocol that we described in Chapter 2. As a consequence we conclude the TDP conceptual models, TDP and the TDP modules, are reasonable to base further DTN experiments.

Where duplicate bundles are a feature of a DTN, TDP will be a better option than the other discovery protocols for discovering them. Also we can go further and state that where reordering and reprocessing queued bundles is a requirement to exploit a contact, TDP will also be a better option than the other discovery protocols. Note that while we do not intend our characterisations of these contexts in which TDP is a better option is complete, these statements reflect the evidence for using TDP that we have demonstrated.

We have defined an application discovery protocol that can be used to exploit an opportunistic contact in order to transfer distinct ADUs in the RHNC. We have also described models used by nodes to organise, transfer and persist the facts about themselves. Furthermore, the architectural constraints, exemplified by TDP, the TDP system and evaluation system architecture modules, when applied as a whole emphasise scalability of loosely coupled systems.

Chapter 6

Conclusions and future work

This thesis presented our approach used to discover facts to enable a node exploit an opportunistic contact in order to transfer distinct application data units (ADUs). Our approach was captured in the definitions of a set of conceptual models, modules, and the TDP, a DTN application discovery protocol. This chapter summarises the most significant contributions of the work described in this thesis and assess its contribution to the state of the art. In addition, suggestions for future work are outlined.

6.1 The contributions

The motivation for the work described in this thesis arose from investigations on the exploitation of less predictable communications opportunities by radically heterogeneous nodes. Such nodes require information in order to exploit contacts, make intelligent routing decisions, trigger message forwarding or make effective scheduling decisions. Routing schemes and data transfer applications can use the information provided by application discovery protocols. We defined the problem in more detail by examining the limitations of an opportunistic contact in the RHNC, availability of time varying node data, and their effect on a nodes ability to transfer ADUs.

A review of existing work on discovery protocols demonstrated that existing work does not completely address the problems tackled in this thesis. There has not yet been a demonstration of the enablement of a node to exploit an opportunistic contact in the RHNC to transfer more ADUs than a node using other discovery protocols. In particular, A number of the projects reviewed, in particular in the IEEE, IETF and IRTF communities, provide discovery mechanisms mostly focused on near real-time, low overhead, discovery of network elements. The family of *IEEE 802* standards and their architectures assume continuous connectivity and negligible latency. For this reason, this thesis, build on protocols specified to tolerate network partition and significant latency. The naming problem is particularly challenging, for this reason this thesis builds on existing work in the IETF to classify a naming facility. Furthermore, many mechanisms provide complex means for the binding of ADUs, or services with names that typically assume underlying communications mechanisms. This is an assumption that is too restrictive and does not solve our problem completely. Provision of a format for generic interchange is also challenging, for this reason this thesis builds on an existing protocols with an appropriate specification for decoupling the representation from the processing program, the transport layer and use of sing for many to many relationships.

We noted that all mechanisms specify a structure of information, and further summarised the characteristics of reviewed discovery mechanisms as operations, structured information, mechanisms, persistence and an exchange format. We then described the hypotheses on which TDP builds that is captured in naming, data and storage models, and system architecture, captured in the *TDP-module*, *TAV-maker* modules and the module which makes decisions based on TDM data, *TDP-decider*.

We then described the modules *ns3_configer* and *ns3_lxc_configer*, required to enable use of ns-3 as a real-time network emulator and the *Analysis* module that extracts information for ns-3 from DTN2 logs. To evaluate TDP, we implemented it encapsulated in RFC 5050 bundles, in order to understand whether mechanisms of a node that utilise information provided by TDP can exploit a contact, as exemplified by contacts described by the 2009, 2010 and 2011 N4C trial data, in order to transfer ADUs.

The usefulness of our hypothesis on which TDP builds was evaluated using the *TDP-module*, *TAV-maker*, and *TDP-decider* modules, by way of the ns-3 modules, *ns3_configer* and *ns3_lxc_configer*, and the automated *Analysis* module. The evaluation demonstrated our discovery approach which can be used by a node to discover facts to exploit a contact, in order to transfer more ADUs than a node using facts discovered by other approaches. Our design of the TDP conceptual models, TDP and the TDP modules, is the substantive contribution of this thesis to the area of DTN.

6.2 Future work

We do not state the the models presented in this thesis are complete but we assert that they serve mostly as a starting point for research in discovery in the RHNC. Therefore, we outline areas where we plan further work.

Formal definitions of TDP and its naming, data and storage models, and formal proofs of their correctness are desirable in order to provide the maximal freedom for implementations along with a coherent meaning

An area that could be extend or improved, is the selection of generic tools used for persisting and querying data. Currently Information is extracted into associative arrays, which are then persisted to the tables of a MySQL database that contain the EAV/CR schema. However, other approaches such as using a distributed "key/value" store or data structure server, such as "Redis" (URL, 87), could be used to build a complete system for TDM to store things as scalars, lists, hashes and sets. The five data structures of Redis each have a key and a value where values represent the actual data associated with the key, which can be anything. Furthermore, an indexing engine such as Lucene (URL, 88) could be used to index TDM data.

The Data model move from EAV to a "MapReduce"(White, 2009) system in which the "Map()" procedure might performs filtering and sorting, such as sorting endpoint elements by name into queues, one queue for each name. A "Reduce()" procedure might then be used to performs a summary operation, such as counting the number of endpoint elements in each queue, and yielding name frequencies. Alternatively a "get-map-prereduce-reduce" procedure of riak pipe (Fink, 2012) as opposed to the "get-map-reduce" procedure might be used to acquire a desirable set of things.

Similarly, the processing of the JSON interchange format could be investigated further, for example it could an represented in an efficient binary serialization format, using msgpack (URL, 105).

We can use our ns-3 modules, *ns3_configer* and *ns3_lxc_configer* to further analyse the performance of TDP. For example we could investigate use of the time varying information listed in Sections 1.4.2 through 1.4.6 to analyse how to distribute fragments for a distributed cache or the effects of fragmentation, in order to exploit contacts.

In addition, while the classes of thing defined in *TAV-maker*, for thing naming, data and storage modelling, allow us to solve numerous applications, their scope is limited, and could be extended in the following ways:

- using the template structures to develop new translators, define more classes of thing to capture more physical or abstract network elements in order to generate TAV structured information, such as system log files.
- as a possible extension to TDM a class could be specified to transform information of the contact analysis module into TDM data in order to use historic information to base decisions.

- extend the set of basic methods which currently can only be used to process list data, physical type representation, and process abstraction type operating system query data.
- extend the capabilities to correlate common elements in order to enable the representation of relationships to be automatically generated and removed. For example, algorithms could be specified to mine TDM data and establish relationships automatically, which could provide decision support for appropriate endpoint selection.

The TDP models may be appropriate for use in other non DTN contexts than described. The architectural constraints of the approach has demonstrated, that when applied as a whole, loosely coupled systems are scalable. Such scalability is applicable in diverse areas, for example, if applied to phases of the "e-discovery" (URL, 86) to enable organisations comply with rules concerning how to identify electronically stored information *ESI*. However, while the thing model can be used to design things for many contexts, interaction with the things are assumed within the described context and their interactions with other context are not covered by this work.

The issues of security and policy, which are not in the scope of this thesis and were consequently not considered could be examined in detail. The authenticity, validation and encoders of the association mechanisms of the naming model, described in Section 2.3.2 and Table 3.3, could be extended using existing full Python bindings for GSSAPI and the Python wrapper for OpenSSL, *M2Crypto* (URL, 89).

As an ultimate goal of this investigation to make a better Internet through eventual incorporation of TDP, the TDP conceptual models, and the TDP modules features and capabilities into the standard suite of Internet protocols, we will determine whether there is wider interest in our approach in the DTNRG. As this thesis seeks to develop protocols that endorse the global testing of those protocols we intend to publish TDP as an Internet Draft.

Bibliography

(1989, dec). Information processing systems - local area networks - part 2: logical link control. *ISO Std 8802-2: 1998; IEEE Std 802.2-1998*.

(2005). Ieee standard for information technology - telecommunications and information exchange between systems - local and metropolitan area networks - specific requirements. - part 15.1: Wireless medium access control (mac) and physical layer (phy) specifications for wireless personal area networks (wpans). *IEEE Std 802.15.1-2005 (Revision of IEEE Std 802.15.1-2002)*, 0–1.

(2008). *A DATA-ORIENTED NETWORK ARCHITECTURE.* Ph. D. thesis.

(2009a, 17). Ieee standard for local and metropolitan area networks– station and media access control connectivity discovery. *IEEE Std 802.1AB-2009 (Revision of IEEE Std 802.1AB-2005)*, 1–190.

(2009b, 21). Ieee standard for local and metropolitan area networks- part 21: Media independent handover. *IEEE Std 802.21-2008*, –1.

