

Addressing Scalability in Mobile Ad hoc Networks

Warren Kenny

A thesis submitted to the University of Dublin, Trinity College
in fulfillment of the requirements for the degree of
Doctor of Philosophy (Computer Science)

October 2012

Declaration

I, the undersigned, declare that this work has not previously been submitted to this or any other University, and that unless otherwise stated, it is entirely my own work.

Warren Kenny

Dated: October 29, 2012

Permission to Lend and/or Copy

I, the undersigned, agree that Trinity College Library may lend or copy this thesis upon request.

Warren Kenny

Dated: October 29, 2012

Acknowledgements

I would like to thank Dr. Stefan Weber for supervising my research and asking the fundamental questions that kept me on my toes all of these years. I would like to thank my mother, Deirdre, and father, David, for their understanding, support and their easy-going attitude towards this long endeavour, and my good friends Shane and Kalle for keeping my spirits up in the final stretch. Last but not least, I would like to thank Serena for sticking with me through the hard times and the good.

Warren Kenny

University of Dublin, Trinity College

October 2012

Abstract

Mobile Ad hoc Networks (MANETs) are wireless networks comprised of nodes which exhibit varying mobility and availability. Such networks are highly volatile, with network membership changing unpredictably over time. As Wifi-enabled mobile devices such as smartphones have become more ubiquitous and powerful, new applications which utilize ad hoc networking have emerged such as mobile multiplayer games and data sharing applications. Such applications currently work within single-hop distances, however, in the future multi-hop connections may be required in order to enable communication at increased distances.

In such a scenario, routing protocols which can operate at high device densities and cope with large networks will be needed. The provision of routing information can, however, have a detrimental effect on application throughput and network scalability due to medium access limitations and radio interference. Attempting to provide relevant routing information without negatively impacting MANET usability represents a significant challenge.

Existing approaches to routing in MANETs attempt to reduce these effects through the use of intelligent forwarding algorithms, flood propagation restrictions, dependence on geographical information or combinations of these approaches. These approaches are based upon wired network routing protocols which attempt to route packets to nodes with specific and unique identities or addresses. In a MANET, however, the identities of nodes may not be known in advance. Thus address or identity-based routing makes little sense in a MANET context. Additionally, the overheads generated by existing protocols when distributing routing information for every node or flooding queries throughout the network can have a significant negative impact on application throughput and network scalability.

In recent years, new communication approaches have emerged, such as protocols designed to dynamically discover services and resources in the network. These protocols attempt to provide information regarding available services to interested clients. This function is necessary as uncertainties regarding network membership and node capability caused by mobility and limited battery capacity make the use static associations between services and specific node

addresses impractical.

Existing works in the area of service discovery attempt to create a mapping between provided services and server node addresses which may then be used by an underlying routing protocol to route traffic between a client and a server. This approach generates significant communication overheads due to the use of both service and route discovery in the same network. More recent service discovery protocols attempt to reduce overheads by integrating directly with an existing routing protocol, in effect 'piggy-backing' their service advertisement or query messages on routing messages. Such cross-layer approaches have reduced overheads compared with separated protocols, however they still have a significant negative impact on application throughput due to their reliance on an underlying routing protocol.

In this thesis we discuss a new approach to improving scalability and usability for MANETs. Our protocol, Service Discovery Routing Protocol (SDRP), is designed to overcome the disadvantages posed by existing solutions in the service discovery space by reducing discovery overheads through the use of novel forwarding techniques and entirely removing reliance on an underlying routing protocol.

In an SDRP-based network, server nodes advertise their services throughout the network using a modified version of the OLSR Multi-Point Relay (MPR) algorithm; an approach which significantly reduces overheads through the avoidance of flooding. Our selective forwarding algorithm, Reduced MPR (RMPR), uses bloom filters to facilitate forwarding node selection and entirely removes the use of periodic beacons in order to reduce network overheads. Crucially, unlike networks based on OLSR, in SDRP networks only nodes that provide services propagate routing information, rather than all nodes. This removal of support for address-based routing significantly reduces overheads while maintaining low end-to-end delay, allowing networks of increased density to be formed without negatively impacting application throughput.

This thesis makes two key contributions: Firstly, a new forwarding algorithm called Reduced Multi-Point Relay which significantly reduces overheads when compared to the traditional approach. Secondly, a new routing protocol for MANETs called Service Discovery Routing Protocol based on service-oriented routing rather than address-based routing. SDRP utilizes Bloom Filters and the RMPR intelligent forwarding algorithm to significantly reduce overheads, thus improving network scalability; particularly in high node density scenarios.

Publications Related to this Ph.D.

- Warren Kenny and Stefan Weber, *Below Cross-Layer: An Alternative Approach to Service Discovery in MANETs*. In Proceedings of the 4th International Conference on Ad Hoc Networks (ADHOCNETS 2012), Paris, France. To Appear 2012.
- Warren Kenny and Stefan Weber, *APSALAR: Ad hoc Protocol for Service-Aligned Location-Aware Routing*. In Proceedings of the 1st International Workshop on Context-Aware Middleware and Services (CAMS 2009), Dublin, Ireland.

Glossary of Terms

- **AODV.** Ad hoc On-demand Distance-Vector routing: a reactive routing protocol for mobile ad hoc networks using on-demand route request flooding.
- **Bloom Filter.** A space-efficient data structure designed to represent sets. A bloom filter is represented by an array of m bits, with bits being set through the hashing of inserted elements using k hash functions.
- **Control Overheads.** The traffic generated by routing and service discovery protocols during their operation. Distinct from application traffic, control overheads are needed to facilitate route and service discovery, however they increase medium contention which negatively impacts application throughput.
- **LSD.** Lightweight Service Discovery protocol. A directory-based service discovery protocol for mobile ad hoc networks with the ability to fall back on peer-to-peer discovery in the absence of directory nodes. Integrates using cross-layer techniques with the Optimized Link-State Routing protocol.
- **Medium Contention.** Network communications involve the transmission of data through a medium between nodes. In the context of wireless networks, this is represented by a shared block of the radio spectrum. Nodes wishing to communicate in a wireless network must do so using this shared medium. This becomes contended as more traffic is transmitted, reducing the available medium access time.
- **MPR.** Multi-Point Relay. An algorithm used by a node disseminating messages to determine the minimum subset of 1-hop neighbours required to reach all 2-hop neighbours. Used by the Optimized Link-State Routing protocol.
- **OLSR.** Optimized Link-State Routing: a proactive routing protocol for mobile ad hoc networks based on selective rebroadcasting of topology control messages.
- **PDP.** Pervasive Discovery Protocol. An application-layer service discovery protocol for mobile ad hoc networks designed to reduce node energy consumption using a combination of on-demand querying and proactive service advertisement.
- **Routing Protocol.** A routing protocol specifies how nodes in a network communicate with each other. Routing protocols for mobile ad hoc networks typically facilitate route discovery over multiple hops.
- **Service.** A resource made available by one node to others in a network. i.e internet gateway, chat server, printer, game server.

- **SMF**. Service Magnetic Field protocol. A cross-layer service discovery protocol for mobile ad hoc networks based on the Ad hoc On-demand Distance Vector protocol. Operates by flooding service requests piggy-backed on AODV messages.
- **ZRP**. Zone Routing Protocol. A hybrid routing protocol for mobile ad hoc networks that combines aspects of proactive and reactive routing protocols.

Contents

Acknowledgements	vii
Abstract	vii
Chapter 1 Introduction	1
1.1 Motivation	1
1.2 Background Information	2
1.2.1 Mobile Ad hoc Networks	2
1.2.2 Routing Protocols	3
1.2.3 Service Discovery	5
1.3 Service Discovery in MANETs	6
1.4 Approach	7
1.5 Objectives	8
1.6 Scope	9
1.7 Contributions	10
1.8 Roadmap	10
1.9 Summary	10
Chapter 2 Related Work	12
2.1 Address-Based Routing Protocols	12
2.1.1 Proactive Routing Protocols	13
2.1.2 Reactive Routing Protocols	21
2.1.3 Alternative Approaches	26
2.2 Service Discovery in MANETs	31
2.2.1 Application-Layer vs Cross-Layer	32
2.3 Directory-Based Service Discovery Protocols	33
2.3.1 Distributed Service Discovery Protocol: Virtual Backbone	35
2.3.2 Lightweight Service Discovery: Peer-to-Peer Fallback	36

2.3.3	Service Rings: Semantic Hierarchical Discovery Overlay	37
2.3.4	SANDMAN: Cluster-Based Energy Efficient Discovery	39
2.3.5	Summary and Analysis	39
2.4	Peer-to-Peer Service Discovery Protocols	41
2.4.1	Mercury: Cross-Layer Service Advertisement	41
2.4.2	SMF: Cross-Layer Service Querying	43
2.4.3	Pervasive Discovery Protocol: Application-Layer Discovery	45
2.4.4	Other Approaches	46
2.4.5	Analysis	47
2.5	Summary	48
Chapter 3	SDRP: Service Discovery and Routing Protocol	50
3.1	Rationale	50
3.2	Approach	52
3.2.1	Design Stages	52
3.3	SDRP Core Design	54
3.3.1	Discovery Approach	54
3.3.2	Service Description	55
3.3.3	Routing	57
3.4	SDRP - Flooding	58
3.4.1	Service Advertisement	58
3.4.2	Example	61
3.4.3	Summary - SDRP-F	62
3.5	SDRP - Multi-Point Relay	63
3.5.1	Optimized MPR Selection	63
3.5.2	Logical Operations on Bloom Filters	65
3.5.3	Bloom Filter-Based MPR Selection	66
3.5.4	Service Advertisement	70
3.5.5	Example	70
3.5.6	Summary - SDRP-MPR	72
3.6	SDRP - Reduced MPR	74
3.6.1	MPR Selection using Variable-Length Bloom Filters	75
3.6.2	HELLO Message Removal	76
3.6.3	Border Node Selection	78
3.6.4	Example	78
3.6.5	Summary - SDRP-RMPR	83
3.7	Summary	85

Chapter 4	Implementation of SDRP	87
4.1	SDRP	88
4.1.1	Core Classes	88
4.1.2	SDRP and Delegate Interface	89
4.1.3	Routing Manager and Route Table	91
4.1.4	Local Area Monitor and MPR Calculation	94
4.1.5	Bloom Filter	99
4.1.6	Summary	101
4.2	The NS-2 Network Simulator	102
4.2.1	Limitations	102
4.2.2	Extending NS-2	104
4.3	SDRPAgent	104
4.4	SCDP	108
4.5	SMF	111
4.6	Summary	112
Chapter 5	Evaluation and Results	114
5.1	Experimental Setup	114
5.1.1	Target Network Configuration and Objectives	115
5.1.2	Mobility and Connection Pattern Generation	115
5.1.3	Experiment Distribution	117
5.1.4	Evaluated Protocols	119
5.1.5	Measured Outputs	121
5.1.6	Experimental Parameters	122
5.2	Network Scale and Density	124
5.2.1	Control Overheads	125
5.2.2	Service Availability	127
5.2.3	Request Delivery Ratio	129
5.2.4	Response Delivery Ratio	130
5.2.5	Application Throughput	132
5.2.6	Summary	133
5.3	Service Agent Density	135
5.3.1	Control Overheads	136
5.3.2	Service Availability	137
5.3.3	Request Delivery Ratio	137
5.3.4	Response Delivery Ratio	139
5.3.5	Application Throughput	140
5.3.6	Summary	141

5.4	User Agent Density	142
5.4.1	Control Overheads	143
5.4.2	Service Availability	143
5.4.3	Request Delivery Ratio	144
5.4.4	Response Delivery Ratio	146
5.4.5	Application Throughput	147
5.4.6	Summary	149
5.5	Mobility	150
5.5.1	Control Overheads	151
5.5.2	Service Availability	152
5.5.3	Request Delivery Ratio	153
5.5.4	Response Delivery Ratio	154
5.5.5	Application Throughput	155
5.5.6	Summary	156
5.6	Results	157
5.7	Summary	159
Chapter 6	Conclusions and Future Work	161
6.1	Achievements	161
6.2	Future Work	162
6.3	Summary	163
Appendix A	Bloom Filters	164

Chapter 1

Introduction

This thesis presents the Service Discovery and Routing Protocol (SDRP), a service-oriented routing protocol for mobile ad hoc networks (MANETs). SDRP is designed to improve MANET scalability and application throughput by reducing the overheads associated with the routing process. By removing support for address-based routing and incorporating message forwarding methods based on the use of novel data structures, SDRP allows services to be located and utilized while producing lower messaging overheads than existing service-discovery protocols.

This chapter first describes the motivations for this work, describes mobile ad hoc networks, routing protocols and service discovery protocols and gives an overview of existing related work. The challenges involved in this work are described as well as the its goal and contributions. Finally, an overview of the rest of this thesis is given in the form of a short road-map.

1.1 Motivation

Mobile devices which incorporate wireless communication capability are becoming increasingly ubiquitous in our everyday environment. Many of these devices are also capable of forming opportunistic wireless networks. Examples of such devices include smartphones (Sung et al., 2010), tablet computers (Peysakhov et al., 2005), PDAs (Sari et al., 2005) and laptops (Ansari, 2009).

Mobile Ad hoc networks (MANETs) have been a popular research topic for a number of years, however real deployments of MANETs outside of a research setting have yet to materialize. As the number of mobile devices in circulation which are capable of wireless communication increases, it is likely that their feature sets will eventually be expanded to include multi-hop

communication capability as well. In this scenario, routing protocols which can operate in large-scale and highly dense networks without impacting application performance will be required.

Current approaches to routing in MANETs attempt to provide routes to all nodes regardless of their function. This approach is similar to that of infrastructure networks; however, unlike infrastructure networks, MANETs are not divided into static stub networks which may assist in the routing process (Giordano, 2002). In addition, mobility and node availability can cause the network topology of a MANET to change dramatically over time. These factors make routing based solely on addresses costly in terms of generated overheads and this impacts negatively on the bandwidth available for real applications (Lye and McEachen, 2007).

Alternative approaches attempt to provide routes to nodes with specific attributes or content through a discovery process. This pattern acknowledges the dynamic nature of MANET topologies and membership, where node identities and capabilities may not be known ahead of time. However, existing discovery protocols operate by mapping attributes or content to node addresses (Flathagen and Øvsthus, 2008; Campo et al., 2006; Li and Lamont, 2005). These addresses are then used by an underlying address-based routing protocol to route messages to the receiver. These two-tiered approaches generate additional overheads and suffer from the dynamic nature of MANETs. This thesis investigates a new approach to this problem where routes are only made available to and from nodes which expose services to the network and control overheads are minimised through a novel use of compact data structures and controlled flooding of advertisements.

1.2 Background Information

In order for nodes to communicate over multiple hops in a wireless network, routes must be discovered which lead to their chosen destination. Routing protocols fulfil this requirement. This section defines the concept of a mobile ad hoc network, of a routing protocol and of a service discovery protocol.

1.2.1 Mobile Ad hoc Networks

Mobile Ad hoc Networks (MANETs) are defined as networks of mobile nodes which communicate over wireless links (Giordano, 2002), typically based on the IEEE 802.11 standard (IEEE, 2011). MANETs are infrastructureless, self-configuring networks, requiring no central coordination entities. Their topology and membership can be highly dynamic and difficult to predict, with nodes moving, joining and dropping out of the network over time. Nodes may also rely on batteries or some other exhaustible means for their energy. As such, MANETs present a far less reliable

medium for communication than wired networks and the challenges involved in providing that communication capability are varied.

MANETs also share the problems experienced in traditional infrastructure networks and wireless networks such as bandwidth limitations, power control, interference and medium contention (Hekmat and Van Mieghem, 2004). Research in the area of MANETs has focused on the development of routing protocols which can cope with unpredictable node availability and distribution (Rahman et al., 2010), as well as schemes for improving quality of service (Marwaha et al., 2008) and node power management (Jones et al., 2001).

Research in MANETs has encompassed a small number of use-cases, specifically battle-field deployments (Pullin and Pattinson, 2008; Canourgues et al., 2007) and disaster recovery scenarios (Aschenbruck et al., 2004; Shibata et al., 2007). These use-cases involve networks being deployed in hostile environments, with nodes moving in an unpredictable fashion across the topology using a limited power supply. More recently, use of smartphones as MANET nodes has been considered, however the same network specification is assumed to apply in this case.

A MANET node is assumed to be a mobile device with limited power, CPU, memory and bandwidth. Links between MANET nodes may not always be bi-directional due to differences in node hardware specification, power level or environmental effects. Nodes are assumed to be failure-prone due to their limited battery capacity and environmental effects. Nodes are not assumed to follow any particular mobility pattern except in specific deployment scenarios such as vehicular ad hoc networks (VANETs) or wireless sensor networks (WSNs), which are subsets of MANET research.