(2010a). n4c-wp2-012-state-of-the-art-11.

(2010b). n4c-wp2-022-pymail-06.

(2012a, 17). Ieee standard for air interface for broadband wireless access systems. *IEEE Std 802.16-2012 (Revision of IEEE Std 802.16-2009)*, 1–2542.

(2012b, 29). Ieee standard for information technology–telecommunications and information exchange between systems local and metropolitan area networks–specific requirements part 11: Wireless lan medium access control (mac) and physical layer (phy) specifications. *IEEE Std 802.11-2012 (Revision of IEEE Std 802.11-2007)*, 1–2793.

(2013). Dtnrg code repositories. `http://www.dtnrg.org/code/`. Accessed 04/04/2013.

Abdelkader, M., M. Hamdi, and N. Boudriga (2012). Unifying identity management for fourth generation wireless networks. *Communications, IET 6*(18), 3222–3230.

Abdulla, M. and R. Simon (2007, March). The impact of the mobility model on delay tolerant networking performance analysis. In *Proc. 40th Annual Simulation Symposium ANSS '07*, pp. 177–184.

Adjie-winoto, W., E. Schwartz, and H. Balakrishnan (1999). An architecture for intentional name resolution and application-level routing. Technical report.

Adjie-Winoto, W., E. Schwartz, H. Balakrishnan, and J. Lilley (2000, April). The design and implementation of an intentional naming system. *SIGOPS Oper. Syst. Rev. 34*(2), 22.

Ahlgren, B., M. D'Ambrosio, M. Marchisio, I. Marsh, C. Dannewitz, B. Ohlman, K. Pentikousis, O. Strandberg, R. Rembarz, and V. Vercellone (2008). Design considerations for a network of information. In *Proceedings of the 2008 ACM CoNEXT Conference*, CoNEXT '08, New York, NY, USA, pp. 66–1. ACM.

Ahlgren, B., C. Dannewitz, C. Imbrenda, D. Kutscher, and B. Ohlman (2012, july). A survey of information-centric networking. *Communications Magazine, IEEE 50*(7), 26–36.

Ali Ghodsi, Teemu Koponen, J. R. P. S. S. S. (2012). Naming in content-oriented architectures.

Aoki, M., M. Saito, H. Aida, and H. Tokuda (2003). Anarch: a name resolution scheme for mobile ad hoc networks. In *Advanced Information Networking and Applications, 2003. AINA 2003. 17th International Conference on*, pp. 723–730.

Atkinson, R., S. Bhatti, and S. Hailes (2010). Evolving the internet architecture through naming. *Selected Areas in Communications, IEEE Journal on 28*(8), 1319–1325.

Azad, S., P. Casari, and M. Zorzi (2012, may). Coastal patrol and surveillance networks using auvs and delay-tolerant networking. In *OCEANS, 2012 - Yeosu*, pp. 1–8.

Aziz, A. and S. Yamada (2012, may). A handoff-based and limited flooding (half) routing protocol in dtn. In *Informatics, Electronics Vision (ICIEV), 2012 International Conference on*, pp. 1220–1225.

Baid, A., T. Vu, and D. Raychaudhuri (2012, march). Comparing alternative approaches for networking of named objects in the future internet. In *Computer Communications Workshops (INFOCOM WKSHPS), 2012 IEEE Conference on*, pp. 298–303.

Balakrishnan, H., K. Lakshminarayanan, S. Ratnasamy, S. Shenker, I. Stoica, and M. Walfish (2004). A layered naming architecture for the internet. In *Proceedings of the 2004 conference on Applications, technologies, architectures, and protocols for computer communications*, SIGCOMM '04, New York, NY, USA, pp. 343–352. ACM.

Balasubramanian, A., B. Levine, and A. Venkataramani (2007a, August). Dtn routing as a resource allocation problem. *SIGCOMM Comput. Commun. Rev. 37*(4), 373–384.

Balasubramanian, A., B. Levine, and A. Venkataramani (2007b). Dtn routing as a resource allocation problem. In *Proceedings of the 2007 conference on Applications, technologies, architectures, and protocols for computer communications*, SIGCOMM '07, New York, NY, USA, pp. 373–384. ACM.

Ballani, H., P. Francis, T. Cao, and J. Wang (2009). Making routers last longer with viaggre. In *Proceedings of the 6th USENIX symposium on Networked systems design and implementation*, NSDI'09, Berkeley, CA, USA, pp. 453–466. USENIX Association.

Baran, P. (1964, march). On distributed communications networks. *Communications Systems, IEEE Transactions on 12*(1), 1–9.

Bari, M. F., S. Rahman Chowdhury, R. Ahmed, R. Boutaba, and B. Mathieu (2012, december). A survey of naming and routing in information-centric networks. *Communications Magazine, IEEE 50*(12), 44–53.

Baugher, M., B. Davie, A. Narayanan, and D. Oran (2012, march). Self-verifying names for read-only named data. In *Computer Communications Workshops (INFOCOM WKSHPS), 2012 IEEE Conference on*, pp. 274–279.

Beitollahi, H. and G. Deconinck (2008, dec.). Dependable overlay networks. In *Dependable Computing, 2008. PRDC '08. 14th IEEE Pacific Rim International Symposium on*, pp. 104–111.

Beitollahi, H. and G. Deconinck (2009, nov.). Making overlay networks more robust to massive failures. pp. 214–221.

Belbekkouche, A., M. M. Hasan, and A. Karmouch (2012, quarter). Resource discovery and allocation in network virtualization. *Communications Surveys Tutorials, IEEE 14*(4), 1114–1128.

Benamar, N., M. Benamar, and J. M. Bonnin (2012, may). Routing protocols for dtn in vehicular environment. In *Multimedia Computing and Systems (ICMCS), 2012 International Conference on*, pp. 589–593.

Berger, S., S. McFaddin, C. Binding, C. Hoertnagl, and A. Ranganathan (2004). Towards pluggable discovery frameworks for mobile and pervasive applications. In *Mobile Data Management, 2004. Proceedings. 2004 IEEE International Conference on*, pp. 308–319.

Berners-Lee, T., R. Fielding, and L. Masinter (1998, August). Uniform Resource Identifiers (URI): Generic Syntax. RFC 2396 (Draft Standard). Obsoleted by RFC 3986, updated by RFC 2732.

Berners-Lee, T., R. Fielding, and L. Masinter (2005, jan). Uniform resource identifier (uri): Generic syntax. Technical Report 3986.

Bierman, A. and K. Jones (2000, September). Physical Topology MIB. RFC 2922 (Informational).

Blanchet, M. (2011, May). Delay-Tolerant Networking Bundle Protocol IANA Registries. RFC 6255 (Informational).

Bohman, D., M. Frank, P. Martini, and C. Scholz (2004). Performance of symmetric neighbor discovery in bluetooth ad hoc networks. In *GI Jahrestagung (1)'04*, pp. 138–142.

Bong, Z. and C. K. Yeo (2011, sept.). Dtn serial convergence layer with multiple access control amp; neighbour discovery. In *Vehicular Technology Conference (VTC Fall), 2011 IEEE*, pp. 1–5.

Booch, G. (1994). *Object-oriented analysis and design with applications (2nd ed.).* Redwood City, CA, USA: Benjamin-Cummings Publishing Co., Inc.

Brandt, A., J. Buron, and G. Porcu (2010, apr). Home automation routing requirements in low-power and lossy networks.

Broch, J., D. A. Maltz, and D. B. Johnson (1999). Supporting hierarchy and heterogeneous interfaces in multi-hop wireless ad hoc networks.

Buford, J., B. Burg, E. Celebi, and P. Frankl (2006, july). Sleeper: A power-conserving service discovery protocol. In *Mobile and Ubiquitous Systems - Workshops, 2006. 3rd Annual International Conference on*, pp. 1–9.

Burgess, J., B. Gallagher, D. Jensen, and B. N. Levine (2006). Maxprop: Routing for vehicle-based disruption-tolerant networks. In *In Proc. IEEE INFOCOM.*

Burleigh, S. (2007, jan). Interplanetary overlay network: An implementation of the dtn bundle protocol. In *2007 4th IEEE Consumer Communications and Networking Conference*, pp. 222–226. IEEE.

Burleigh, S. (2011, May). Compressed Bundle Header Encoding (CBHE). RFC 6260 (Experimental).

Burleigh, S., M. Ramadas, and S. Farrell (2008, sep). Licklider transmission protocol - motivation.

Burleigh, S. C. (2006). Jpl's bundle protocol implementation : interplanetary overlay network (ion). *Pasadena, CA : Jet Propulsion Laboratory, National Aeronautics and Space Administration, 2006.*.

Bush, R. and D. Meyer (2002, dec). Some internet architectural guidelines and philosophy.