1.2.2 Routing Protocols

Routing protocols are designed to facilitate communication between mobile nodes, using other nodes in the network as relays for their messages (Corson and Macker, 1999) if necessary. Communication in this case is defined as the transmission of data packets between nodes as requested by an application or a component of the node's operating system. Routing protocols operate by mapping a destination address to a route, where a route is defined as a series of hops leading to the requested destination node. A hop is defined as one portion of the path between nodes, represented by two nodes which can achieve a bi-directional link. Thus MANET routing protocols are designed to discover available routes in the network in order to facilitate communication between nodes. Route discovery may be accomplished using a number of methods. The three most common methods are proactive, reactive and hybrid route discovery.

Proactive or table-driven routing protocols attempt to discover available routes before any communication attempt is made. This is typically accomplished by having each node period-

ically advertise its link-state or routing table throughout the network using broadcast packets which are relayed by neighbouring nodes. This approach is known as flooding (Giordano, 2002). This process allows nodes that receive advertisements to build routes to any destination. Proactive protocols allow routes to be used immediately when available, however they tend to produce greater control overheads than other approaches. Control overheads are defined as the messaging traffic produced by a routing protocol during its operation. The higher a node's control overheads, the more they may interfere with traffic from applications (Hekmat and Van Mieghem, 2004), reducing throughput and usability. Various optimizations can be applied to this process to reduce the number of broadcasts required and to allow advertisement receivers to better choose the optimal route to a source node. Protocols which follow the proactive route discovery pattern include the Optimized Link-State Routing protocol (Jacquet et al., 2001) and Destination-Sequenced Distance Vector routing protocol (Perkins and Bhagwat, 1994).

Reactive routing protocols attempt to discover routes on request rather than before a communication attempt is made. Routes are discovered by flooding queries throughout the node network until they reach the destination node. A response is then returned to the source through one or more routes, depending on the protocol's implementation. This allows the source node to form routes which may be used to send packets. Reactive routing protocols tend to produce lower control overheads than proactive protocols due to the fact that they only flood when needed rather than periodically. However, a route request typically takes longer to fulfil than when using a proactive protocol due to the delay involved in on-demand route discovery. Examples of reactive routing protocols include AODV (Perkins et al., 1999) and DSR (Johnson and Maltz, 1996).

Hybrid protocols combine aspects of reactive and proactive routing protocols. Approaches such as the Zone Routing Protocol (Haas, 1997) establish a routing zone around a node, with a radius measured in hop distance, in which routing messages are proactively distributed. Nodes which receive these messages update their routing tables and can then route packets to the source as required. If a destination node is not present in a source's local zone, a query is sent to nodes on the periphery of the local zone. These nodes propagate the query to neighbouring zones until the zone containing the destination is contacted. A response is sent containing the forwarding path to the destination and the source can then send its messages. This approach combines advantages from both reactive and proactive protocols. Overheads generated by the proactive topology message advertisement process are isolated to the routing zone surrounding each node and are thus reduced compared to a full proactive protocol. Overheads are further reduced by ensuring that reactive queries are sent to zone periphery nodes only, rather than flooded in an uncontrolled manner.

The approaches described all use node addresses in order to function and attempt to

provide routes to nodes regardless of their role in the network.

1.2.3 Service Discovery

Current network usage patterns typically revolve around communication between a client and a server. Basic network functions such as DNS and DHCP involve a client querying a service which resides on a server for required information and receiving a response while higher-level applications such as web browsing, networked chat systems and multiplayer games similarly follow the client-server model.

1.2.3.1 Services

A service may be defined as an application or device hosted on a server; a connected computer or some other device capable of communication on a network, which provides some functionality to other nodes in a network. Examples of typical services in a network include internet gateways, databases, proxy servers, network printers and DHCP servers.

In order for services to be utilized, they must either be discovered indirectly or the address of the host must be known in advance. Various protocols exist to facilitate service discovery in infrastructure networks and these will be discussed in the following section.

1.2.3.2 Service Discovery Protocols

Service discovery is the process of locating services within a network so that they may be used. Service discovery is an active area of research and has produced a number of approaches in recent years. Many service discovery protocols have been widely adopted in business and consumer networks, including Apple Bonjour (Apple Inc., 2002), Microsoft's Universal Plug and Play (Microsoft, 2008) and the bluetooth service discovery protocol (Mettala, 1999). Other examples of service discovery protocols deployed in infrastructure networks include the Service Location Protocol (SLP) (Veizades et al., 1997) and the Simple Service Discovery Protocol (SSDP) (Microsoft, 2008). These two protocols demonstrate the three main mechanisms for service discovery in infrastructure networks; *directory-based* discovery, *query-based* discovery and *advertisement-based* discovery.

SLP can operate using either directory-based discovery mode or query-based discovery mode. In directory mode, it designates nodes as either User Agents (UA), Service Agents (SA) or Directory Agents (DA). Service Agents have one or more services to provide to the network. They register their services and addresses with Directory Agents. Directory Agents respond to

queries for services by User Agents; nodes which are seeking particular services, by providing a list of SAs which can provide that service. UAs then connect directly to SAs in order to utilize their service. Directory Agents periodically advertise their presence through multicasting, allowing User Agents to discover and query them and allowing Service Agents to register their services. If directory nodes are not present in a network, SLP will utilize a query-based discovery approach instead. In this mode, UAs send multicast queries for services throughout the network until an SA which can provide the sought service responds.

This approach attempts to maximize service discovery success rates while minimizing impact on the network by reducing the overheads associated with discovery traffic. By restricting multicasting activity to a set of directories when possible, SLP can avoid having User Agents flood the network with queries. However, SLP allows query flooding to be used if no directories can be found in order to ensure that services are ultimately located.

SSDP operates using an advertisement-based discovery mechanism in which servers multicast advertisements for their services through the network. User agents seeking a service listen for these broadcasts and make a unicast connection to the source when the advertised services contain the one being sought. This approach has the advantage of not requiring a directory be present in the network, and scales well when there are few services in the network but many clients as multicasting is only performed by servers.

The query-based and advertisement-based discovery mechanisms described above are similar to the reactive and proactive routing approaches described in section 1.2.2. This similarity is exploited by a number of MANET service discovery protocols to optimize the query or advertisement process through direct integration with the underlying routing layer, as described in the following section.

1.3 Service Discovery in MANETs

Mobile ad hoc networks are composed of nodes which move unpredictably and may be unreachable due to a lack of energy or environmental factors. The identities of nodes may also be unknown at deployment time and the network may be open to the addition of new nodes whose identities and capabilities are unknown. In order for nodes to utilize shared resources in the network, they must be able to discover and communicate with them. Current service discovery protocols for MANETs accomplish this task using mechanisms similar to those used in the infrastructure network protocol described in section 1.2.3.2.

Increased medium contention in wireless networks leads to a reduction in capacity for application traffic (Hekmat and Van Mieghem, 2004). In order to reduce medium contention

caused by query and advertisement flooding, a number of service discovery protocols for MANETs have adopted the directory-based discovery mechanism described in section 1.2.3.2, such as those proposed by Li and Lamont (2005) and Sailhan and Issarny (2005).

This approach typically involves directory nodes advertising their presence in the network using flooding so that user and service agents can connect for service registration and requests. These protocols assume that the number of directories is lower than the number of service agents, otherwise direct advertisement by service agents would be more efficient. The placement of directory agents in the network is a key concern, as the inability for user and service agents to locate directories can lead to an inability for the protocol to operate or cause the protocol to fall back on a direct query or advertisement approach. Directory placement also increases service request latency, as user agents must first contact a directory agent in order to locate a server before the service can be used.

An alternative approach to service discovery involves directly query for or advertising services. This approach, used in protocols such SMF (Zhou et al., 2011), PDP (Campo et al., 2006) and Mercury (Flathagen and Øvsthus, 2008), avoids the designation of central coordinators; thus improving service request latency and robustness. However, direct querying or advertisement can generate excessive overheads when the number of clients or servers in the networks is high due to the broadcast storm effect (Tonguz et al., 2006).

The protocols described above operate by mapping service requests to network addresses which are then used by an underlying routing protocol to route messages between the client and server. This two-tier approach generates unnecessary overheads which may interfere with application traffic. A number of protocols, including SMF and Mercury, attempt to reduce this effect by directly integrating with the underlying routing protocol and piggybacking service query or advertisement messages onto routing protocol packets. This *cross-layer* approach results in reduced overheads when compared to application-layer protocols, however the overheads generated by the support of arbitrary node-to-node routing remain a concern; particularly as network density increases (Arora et al., 2012; Lee et al., 2003).

1.4 Approach

It has been shown that existing routing approaches generate high overheads in large-scale networks (Arora et al., 2012; Lee et al., 2003). These overheads interfere with application traffic, reducing throughput and increasing delay (Hekmat and Van Mieghem, 2004). In the case of proactive link-state protocols such as OLSR (Jacquet et al., 2001), while proactive routing allows nodes to immediately route messages to their destinations without route discovery delay, the overheads generated by advertising link state information throughout the network for all nodes

are significant and increase with network scale. This implies that support for arbitrary routes between nodes, without regard for node function in the network, can impact on network scalability. We refer to this approach as address-based routing. Support for address-based routing makes little sense in most MANET deployment scenarios, as nodes tend not to have knowledge of network membership or peer capability before joining the network.

Routing approaches based on service discovery acknowledge this by allowing nodes to discover the services made available by their peers. However, these approaches function by first discovering the address of a server and then using an underlying address-based routing protocol to facilitate communication between the client and server. For service discovery protocols based on proactive link-state protocols such as OLSR, scalability remains a concern. Similarly, approaches based on reactive protocols suffer from scalability problems due to query flooding, particularly when the number of clients in the network is greater than the number of servers.

Instead of utilizing an underlying address-based routing protocol, our approach facilitates routing only between clients and servers, with other nodes acting only as relays for communication. This reduces protocol messaging overheads and improves network scalability. Our approach designates nodes as either clients or servers, with servers advertising their services to clients using a proactive mechanism based on OLSR's multi-point relay algorithm (Jacquet et al., 2001). A proactive approach was used in order to reduce discovery delays and messaging overheads associated with query flooding. It is also well-suited to scenarios in which the number of clients is greater than the number of servers, which is assumed to be the case.

Clients receive these advertisements, record the included services and the route to the source server before rebroadcasting to a subset of their neighbours. Advertisements are further optimized through a novel use of bloom filters both as containers for service descriptors and link-state information, reducing their size. An additional optimization which avoids the use of HELLO messages for conveying link-state information further reduces messaging overheads.

Our approach is designed to improve network scalability and application throughput through a reduction in messaging overheads associated with service discovery. The contributions of this thesis include the SDRP non-address-based routing model and the bloom filter-based link-state dissemination mechanism which is used to facilitate service advertisement.

1.5 Objectives

As consumer smart devices become increasingly ubiquitous, new potential deployment scenarios for MANETs involving network densities and scales beyond those previously studied have begun to emerge. Such deployment scenarios surpass the scalability of existing routing approaches

which use address-based routing.

In this thesis, we will explore the limitations of existing routing approaches for MANETs; particularly in scenarios involving large-scale and high density networks. We will also evaluate the performance of existing service discovery approaches in such networks, with an emphasis on application throughput and service discovery success rates.

We will then compare these approaches to our own service discovery protocol, SDRP, in order to demonstrate that support for address-based routing in MANETs hinders their usability in high-density networks.

1.6 Scope

This thesis describes a service-oriented routing protocol for high-density MANETs. This includes a space-efficient approach to service advertisement based on bloom filters. The performance of this protocol is evaluated using a quantitative approach and is compared against other service-oriented routing protocols in simulations involving large-scale and high-density networks. This work does not present a formal analysis of the protocol.

SDRP improves scalability by removing support for routing between arbitrary nodes, instead only supporting routes between clients and servers only. It is assumed that client-server communication is the primary form of interaction between nodes in a network in which SDRP is deployed. This assumption is discussed informally, however a formal proof of this assumption is considered to be outside the scope of this work. SDRP operates using an advertisement-based approach which best suits networks in which there are fewer servers than clients.

While it is assumed that the reduction in protocol overheads gained through the use of SDRP results in improved energy efficiency for mobile nodes, an analysis of node energy consumption is not included in this work and is considered to be outside the scope of this thesis.

SDRP is not well-suited to the advertisement of complex service descriptors which include optional elements such as attributes. The use of a bloom filter as a service descriptor container in advertisements means that only membership queries by clients are supported; the services included in the filter cannot be deserialized and inspected. Complex service ontologies are not supported by the protocol and are outside the scope of this work.

Finally, it should be noted that issues of node trust and security are not considered in this thesis.

1.7 Contributions

The contributions of this work are as follows: first, we present a novel approach to service discovery that does not rely on an underlying address-based routing protocol to facilitate communication between user and service agents. Current work in the area of service discovery is focused on cross-layer approaches, however we aim to demonstrate that a protocol which forgoes reliance on address-based routing greatly outperforms these approaches and significantly enhances the viability of certain MANET deployment scenarios.

Secondly, we present a novel approach to selective message rebroadcasting based on bloom filters that significantly decreases generated control overheads when compared with existing methods. Finally, we present a comprehensive evaluation of our approach and a comparison with a number of existing service discovery protocols.

1.8 Roadmap

The remainder of this thesis is organized as follows. Chapter 2 presents a review of existing routing approaches for mobile ad hoc networks, including address-based and service-oriented approaches. Chapter 3 describes the SDRP, including a description of its advertisement mechanism, messages and interfaces.

Chapter 4 describes the implementation of SDRP as well as a description of how SDRP can be deployed outside of a simulated network. It also describes the modifications made to the ns-2 simulator and other evaluated routing protocols in order to support thorough measurements of metrics such as application throughput and message delivery success rates.

Chapter 5 presents an evaluation of SDRP and comparison with existing routing approaches in a number of simulated network configuration. The chapter also includes a discussion of the results obtained. Finally, chapter 6 concludes this thesis and discusses possible future work.

1.9 Summary

This chapter outlines the objectives and scope of this thesis. This chapter began with an outline of the motivation for this work and then presented a high-level overview of mobile ad hoc networks. This overview included a discussion of new deployment scenarios for MANETs which are expected to become more relevant in the coming years. An outline of existing routing approaches for

MANETs was then presented which demonstrates the limitations of such approaches in these deployment scenarios.

Finally, we concluded this chapter by detailing our approach to solving these limitations and discussing the objectives scope and contribution of this thesis.

Chapter 2

Related Work

This thesis deals with routing for mobile ad hoc networks. Routing is a key concern in MANETs, where node mobility and availability causes the network topology to change over time. As outlined in Chapter 1, current approaches to routing have difficulty performing acceptably in the network deployment scenarios which have recently begun to emerge, particularly due to their support for routing between all nodes, regardless of their function.

In this chapter, we review existing approaches to routing for MANETs, focusing initially on address-based routing protocols before moving on to current approaches to service discovery. This work attempts to demonstrate that reliance on an underlying address-based routing protocol is detrimental to the performance of existing service discovery approaches. Section 2.1 explores address-based routing protocols, focusing in particular on those protocols upon which related service discovery approaches are based.

Section 2.2 begins the exploration of service discovery in mobile ad hoc networks by giving a brief overview of the area before moving into a deeper explanation of the different approaches to integration with address-based routing protocols. Section 2.3 details directory-based service discovery, including virtual backbone and cluster-based approaches. In section 2.4, peer-to-peer approaches, including advertisement and query-based protocols, are discussed. Finally, section 2.5 summarises this chapter.

2.1 Address-Based Routing Protocols

The MANET research community's traditional focus when developing routing protocols has been the replication of infrastructure routing capabilities and usage patterns, including the use

of IP addresses for node identification and the provision of routing for all nodes regardless of their function. Address-based routing protocols attempt to discover routes to node's possessing particular IP addresses.

Proactive routing protocols attempt to discover and maintain routing information at all times, propagating routing messages throughout the network and resulting in constant network overheads. Reactive protocols attempt route discovery upon request using query dissemination, leading to a delay before discovery completes. Hybrid approaches are a mix of proactive and reactive mechanisms, with proactive route maintenance taking place in certain zones of the network and reactive discovery being used outside those zones. Protocols may be further categorized as link-state or distance-vector approaches.

Link-state protocols formulate routes through calculations based on link-states between nodes, gathered using one of the above mechanisms. Nodes typically calculate the shortest path to a given destination using Dijkstra's algorithm (Dijkstra, 1959) or variants thereof.

Distance-vector protocols calculate routes through the sharing of routing tables between nodes. With knowledge of peer routing tables, which include a distance metric to known destinations, nodes can calculate the shortest path to a given destination based on the known distances between nodes.

Link-state protocol shortest-path-calculation algorithms typically result in greater processing overheads but require lower messaging overheads, as link-state messages only include data on nearest neighbours. Distance vector approaches typically produce greater messaging overheads due to the requirement for sharing whole routing tables, which may include routes to distant nodes as well as neighbours, while needing lower processing overheads due to the inclusion of distance metrics in recorded routing tables.

In this section we will describe a number of existing address-based routing protocols based on the above approaches, with a particular focus on the most widely-used protocols in each category and those upon which service-discovery protocols tend to be built.

2.1.1 Proactive Routing Protocols

Proactive routing protocols attempt to maintain up-to-date routing information between all nodes in a network at all times. Nodes maintain routing tables which are updated upon receipt of new updates from peers in the network (Royer and Toh, 1999). Protocols in this family differ in terms of how updates are propagated, the content of updates, how they respond to node movement and their route table structure.

Proactive protocols can respond immediately to route requests, reducing the delay as-

sociated with route discovery, as routing information is consolidated before requests are made. However overheads are generated by route information propagation regardless of request activity. Thus, in networks with infrequent route requests, proactive protocols are inefficient in terms of bandwidth and energy usage.

Similarly, high-density networks are also unsuitable for proactive protocol deployment as the periodic propagation of route information by all nodes results in an increase in overheads and medium contention, reducing application throughput. This was demonstrated recently by Arora et al. (2012) in a statistically-rigorous study in which the optimized link-state routing protocol (OLSR) was evaluated in scenarios involving high node densities. OLSR is considered to be one of the most efficient proactive routing protocols in terms of generated overheads due to its optimization of route information propagation.

2.1.1.1 Local Advertisement Protocols

Proactive protocols must periodically distribute routing data to their peers in order to build reliable route tables. In order to avoid flooding route information, a number of protocols instead broadcast updates to their local neighbours periodically, with routing information thus cascading through the network over time rather than flooding immediately. This approach reduces network overheads and avoids the broadcast storm problem; however, it also tends to reduce routing table accuracy as route information takes more time to propagate; a problem exacerbated in large-scale networks.

Destination-Sequenced Distance Vector (DSDV) routing protocol (Perkins and Bhagwat, 1994) is one of the earliest examples of a distance-vector proactive routing protocol. DSDV nodes broadcast the contents of their routing tables to their immediate neighbours both periodically and in response to topology changes, allowing receiving nodes to build a more complete map of the network.

DSDV nodes broadcast either a full routing table or a set of changes since the last broadcast, depending on changes to the local network topology. The protocol also specifies that broadcasts be sent in response to topology changes, resulting in broadcast frequency increasing with node mobility, interference or power loss.

Performance studies have shown that DSDV tends to perform poorly in scenarios involving frequent topology changes, both due to the increased overheads produced by more frequent full-table dumps and lowered throughput (Johansson et al., 1999). Drops in throughput have been shown to occur due to route invalidation. As DSDV nodes propagate their routing tables through periodic local broadcasts, additional time is required to rectify errors in routing tables produced by topology changes when compared to approaches which quickly flood updates throughout the

network.

The Wireless Routing Protocol (WRP) operates similarly to DSDV, with nodes exchanging routing information with direct neighbours (Murthy and Garcia-Luna-Aceves, 1995). WRP nodes maintain four tables containing packet retransmission lists, distance tables and routing information. In large-scale networks these tables can consume significant amounts of device memory. WRP attempts to improve robustness in the presence of interference or jamming by including an acknowledgement mechanism for route information updates into its messaging system. Nodes retransmit route updates until neighbours include an acknowledgement for that update's sequence number in their own periodic update messages, the targeted neighbours can no longer be detected or the message retransmission limit is reached. This mechanism ensures that nodes receive route updates even in the presence of interference, however, it also increases message sizes due to the presence of acknowledgements and overheads due to retransmissions.

The Global State Routing (GSR) protocol, like DSDV and WRP, disseminates routing information using local transmissions only (Chen and Gerla, 1998). GSR, however, is a link-state protocol, meaning that disseminated information contains link-state information instead of full or partial route table dumps. Thus, GSR nodes must calculate routes to destinations.

Fish-eye State Routing (FSR) refines the algorithm used by GSR with an aim to improve protocol scalability. With increasing network scale, the size of link-state message can balloon and consume large amounts of network bandwidth. FSR reduces this effect by including link-state entries in update messages more frequently for nodes which are nearby and less frequently for distant nodes. Thus, all update messages may contain link-state information for nodes within, for example, 2 hops but nodes further than this will have their link-states included less frequently. This approach has the effect of reducing network overheads but also reduces route accuracy at greater distances to the destination, however routes become more accurate as the packet approaches the destination, as link-state information becomes fresher and is updated more frequently.

2.1.1.2 Flooding Protocols

As detailed in the previous section, approaches based on local advertisements avoid the broadcast storm problem resulting from message flooding. However, such approaches suffer from degrading route accuracy with greater network scale, as updates take time to propagate across the network. Approaches based on route information flooding have been developed and attempt to avoid the broadcast storm problem in a number of ways.

Distance Routing Effect Algorithm for Mobility (DREAM) is a proactive protocol that relies on the distribution on geographical information rather than distance-vector or link-state (Basagni

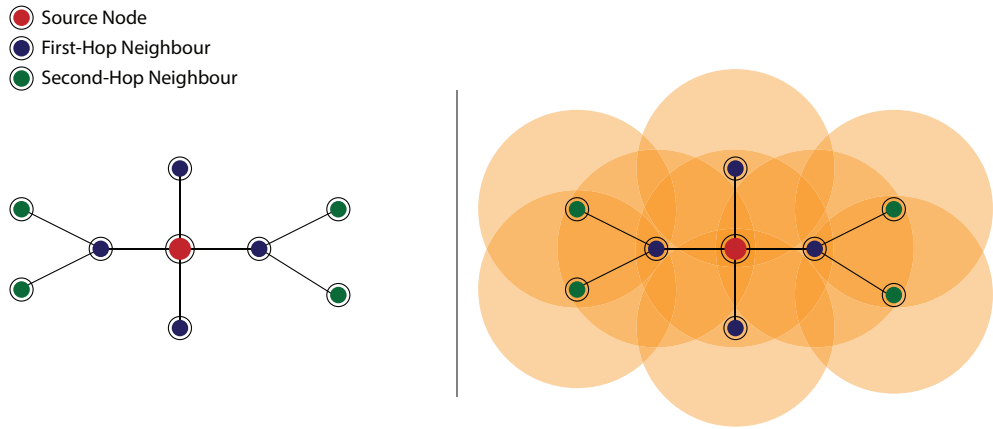
et al., 1998). Thus the protocol assumes that nodes have the ability to determine their position using a mechanism such as GPS. DREAM nodes flood control packets containing their current position at a frequency determined by their mobility. As position information is assumed to be available, mobility can be determined easily. Mobility also determines control packet forwarding distance, with control packets being dropped at a distance from the source specified in the packet itself.

Immobile nodes send control packets infrequently and with a high distance limit while mobile nodes send control packets frequently and with low distance limits. This results in control overheads that are well-suited to the dynamics of the network. Control packets are also small when compared to distance-vector or link-state packets used by other protocols. DREAM nodes route packets by sending them to neighbours that are in the direction of the packet destination. DREAM nodes may forward packets to more than one neighbour in order to improve reliability, however this results in greater overheads when compared to the direct unicasting used in other protocols that operate with greater certainty of route composition. Thus while DREAM's control overheads are low compared to alternative approaches, its packet forwarding strategy is relatively inefficient and its reliance on position determination capability for nodes reduces its deployment potential.

The Optimized Link-State Routing Protocol (OLSR) (Jacquet et al., 2001) is an approach based on the flooding of link-state information. Rather than simple message flooding, in which nodes broadcast link-state messages which are then rebroadcasted by all recipients, OLSR attempts to determine the minimum subset of a node's neighbours required to reach all second-hop neighbours through rebroadcasting. OLSR is a highly extensible protocol, providing a unified message format that allows new capabilities to be added to the protocol over time. This feature has been used by a number of service discovery protocols to implement cross-layer discovery on top of OLSR (Flathagen and Øvsthus, 2008; Li and Lamont, 2005). As such, and due to the fact that the OLSR protocol's flooding approach is of particular relevance to this thesis, it will be elaborated upon in greater detail.

As shown in figure 2.1, the simple flooding pattern results in redundant rebroadcasts by receiving nodes. This is due to the fact that the coverage area of wireless antennas are very likely to overlap with each other, particularly in dense networks, resulting in an area being covered by the same broadcast packet several times. This issue is known as the broadcast storm problem, as described by Tseng et al. (2002).

The north and south first-hop neighbours illustrated in figure 2.1 do not need to rebroadcast the received routing information as they are not in range of any second-hop neighbours. Protocols based on simple flooding approaches do not take this into account, resulting in unnecessary overheads which increase medium contention. This problem is exacerbated in high-density



55

Fig. 2.1: In a simple flooding pattern, advertisement broadcast by the source are rebroadcast by all receivers. This can result in a broadcast storm.

networks, where the problem of redundant rebroadcasts is worsened.

OLSR determines the minimum set of neighbours which should rebroadcast link-state advertisements using an algorithm called multi-point relay (MPR). OLSR nodes periodically broadcast beacons or HELLO messages to their immediate neighbours. These messages contain the source node's network address and a list of IP addresses from neighbour nodes, compiled from recently received HELLO messages. In order to further decrease redundant broadcasts, OLSR is capable of sending several message types in a single packet. Thus a given packet may contain a HELLO message, a TC message and the payload from a protocol extension.

8		16		24		32	
Reserved				HELLO Message Interval		Willingness to Forward Traffic	
Link Code		Reserved		Link Message Size			
Neighbour Interface Address							
Neighbour Interface Address							
...							
Link Code		Reserved		Link Message Size			
Neighbour Interface Address							
Neighbour Interface Address							
...							

Fig. 2.2: OLSR HELLO Messages contain link-state information for all neighbours, allowing receivers to calculate an MPR set.

The link state of each neighbour is also included in the message, as shown in figure 2.2, allowing receiving nodes to ascertain whether their link to the source of the HELLO message is symmetric. This is a useful feature, as differences in hardware may result in differing wireless coverage among nodes. Lists of neighbour interface addresses are preceded in the packet by a link code, denoted in green, indicating the link state between the packet source and that neighbour. The link code can indicate one of the following link states:

- Symmetric - The source and neighbour can receive each other's packets
- Asymmetric - The source can receive the neighbour's packets, but the neighbour has not listed the source in recent HELLO messages.
- Lost - No HELLO messages have been received from the specified neighbour after a pre-defined span of time, indicating link loss.
- MPR - The source has designated the neighbour as a multi-point relay.

HELLO messages are not rebroadcasted by receiving nodes, they are instead used to calculate an MPR set: the minimum set of one-hop neighbours required to reach all two-hop neighbours.

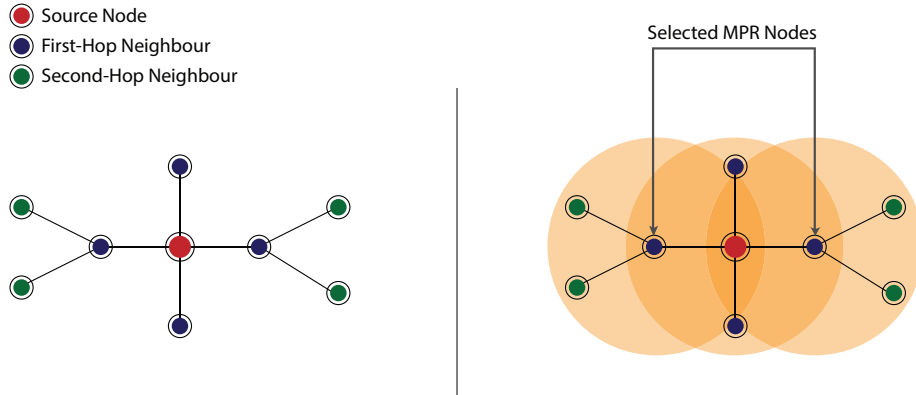
OLSR nodes recalculate their MPR sets in response to changes in the local network topology observed in received HELLO data. Using the link-state data contained in received HELLO messages, a set of two-hop neighbours with bi-directional links with one or more bi-directionally linked one-hop neighbours is constructed.

Any one-hop neighbours with unique links to one or more two-hop neighbours are designated as MPRs. Then, nodes are added to the MPR set in order of their coverage of two-hop neighbours, from largest to smallest, until all two-hop neighbours are covered by the nodes contained in the MPR set.

Neighbours selected as multi-point relays have their link-code set to *MPR* in subsequent HELLO messages, indicating that they should re-broadcast topology control messages from the source. The MPR algorithm reduces the number of rebroadcasts required to propagate link-state information, resulting in lower messaging overheads and medium contention when compared to simple flooding. This pattern is illustrated in figure 2.3.

Link-state information is distributed in the form of topology control (TC) messages. Only nodes that have been selected as MPRs send TC messages. This measure reduces network overheads associated with TC distribution while ensuring that full link-state information is still conveyed.

As shown in figure 2.4, TC messages contain a list of the source node's neighbour addresses along with a sequence number. If a message size restriction exists, the TC message must at least



55

Fig. 2.3: OLSR optimizes broadcasting by designating certain nodes as multi-point relays - nodes which may rebroadcast topology control messages.

8	16	24	32
Sequence Number		Reserved	
Advertised Neighbour Address			
Advertised Neighbour Address			
Advertised Neighbour Address			
...			

Fig. 2.4: OLSR TC Messages contain the link-state of the source node.

contain the addresses of any nodes that have selected the originator as an MPR. If the number of symmetric neighbours exceeds the maximum packet size for the network, additional TC messages containing the remaining neighbour addresses are sent.

Receiving nodes record the sequence number and will not rebroadcast a TC message with a source-sequence number pairing that they have already received. This prevents loops in rebroadcasts and is a common strategy for routing protocols.

In order to further reduce messaging overheads, only those nodes that have been selected as MPRs by one or more neighbours broadcast TC messages. This restriction results in fewer TC messages being distributed while still ensuring that all nodes are included in link-state advertisements. Upon receipt of a topology control message, nodes validate the message by checking the included sequence number against the last sequence number received from the same originating address. If the new sequence number is greater than the one last recorded, this TC

message is considered to be 'fresher'. Finally, if the link with the message's last-hop sender is symmetric, the message is considered valid and the receiver's routing table is updated using a shortest-path algorithm.

2.1.1.3 Analysis

OLSR's MPR algorithm reduces the detrimental effects of message flooding, however this comes at the cost of requiring periodic HELLO message broadcasts by all nodes in order to establish local link states prior to MPR selection. OLSR has also been shown to suffer performance drops in dense networks (Arora et al., 2012), particularly with lower topology control message send intervals, despite the optimizations granted by the MPR algorithm.

Greater topology control send intervals can result in poor route discovery success rates, particularly in larger networks where topology changes compound invalidation of link-state caches over greater distances. Thus OLSR exhibits a weakness in larger, more dense networks due to the need for lower TC intervals that simultaneously produce large increases in network overheads, reducing throughput for non-routing traffic.

DSDV exhibits similar issues in large or mobile networks, particularly in terms of route table convergence over large distances. This is due to the fact that route table updates effectively cascade through the network at a rate which is dependent on the frequency of table dump and update broadcasts. While DSDV does react to topology changes by broadcasting updates, nodes first cache and consolidate received updates before broadcasting; delaying convergence with increasing distance (Broch et al., 1998).

Mobility contributes to overhead generation in DSDV networks due to the increased incidence full-dump broadcasts in response to topology changes (Broch et al., 1998). Larger networks also experience higher routing overheads due to the increased size of DSDV messages required to contain routes for all nodes. OLSR experiences similar increases in overheads in high-density networks due to the increase in HELLO and TC message sizes required to contain link-state information for a greater number of neighbours.

GSR and WRP's periodic broadcast strategy worsens the convergence problem when compared to DSDV. WRP, however, has advantages over DSDV such as its reliability mechanism. Like DSDV, WRP sends updates when possible rather than full table dumps, reducing bandwidth consumption. GSR link-state messages are effectively full dumps and consume large amounts of bandwidth as the network scale increases. Thus GSR has no advantages compared to the other protocols described in this section, it is a very early and relatively simple proactive protocol.

FSR attempts to overcome GSR's scalability issues using an interesting mechanism that reduces bandwidth usage at the cost of reduced accuracy with increasing route length. DREAM's

approach is novel and effective, however its assumption that a positioning system is available for all nodes limits its possible deployment scenarios, particularly in truly ad-hoc networks where devices may have widely differing capabilities and may not have positioning devices.

Protocol	Dissemination Mechanism	Update Content	Update Frequency	Advantages	Disadvantages
DSDV	Local	Distance Vector	Periodic and Reactive	Partial Update Support, Reacts to Topology Changes	High Messaging Overheads at High Scale
WRP	Local	Distance Vector	Periodic	Reliability through Update Acknowledgements	High Messaging Overheads and Poor Convergence at High Scale
GSR	Local	Link-State	Periodic		High Messaging Overheads and Poor Convergence at High Scale
FSR	Local	Link-State	Periodic	Low Control Overheads	Poor Convergence with Scale
DREAM	Flooding	Geographical	Mobility Based	Low Control Overheads	Requires Positioning Information
OLSR	Flooding	Link-State	Periodic	Efficient Flooding, Fast Convergence	Requires HELLO Packets for Local Link-State and MPR Calculation

Fig. 2.5: Proactive Routing Protocol Comparison

The above protocols clearly demonstrate the cost of supporting route discovery for all nodes in a network, particularly at increasing network scales. Each protocol must make some compromises in order to perform acceptably which can impact their quality of service in various scenarios. Table 2.5 outlines the operation, advantages and disadvantages of these protocols in a more concise form.