Cai, T., S. Ju, J. Zhao, and W. Zhong (2006, oct.). Self-certifying object-based network storage. In *Grid and Cooperative Computing Workshops, 2006. GCCW '06. Fifth International Conference on*, pp. 419–423.

Cain, B., S. Deering, I. Kouvelas, B. Fenner, and A. Thyagarajan (2002, October). Internet Group Management Protocol, Version 3. RFC 3376 (Proposed Standard). Updated by RFC 4604.

Campo, C., M. Munoz, J. C. Perea, A. Mann, and C. Garcia-Rubio (2005, march). Pdp and gsdl: a new service discovery middleware to support spontaneous interactions in pervasive systems. In *Pervasive Computing and Communications Workshops, 2005. PerCom 2005 Workshops. Third IEEE International Conference on*, pp. 178–182.

Carpenter, B. (1996, jun). Architectural principles of the internet. Updated by RFC 3439.

Carroll, L. *Alice's adventures in wonderland / by Lewis Carroll.* London : MacMillan, 1872.

Carzaniga, A., M. J. Rutherford, and A. L. Wolf (2004). A routing scheme for content-based networking. In *INFOCOM 2004. Twenty-third AnnualJoint Conference of the IEEE Computer and Communications Societies*, Volume 2, pp. 918–928.

Case, J., R. Mundy, D. Partain, and B. Stewart (2002, December). Introduction and Applicability Statements for Internet-Standard Management Framework. RFC 3410 (Informational).

Castignani, G., A. Arcia, and N. Montavont (2011, mar). A study of the discovery process in 802.11 networks. *SIGMOBILE Mob. Comput. Commun. Rev. 15*(1), 25–36.

Castro, M., P. Druschel, A. Ganesh, A. Rowstron, and D. S. Wallach (2002). Secure routing for structured peer-to-peer overlay networks. *SIGOPS Oper. Syst. Rev. 36*(SI), 299–314.

Cerf, V., S. Burleigh, A. Hooke, L. Torgerson, R. Durst, K. Scott, K. Fall, and H. Weiss (2007, apr). Delay-tolerant networking architecture. Technical Report 4838.

Cerf, V. and R. Kahn (1974, may). A protocol for packet network intercommunication. *Communications, IEEE Transactions on 22*(5), 637–648.

Cerf, V. G. (2013). Abstraction, federation, and scalability. *Internet Computing, IEEE 17*(1), 96–3.

Chakraborty, D., A. Joshi, Y. Yesha, and T. Finin (2002). Gsd: a novel group-based service discovery protocol for manets. In *Mobile and Wireless Communications Network, 2002. 4th International Workshop on*, pp. 140–144.

Charron-Bost, B. and A. Schiper (2007, mar). Harmful dogmas in fault tolerant distributed computing. *SIGACT News 38*(1), 53–61.

Cheng, L. (2005, april). Lightweight service advertisement and discovery in mobile ad hoc networks. In *Wireless and Optical Communications, 2005. 14th Annual WOCC 2005. International Conference on*, pp. 46.

Cheriton, D. R. and M. Gritter (2000). Triad: A scalable deployable nat-based internet architecture. Technical report.

Chi, J., Y. Zhang, J. Peng, H. Tang, and X. Zhou (2011, oct.). Design and analysis on routing by hierarchical and flat names. In *Pervasive Computing and Applications (ICPCA), 2011 6th International Conference on*, pp. 452–457.

Choi, B. J. and X. Shen (2010, may). Distributed clock synchronization in delay tolerant networks. In *Communications (ICC), 2010 IEEE International Conference on*, pp. 1–6.

Choi, J., J. Han, E. Cho, H. Kim, T. Kwon, and Y. Choi (2009, oct.). Performance comparison of content-oriented networking alternatives: A tree versus a distributed hash table. In *Local Computer Networks, 2009. LCN 2009. IEEE 34th Conference on*, pp. 253–256.

Choi, J., J. Han, E. Cho, T. Kwon, and Y. Choi (2011, march). A survey on content-oriented networking for efficient content delivery. *Communications Magazine, IEEE 49*(3), 121–127.

Chowdhury, S. I., W.-I. Lee, Y.-S. Choi, G.-Y. Kee, and J.-Y. Pyun (2011, oct.). Performance evaluation of reactive routing protocols in vanet. In *Communications (APCC), 2011 17th Asia-Pacific Conference on*, pp. 559–564.

Chuang, P.-J., P.-H. Yen, and T.-Y. Chu (2012, march). Efficient route discovery and repair in mobile ad-hoc networks. In *Advanced Information Networking and Applications (AINA), 2012 IEEE 26th International Conference on*, pp. 391–398.

Cicconetti, C., F. Galeassi, and R. Mambrini (2010, dec.). Network-assisted handover for heterogeneous wireless networks. In *GLOBECOM Workshops (GC Wkshps), 2010 IEEE*, pp. 1–5.

Clare, L., S. Burleigh, and K. Scott (2010, mar.). Endpoint naming for space delay / disruption tolerant networking. pp. 1–10.

Clark, D., R. Braden, A. Falk, and V. Pingali (2003, August). Fara: reorganizing the addressing architecture. *SIGCOMM Comput. Commun. Rev. 33*(4), 313–321.

Colella, R., R. Callon, E. Gardner, and Y. Rekhter (1994, May). Guidelines for OSI NSAP Allocation in the Internet. RFC 1629 (Draft Standard).

Cooper, D., S. Santesson, S. Farrell, S. Boeyen, R. Housley, and W. Polk (2008, May). Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile. RFC 5280 (Proposed Standard). Updated by RFC 6818.

Cristian, F. (1991, feb). Understanding fault-tolerant distributed systems. *Commun. ACM 34*(2), 56–78.

Crockford, D. (2006, July). The application/json Media Type for JavaScript Object Notation (JSON). RFC 4627 (Informational).

D. Ellard, R. Altmann, A. G. D. B. (2012, November). Dtn ip neighbor discovery (ipnd). draft-irtf-dtnrg-ipnd-02.

D. Farinacci, V. Fuller, D. M. D. L. (2011). Locator/id separation protocol (lisp). (July).

Daly, E. M. and M. Haahr (2007). Social network analysis for routing in disconnected delay-tolerant manets. In *Proceedings of the 8th ACM international symposium on Mobile ad hoc networking and computing*, MobiHoc '07, New York, NY, USA, pp. 32–40. ACM.

Deering, S. (1989, August). Host extensions for IP multicasting. RFC 1112 (Standard). Updated by RFC 2236.

Del Duca Almeida, V., A. B. Oliveira, D. F. Macedo, and J. M. S. Nogueira (2012, nov.). Performance evaluation of manet and dtn routing protocols. In *Wireless Days (WD), 2012 IFIP*, pp. 1–6.

Demmer, M., E. Brewer, K. Fall, M. Ho, R. Patra, S. Jain, and R. Patra (2003). Implementing delay tolerant networking. Technical report.

Demmer, M. and K. Fall (2007). Dtlsr: Delay tolerant routing for developing regions.

Demmer, M., K. Fall, T. Koponen, and S. Shenker (2007). Towards a modern communications api. In *In Proceedings of the 6 th Workshop on Hot Topics in Networks (HotNets-VI*.

Demmer, M. J. (2008). *A delay tolerant networking and system architecture for developing regions*. Ph. D. thesis, Berkeley, CA, USA. AAI3388269.

DePaoli, F. and L. Mariani (2004, july-aug.). Dependability in peer-to-peer systems. *Internet Computing, IEEE 8*(4), 54–61.

Deugd, S. d., R. Carroll, K. Kelly, B. Millett, and J. Ricker (2006). Soda: Service oriented device architecture. *IEEE Pervasive Computing 5*, 94–96.

Dinuab, V. and P. Nadkarnia (2007, nov). Guidelines for the effective use of entity–attribute–value modeling for biomedical databases. *International Journal of Medical Informatics 76*(11-12), 769–779.

Doering, M., S. Lahde, J. Morgenroth, and L. Wolf (2008). Ibr-dtn: an efficient implementation for embedded systems. In *Proceedings of the third ACM workshop on Challenged networks*, CHANTS '08, New York, NY, USA, pp. 117–120. ACM.

Dohler, M., T. Watteyne, T. Winter, and D. Barthel (2009, may). Routing requirements for urban low-power and lossy networks.

Du, J., E. Kranakis, and A. Nayak (2012, june). Cooperative neighbor discovery protocol for a wireless network using two antenna patterns. In *Distributed Computing Systems Workshops (ICDCSW), 2012 32nd International Conference on*, pp. 178–186.

Eddy, W. and E. Davies (2011, May). Using Self-Delimiting Numeric Values in Protocols. RFC 6256 (Informational).