2.1.2 Reactive Routing Protocols

Reactive protocols discover routes on-demand rather than through the periodic exchange of route table or link-state messages (Giordano, 2002). Unlike proactive protocols, reactive routing protocols do not require periodic broadcasts or floods in order to discover routes, instead route requests are flooded throughout the network until the destination node, or a node with a valid route to the destination, is reached. Request responses are then used to select the best route.

Dynamic Source Routing (DSR), an early and heavily investigated reactive protocol, is based on a route request flooding strategy. DSR is a source routing protocol, meaning that nodes in a DSR network specify the entire route that should be taken by sent packets by including a source route in a special packet header. Intermediate nodes on a source route send forward received packets to the next address specified in this header until the destination is reached, as shown in figure 2.6. This method causes the packet size to increase with increasing path length and may lead to packet fragmentation depending on the size of the packet payload and maximum packet size.

Source routes are discovered by flooding route requests on-demand throughout the net-

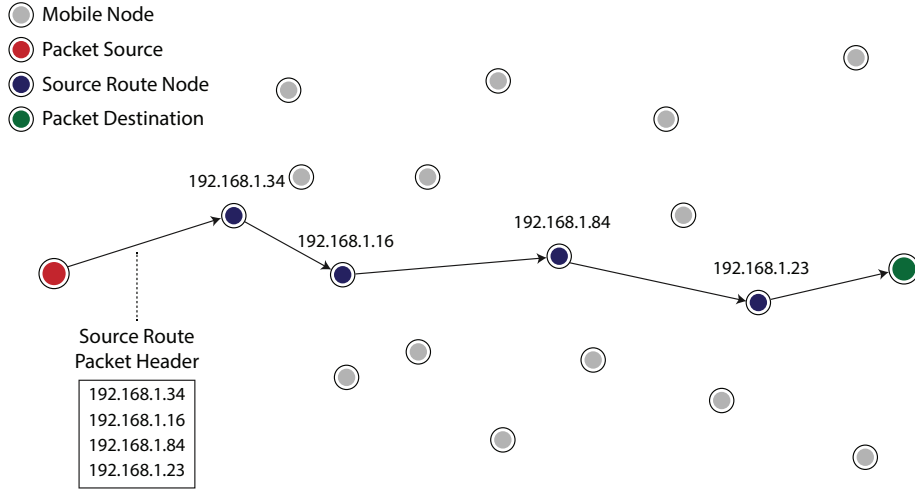


Fig. 2.6: The path to be taken by data packets is specified in the DSR source route header. Intermediate nodes in the path forward the packet based on this route.

work. Flooded route requests contain a sequence number that is used to prevent request loops. Upon receipt of a route request, intermediate nodes check whether the source address-sequence number pair has been previously received and discard it if it has. Otherwise, the intermediate node appends its own address to the source route header in the route request and rebroadcasts the packet.

The route request proliferates throughout the network until the destination node, or a node with a cached route to the destination, is reached. If the destination node, or a receiving node with a valid route to the destination, does not have a source root to the request originator, it will piggyback a route reply containing the requested source route onto a route request for the originator's address. Otherwise, if a source route to the originator is available, it will be sent directly.

DSR specifies a number of optimizations to the protocol designed to shorten the distance travelled by packets and speed up route requests. As mentioned above, if a node receiving a route request already has a source route to the requested destination, it can directly reply to the originator rather than forward the packet to the destination. If nodes have the ability to operate in promiscuous mode, in which they receive and act upon packets for which they are not the designated receiver, they may forward packets to their ultimate destination if their route cache contains a shorter route than the packet's source route. Nodes will also send a route reply message specifying the improved route to the packet source in order to update its route cache.

Ad-hoc On-demand Distance Vector (AODV) acts as the basis for a number of service discovery protocols (Zhou et al., 2011; Garcia-Macias and Torres, 2005) and is one of the most widely-researched reactive protocols for MANETs (Perkins et al., 1999). AODV operates by flooding route requests throughout the network on-demand. Once a request packet has reached the target node, or nodes with a valid route to the target, route requests are returned to the request source via unicast along the temporary route formed by the request. Each intermediate node records the next-hop address towards the requested destination and increments the response packet's hop-count field.

Upon receipt of route responses, request originators choose the route with the shortest path to the destination based on hop count. Data packets are then sent to neighbour acting as the next hop along the chosen route. While AODV is a distance-vector protocol, it does not operate by sharing entire route tables through protocol messages. Instead, intermediate nodes on the route traversed by request and response packets record packet sources, destinations and next hop addresses in order to construct a routing table. These tables are then used by intermediate nodes to route data packets on a per-hop basis.

AODV implements an error handling mechanism in which nodes maintain awareness of links for which they are acting as relays and, upon loss of contact with the next hop on an active link, send error messages back along the path to the source indicating that the link is no longer valid. Sources may then initiate a route request to regain contact with the destination node.

The Temporally Ordered Routing Algorithm (TORA) is a query-flooding algorithm that operates using directed acyclic graphs (DAC) in order to facilitate route repair (Park and Corson, 1997). TORA is based on an earlier protocol, Lightweight Mobile Routing protocol (LMR), that incorporates the ability to repair route failures due to broken links using a process of link reversal (Corson and Ephremides, 1995). This results in cheap route maintenance as control packets resulting from link reversal and route repair occur in a small area local to the broken link.

Unlike AODV, TORA maintains multiple routes to a given destination, allowing it to ignore route breakages as long as alternative routes remain available. TORA has been shown to perform well in large-scale, network scenarios, however its route maintenance and partition detection algorithms tend to produce excessive overheads, making it less suitable for more mobile networks (Gupta et al., 2010).

AODV and DSR select routes based on a distance metric; i.e hops, while TORA takes a similar approach based on a virtual 'height' metric measured during the formation of DACs. Associativity-Based Routing (ABR) uses a similar query-flooding approach to route discovery, however routes are selected on the basis of stability rather than distance to the destination (Toh, 1996).

Stability is measured using an algorithm whereby nodes periodically broadcast HELLO messages containing their address and an associativity 'tick' value that is decremented as the node's local topology changes and increments the longer it remains stable. The obvious disadvantage to this approach is that the selected route may not be the shortest, thus increasing end-to-end delay; however more stable routes are likely to achieve greater overall throughput due to the decreased likelihood of link breakage. While ABR does not cache multiple routes to a given destination, like TORA it has the ability to initiate localized route repairs, reducing control overheads when compared to AODV and DSR.

2.1.2.1 Request Flooding

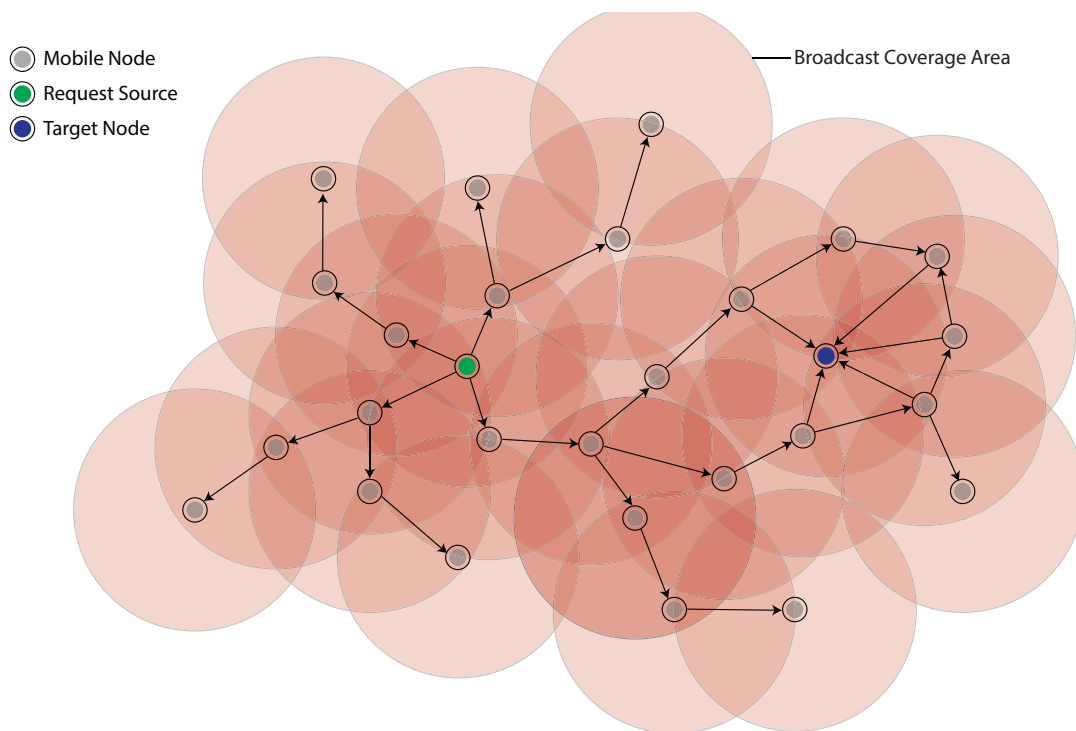


Fig. 2.7: Reactive Route Discovery Pattern. While reactive protocols typically do not require periodic message exchanges in order to function, this advantage is offset by their naive and inefficient request flooding pattern.

The on-demand approach to query flooding reduces overheads when compared to periodic approaches used in proactive protocols, particularly in scenarios in which there are few nodes requesting routes. However, in dense networks and networks in which route requests are frequent,

this naive flooding approach generates unnecessary interference that impacts delivery of application traffic. This is illustrated in figure 2.7, where it can be seen that the coverage areas for request broadcasts frequently overlap and requests are likely to propagate away from the request target, compounding the interference problem. Flooded route requests also have the potential to interfere with each other, reducing the reliability of the route request process and potentially resulting in failure.

This effect is worsened with increasing network density and mobility. Both AODV and DSR react to link breakage with additional route request floods. AODV attempts to limit this effect by placing a time-to-live value on the propagation distance of route requests in terms of number of hops traversed, however this does not address the interference problem experienced in dense networks.

2.1.2.2 Analysis

Reactive protocols tend to out-perform proactive protocols in terms of control overheads and throughput in scenarios involving large-scale, low-density networks with low route request frequencies. This is due to the periodic route advertisement broadcasts initiated by proactive protocols. As reactive protocols flood only on-demand, control overhead responds to request frequency rather than to the number of nodes in the network. Lower network density also reduces redundant coverage during request floods and reduces the likelihood of interference causing request failures.

However, reactive protocols experience greater end-to-end delays when compared to proactive protocols due to the wait period between route request initiation and the return of route responses. Proactive protocols cache available routes before requests take place, allowing immediate forwarding to take place.

Protocol	Response Content	Multiple Routes	Route Repair	Advantages	Disadvantages
DSR	Source Route	Yes	Erase and Retry	Multiple Routes Improve Redundancy, Loop-Free	Scalability Problems, Large Message Size
AODV	Hop Count	No	Erase and Retry	Small Message Size, Adaptive	Expensive Route Repair Algorithm, Lack of Route Redundancy
TORA	DAC Neighbour Height	Yes	Local Repair	Multiple Routes Improve Redundancy, Local Route Repair	Complex, High Overheads During Partitions and Route Repair
ABR	Link Stability	No	Local Repair	Small Message Size, Route Stability	Non-Optimal Route Length, Lack of Route Redundancy

Fig. 2.8: Reactive Routing Protocol Comparison

Proactive protocols such as DSDV and GSR can avoid broadcast storms through local exchange of route tables. Similarly, OLSR can avoid naive flooding through optimizations based

on knowledge of the link-states of neighbouring nodes. As reactive protocols are designed to send messages only on-demand, these mechanisms cannot be used, resulting in the use of simple flooding.

This approach leads to redundant rebroadcasts and unnecessary medium contention. This problem is further exacerbated as network density increases, with redundant rebroadcasts occurring more frequently and with more deleterious effects. Previous performance studies have by Adam et al. (2010) have shown that both AODV and DSR experience large increases in routing overhead with increasing node densities. Reactive protocols also suffer from greater end-to-end delay than proactive approaches, as routes must be discovered on-demand. By contrast, proactive approaches obtain knowledge of available routes before they're required, eliminating discovery delay. Table 2.8 outlines the functionality, advantages and disadvantages of the described protocols.

2.1.3 Alternative Approaches

A number of alternative approaches to routing in MANETs have been developed in recent years that diverge from the proactive/reactive and flat-network approaches used in earlier protocols.

2.1.3.1 Cluster-Based Routing

Clustering mechanisms for MANETs are designed to promote scalability through the division of mobile nodes into virtual groups, allowing for effective topology control (Rajaraman, 2002) and routing coordination (Yu and Chong, 2005). Unlike infrastructure networks, which cluster based on abstract characteristics such as subnet membership, mobile node mobility and the cost of multi-hop coordination requires that clustering occur on a geographical basis in MANETs. Clustering can allow the coordination of routing through a special node, typically called a cluster-head, which acts as the center for a cluster, with intra-cluster packet forwarding being facilitated by nodes on cluster boundaries.

Clusterhead Gateway Switch Routing (CGSR) is an early example of a cluster-based routing protocol for MANETs (Chiang et al., 1997). CGSR aggregates mobile nodes into clusters through the election of cluster-head nodes; nodes within transmission range of cluster-heads are a part of its cluster. Cluster-heads are elected using the least-clusterhead change (LCC) algorithm, designed to minimize cluster-head changes in order to maintain stability.

The formation of clusters simplifies routing when compared to flat-network protocols. Nodes within transmission range of two or more cluster-heads act as gateway nodes, able to forward packets between clusters. Routing information mapping node addresses to clusters is

distributed using DSDV as the underlying protocol. Nodes attempting to send packets first transmit them to their cluster-head. Packets sent to nodes within the same cluster are immediately forwarded to the destination by the cluster-head. Packets sent to nodes outside the cluster are directed to the first gateway node on the shortest route to the target cluster. Packets are then forwarded from gateway to cluster-head to gateway until they reach the target cluster, where they are then forwarded to the destination by the local cluster-head. This forwarding pattern is illustrated in figure 2.9.

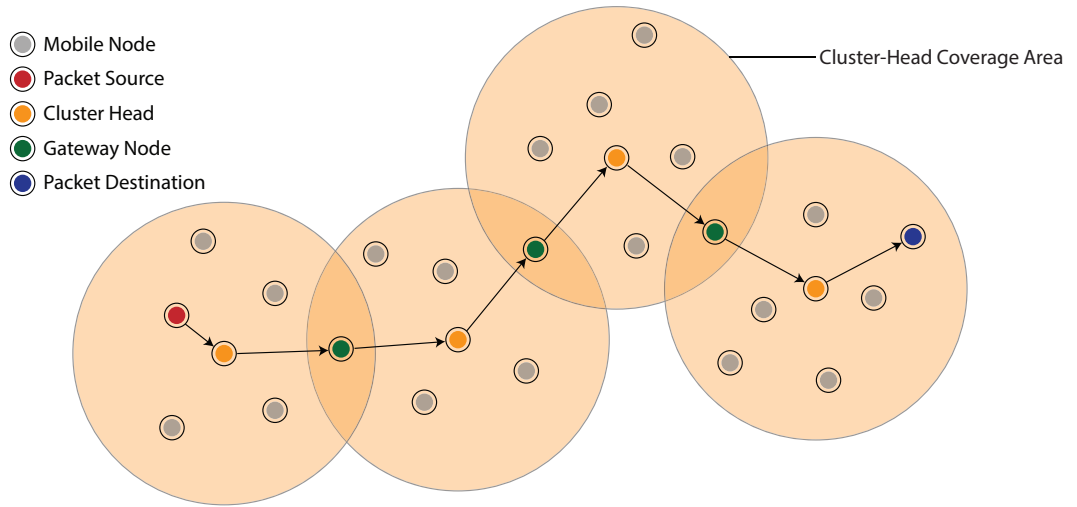


Fig. 2.9: Packets destined for nodes in non-local clusters follow a path along intermediate cluster-heads and gateways.

The pattern followed by CGSR is typical for hierarchical routing protocols. Alternative approaches differ in terms of clustering algorithms used, with a number of protocols focusing on cluster stability over route length minimization (Chiang et al., 1997). Other approaches attempt to find a minimum dominating set (DS); the minimum number of nodes required for route search and table maintenance (Wu et al., 2001; Chen and Liestman, 2002). Members of the DS set are selected as cluster-heads, forming an optimal cluster distribution within the network. Such approaches perform well given minimum changes to the network topology over time, however in highly mobile networks the computational and messaging overhead generated by DS recalculation becomes significant (Yu and Chong, 2005).

Cluster size also varies among existing implementations, with some attempting to load-balance clusters by restricting the number of nodes in a cluster to a specified range (Amis and Prakash, 2000). Extending cluster size beyond single-hop distances has also been explored in

order to normalize cluster sizes and limit the computational and communication overheads for cluster-heads (Ohta et al., 2003).

While cluster-based routing has benefits in terms of scalability and routing simplification, maintenance of coherent cluster membership typically comes at a cost in terms of additional overheads required for coordination. These costs are often increased when clusters must be reformed around a new geographical location or node as a result of mobility; a process which can temporarily disrupt routing activity. This process can cascade throughout the network, generating significant overheads while clusters are reformed.

2.1.3.2 Hybrid Routing

Hybrid protocols such as those developed by Haas (1997) combine aspects of both reactive and proactive protocols. The area within a specified number of hops of each node is designated as a proactive coverage zone within which route advertisements are disseminated, with nodes outside of this zone being discoverable through a reactive query-flood approach.

This scheme limits the negative effects of periodic advertisements to a smaller area than that used in fully proactive protocols while still allowing for route discovery outside of the proactive zone through the use of reactive querying.

In the case of the Zone Routing Protocol (ZRP), the query mechanism is enhanced using the knowledge of the local topology gathered by the intra-zone routing protocol (Haas, 1997). Queries are sent to zone border nodes rather than flooded using a naive approach. This approach, called border-casting, reduces redundant rebroadcasts typically seen in naive flooding by leveraging topology information gained by the intra-zone routing protocol.

Border nodes check their intra-zone routing tables for the requested destination node and forward the request to their own border nodes if no route is available. This process continues until the request reaches a border node with an intra-zone route to the destination or the request's time-to-live value is exceeded. If the destination is found, request replies are then returned to the source over the same paths used during the request phase. Border-casting is illustrated visually in figure 2.10.

The performance advantage of hybrid protocols such as ZRP depends on the selection of an optimal zone radius r_{zone} - the number of hops within which a proactive routing approach is used. For smaller values of r_{zone} , the performance of ZRP approaches that of a reactive routing protocol, resulting in an increase in inter-zone query traffic. Larger values of r_{zone} cause ZRP to perform similarly to a fully proactive protocol; with increased overheads due to greater coverage by periodic advertisement messages (Pearlman and Haas, 1999).

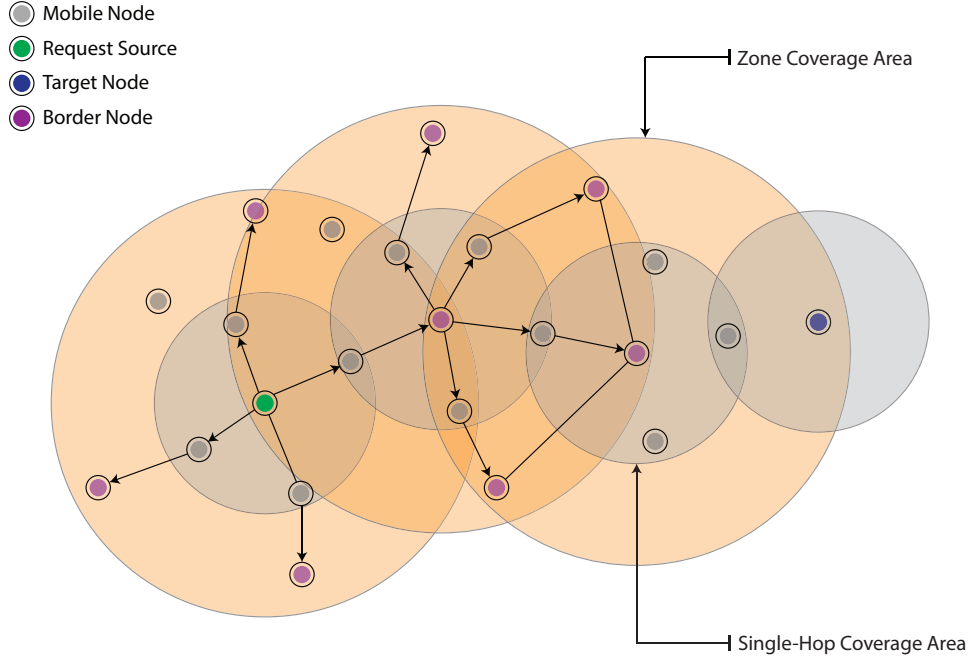


Fig. 2.10: ZRP Border-Casting leverages intra-zone topology information to disseminate route requests to neighbouring zones efficiently. This example shows ZRP nodes operating with $r_{zone} = 2$ hops.

Independent Zone Routing (IZR) (Samar et al., 2004), an enhancement of ZRP, attempts to dynamically adapt the zone radius for each node such that the ratio of reactive and proactive traffic volume remains at an optimal level. IZR has been shown to dramatically reduce control overheads when compared with standard ZRP configured with a static zone radius. This approach is similar to that of Ohta et al. (2003) in which cluster sizes are adjusted in response to topology changes.

2.1.3.3 Probabilistic Routing

The address-based routing protocols outlined in previous sections attempt to facilitate route discovery with a high degree of reliability while reducing overheads and the effect of broadcast storms using various mechanisms based on knowledge of network topology, message size reduction or increased redundancy. These approaches may be broadly categorized as deterministic routing approaches.

An alternative approach, known as probabilistic routing, attempts to facilitate route discovery while avoiding the broadcast storm problem by rebroadcasting queries or advertisements based on a probability value which may be calculated in a number of ways.

Haas et al. (2002) have proposed a number of solutions using probabilities and parameters based on simulation. Initial versions of their protocol simply rebroadcast with probabilities in the range 0.6-0.75; determined as resulting in the best tradeoff between coverage and redundant rebroadcasts. Further enhancements include additional parameters that determine the number of hops that a broadcast must traverse before probabilistic rebroadcasting is used, as well as parameters that use different rebroadcast probabilities depending on the local network density of each node. Nodes may also rebroadcast if a certain number of their neighbours are not observed to rebroadcast within some timeout value. The weakness of this approach is that the various parameters are not adapted to actual network conditions but are instead set before deployment time. A network which diverges from the configuration used to determine those parameters will experience decreased performance as nodes fail to receive broadcasts and are thus unable to route packets.

Cartigny and Simplot (2003) propose a solution wherein nodes determine their distance from the source of a broadcast by calculating the intersection between their own neighbour set and the neighbour set of the source. The neighbour set, a list of all nodes within duplex transmission range, is inserted into the broadcast packet. A small degree of intersection indicates that the source and receiver are far away from each other, while a large degree of intersection indicates that they are close. The degree of intersection between neighbour sets is used to determine rebroadcast probability; with smaller intersections resulting in a greater rebroadcast probability. This is illustrated visually in figure 2.11.

This approach works similarly to OLSR's MPR algorithm and ZRP's border-casting, where only the required subset of neighbours required to reach distant nodes rebroadcast messages. However, unlike OLSR and ZRP, the approach proposed by Cartigny and Simplot (2003) does not require periodic broadcasting of full neighbour sets in HELLO messages or the establishment of proactive zones allowing rebroadcasters to be determined directly. Thus, overheads are significantly reduced at the cost of reliability; particularly at greater distances from the source as broadcast coverage becomes compromised by non-optimal rebroadcasts. Decreasing the intersection threshold partially alleviates this problem at the cost of greater overheads, as more neighbours choose to rebroadcast.

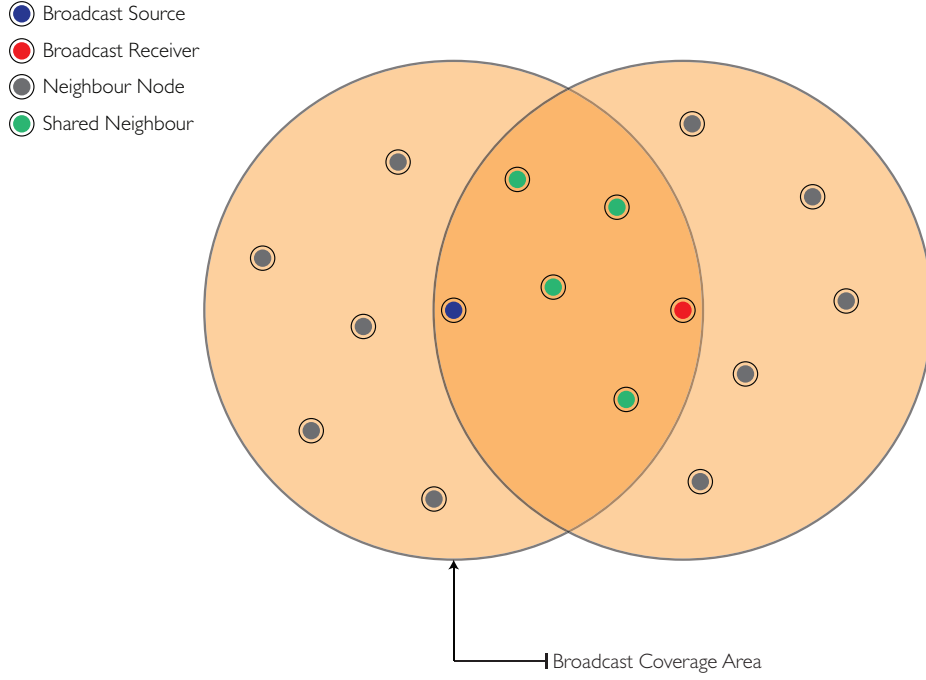


Fig. 2.11: Probabilistic border rebroadcasting is based on the degree of intersection between the source’s neighbour set and that of the receiver. Smaller intersections indicate greater distance between the source and receiver.

2.2 Service Discovery in MANETs

Numerous deployment scenarios envisioned for MANETs assume a lack of knowledge of node identities prior to deployment. In this situation, address-based routing is inadequate for network utilization as nodes are unaware of the resources made available by their peers. Thus the development of resource discovery approaches has emerged as an important area of research in MANETs.

Service discovery has been identified as a crucial area of research in MANETs, with a number of protocols having been developed in recent years. A key assumption made by each implementation is the presence of an underlying address-based routing protocol that can facilitate communication between clients and servers once service discovery has completed. More recently, performance gains have been made by directly integrating service discovery with address-based routing protocols using approaches that piggy-back service advertisement or discovery messages onto route advertisement and request packets. These *cross-layer* protocols tend to improve

performance when compared to application-layer approaches, however they still suffer from the limitations imposed by the routing protocols upon which they are based.

As outlined in previous sections, the various existing approaches to address-based routing impose limitations on network scalability, mobility, reliability and density. This work aims to demonstrate that the use of such protocols is unnecessary in a service-oriented context and can be detrimental to performance.

2.2.1 Application-Layer vs Cross-Layer

Current approaches to service discovery assume the presence of an underlying routing protocol. Early implementations did not assume that any particular routing protocol was installed on the local node, only that the protocol supported the features required such as unicasting or multicasting (Kozat and Tassiulas, 2004; Campo et al., 2006; Helal et al., 2003). Protocols effectively acted like applications; existing at the application layer of the protocol stack and exposing a set of functions that allow other applications to ascertain the address associated with a particular service's server.

This approach mimics the strict layering of network protocols observed in infrastructure networks and results in good flexibility; service discovery protocols can be deployed without consideration for the underlying routing protocol installed on nodes. However, this layered approach is not flexible enough to allow for the sorts of performance optimizations that can be engineered through cross-layer data sharing and cooperation (Conti et al., 2004). An illustration of these differing approaches is shown in figure 2.12.

Allowing service discovery protocols to become aware of link breakages or changes in the network topology through information sharing with the underlying routing results in improved responsiveness and success rates. Similarly, the ability to append service discovery messages to routing protocol packets can reduce bandwidth consumption through reduced message duplication. Energy usage and interference can also be reduced and the discovery protocol is also able to take advantage of any efficient message dissemination algorithms used by the underlying routing protocol.

The MANET research community has shown great interest in this area of research, with many new protocols incorporating cross-layer design having been developed in recent years (Garcia-Macias and Torres, 2005; Li and Lamont, 2005; Flathagen and Øvsthus, 2008). In the following sections we will explore the two main approaches to service discovery; directory-based and peer-to-peer, and will describe protocols in each category that adhere to both application-layer and cross-layer design.

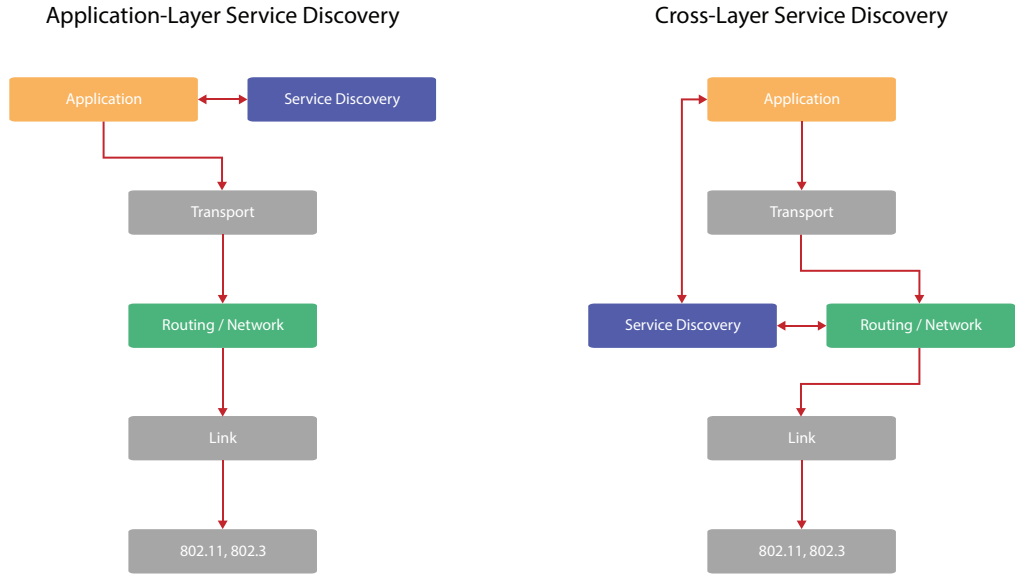


Fig. 2.12: Application-layer approaches do not integrate with the underlying routing protocol, allowing for greater deployment flexibility. Cross-layer integration, however, results in improved performance.

2.3 Directory-Based Service Discovery Protocols

Directory-based service discovery protocols may be defined as service discovery approaches in which a subset of nodes in the network are designated as service directories. Service directories are responsible for the storage of mappings of service identifiers to the addresses of server nodes. Server nodes connect to directories in order to register their services so that future queries for those services by client nodes may be honoured. Client nodes connect to known directories to query for the addresses of nodes that provide services. This pattern is illustrated in figure 2.13.

Work in the area of directory-based discovery progressed from earlier protocols developed for infrastructure networks such as Universal Description, Discovery and Integration (UDDI) (Microsoft, IBM, 2000), JINI (Sun Microsystems, 1998) and Service Location Protocol (SLP) (Veizades et al., 1997). Infrastructure network service discovery protocols are able to take advantage of static network topologies, high bandwidth links and high node reliability. UDDI, for example, relies on a single centralized server to coordinate discovery. Both SLP and JINI allow for multiple directories to be deployed in a given environment, however directories do not coordinate among themselves; it is up to service providers to register their services against multiple directories as

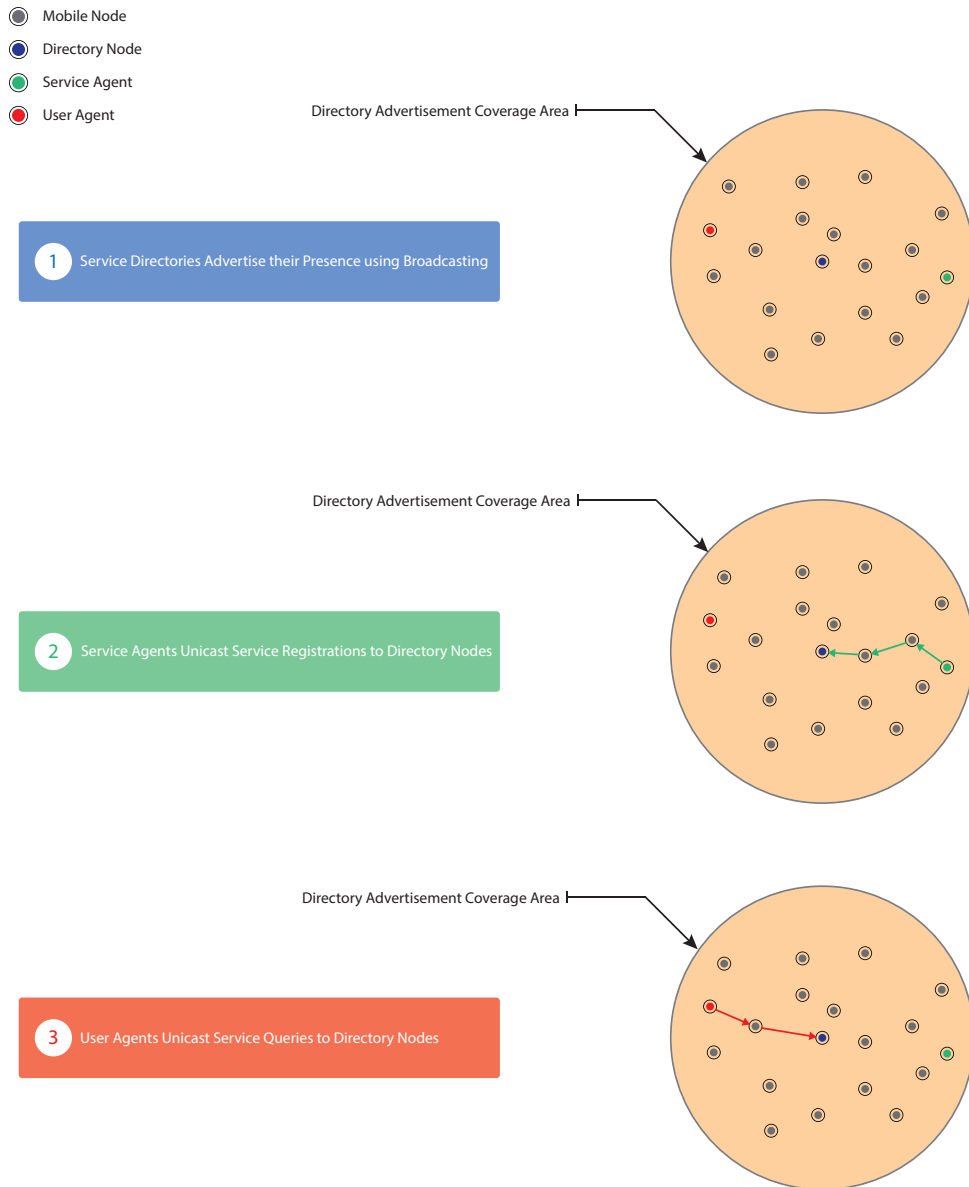


Fig. 2.13: In directory-based architectures, directory nodes act as repositories for service information. Service agents register their services with directories so that user agents can query for the addresses of nodes providing required services.

needed.