Elizabeth, Y. S., Y. Sun, E. M. Belding-Royer, and C. E. Perkins (2002). Internet connectivity for ad hoc mobile networks. *INTERNATIONAL JOURNAL OF WIRELESS INFORMATION NETWORKS 9*(2), 75–88.

Elwyn Davies, A. D. (2010). *n4c-wp2-023-dtn-infrastructure-fs-12* (n4c-wp2-023 ed.).

Erl, T. (2004). *Service-Oriented Architecture: A Field Guide to Integrating XML and Web Services*. Upper Saddle River, NJ, USA: Prentice Hall PTR.

Erramilli, V., M. Crovella, A. Chaintreau, and C. Diot (2008). Delegation forwarding. In *Proceedings of the 9th ACM international symposium on Mobile ad hoc networking and computing*, MobiHoc '08, New York, NY, USA, pp. 251–260. ACM.

Evdokimov, S., B. Fabian, S. Kunz, and N. Schoenemann (2010, june). Comparison of discovery service architectures for the internet of things. In *Sensor Networks, Ubiquitous, and Trustworthy Computing (SUTC), 2010 IEEE International Conference on*, pp. 237–244.

Fall, K. (2003). A delay-tolerant network architecture for challenged internets. In *Proceedings of the 2003 conference on Applications, technologies, architectures, and protocols for computer communications*, SIGCOMM '03, New York, NY, USA, pp. 27–34. ACM.

Fall, K. (2012). Comparing information-centric and delay-tolerant networking. In *Local Computer Networks (LCN), 2012 IEEE 37th Conference on.*

Fall, K. R. and S. Farrell (2008). Dtn: an architectural retrospective. *IEEE Journal on Selected Areas in Communications 26*(5), 828–836.

Farrell, S. (2010, sept.). Endpoint discovery and contact graph routing in space and terrestrial dtns. In *Advanced satellite multimedia systems conference (asma) and the 11th signal processing for space communications workshop (spsc), 2010 5th*, pp. 89–93.

Farrell, S. and V. Cahill (2006a). *Delay- and Disruption-Tolerant Networking.* Norwood, MA, USA: Artech House, Inc.

Farrell, S. and V. Cahill (2006b). Security considerations in space and delay tolerant networks. pp. 8–38.

Farrell, S., M. Ramadas, and S. Burleigh (2008, sep). Licklider transmission protocol - security extensions.

Fawaz, K. and H. Artail (2012). Dcim: Distributed cache invalidation method for maintaining cache consistency in wireless mobile networks. *Mobile Computing, IEEE Transactions on PP*(99), 1.

Fielding, R., J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, and T. Berners-Lee (1999, June). Hypertext Transfer Protocol – HTTP/1.1. RFC 2616 (Draft Standard). Updated by RFCs 2817, 5785, 6266, 6585.

Fielding, R. T. (2000). *Architectural styles and the design of network-based software architectures.* Ph. D. thesis. AAI9980887.

Fink, B. (2012). Distributed computation on dynamo-style distributed storage: riak pipe. In *Proceedings of the eleventh ACM SIGPLAN workshop on Erlang workshop*, Erlang '12, New York, NY, USA, pp. 43–50. ACM.

for Space Data Systems (CCSDS), C. C. (2011). *Asynchronous Message Service* (Issue 1 ed.). Blue Book.

Fox, A. and E. A. Brewer (1999). Harvest, yield, and scalable tolerant systems. In *Hot Topics in Operating Systems, 1999. Proceedings of the Seventh Workshop on*, pp. 174–178.

Garcia-Macias, J. A. and D. A. Torres (2005, june). Service discovery in mobile ad-hoc networks: better at the network layer? In *Parallel Processing, 2005. ICPP 2005 Workshops. International Conference Workshops on*, pp. 452–457.

Gluhak, A., S. Krco, M. Nati, D. Pfisterer, N. Mitton, and T. Razafindralambo (2011, november). A survey on facilities for experimental internet of things research. *Communications Magazine, IEEE 49*(11), 58–67.

Greenberg, A., G. Hjalmtysson, D. A. Maltz, A. Myers, J. Rexford, G. Xie, H. Yan, J. Zhan, and H. Zhang (2005, oct). A clean slate 4d approach to network control and management. *SIGCOMM Comput. Commun. Rev. 35*(5), 41–54.

Greifenberg, J. and D. Kutscher (2008). Efficient publish/subscribe-based multicast for opportunistic networking with self-organized resource utilization. *Advanced Information Networking and Applications Workshops, International Conference on 0*, 1708–1714.

Grundy, A. and M. Radenkovic (2010a, oct). Promoting congestion control in opportunistic networks. In *2010 IEEE 6th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob)*, pp. 324–330. IEEE.

Grundy, A. and M. Radenkovic (2010b). Promoting congestion control in opportunistic networks. In *Wireless and Mobile Computing, Networking and Communications (WiMob), 2010 IEEE 6th International Conference on*, pp. 324–330.

Gurses, E. and A. N. Kim (2008). Maximum utility peer selection for p2p streaming in wireless ad hoc networks. In *Global Telecommunications Conference, 2008. IEEE GLOBECOM 2008. IEEE*, pp. 1–5.

Hadzilacos, V. and S. Toueg (1993). Fault-tolerant broadcasts and related problems. Chapter Distributed systems (2nd Ed.), pp. 97–145. New York, NY, USA: ACM Press/Addison-Wesley Publishing Co.

Helal, S., N. Desai, V. Verma, and C. Lee (2003, march). Konark - a service discovery and delivery protocol for ad-hoc networks. In *Wireless Communications and Networking, 2003. WCNC 2003. 2003 IEEE*, Volume 3, pp. 2107–2113.

Hodges, J. and R. Morgan (2002, September). Lightweight Directory Access Protocol (v3): Technical Specification. RFC 3377 (Proposed Standard). Obsoleted by RFC 4510.

Houidi, I., W. Louati, D. Zeghlache, and S. Baucke (2009, june). Virtual resource description and clustering for virtual network discovery. In *Communications Workshops, 2009. ICC Workshops 2009. IEEE International Conference on*, pp. 1–6.

Itu, T. (2005, aug). *Recommendation X.500, "The Directory: Overview of Concepts, Models and Service"*.

Ivancic, W. D. (2010, march). Security analysis of dtn architecture and bundle protocol specification for space-based networks. In *Aerospace Conference, 2010 IEEE*, pp. 1–12.

J. Wyllie, W. Eddy, J. I. W. I. S. O. (2007). Automated bundle agent discovery for delay/disruption-tolerant networking.

Jackson, J. (2005, aug.). The interplanetary internet [networked space communications]. *Spectrum, IEEE 42*(8), 30–35.

Jain, S., K. Fall, and R. Patra (2004). Routing in a delay tolerant network. In *Proceedings of the 2004 conference on Applications, technologies, architectures, and protocols for computer communications*, SIGCOMM '04, New York, NY, USA, pp. 145–158. ACM.

Jeon, H. and J. Jee (2006, feb.). Ipv6 neighbor discovery protocol for common prefix allocation in ieee 802.16. Volume 3, pp. 1661–1663.

Jindal, A. (2006). Optimizing multi-copy routing schemes for resource constrained intermittently connected mobile networks. In *In Proceedings of IEEE Asilomar Conference on Signals, Systems and Computers*.

Jonsson, U., F. Alriksson, T. Larsson, P. Johansson, and . J. Maguire, G. Q. (2000). Mipmanet-mobile ip for mobile ad hoc networks. In *Mobile and Ad Hoc Networking and Computing, 2000. MobiHOC. 2000 First Annual Workshop on*, pp. 75–85.

Jun, H., M. H. Ammar, M. D. Corner, and E. W. Zegura (2009, oct). Hierarchical power management in disruption tolerant networks using traffic-aware optimization. *Comput. Commun. 32*(16), 1710–1723.

Kafle, V. P. and M. Inoue (2012, march). Introducing multi-id and multi-locator into network architecture. *Communications Magazine, IEEE 50*(3), 104–110.

Khabbaz, M., H. Alazemi, and C. Assi (2013). Delay-aware data delivery in vehicular intermittently connected networks. *Communications, IEEE Transactions on PP*(99), 1–10.

Khabbaz, M. J., C. M. Assi, and W. F. Fawaz (2012, quarter). Disruption-tolerant networking: A comprehensive survey on recent developments and persisting challenges. *Communications Surveys Tutorials, IEEE 14*(2), 607–640.

Khan, A., D.-W. Kum, J.-C. Nam, and Y.-Z. Cho (2012, oct.). Minimum delay routing protocol for dtn-enabled vanets. In *Communications (APCC), 2012 18th Asia-Pacific Conference on*, pp. 534–538.