These characteristics make such protocols unsuitable for MANETs, where multi-hop communication is costly and low node reliability and unpredictable network topologies make centralized approaches untenable. Directory-based service discovery protocols for MANETs must take these limitations into account by improving redundancy and devising intelligent schemes for directory deployment in order to reduce request path lengths while maintaining low overheads (Ververidis and Polyzos, 2008).

2.3.1 Distributed Service Discovery Protocol: Virtual Backbone

Kozat and Tassiulas (2004) proposed a protocol named Distributed Service Discovery Protocol (DSDP) in which a virtual backbone, or dominating set, of *service brokers* is formed within the network, such that all nodes are either part of the backbone or at most one hop away from a node that is part of the backbone. A service broker as defined by DSDP is the same as a service directory. Service brokers may be separated by at most one or two *virtual access points* (VAPs) that act as gateways to service brokers for nodes which are outside of immediate broadcast range of a broker. The virtual backbone is formed and maintained through the use of beacons or HELLO messages.

Server nodes register their services with their nearest broker, possibly through a VAP depending on the network topology. DSDP also provisions for multiple registrations against more distant brokers, however it assumes the presence of an underlying broadcast or multicast routing protocol to facilitate this. Server nodes can also act as their own broker if they are already part of the virtual backbone. User agents request services by forwarding requests to their closest service broker. If the service request cannot be satisfied by the local broker, the broker can then multicast or broadcast requests to other service brokers in the network, depending on the underlying routing protocol.

DSDP attempts to assist in this process by designating brokers their VAPS and nearby nodes as single multicast destinations, effectively forming clusters in a manner similar to the protocols described in section 2.1.3.1. Requests are routed to neighbouring clusters and proliferated using a source routing approach until a cluster containing the required service is found, at which point a reply is unicasted to the source cluster. This clustering approach is also used to facilitate service registrations against more distant brokers.

DSDP's clustering algorithm is robust and responds well to changes in the network topology, with clusters able to reform without producing cascading changes in the rest of the network. However DSDP makes the assumption that some underlying routing protocol capable of supporting multicasting or broadcasting is available. Without such a protocol, DSDP only supports

querying for services within the local cluster, greatly reducing its usability in large-scale networks. In addition, DSDP does not attempt to integrate its messaging with that of the underlying protocol; resulting in greater control overheads than might otherwise be needed, particularly in the case of a routing protocol that employs its own HELLO message broadcasts.

2.3.2 Lightweight Service Discovery: Peer-to-Peer Fallback

The absence of available directory nodes results in the inability for user agents to discover server addresses and subsequently utilize available services. Approaches such as those of Kozat and Tassiulas and Sailhan and Issarny elect directories dynamically based on changes in the network topology in order to overcome this limitation. An alternative approach is used by the Lightweight Service Discovery (LSD) protocol (Li and Lamont, 2005), a directory-based approach which allows user agents and service agents to fall-back on a peer-to-peer discovery mechanism in the absence of directory nodes.

The advantage in reliability conferred by this feature is further improved by direct cross-layer integration with the underlying protocol, in this case the optimized link-state routing (OLSR) protocol. OLSR is described in detail in section 2.1.1. LSD takes advantage of OLSR's multi-point relay (MPR) algorithm and message extension features to efficiently disseminate directory advertisements, service queries and service advertisements throughout the network in Service Location Extension (SLE) messages. SLE's act as extensions to OLSR's topology control (TC) messages, allowing service discovery and routing information to share the same packet space; thus reducing message duplication and bandwidth consumption.

LSD does not include a dynamic directory election process, instead assuming that directories are designated manually or using some external method. Directory nodes broadcast advertisements every $t_{advertise}$ seconds using OLSR's MPR algorithm. User agents and service agents overhearing these advertisements cache the directory's address. Service agents immediately unicast a list of their services to the directory, allowing user agents to query for those services.

In the absence of directory nodes, indicated by a lack of advertisements, user agents will broadcast a query for available directories when a service request is made. If no response is received within a configured time limit, user agents will then broadcast a direct query for service agents providing the required service. Service agents providing the required service then unicast a positive response, allowing the user agent to unicast traffic to the service agent. Similarly, if service agents do not detect directory advertisements and have not cached recent directory addresses, they may begin to advertise their services directly. User agents receiving these advertisements then cache the address of the server, allowing them to utilize the advertised services

immediately upon request without resorting to querying.

LSD has a number of advantages over DSDP, particularly in terms of required overheads for service discovery. Unlike DSDP, LSD does not require the formation of a virtual backbone of directory servers and does not suffer from the increased control overheads that backbone formation generates, particularly in highly mutable topologies. In addition, LSD's direct integration with OLSR allows it to take advantage of an optimized broadcast dissemination mechanism and reduce packet duplication; resulting in decreased overheads when compared to the more separated approach used by DSDV. This comes at the cost of flexibility, however, as LSD requires direct integration with OLSR and thus cannot be deployed on MANETs utilizing a different routing protocol. LSD also suffers from the lack of scalability inherent to the OLSR protocol. OLSR makes routes to all nodes available, regardless of their role in the network. Topology control messages are disseminated by nodes regardless of their participation in client-server communication, reducing usability for real traffic by causing medium contention and wireless interference.

2.3.3 Service Rings: Semantic Hierarchical Discovery Overlay

Service Rings (Klein et al., 2003) attempts to facilitate efficient service discovery by forming a structured overlay. Nodes that are physically close to each other and share similar services are grouped together into virtual rings. Nodes within a ring have a designated predecessor and successor node with which they attempt to maintain contact, with the first node in a ring connected to both its own successor and the last node, thus forming the circular ring structure. Successor/predecessor connections may span multiple hops, however nodes joining a ring will attempt to choose successors and predecessors that are as physically close to them as possible in order to reduce ring maintenance overheads.

The first node in a ring is designated as a Service Access Point (SAP) and is responsible for recording the services offered by all nodes in the local ring. Nodes aiming to register their services send a *Service Offer* message to their successor, which then forwards it to its own successor and so on until the SAP is reached. As illustrated in figure 2.14, higher level rings may be formed by the SAPs of lower-level rings. This feature facilitates service discovery by collecting service information in a hierarchical fashion, with the higher-level ring SAPs being responsible for recording the services available in all of the rings below their own.

Service requests made by level 0 ring members are sent from successor to successor until an SAP is reached. If the SAP is unable to satisfy the request, it is then forwarded to the SAP for the next level ring. This algorithm is repeated until a SAP that stored a service descriptor matching the required service is contacted. The request then descends through the service rings through successive SAPs until the service provider is reached. A response is then sent to the requesting

user agent. If a service requests reaches the top of the ring hierarchy without encountering a SAP with a matching service record, or exceeds the ring step limit specified in the request, a failure response is sent to the user agent.

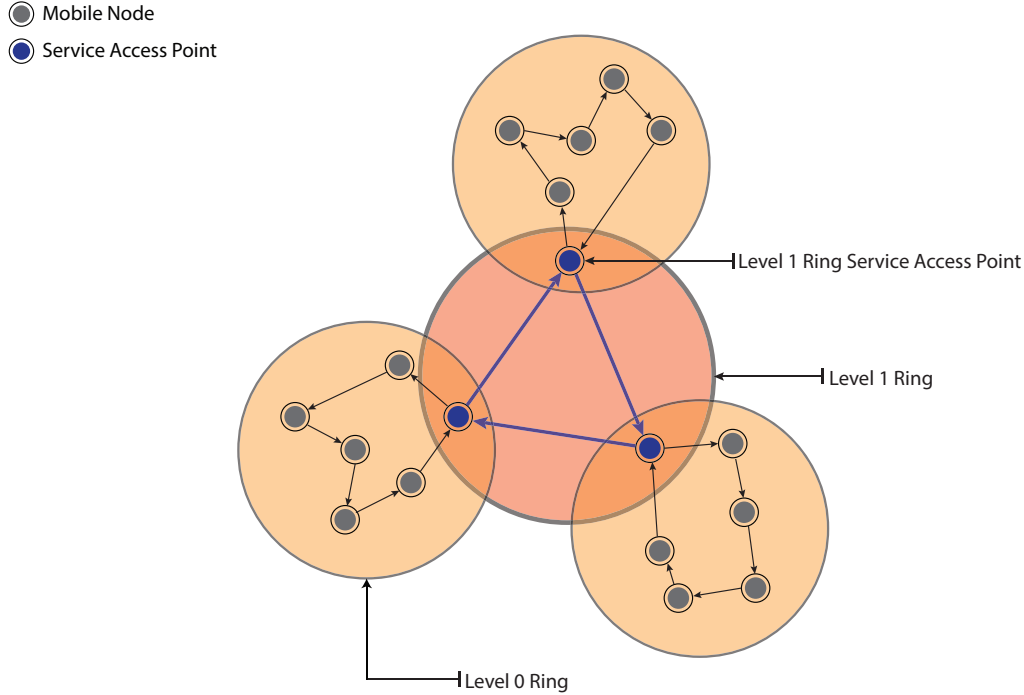


Fig. 2.14: Service Rings are formed in order to facilitate efficient service discovery. Service Access Points act as directories for their local ring and can respond to service queries from user agents.

Service rings demonstrates a number of advantages when compared with approaches such as LSD and DSDP; its structured ring approach allows services to be registered and located entirely through unicasting between nodes and their successors rather than through multicasting or broadcast flooding. This reduces the impact of service discovery on the network in terms of induced control overheads. The ring hierarchy also demonstrates an efficient method for collecting service descriptor summaries at key nodes in order to facilitate non-local discovery.

While service rings does make some attempt to maintain the locality of rings in order to reduce the impact of multi-hop predecessor-successor link maintenance, network topologies that display high mobility and variability are likely to cause links between successive overlay nodes to lengthen and break. In addition, the mechanisms for splitting rings that become too large and those used to adapt to ring breakages induce additional overheads.

Service rings is optimized for scenarios where there is a correlation between the services provided by nodes and their geographic location, as this results in a structured overlay in which requests may be routed towards rings containing semantically similar service summaries. However, this assumption may not hold true in many deployment scenarios, resulting in a less efficient overlay in which searches propagate over a larger area; making the protocol unsuitable outside of specialized deployment scenarios.

Finally, like the protocols mentioned above, service rings makes the implicit assumption that a routing protocol capable of allowing overlay nodes to maintain their virtual links is available. This results in added costs in terms of bandwidth usage and overheads on top of those required for overlay formation; reducing network usability for client-server traffic.

2.3.4 SANDMAN: Cluster-Based Energy Efficient Discovery

Schiele et al. (2004) describe an approach called SANDMAN based on clustering in which service agents register their services with the cluster head. Cluster heads thus act as directory nodes. One novel aspect of this approach is its focus on energy efficiency; all designated cluster heads are prohibited from entering standby mode while cluster members enter standby mode periodically when idle. Service agents register their standby time with the cluster head, allowing user agents to become aware of the times at which services available. User agents can then enter standby mode until the service agent wakes, saving energy.

SANDMAN's approach has disadvantages in terms of service availability; while a network may contain service agents that provide a required service, user agents may have to wait before requests can be sent due as those service agents may be in standby mode. Schiele et al. propose that clustering take place at the application layer, independent of the underlying routing protocol, however their evaluations make use of statically-defined clusters with immobile nodes. The underlying routing protocol used is also unspecified. An approach that integrates with an existing cluster-based routing protocol such as CGSR in a cross-layer design would allow clusters to adapt to network conditions.

2.3.5 Summary and Analysis

The protocols described in this section outline the main approaches to directory-based service discovery for MANETs; virtual backbone formation, manual directory deployment with peer-to-peer fallback capability, overlay formation and clustering. All rely to some degree on address-based routing in order to facilitate discovery.

In the case of DSDP, a routing protocol is required for service discovery outside of the small range covered by service brokers and their immediate virtual access points. LSD directly integrates with OLSR using the cross-layer design pattern, allowing service advertisement and query messages to piggy-back on OLSR TC messages and take advantage of its efficient MPR algorithm for message dissemination.

Service rings requires an address-based protocol for ring maintenance, hierarchical ring formation and all discovery activities. SANDMAN requires a routing protocol for all multi-hop communications, however its evaluation makes use of statically-defined routes between clusters. LSD's approach is the most efficient, as it does not require additional overheads for overlay formation and can optimize broadcasts using OLSR's MPR algorithm.

However, LSD's reliance on pre-assigned directories rather than the dynamic assignment methods used by the other protocols means that in certain network configurations it will be forced to resort to peer-to-peer discovery. As LSD's peer-to-peer fallback strategy involves the dissemination of both advertisements and queries, this can result in a comparative increase in control overheads; particularly with increasing numbers of service and user agents, as well as increasing node density and network scale.

Overlay-based approaches such as those employed by DSDP, Service Rings and, to an extent, SANDMAN incur a cost in terms of overheads during overlay formation and maintenance. This cost is further increased in the presence of high mobility when the overlay must adapt to topology changes. Initial and maintenance costs also tend to increase with increasing node density and network scale as more nodes become integrated into the overlay.

Protocol	Design	Directory Designation	Control Overheads	Advantages	Disadvantages
DSDP	Application-Layer, Overlay-Based	Automatic	High	Avoids Cascading Overlay Changes	Requires Multicast-Capable Routing Protocol.
LSD	Cross-Layer with OLSR, P2P Fallback	Manual	Low	Falls Back to P2P Discovery in Absence of Directory Nodes	Manual Directory Designation Results in Higher Overheads
Service Rings	Application-Layer, Overlay-Based	Automatic	High	Structured Discovery using Ring Hierarchy	Overlay Formation and Maintenance Costs
SANDMAN	Application-Layer, Overlay-Based	Automatic	High	High Energy Efficiency	Discovery Delay due to Node Standby Mechanism

Fig. 2.15: Directory-Based Discovery Protocol Comparison

The features, advantages and disadvantages of the reviewed directory-based protocols are summarised in table 2.15.

2.4 Peer-to-Peer Service Discovery Protocols

Peer-to-peer (P2P) service discovery protocols operate through direct service advertisement by service agents or direct querying for services by user agents. Such approaches typically avoid the formation of complex overlays, instead focusing on efficient mechanisms for direct query and advertisement dissemination. Work in the area of P2P discovery emerged from the decentralized discovery approaches developed for infrastructure networks such as UPnP (Microsoft, 2008) and Apple’s Bonjour protocols (Apple Inc., 2002). P2P discovery approaches are well-suited to MANETs, where frequent topology changes can disrupt overlays or partition networks, isolating agents from directories.

The area of P2P discovery shares many challenges with the area of routing for MANETs. In both areas, work has focused on allowing control messages to be efficiently disseminated without disrupting network traffic while still providing maximum availability. These shared aims have resulted in much work in the area of cross-layer P2P service discovery protocols, with a number of such approaches having emerged in recent years (Flathagen and Øvsthus, 2008; Zhou et al., 2011).

In this section we will review a number of such protocols, focusing on both cross-layer and application-layer approaches.

2.4.1 Mercury: Cross-Layer Service Advertisement

Mercury (Flathagen and Øvsthus, 2008) is a cross-layer protocol that facilitates service discovery through periodic service advertisement. Service agents append service advertisements to OLSR packets using OLSR’s extensible messaging format and disseminate them through the network using OLSR’s MPR algorithm, allowing advertisements to reach distant nodes without the use of naive flooding. Mercury attempts to reduce the control overheads associated with service advertisements by utilizing bloom filters (Bloom, 1970); a space-efficient probabilistic data structure that supports queries against its contents.