Kim, D., D. Shin, and H. Yoo (2008, 23–25 July). Providing service-connectivity in delay tolerant networks. In *Proc. International Conference on Advanced Language Processing and Web Information Technology ALPIT '08*, pp. 471–476.

Kim, S.-H. and S.-J. Han (2012, jan.). Contour routing for peer-to-peer dtn delivery in cellular networks. In *Communication Systems and Networks (COMSNETS), 2012 Fourth International Conference on*, pp. 1–9.

Kozat, U. C. and L. Tassiulas (2003, march-3 april). Network layer support for service discovery in mobile ad hoc networks. In *INFOCOM 2003. Twenty-Second Annual Joint Conference of the IEEE Computer and Communications. IEEE Societies*, Volume 3, pp. 1965–1975.

Kozat, U. C. and L. Tassiulas (2004, jan). Service discovery in mobile ad hoc networks: an overall perspective on architectural choices and network layer support issues. *Ad Hoc Networks 2*(1), 23–44.

Kushalnagar, N., G. Montenegro, and C. Schumacher (2007, aug). Ipv6 over low-power wireless personal area networks (6lowpans): Overview, assumptions, problem statement, and goals.

Lam, T., J. J. Ding, and J. C. Liu (2008). Xml document parsing: Operational and performance characteristics. *Computer 41*(9), 30–37.

Leach, P., M. Mealling, and R. Salz (2005, July). A Universally Unique IDentifier (UUID) URN Namespace. RFC 4122 (Proposed Standard).

Lee, J.-C., Y.-H. Han, M.-K. Shin, H.-J. Jang, and H.-J. Kim (2006, feb.). Considerations of neighbor discovery protocol over ieee 802.16 networks. In *Advanced Communication Technology, 2006. ICACT 2006. The 8th International Conference*, Volume 2, pp. 951–955.

Li, L. and L. Lamont (2005, march). A lightweight service discovery mechanism for mobile ad hoc pervasive environment using cross-layer design. In *Pervasive Computing and Communications Workshops, 2005. PerCom 2005 Workshops. Third IEEE International Conference on*, pp. 55–59.

Li, X., F. Qiu, P. Dong, and H. Zhang (2010, sept.). One-hop dht resolution of locator/id mapping. In *Network Infrastructure and Digital Content, 2010 2nd IEEE International Conference on*, pp. 417–423.

Li, Z., L. Sun, and E. C. Ifeachor (2006, 27 2006-dec. 1). Wsn10-5: Adaptive multi-copy routing for intermittently connected mobile ad hoc networks. In *Global Telecommunications Conference, 2006. GLOBECOM '06. IEEE*, pp. 1–6.

Lindgren, A., A. Doria, E. Davies, and S. Grasic (2012, August). Probabilistic Routing Protocol for Intermittently Connected Networks. RFC 6693 (Experimental).

Lindgren, A., A. Doria, and O. Schelén (2003). Probabilistic routing in intermittently connected networks. *SIGMOBILE Mob. Comput. Commun. Rev. 7*(3), 19–20.

Liu, C. and J. Wu (2008). Routing in a cyclic mobispace. In *Proceedings of the 9th ACM international symposium on Mobile ad hoc networking and computing*, MobiHoc '08, New York, NY, USA, pp. 351–360. ACM.

Liu, C. and J. Wu (2009). An optimal probabilistic forwarding protocolin delay tolerant networks. In *Proceedings of the tenth ACM international symposium on Mobile ad hoc networking and computing*, MobiHoc '09, New York, NY, USA, pp. 105–114. ACM.

Liu, C., J. Wu, and I. Cardei (2009, oct.). Message forwarding in cyclic mobispace: the multi-copy case. In *Mobile Adhoc and Sensor Systems, 2009. MASS '09. IEEE 6th International Conference on*, pp. 413–422.

Luo, H., Y. Qin, and H. Zhang (2009, dec.). A dht-based identifier-to-locator mapping approach for a scalable internet. *Parallel and Distributed Systems, IEEE Transactions on 20*(12), 1790–1802.

Lynch, N. A. (1996). *Distributed Algorithms.* San Francisco, CA, USA: Morgan Kaufmann Publishers Inc.

Majumder, K., S. Ray, and S. K. Sarkar (2011, may). Implementation and performance analysis of the gateway discovery approaches in the integrated manet-internet scenario. In *Communication Software and Networks (ICCSN), 2011 IEEE 3rd International Conference on*, pp. 601–605.

Marias, G. F., N. Fotiou, and G. C. Polyzos (2012, june). Efficient information lookup for the internet of things. In *World of Wireless, Mobile and Multimedia Networks (WoWMoM), 2012 IEEE International Symposium on a*, pp. 1–6.

Martinez Tornell, S., C. T. Calafate, J. Cano, and P. Manzoni (2012, june). A map-based sensor data delivery protocol for vehicular networks. In *Ad Hoc Networking Workshop (Med-Hoc-Net), 2012 The 11th Annual Mediterranean*, pp. 1–8.

Martocci, J., P. D. Mil, N. Riou, and W. Vermeylen (2010, jun). Building automation routing requirements in low-power and lossy networks.

McMahon, A. and K. Fall (2010). draft-mcmahon-dtnrg-dtn-edp-00. *Internet Draft*.

McMahon, A. and S. Farrell (2009, nov.). Delay- and disruption-tolerant networking. *Internet Computing, IEEE 13*(6), 82–87.

McMahon, A., S. Farrell, A. Lynch, S. Weber, and K. Hartnett (2011, September). Dtn trials and router updates.

McMahon, A., S. Farrell, E. Meehan, S. Weber, and K. Hartnett (2011, July). Report on an arctic summer dtn trial. *Wirel. Netw. 17*(5), 1127–1156.

McMahon, A., S. Farrell, S. Weber, K. Hartnett, and A. Lynch (2011, oct). Report on dtn applications during arctic summer 2010 trial. In *International Workshop on Opportunistic and Delay/disruption-Tolerant Networking in conjunction with the 14th International Symposium on Wireless Personal Multimedia Communications (WODTN 2011 (WPMC 2011 Workshop))*, Brest, France.

McMahon, A., S. Farrell, S. Weber, E. Meehan, and K. Hartnett (2009, September). An n4c dtn router node design. http://www.extremecom.org/.

McMahon, A., S. Farrell, S. Weber, E. Meehan, and K. Hartnett (2010, September). N4c dtn router node: 2009 results, 2010 plans.

Mehar, S., S. M. Senouci, and G. Remy (2012, nov.). Dissemination protocol for heterogeneous cooperative vehicular networks. In *Wireless Days (WD), 2012 IFIP*, pp. 1–6.

Mekbungwan, P., A. Tunpan, L. F. S. Borlido, N. Khaitiyakun, and K. Kanchanasut (2011, aug.). A dtn routing on olsr for vanet: A preliminary road experiment. In *Global Information Infrastructure Symposium (GIIS), 2011*, pp. 1–6.

Menth, M., M. Hartmann, and M. Hofling (2010, october). Firms: A mapping system for future internet routing. *Selected Areas in Communications, IEEE Journal on 28*(8), 1326–1331.

Merani, D., A. Berni, J. Potter, and R. Martins (2011, feb.). An underwater convergence layer for disruption tolerant networking. In *Internet Communications (BCFIC Riga), 2011 Baltic Congress on Future*, pp. 103–108.

Meshkova, E., J. Riihijärvi, M. Petrova, and P. Mähönen (2008). A survey on resource discovery mechanisms, peer-to-peer and service discovery frameworks. *Computer Networks 52*(11), 2097–2128.

Meyer, D., L. Zhang, and K. Fall (2007, September). Report from the IAB Workshop on Routing and Addressing. RFC 4984 (Informational).

Mills, D. (1992, March). Network Time Protocol (Version 3) Specification, Implementation and Analysis. RFC 1305 (Draft Standard). Obsoleted by RFC 5905.

Misic, J., N. Khan, H. Khojasteh, and V. B. Misic (2012, april). Cscd: A simple channel scan protocol to discover and join a cognitive pan. In *Wireless Communications and Networking Conference (WCNC), 2012 IEEE*, pp. 2643–2647.

Moats, R. (1997, May). URN Syntax. RFC 2141 (Proposed Standard).

Mockapetris, P. (1987, November). Domain names - implementation and specification. RFC 1035 (Standard). Updated by RFCs 1101, 1183, 1348, 1876, 1982, 1995, 1996, 2065, 2136, 2181, 2137, 2308, 2535, 2845, 3425, 3658, 4033, 4034, 4035, 4343, 5936, 5966, 6604.

Moskowitz, R. and P. Nikander (2006, may). Host identity protocol (hip) architecture.

Musolesi, M. and C. Mascolo (2009, feb.). Car: Context-aware adaptive routing for delay-tolerant mobile networks. *Mobile Computing, IEEE Transactions on 8*(2), 246–260.