A Bloom Filter is represented as an array of m bits, initially set to 0. Set elements are inserted into a filter by hashing them using a set of k hashing functions to produce k offsets into the bit array. Bits at the resulting offsets are then set to 1.

Queries against the filter are performed by hashing an element using the same hash functions and checking whether the resulting offsets are set to 1. If any of the bit offsets are set to 0, the element is definitely not contained in the filter. If all of the bit offsets are set to 1, either the element is in the set or the offsets were previously set by one or more past insertions; resulting in a false-positive. As more elements are inserted into the filter, the probability of a false-positive

increases. This probability is reduced with increasing filter size and adjustment of the number of hash functions used to compute bit offsets. Thus a trade-off must be made between query reliability and filter size. Bloom filters are an integral part of our service discovery approach, SDRP, and are elaborated upon in Appendix A.

8	16	24	32
Packet Length		Reserved	
Message Type	Valid Time	Message Size	
Originator Address			
Time to Live	Hop Count	Sequence Number	
Type	Filter Length	Reserved	
Service Filter			
...			

Fig. 2.16: Mercury service advertisement message. OLSR header fields are highlighted in grey. The highlighted message type field is set to a constant indicating that the message payload is an extension. Blue fields are part of the mercury advertisement message.

Mercury uses bloom filters as a space-efficient structure for representing sets of available services. User agents insert *service filters* containing available services into advertisement messages. The mercury message format, including the basic OLSR message header, is illustrated in figure 2.16.

Flathagen and Øvsthus specify a filter size of 128 bits; this represents a compromise between packet size and filter query reliability. The MD5 hash function is used to produce bit offsets for insertion and query operations.

Mercury does not attempt to vary service filter size according to the number of services made available by the advertising service agent. This may result in false-positive queries when service agents serve a large number of services or space wastage when there are few services. Mercury assumes that services are described in string form, however any data format that can be hashed is also suitable.

Mercury service agents flood service advertisements throughout the network periodically and in response to received service requests. Nodes receiving service advertisements cache the originator address and the contained service filter. In order to reduce false-positives resulting from the use of cache entries for service agents which are no longer available, or for services which are no longer available on a particular service agent node, entries are cached for a time period specified as t_{cache} , after which they are deleted. Mercury also consults the OLSR routing table and deletes cache entries for service agents which are no longer reachable.

User agents discover services by consulting their service filter caches. The descriptor for the required service is used to query the cached service filters. Mercury assumes that a positive query result indicates the availability of a service, user agents then attempt to route requests to the originator address associated with the queried filter using OLSR.

Mercury has been shown to perform well when compared to two application-layer approaches, namely PDP (Campo et al., 2006) and a version of SLP (Veizades et al., 1997) adapted to MANETs. Mercury’s use of bloom filters for service description can be expected to result in reduced control overheads in situations involving service-dense service agents, especially when compared to protocols where service descriptors are verbose and are serialized into packets without compression. However, Mercury does not dynamically adapt bloom filter sizes to their content, which may result in increased false-positive rates or wasted packet space depending on the number of service descriptors to be contained.

Mercury’s cross-layer design presents advantages in terms of control overhead reduction, as it allows service advertisements to be disseminated more efficiently through the use of the OLSR MPR protocol. However, the use of OLSR results in the dissemination of topology control messages by all MPR nodes, regardless of whether those nodes act as service agents. This makes little sense in networks where node identities and characteristics are unknown prior to deployment and results in greater interference and control overheads than may be otherwise necessary.

2.4.2 SMF: Cross-Layer Service Querying

Service Magnetic Field (SMF) is a recently developed cross-layer discovery approach integrated with the AODV reactive routing protocol (Zhou et al., 2011). Unlike Mercury, which has service agents append advertisements to protocol messages in order to inform user agents of available services, SMF user agents append service queries to flooded AODV route request messages. This presents advantages in situations where the number of service requests is expected to be low, as it avoids the unnecessary overheads resulting from periodic advertisement messages associated with proactive approaches. Such behaviour is also desirable in deployment scenarios involving large numbers of servers and small numbers of user agents, as the overheads associated with queries would be expected to be lower than those of periodic advertisements.

The novelty of SMF’s approach lies in its service selection method, rather than its cross-layer query mechanism. SMF defines a quantity, *Service Magnetic Field Strength* (SMFS) based on the distance from a user agent to a service agent. As the distance in hops from a user agent to a service agent increases, the SMFS decreases by a quantity γ , which is proportional to the overall network diameter. γ is defined prior to deployment and is not updated to account for changes in the network’s topology. The initial SMFS at 0 hops from the service agent is 1. The

definition of γ allows the maximum distance for client-server communications to be defined as needed. Small values of γ allow for a larger hop distance between client and server which may result in decreased performance, particularly in dense networks where interference has a greater effect. SMF assumes that services are described using variable-length strings; this can have an adverse effect on performance when service descriptors are long or numerous and is inefficient when compared with Mercury’s more compact approach.

SMF requires that AODV be configured to send periodic HELLO packets. Both local and query result service information is added to HELLO packets so that nodes can build tables describing the service available in the network and their SMFS. Nodes receiving HELLO packets copy new service entries into their route tables and decrement associated SMFS values by γ .

HELLO packets are also used to determine the link state of intermediate hops along routes between clients and servers; if HELLO packets are not received after a pre-determined period of time, the link with the originating node is assumed to be lost and a new route may have to be discovered. SMFS table entries and route tables containing entries for the originator are also deleted.

AODV’s route request packet type is extended to include the descriptor for the service being sought. Route requests are disseminated through the network using AODV’s naive flooding approach. When a node containing an SMFS table entry for the sought service receives the route request, it sends an extended route reply packet to the originating node containing the network address of the service agent and the service’s associated SMFS. Querying nodes receiving multiple route reply packets choose the route to the service agent with the highest associated SMFS in order to maximize route reliability and throughput. Intermediate nodes receiving route reply packets add information about the contained services to their own SMFS tables.

SMF was compared with a custom-written application-layer service discovery approach based on query broadcasting and was shown to perform well in terms of discovery delay, success rate and control overheads. While SMF’s approach to service selection does have advantages in terms of its ability to account for network diameter, the fact that γ is not updated during runtime limits its usefulness in highly dynamic networks and may, in fact, result in lower discovery success rates as service agents are needlessly avoided due to low SMFS; despite network conditions that would otherwise make them viable. SMF’s use of strings as service descriptors introduces the potential for high overheads due to the increase in packet sizes, particularly when large numbers of services are present in the network. HELLO packets would be expected to contribute significantly to these overheads, as they contain full dumps of local SMFS tables.

2.4.3 Pervasive Discovery Protocol: Application-Layer Discovery

The Pervasive Discovery Protocol (PDP) is an application-layer approach that utilizes a mixture of service queries and advertisements to facilitate discovery (Campo et al., 2006). PDP is not designed specifically for MANETs, but rather for pervasive computing devices with wireless capability. Recognizing the disadvantages of approaches based exclusively on query and advertisement mechanisms, Campo et al. propose a protocol in which service agents do not initially advertise their services, but rather begin only in response to interest from user agents. Once a service agent has received a query for a service from a user agent, it begins advertising that service using local broadcasts, ensuring that only those services that are in demand consume network resources.

PDP requires the installation of a multicast-capable routing protocol. All nodes participating in the discovery process must be part of a multicast group in order to facilitate query and response functionality. PDP user agents multicast service requests to their multicast group, service agents also multicast their responses. This approach may be inefficient, however, when compared to a query broadcast and response unicast approach, as responses are received by all group members regardless of their interest in the contained service. This allows receiving nodes to update their service tables, but also consumes additional bandwidth.

PDP attempts to allow for energy scarcity by having services and their agents advertise an availability time, indicating the time for which a node or service will be available; corresponding to the time until the node enters standby, runs out of power, or a service is disabled. This metric must be taken into account by user agents when evaluating advertisements and query responses. PDP also allows user agents to discover all service agents possessing a specific service using its one-query-multiple-response feature. In this query mode, PDP user agents send service queries in response to application requests and wait a pre-determined length of time before responding to the application. All query replies received are collected into the application response. In one-query-one-response mode, the first reply received is passed to the application.

PDP service agents take their availability time into account when responding to service requests; service agents delay their responses to one-query-one-response requests by an amount inversely proportional to their availability time. This ensures that devices with higher availability times are the first to respond to requests, reducing the load on those devices with a lower availability.

PDP has advantages when compared to previously detailed approaches, particularly in terms of energy efficiency and adaptation to node availability. However, PDP's requirement of a multicast-capable routing protocol, its use of multicasting for all queries and responses and the delay induced by its availability metric during responses can all be expected to lower its overall

performance in terms of throughput, success rate and control overheads. Lack of integration with the underlying routing protocol can also be expected to reduce its performance, particularly when compared to the cross-layer approaches detailed in previous sections.

2.4.4 Other Approaches

Ververidis and Polyzos (2009) have proposed a cross-layer discovery protocol integrated with ZRP called E-ZRP. ZRP's HELLO, IERP and IARP messages are extended to include service identifiers expressed as Unique Universal Identifiers (UUIDs); thus nodes must have a-priori knowledge of the mappings between services and UUIDs. The inclusion of service UUIDs in IARP messages allows a node's local zone to become aware of its offered services proactively while their inclusion in IERP queries means that services that are unavailable in the local zone can be queried for in remote zones using ZRP's efficient inter-zone query algorithm.

E-ZRP has been shown to perform well when compared to a simple algorithm based on naive advertisement flooding, with comparable service discovery success rates, lower overheads and improved energy efficiency.

KONARK (Helal et al., 2003) is an early application-layer discovery protocol with both query and advertisement capability, similar to PDP. Like PDP, KONARK assumes the presence of a multicasting-capable routing protocol and assumes that all nodes participating in the discovery process are part of the multicast group. KONARK service agents may be configured to advertise their service periodically to their multicast group or in response to some external event. User agents also multicast their service queries to the group, with service agents responding if a local service matches the one requested.

KONARK's main contribution is its tree-based service description approach. Services are distributed in a hierarchy, with higher-level nodes denoting categories and leaves representing specific services. This allows KONARK service agents to easily match requests against local services and advertise services in groups if required. XML is used to describe service trees in packets and in the local registry.

KONARK's service tree description approach is elegant and allows for efficient matching in the presence of many services. However, storage and transmission of XML is costly when compared to approaches based on binary serialization and can be expected to increase control overheads. KONARK service agents run local HTTP servers in order to provide services, thus services must communicate using HTTP only rather than a variety of protocols. This restricts the usefulness of KONARK when compared with approaches that do not tightly couple with the communication protocol used by local services. Finally, like PDP, KONARK's lack of integration with the underlying routing protocol and its requirement for multicasting capability reduces both

its performance and flexibility when compared to cross-layer approaches or application-layer approaches without that requirement.

2.4.5 Analysis

Among the described approaches, Mercury, E-ZRP and SMF are the best-performing protocols due to their cross-layer design. Tight integration with the messaging mechanism of underlying routing protocols allows for the optimization of query and advertisement dissemination; reducing overheads when compared to a two-tier application-layer approach.

PDP and KONARK, while more inefficient than the above approaches in terms of generated control overheads and overall performance, have advantages of their own. PDP's energy efficiency makes it more suitable for deployment in scenarios involving nodes with low energy capacity. KONARK's service description mechanism is well-suited to services that are easily arranged categorically and allows for efficient service matching and more detailed description. Both approaches, however, are less flexible than application-layer protocols that do not require multicasting capability.

Protocol	Design	Control Overheads	Advantages	Disadvantages
Mercury	Cross-Layer (OLSR), Advertisement	Low	Compact Service Description with Bloom Filters	Service Filter Size is not Dynamic, May Lead to False Positives
SMF	Cross-Layer (AODV), Query	Low	Adaptable Selection Metric through SMFS	γ is not Dynamic, Must be Set at Deployment Time
PDP	Application-Layer (MultiCast), Mixed	High	Energy Efficiency	Requires Multicast-Capable Routing Protocol.
E-ZRP	Cross-Layer (ZRP), Mixed	Low	Efficient Discovery using ZRP's IARP and IERP Routing	Performance Dependent on Zone Radius Configuration
KONARK	Application-Layer (MultiCast), Mixed	High	Tree-Based Service Description Approach	Requires Multicast-Capable Routing Protocol.

Fig. 2.17: Peer-to-Peer Discovery Protocol Comparison

The protocols described in this section are quite different from the directory-based protocols described in previous sections. The formation of structured overlays is less popular for peer-to-peer architectures, most likely due to the limitations of their two-role approach. This lack of reliance on third-party nodes allows peer-to-peer protocols to better adapt to rapidly changing topologies. Similarly, the costs for overlay-based protocols in terms of overlay maintenance overheads can be avoided in peer-to-peer architectures. However, networks with stable topologies can be expected to benefit from the more structured discovery capability afforded by the presence of an overlay.

An outline of the described protocols is presented in tabular form in figure 2.17.

2.5 Summary

In this section we described current approaches to both address-based routing and service discovery in MANETs. As we have seen, both areas are tightly-coupled, with all of the described discovery protocols relying on an underlying address-based routing protocol to some extent.

Gains in performance and efficiency have been made in recent years with the development of cross-layer discovery protocols; allowing routing and discovery messages to be combined in order to reduce control overheads and improve reliability. However, even this design exhibits inefficiency due to its continued reliance on an underlying routing protocol.

Protocols that are based on proactive routing protocols, such as LSD and Mercury, are able to efficiently disseminate their service advertisements throughout the network using OLSR's MPR algorithm. However, those advertisement must contend with normal routing advertisements for transmission time in the medium, regardless of whether the originator is a service agent or plays any role in client-server communications.

Protocols based on reactive routing approaches such as SMF and AODV-SD (Garcia-Macias and Torres, 2005) do not share this problem, however they cannot benefit from the more efficient message dissemination mechanisms offered by protocols such as OLSR, instead requiring that service queries be flooded throughout the network. This approach can be expected to perform poorly as user agents request services with increasing frequency or in deployment scenarios in which user agents outnumber service agents; a common distribution in infrastructure networks.

The protocols reviewed in this section, along with the links between the various cross-layer approaches, are detailed in figure 2.18.

Our approach, SDRP, attempts to solve the problems presented by reliance on address-based routing. SDRP service agents advertise their services proactively using an efficient dissemination mechanism similar to that of OLSR. However unlike, Mercury, LSD and similar approaches, in SDRP only service agents disseminate advertisements; significantly reducing control overheads and improving throughput for client-server communications. In the next section, we will describe SDRP in detail.

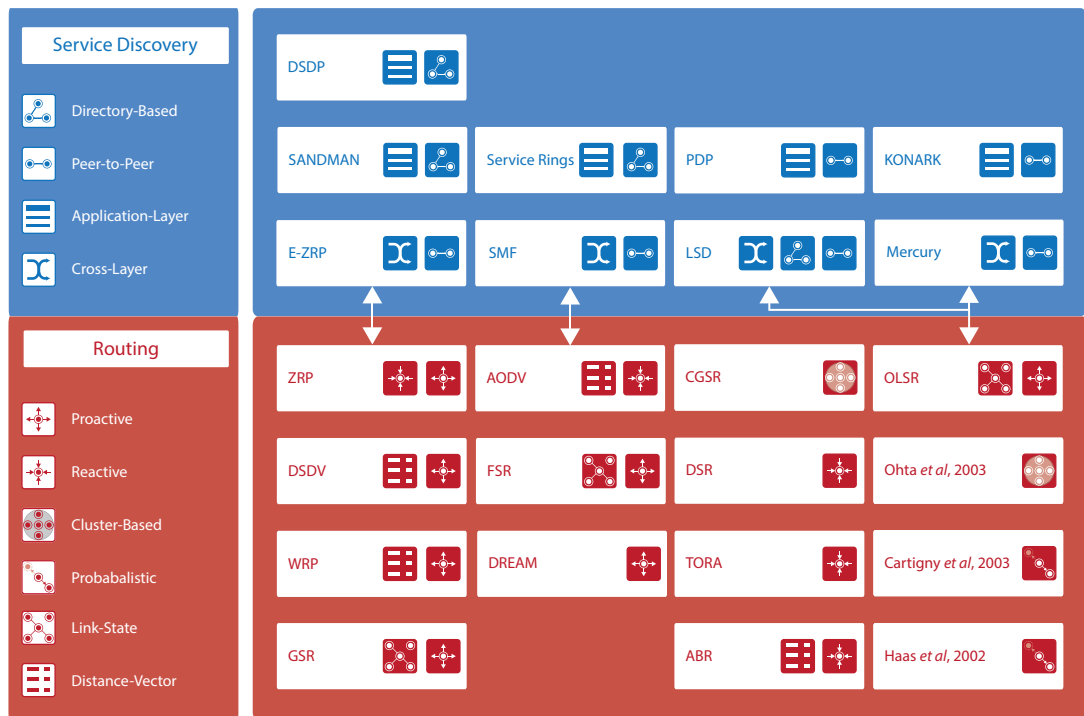


Fig. 2.18: Summary of reviewed protocols. All of the reviewed discovery approaches rely on an underlying routing protocol, with cross-layer approaches having tight integration with specific protocols as shown.