Nadkarni, P. M., L. N. Marenco, R. Chen, E. Skoufos, G. M. Shepherd, and P. L. Miller (1999). Application of information technology: Organization of heterogeneous scientific data using the eav/cr representation. *JAMIA 6*(6), 478–493.

Nayebi, A., H. Sarbazi-Azad, and G. Karlsson (2009, may.). Routing, data gathering, and neighbor discovery in delay-tolerant wireless sensor networks. pp. 1–6.

Nguyen, H. S., Y. Tan, and Y. Shinoda (2012, march). Building distributed attribute-value trees on dht-based networks. In *Advanced Information Networking and Applications (AINA), 2012 IEEE 26th International Conference on*, pp. 795–802.

Nidd, M. (2001, aug). Service discovery in deapspace. *Personal Communications, IEEE 8*(4), 39–45.

Niven-Jenkins, B., F. L. Faucheur, and N. Bitar (2012, September). Content Distribution Network Interconnection (CDNI) Problem Statement. RFC 6707 (Informational).

Oliver, E. and H. Falaki (2007). Performance evaluation and analysis of delay tolerant networking. In *Proceedings of the 1st international workshop on System evaluation for mobile platforms*, MobiEval '07, New York, NY, USA, pp. 1–6. ACM.

Pan, J., S. Paul, and R. Jain (2011, july). A survey of the research on future internet architectures. *Communications Magazine, IEEE 49*(7), 26–36.

Park, H.-S., J.-H. Jang, S.-H. Lee, and J.-D. Kim (2012, may). Position-based dtn routing in metropolitan bus network. In *Systems and Informatics (ICSAI), 2012 International Conference on*, pp. 1449–1453.

Paul, S., J. Pan, and R. Jain (2011, jan). Architectures for the future networks and the next generation internet: A survey. *Comput. Commun. 34*(1), 2–42.

Pedrasa, J. R. and A. P. Seneviratne (2011). Determining network availability on the move. In *Communications (APCC), 2011 17th Asia-Pacific Conference on*, pp. 301–306.

Ping, Z., H. Ronglei, F. Yong, Y. Jianxi, and L. Yue (2009, sep.). A key management scheme for ad hoc networks. pp. 1–5.

Pister, K., P. Thubert, S. Dwars, and T. Phinney (2009, oct). Industrial routing requirements in low-power and lossy networks.

Pitkanen, M., T. Karkkainen, and J. Ott (2012, march). Mobility and service discovery in opportunistic networks. In *Pervasive Computing and Communications Workshops (PERCOM Workshops), 2012 IEEE International Conference on*, pp. 204–210.

Pro, O. D. (2010a). *"associated". Oxford Dictionaries Pro. April 2010.* Oxford University Press (http://english.oxforddictionaries.com).

Pro, O. D. (2010b). *"identifier". Oxford Dictionaries Pro. April 2010.* Oxford University Press (http://english.oxforddictionaries.com).

Ramadas, M., S. Burleigh, and S. Farrell (2008, sep). Licklider transmission protocol - specification.

Ratnasamy, S., P. Francis, M. Handley, R. Karp, and S. Shenker (2001, aug). A scalable content-addressable network. *SIGCOMM Comput. Commun. Rev. 31*(4), 161–172.

Ratsimor, O., D. Chakraborty, A. Joshi, and T. Finin (2002). Allia: Alliance-based service discovery for ad-hoc environments. In *in Proc. of ACM Mobile Commerce Workshop*, pp. 1–9.

Reschke, J. (2011, June). Use of the Content-Disposition Header Field in the Hypertext Transfer Protocol (HTTP). RFC 6266 (Proposed Standard).

Rodrigues, P., Y.-D. Bromberg, L. Réveillère, and D. Négru (2012). Zigzag: a middleware for service discovery in future internet. In *Proceedings of the 12th IFIP WG 6.1 international conference on Distributed Applications and Interoperable Systems*, DAIS'12, Berlin, Heidelberg, pp. 208–221. Springer-Verlag.

Roussos, G. and P. Chartier (2011, oct.). Scalable id/locator resolution for the iot. In *Internet of Things (iThings/CPSCom), 2011 International Conference on and 4th International Conference on Cyber, Physical and Social Computing*, pp. 58–66.

Rowstron, A. and P. Druschel (2001). Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems.

Sailhan, F. and V. Issarny (2005, march). Scalable service discovery for manet. In *Pervasive Computing and Communications, 2005. PerCom 2005. Third IEEE International Conference on*, pp. 235–244.

Sarvanko, H., M. Höyhtyä, M. Katz, and F. Fitzek (2010). Distributed resources in wireless networks: Discovery and cooperative uses.

Schiele, G., C. Becker, and K. Rothermel (2004). Energy-efficient cluster-based service discovery for ubiquitous computing. In *Proceedings of the 11th workshop on ACM SIGOPS European workshop*, EW 11, New York, NY, USA. ACM.

Schmidt, A., K. A. Aidoo, A. Takaluoma, U. Tuomela, K. V. Laerhoven, and W. V. d. Velde (1999). Advanced interaction in context. In *Proceedings of the 1st international symposium on Handheld and Ubiquitous Computing*, HUC '99, London, UK, UK, pp. 89–101. Springer-Verlag.

Schmidt, B. A. and A. R. Butt (2009, mar.). Facilitating intermediate node discovery for decentralized offloading in high performance computing centers. pp. 289–294.

Scholl, J., L. Lambrinos, and A. Lindgren (2009). Rural telemedicine networks using store-and-forward voice-over-ip. In K.-P. Adlassnig, B. Blobel, J. Mantas, and I. Masic (Eds.), *Medical Informatics in a United and Healthy Europe - Proceedings of MIE 2009, The XXIInd International Congress of the European Federation for Medical Informatics, Sarajevo, Bosnia and Herzegovina, Agust 30 - September 2, 2009*, Volume 150 of *Studies in Health Technology and Informatics*, pp. 448–452. IOS Press.

Schurgot, M. R., C. Comaniciu, and K. Jaffres-Runser (2012, july). Beyond traditional dtn routing: social networks for opportunistic communication. *Communications Magazine, IEEE 50*(7), 155–162.

Scott, J., P. Hui, J. Crowcroft, and C. Diot (2006, jan). Haggle: A networking architecture designed around mobile users. *Invited paper at The Third Annual IFIP Conference on Wireless On-demand Network Systems and Services (WONS 2006).*

Scott, K. and S. Burleigh (2007, nov). Bundle protocol specification. Technical Report 5050.

Shaw, M. (1990, July). Toward higher-level abstractions for software systems. *Data Knowl. Eng. 5*(2), 119–128.

Siglin, P. W. and G. Senn (1961, august). Courier communication satellite. *Electrical Engineers, Journal of the Institution of 7*(80), 504–508.

Singla, A., B. Godfrey, K. R. Fall, G. Iannaccone, and S. Ratnasamy (2010). Scalable routing on flat names. In *CoNEXT'10*, pp. 20–20.

Sollins, K. (1998, January). Architectural Principles of Uniform Resource Name Resolution. RFC 2276 (Informational). Updated by RFC 3401.

Sollins, K. and L. Masinter (1994, December). Functional Requirements for Uniform Resource Names. RFC 1737 (Informational).

Spyropoulos, T., T. Turletti, and K. Obraczka (2007, june). Utility-based message replication for intermittently connected heterogeneous networks. In *World of Wireless, Mobile and Multimedia Networks, 2007. WoWMoM 2007. IEEE International Symposium on a*, pp. 1–6.

Stenstrom, P. (1990). A survey of cache coherence schemes for multiprocessors. *Computer 23*(6), 12–24.

Stoica, I., R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan (2001). Chord: A scalable peer-to-peer lookup service for internet applications. In *Proceedings of the 2001 conference on Applications, technologies, architectures, and protocols for computer communications*, SIGCOMM '01, New York, NY, USA, pp. 149–160. ACM.

Su, J. and W. Guo (2008, may). A survey of service discovery protocols for mobile ad hoc networks. In *Communications, Circuits and Systems, 2008. ICCCAS 2008. International Conference on*, pp. 398–404.

Sullivan, P. M. (2000). The totality of facts. *Proceedings of the Aristotelian Society 100*, 175–192.

Symington, S. (2011a, May). Delay-Tolerant Networking Metadata Extension Block. RFC 6258 (Experimental).

Symington, S. (2011b, May). Delay-Tolerant Networking Previous-Hop Insertion Block. RFC 6259 (Experimental).

Symington, S., S. Farrell, H. Weiss, and P. Lovell (2011, May). Bundle Security Protocol Specification. RFC 6257 (Experimental).

Tan, L. and N. Wang (2010, aug.). Future internet: The internet of things. In *Advanced Computer Theory and Engineering (ICACTE), 2010 3rd International Conference on*, Volume 5, pp. –5.

Tim Bray, Dave Hollander, A. L. R. T. H. S. T. (2009, December). Namespaces in xml 1.0 (third edition). http://www.w3.org/TR/2009/REC-xml-names-20091208.

Torres, J., M. Nogueira, and G. Pujolle (2012). A survey on identity management for the future network. *Communications Surveys Tutorials, IEEE PP*(99), 1–16.

Uddin, M., H. Ahmadi, T. Abdelzaher, and R. Kravets (2012). Inter-contact routing for energy constrained disaster response networks. *Mobile Computing, IEEE Transactions on PP*(99), 1.

URL (1). Association lists. `http://www.gnu.org/software/emacs/manual/html_node/elisp/Association-Lists.html`. Accessed 04/04/2013.

URL (100). Osbridge. `http://www.osbridge.com/?q=en/node/93`. Accessed 04/04/2013.

URL (102, 2). Rdf semantics.

URL (103). Predicate logic terms and symbols. `http://legacy.earlham.edu/~peters/courses/log/terms3.htm`. Accessed 04/04/2013.

URL (104). Yet another network simulator. `http://cutebugs.net/files/wns2-yans.pdf`. Accessed 17/07/2013.

URL (105). messagepack. `http://msgpack.org/`. Accessed 22/07/13.

URL (106). dieselnet. `http://prisms.cs.umass.edu/dome/umassdieselnet`. Accessed: 23/07/2013.

URL (107). Ipnd bloom filter. `http://tools.ietf.org/html/draft-irtf-dtnrg-ipnd-01#section-2.6.2`. Accessed 24/07/13.

URL (12). Jboss community. `https://community.jboss.org`. Accessed 04/04/2013.

URL (13). mibdepot. `http://www.mibdepot.com`. Accessed 04/04/2013.

URL (14). Standard generalised markup language (sgml). `http://www.w3.org/MarkUp/SGML/`. Accessed 04/04/2013.

URL (15). The world wide web consortium (w3c). `http://www.w3.org`. Accessed 04/04/2013.

URL (16). Universal description, discovery and integration v3.0.2 (uddi) xml schema. `http://uddi.org/pubs/uddi_v3.htm`. Accessed 04/04/2013.

URL (17). The extensible resource descriptor (xrd) version 1.0. `http://docs.oasis-open.org/xri/xrd/v1.0/xrd-1.0.html`. Accessed 04/04/2013.

URL (18). Xep-0030. `http://xmpp.org/extensions/xep-0030.html`. Accessed 04/04/2013.

URL (19). Near field communication data exchange format (ndef). `http://www.nfc-forum.org/specs/`. Accessed 04/04/2013.

URL (2). The freenet project. `https://freenetproject.org`. Accessed 04/04/2013.

URL (20). Content centric networking (ccn). `http://www.parc.com/work/focus-area/networking/`. Accessed 04/04/2013.

URL (21). The 4ward project (4ward). `http://www.4ward-project.eu/`. Accessed 04/04/2013.

URL (22). Named data networking (ndn). `http://www.named-data.net/`. Accessed 04/04/2013.

URL (23). Network of information (netinf). `http://www.netinf.org/`. Accessed 04/04/2013.

URL (24). Scalable architecture for interactive learning (sail). http://www.sail-project.eu/. Accessed 04/04/2013.

URL (25). Content mediator architecture for content-aware networks (comet). http://www.comet-project.org. Accessed 04/04/2013.

URL (26). Publish-subscribe internet routing paradigm (psirp). http://www.psirp.org/. Accessed 04/04/2013.

URL (27). Content-oriented networking: a new experience for content transfer (connect). urlhttp://www.anr-connect.org/. Accessed 04/04/2013.

URL (28). Pursuit. http://www.fp7-pursuit.eu/. Accessed 04/04/2013.

URL (29). The convergence project. http://www.ict-convergence.eu/. Accessed 04/04/2013.

URL (3). Information-centric networking (icn). http://www.neclab.eu/icn-2011. Accessed 04/04/2013.

URL (30). Data orientated-network architecture (dona). https://buffy.eecs.berkeley.edu/PHP/resabs/resabs.php?f_year=2007&f_submit=one&f_absid=101594. Accessed 04/04/2013.

URL (31). The host identity protocol (hip) experiment report. https://datatracker.ietf.org/doc/draft-irtf-hip-experiment/. Acceessed 04/04/2013.

URL (32). Intentional naming system (ins). http://nms.csail.mit.edu/projects/ins/. Accessed 04/04/2013.

URL (33). Internet corporation for assigned names and numbers (icann). http://www.icann.org. Accessed 04/04/2013.

URL (34). Internet assigned numbers authority (iana). http://www.iana.org/assignments/smi-numbers. Accessed 04/04/2013.

URL (36). Delay-tolerant networking research group (dtnrg). http://www.dtnrg.org. Accessed 04/04/2013.

URL (37). Icn. http://www.dagstuhl.de/10492. Accessed 04/04/2013.

URL (4). The institute of electrical and electronics engineers standards association (ieee-sa). https://standards.ieee.org. Accessed 04/04/2013.

URL (41). Information-centric networking (icn) architectures. http://www.dagstuhl.de/10492. Accessed 04/04/2013.

URL (42). Resource location and discovery (reload) base protocol. http://tools.ietf.org/html/draft-ietf-p2psip-base-18. Accessed 04/04/2013.

URL (43). haggleproject. http://www.haggleproject.org/. Accessed 04/04/2013.

URL (44). Uppsala university, department of information technology. http://www.it.uu.se. Accessed 04/04/2013.

URL (45). Haggle wiki faq. http://code.google.com/p/haggle/wiki/FAQ. Accessed 04/04/2013.

URL (46). Contact graph routing (cgr). http://tools.ietf.org/html/draft-burleigh-dtnrg-cgr-01. Accessed 04/04/2013.

URL (47). Sensor networking with delay tolerance (sendt). http://down.dsg.cs.tcd.ie/sendt/. Accessed 04/04/2013.

URL (48). The technology and infrastructure for developing regions (tier. http://tier.cs.berkeley.edu/wiki/Home.

URL (49). daknet. http://www.firstmilesolutions.com/documents/DakNet_IEEE_Computer.pdf. Accessed 04/04/2013.

URL (5). Ieee std 802-2001. http://standards.ieee.org/getieee802/download/802-2001.pdf. Accessed 04/04/2013.

URL (50). Acommsnet10. http://www.ua-net.eu/UAN10. Accessed 04/04/2013.

URL (51). Consultative committee on space data systems (ccsds). http://www.ccsds.org. Accessed 04/04/2013.

URL (52). dtn-interest. https://www.irtf.org/mailman/listinfo/dtn-interest/. Accessed 04/04/2013.

URL (53). dtn-users. https://www.irtf.org/mailman/listinfo/dtn-users/. Accessed 04/04/2013.

URL (54). Nasa space dtn project announcement. http://prod.nais.nasa.gov/cgi-bin/eps/synopsis.cgi?acqid=146586. Accessed 04/04/2013.

URL (55). The institute of operating systems and computer networks. http://www.ibr.cs.tu-bs.de. Accessed 04/04/2013.

URL (56). Ibr-dtn. http://www.ibr.cs.tu-bs.de/projects/ibr-dtn/. Accessed 04/04/2013.

URL (57). Ohio university. https://ion.ocp.ohiou.edu/index.php. Accessed 04/04/2013.

URL (58). Ion code release. http://sourceforge.net/projects/ion-dtn/. Accessed 04/04/2013.

URL (59). Jdtn. http://jdtn.sourceforge.net/. Accessed 04/04/2013.

URL (60). The bytewalla project. http://www.tslab.ssvl.kth.se/csd/projects/092106/home. Accessed 04/04/2013.

URL (61). Telecommunication systems laboratory (tslab). http://www.tslab.ssvl.kth.se/. Accessed 04/04/2013.

URL (62). Interplanetary internet (ipn): Architectural definition. http://tools.ietf.org/html/draft-irtf-ipnrg-arch-00. Accessed 04/04/2013.

URL (63). Wired. http://www.wired.com/wired/archive/8.01/solar.html. Accessed 04/04/2013.

URL (64). Delay-tolerant network architecture: The evolving interplanetary internet. http://tools.ietf.org/id/draft-irtf-ipnrg-arch-01.txt. Accessed 04/04/2013.

URL (65). The postellation project. http://postellation.viagenie.ca/index.html. Accessed 04/04/2013.

URL (66). Viagenie. http://www.viagenie.ca. Accessed 04/04/2013.

URL (67). Nasa disruption tolerant networking program. http://www.nasa.gov/mission_pages/station/research/experiments/DTN.html. Accessed 04/04/2013.

URL (68). Magento eav data model. http://www.magentocommerce.com/wiki/2_-_magento_concepts_and_architecture/magento_database_diagram#the_magento_eav_data_model. Accessed 04/04/2013.

URL (69). Json. http://www.json.org/. Accessed 04/04/2013.

URL (7). The internet engineering task force (ietf). http://www.ietf.org/. Accessed 04/04/2013.

URL (70). Eav at senselab. http://senselab.med.yale.edu/_siteNET/dsAdmin/. Accessed 04/04/2013.

URL (71). Eav/cr model. http://ycmi.med.yale.edu/nadkarni/eav_cr_frame.htm. Accessed 04/04/2013.

URL (72). Python multiprocessing module. http://docs.python.org/2/library/multiprocessing.html. Accessed 04/04/2013.

URL (73). Python thread module. http://docs.python.org/2/library/thread.html#module-thread. Accessed 04/04/2013.

URL (74). Python threading module. http://docs.python.org/2/library/threading.html#module-threading. Accessed 04/04/2013.

URL (75). Python queue module. http://docs.python.org/2/library/queue.html. Accessed 04/04/2013.

URL (76). Python select module. http://docs.python.org/2/library/select.html. Accessed 04/04/2013.

URL (77). Python mysqldb module. http://mysql-python.sourceforge.net/MySQLdb-1.2.2/. Accessed 04/04/2013.

URL (78). Python os module. http://docs.python.org/2/library/os.html. Accessed 04/04/2013.

URL (79). Python string module. http://docs.python.org/2/library/string.html. Accessed 04/04/2013.

URL (8). The wimax forum. http://www.wimaxforum.org. Accessed 04/04/2013.

URL (80). Python uuid module. http://docs.python.org/2/library/uuid.html. Accessed 04/04/2013.

URL (81). Dtn2 api. http://www.dtnrg.org/docs/code/DTN2/doc/doxygen/html/. Accessed 04/04/2013.

URL (82). Dtn2 open call. http://www.dtnrg.org/docs/code/DTN2/doc/doxygen/html/dtn__api_8c.html#a3afc66b2e2c75fae07a45467df9f74e9. Accessed 04/04/2013.

URL (83). Dtn2 send call. http://www.dtnrg.org/docs/code/DTN2/doc/doxygen/html/dtn__api__wrap_8cc_source.html#l00205. Accessed 04/04/2013.

URL (84). Dtn2 receive call. http://www.dtnrg.org/docs/code/DTN2/doc/doxygen/html/dtn__api_8c_source.html#l00508. Accessed 04/04/2013.

URL (85). Numpy. http://www.numpy.org/. Accessed 04/04/2013.

URL (86). The electronic discovery reference model. http://www.edrm.net/. Accessed 04/04/2013.

URL (87). Redis. http://redis.io/. Accessed 04/04/2013.

URL (88). Apache lucene. http://lucene.apache.org/. Accessed 04/04/2013.

URL (89). Metoocrypto. http://chandlerproject.org/Projects/MeTooCrypto. Accessed 04/04/2013.

URL (9). The wi-fi alliance. http://www.wi-fi.org. Accessed 04/04/2013.

URL (90). Padjelanta national park. http://www.padjelanta.com/en/karta/index.asp. Accessed 04/04/2013.

URL (91). The hardware and software used during the n4c 2010 summer trial with references to full specifications. http://dtn.dsg.cs.tcd.ie/n4c-summer10/. Accessed 04/04/2013.

URL (92). N4c 2010 summer trial informal report. http://www.n4c.eu/Download/8.4/TCD_N4C%20Summer%202010%20DTN%20Trial%20%28Informal%20Report%29.pdf. Accessed 04/04/2013.

URL (95). Dtn2 s10 logging. http://dtn.hg.sourceforge.net/hgweb/dtn/DTN2/rev/a9cc6a7c7803. Accessed 04/04/2013.

URL (96). Extremecom 2009. http://www.extremecom.org/2009. Accessed 04/04/2013.

URL (97). Openvpn. http://www.openvpn.net/. Accessed 04/04/2013.

URL (98). Itelite omni directional antenna. http://www.itelite.net/products-desc.php?id=274. Accessed 04/04/2013.

URL (99). Openwrt. http://code.n4c.eu/code/kamikaze-dtn/. Accessed 04/04/2013.

Vahdat, A. and D. Becker (2000). Epidemic routing for partially-connected ad hoc networks. Technical report.

Veríssimo, P. and C. Almeida (1995). Quasi-synchronism: a step away from the traditional fault-tolerant real-time system models.

Ververidis, C. N. and G. C. Polyzos (2005, july). Extended zrp: a routing layer based service discovery protocol for mobile ad hoc networks. In *Mobile and Ubiquitous Systems: Networking and Services, 2005. MobiQuitous 2005. The Second Annual International Conference on*, pp. 65–72.

Vilardi, R., L. A. Grieco, C. Barakat, and G. Boggia (2013). Lightweight enhanced monitoring for high speed networks. *ETT, Transactions on Emerging Telecommunications Technologies, Wiley*. DOI: 10.1002/ett.2637.

Vinoski, S. (2004). Dark matter revisited. *Internet Computing, IEEE 8*(4), 81–84.

Walfish, M., H. Balakrishnan, and S. Shenker (2004). Untangling the web from dns. In *Proceedings of the 1st conference on Symposium on Networked Systems Design and Implementation - Volume 1*, NSDI'04, Berkeley, CA, USA, pp. 17–17. USENIX Association.

Wang, W., M. Motani, and V. Srinivasan (2009, oct). Opportunistic energy-efficient contact probing in delay-tolerant applications. *IEEE/ACM Trans. Netw. 17*(5), 1592–1605.

Wang, Y., J. Bi, and J. Wang (2012, 30 2012-aug. 2). Towards an aggregation-aware internet routing. In *Computer Communications and Networks (ICCCN), 2012 21st International Conference on*, pp. 1–7.

White, T. (2009). *Hadoop: The Definitive Guide* (1st ed.). O'Reilly Media, Inc.

Williams, N., L. Johansson, S. Hartman, and S. Josefsson (2012, August). Generic Security Service Application Programming Interface (GSS-API) Naming Extensions. RFC 6680 (Proposed Standard).

Winston, P. H. (1984). *Artificial Intelligence.* Reading, MA: Addison-Wesley.

Wittgenstein, L. (1922). Tractatus logico-philosophicus. *London: Routledge, 1981.*

Wood, L., W. M. Eddy, and P. Holliday (2009, march). A bundle of problems. In *Aerospace conference, 2009 IEEE*, pp. 1–17.

Wright, W. and D. Moore (2006). Agile language development: the next generation. In *Aerospace Conference, 2006 IEEE*, pp. 6.

Wu, H., F. Lin, and H. Zhang (2009, oct.). A novel data-oriented name service for next generation internet. In *Communications Technology and Applications, 2009. ICCTA '09. IEEE International Conference on*, pp. 790–795.

Xi, Y., M. Chuah, and K. Chang (2007, dec). Performance evaluation of a power management scheme for disruption tolerant network. *Mob. Netw. Appl. 12*(5), 370–380.

Xia, R. L. and J. K. Muppala (2010). A survey of bittorrent performance. *Communications Surveys Tutorials, IEEE 12*(2), 140–158.

Yang, D., J. Shin, J. Kim, and C. Kim (2009, mar.). Asynchronous probing scheme for the optimal energy-efficient neighbor discovery in opportunistic networking. pp. 1–4.

Yang, P. and M. C. Chuah (2006). Context-aware multicast routing scheme for disruption tolerant networks. In *Proceedings of the 3rd ACM international workshop on Performance evaluation of wireless ad hoc, sensor and ubiquitous networks*, PE-WASUN '06, New York, NY, USA, pp. 66–73. ACM.

Ye, Q. and L. Cheng (2008, june). Dtp: Double-pairwise time protocol for disruption tolerant networks. In *Distributed Computing Systems, 2008. ICDCS '08. The 28th International Conference on*, pp. 345–352.

Zhao, W., M. Ammar, and E. Zegura (2004). A message ferrying approach for data delivery in sparse mobile ad hoc networks. In *Proceedings of the 5th ACM international symposium on Mobile ad hoc networking and computing*, MobiHoc '04, New York, NY, USA, pp. 187–198. ACM.

Zhuang, S., K. Lai, I. Stoica, R. Katz, and S. Shenker (2005, November). Host mobility using an internet indirection infrastructure. *Wirel. Netw. 11*(6), 741–756.

Znati, T. B. and J. Molka (1992, apr). A simulation based analysis of naming schemes for distributed systems. In *Simulation Symposium, 1992. Proceedings. 25th Annual*, pp. 42–51.