Chapter 3

SDRP: Service Discovery and Routing Protocol

The recent and rapid rise in ownership of smart connected devices represents a challenging new deployment scenario for MANETs; one in which multi-hop routing must be facilitated without significantly impacting usability and performance, even at network densities beyond those traditionally studied. Our approach, the Service Discovery and Routing Protocol (SDRP), is designed for use in these challenging new deployment scenarios. SDRP facilitates service discovery and routing while generating low control overheads and maintaining the low end-to-end delay associated with proactive routing protocols.

The first section of this chapter describes our rationale for developing SDRP. Section 3.2 describes our approach to service discovery and routing, and the three-stage design of SDRP. Section 3.3 describes the design elements that are common to all three version of SDRP. Section 3.4 describes SDRP-Flooding, the first version of SDRP. Section 3.5 describes SDRP Multi-Point Relay, a version of SDRP designed to further reduce control overheads using methods similar to those seen in proactive routing protocols. Section 3.6 describes SDRP Reduced-MPR, the final version of SDRP. Finally, the chapter is summarized in section 3.7.

3.1 Rationale

Traditional deployment scenarios for MANETs, such as battlefields and disaster zones, typically comprise a small number of nodes or low network densities. In such scenarios, it is reasonable to assume that prior knowledge of the identities and capabilities of the deployed nodes is available.

Thus, routing patterns based on identity, rather than characteristics, content or functionality, are viable.

However, emerging scenarios such as the formation of an ad-hoc network composed of smartphones in an urban setting, are very different from those described above. In such a network, prior knowledge of other nodes' capabilities would not be available, thus rendering node addresses useless except as unique identifiers. In addition, high network densities and scales make the provision of routing for all nodes more costly, as interference from control overheads impacts a greater number of nodes than in low-density networks (Arora et al., 2012; Lee et al., 2003).

Current approaches to communication in MANETs are centered around the provision of routing to and from all nodes, regardless of their role or function. As outlined in Chapter 2, this approach introduces performance costs in the form of high control overheads, end-to-end delay or some combination thereof; costs that must be avoided in these emerging deployment scenarios.

New approaches to communication have emerged that revolve around service discovery. These approaches are well-suited to networks in which prior knowledge of available resources is not available at deployment time. In such scenarios, service discovery approaches allow users to dynamically ascertain what services are available in the network and utilize them. However, a design pattern has emerged in service discovery whereby protocols map service descriptors to network addresses and rely on an underlying address-based routing protocol to facilitate routing.

This two-tier approach introduces unnecessary costs, particularly in scenarios in which only a subset of network nodes act as user agents or service agents. The ability to route between arbitrary non-agent nodes is unnecessary for networks in which there is no prior knowledge of network resources, and the provision of this arbitrary routing information impacts on communication performance and network scalability.

Our approach, the Service Discovery and Routing Protocol (SDRP) provides service discovery without incurring the performance cost associated with address-based routing. We accomplish this by removing provision for routing to and from arbitrary nodes, instead providing routes only between user and service agents. The performance of this approach is further improved through a combination of proactive and probabilistic service advertisement that further decreases control overheads, reducing generated control overheads while maintaining low end-to-end delay for client-server communications.

In the following sections, we will elaborate on the rationale for our approach and describe how SDRP overcomes the performance limitations of existing service discovery protocols in greater detail.

3.2 Approach

Our approach, SDRP, acknowledges the cost of address-based routing and attempts to provide service discovery capability without the use of an underlying routing protocol. SDRP facilitates routing only between user agents and service agents; reducing the cost of service discovery when compared with existing approaches. Our aim during the design of SDRP was two-fold; first, to investigate the viability of a service discovery protocol that does not rely on address-based routing. Secondly, we aimed to develop an extremely lightweight protocol that would minimize the impact on the network in terms of control overheads in order to maximize application throughput and network scalability.

SDRP combines optimizations from existing service discovery protocols and classical address-based routing protocols to produce a discovery approach that significantly reduces control overheads while still providing high discovery success rates.

3.2.1 Design Stages

SDRP was developed in stages in order to investigate the effectiveness of particular design decisions and optimizations. The initial version of SDRP, SDRP-Flooding, is a simple discovery approach based on advertisement flooding by service agents. It is designed to investigate the viability of discovery without the use of an address-based routing protocol.

SDRP-Flooding includes basic measures for control message overhead reduction that are further improved upon by later versions. The second version of SDRP, SDRP-Multi-Point Relay, reduces the control overheads associated with advertisement flooding by adapting OLSR's multi-point relay (MPR) algorithm to service discovery. SDRP-MPR includes an MPR selection algorithm based on bloom filters that reduces control overheads when compared to the classical algorithm at the cost of an increased probability of incorrect MPR selection. SDRP-MPR is designed to investigate the performance and success rate of our low-overhead MPR selection algorithm.

The final version of SDRP, SDRP-Reduced MPR, further optimizes the advertisement algorithm used by SDRP-MPR by removing the use of periodic beacons in favour of an iterative probabilistic flooding optimization approach, again using bloom filters. SDRP-RMPR includes additional optimizations for scenarios involving small numbers of services in large-scale networks. This version of SDRP diverges significantly from OLSR's MPR algorithm in order to further reduce control overheads, coming closer to the probabilistic approaches discussed in section 2.1.3.3. The expected control overheads produced by each version of SDRP, compared with existing approaches based on reactive and proactive routing protocols, is illustrated in figure 3.1.

Expected Control Overheads with Increasing Network Density

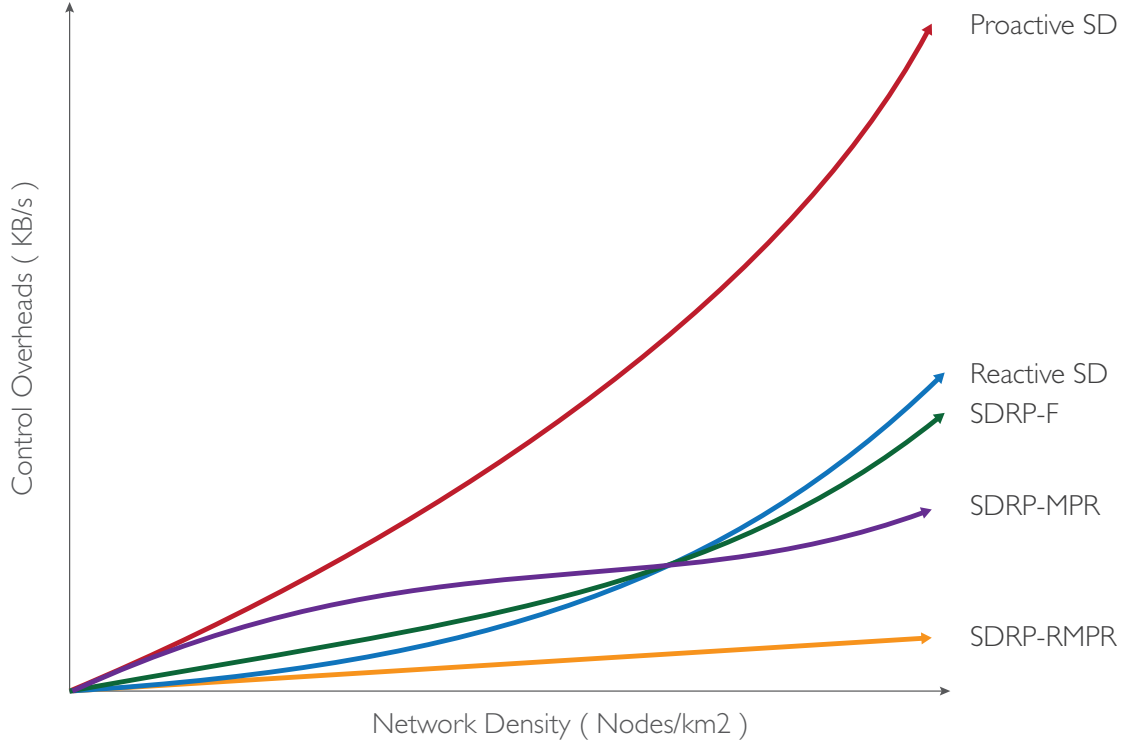


Fig. 3.1: The control overheads produced by all versions of SDRP are expected to be lower than protocols incorporating full address-based routing as network density increases. Proactive approaches such as Mercury can be expected to generate more control overheads as TC and HELLO messages are sent by an increasing number of nodes. Reactive approaches that use naive flooding such as SMF are initially efficient but their approach suffers in highly dense networks, particularly as the number of user agents increases. The SDRP family of protocols incorporate control overhead reduction measures and remove reliance on address-based routing.

To summarize, SDRP was developed in distinct stages so that each service discovery optimization introduced by our approach could be evaluated separately. Were we to directly evaluate the final version of SDRP, SDRP-RMPR, it would be difficult or impossible to separate the statistical effect of our advertisement distribution algorithm from the effect due to the removal of address-based routing. The following sections will describe these design stages in greater detail.

3.3 SDRP Core Design

The three versions of SDRP share a number of core design choices; this section will describe those common elements in greater detail. Section 3.3.1 describes the service discovery approach used by SDRP. Section 3.3.2 describes SDRP's service description method. Finally, section 3.3.3 describes how SDRP routes packets between service agents and user agents.

3.3.1 Discovery Approach

The core function of a service discovery protocol is facilitating the discovery of services by user agents. Current approaches to service discovery operate through direct service advertisement by service agents, direct service querying by user agents, indirect discovery using directory nodes or discovery using a structured overlay.

SDRP service agents directly advertise their services to user agents. This approach was chosen for a number of reasons: firstly, the delay between an application requesting a service and SDRP responding is significantly reduced when compared to user agent or directory querying as service advertisements are cached ahead of time. Secondly, we assume that the number of user agents in the network will be greater than the number of service agents; thus service agents acting as sources for control messages is more efficient than querying by many user agents. Thirdly, an approach based on proactive service advertisement provides more scope for optimization using selective rebroadcasting, as seen in OLSR and probabilistic routing protocols. Finally, the complication and control overheads associated with the formation of a structured overlay made that approach unsuitable for our target deployment scenario.

SDRP service agents periodically disseminate service advertisements throughout the network. Advertisement range may be limited using a time-to-live (TTL) value inserted into the message header. An SDRP service advertisement contains a summary of the services provided by the source service agent, the agent's identifier and, depending on the version of SDRP used, additional fields that assist in the dissemination process. The word "identifier" is used here, rather than address, in order to distinguish between SDRP's service-oriented routing approach and the approach used by address-based routing protocols.

Nodes that receive a service advertisement may rebroadcast that advertisement to their own neighbours, depending on the SDRP version used. Upon receipt of an advertisement, nodes create a route record containing the identity of the source service agent, the services offered by that service agent, the distance to the service agent in hops and the next hop along the route to the service agent. The next hop is simply the identity of the node from which the advertisement was received. Redundant advertisement rebroadcasts are avoided through the use of sequence

numbers. Service agents insert an incrementing sequence number into advertisements. Nodes receiving an advertisement will record the sequence number and associate it with the source service agent. When a new advertisement is received from a neighbour, the sequence number is checked against the last number received from the source service agent. If the sequence number is less than or equal to the last number received, the message has already been received and is not rebroadcast.

Through the advertisement processes, nodes gain knowledge of the services available in the network and are able to route messages to those services by sending them to the next hop back along the path taken by received service advertisements. In the next section, the service description approach used by SDRP is described.

3.3.2 Service Description

SDRP's service description approach is similar to that of Mercury; the set of services provided by a service agent are summarized using a bloom filter and inserted into service advertisements. Bloom filters are a space-efficient data structure used to represent sets. A bloom filter is initially represented by an array of m bits, all set to 0. Elements are inserted into a bloom filter by hashing them using k hash functions, producing offsets into the bit array which are then set to 1. Bloom filters support queries for elements by hashing the element to be searched for and checking whether the resulting bit offsets are set to 1. If the offsets are set to 1, the element is probably in the filter. If the offsets are set to 0, the element definitely is not in the filter. For a more detailed explanation of bloom filters, see Appendix A.

The advantage of this approach is that summaries of provided services are compact, particularly when compared to approaches that directly serialize service descriptors into advertisements or queries. Some approaches, such as SMF, describe services using strings; this approach can result in significantly larger protocol messages, particularly for nodes providing many services.

Bloom filters have the added advantage of being able to contain service descriptors of varying types. Whether services are described using integers, strings or some other data type, they can be inserted into a bloom filter using the same hashing process. When disseminating service advertisements, SDRP forms a bloom filter containing the services provided by the service agent. This bloom filter is referred to as the *service filter*. The size of the constructed service filter depends on the number of services to be advertised and a desired false-positive probability, configured at deployment time. Given a number of service identifiers, n , and a false-positive probability, p , the required length for the bloom filter is given by the following equation:

$$m = -\frac{n \ln p}{(\ln 2)^2} \quad (3.1)$$

The optimal number of hash functions, k , that minimizes the probability of a false-positive membership query is given by the following equation:

$$k = \frac{m}{n} \ln 2 \quad (3.2)$$

With a lower false-positive probability, messages become larger due to an increase in filter size. However, configuring a higher false-positive value can potentially result in user agents mistakenly determining that a service agent is providing a service when this is not the case. Thus, the false-positive probability must be calibrated depending on the requirements of the user.

Input: Target Service Descriptor

Output: Routes to Service Agents Providing the Target Service

```

1 targetService ← target service descriptor;
2 matches ← Route Set;
3 foreach Route r in Route Cache do
4   | serviceFilter ← r.services;
5   | if serviceFilter.contains( targetService ) then
6   |   | matches.insert( r );
7   | end
8 end
9 return matches

```

Algorithm 1: SDRP Service Cache Search Algorithm

User agents cache the service filters advertised by service agents and associate them with the recorded route. When an application requests a service, the user agent iterates over the cached filters and queries them using the service descriptor provided by the application. The results of this search are then used to route application messages to the target service, typically based on route length. This algorithm is described in Algorithm 1.

Routes and cached service filters are periodically purged so that stale records are not used. The maximum lifetime of a route is based on the service advertisement interval used by service agents.

A weakness of this service description approach, when compared with direct serialization of raw descriptors into messages, is that the descriptors contained in the resulting bloom filter cannot be iterated over. Bloom filters support queries for the presence of elements, however an

element that is inserted into a bloom filter cannot be extracted. Thus user agents can only query service filters for the presence of service descriptors. In the next section, the packet routing approach used by SDRP will be described.

3.3.3 Routing

SDRP is not designed to support address-based routing, however it must facilitate routing between user agents and service agents in order to operate. As outlined in section 3.3.1, SDRP's service advertisement mechanism allows messages from user agents to be routed to service agents by reversing the path taken by service advertisements. Packets are routed on a hop-by-hop basis rather than through the inclusion of the entire route in the packet header, as seen in protocols such as DSR.

Intermediate nodes along the route to a service agent expect the packet to contain a service descriptor indicating which service the packet is being routed towards and the identifier of the target service agent. Using the service descriptor, nodes search their route cache and select the route with the fewest hops to the destination service agent. The packet is then sent to the next node along that route. This process continues until the service agent receives the packet.

Messages from a service agent to a user agent are routed back along the packet taken by packets sent to the service agent from the user agent. Thus, SDRP expects that communication between a user agent and a service agent will be initiated by the user agent. This pattern is similar to that used by current applications, such as HTTP and chat servers. Intermediate nodes along a route will record a *reverse route* when relaying a message between a user agent and a service agent. This reverse route contains the identifiers of the user and service agent endpoints of the routed data packet.

If a data packet is received that does not contain a destination service descriptor, it is assumed that this is a packet being sent to a user agent and the reverse route cache is searched using the packet source and destination node identifiers, as outlined in Algorithm 2.

The weakness of this approach is that reliable routing between service agents and service agents relies on frequent packet traffic between user agents and service agents in order to establish reverse routes. If no message is received from a user agent and node mobility or power loss causes breaks in the links that comprise a reverse route, a service agent wishing to communicate with a user agent must wait for a new route to be established by a message sent to it from that user agent. Thus, SDRP is best suited to traffic that matches a request-response pattern, or for networks with low node mobility. In the next section, the first version of SDRP, SDRP-Flooding, will be described.

Input: Data Packet

Output: Routes to Destination Node

```
1  $p \leftarrow$  Data Packet;
2  $matches \leftarrow$  Route Set;
3 if  $p$  contains target service descriptor then
4   | search route cache
5 else
6   |  $src \leftarrow$  packet source identifier;
7   |  $dst \leftarrow$  packet destination identifier;
8   | foreach Route  $r$  in Reverse Route Cache do
9     |   if  $r.src == src$  AND  $r.dst == dst$  then
10      |   |  $matches.insert( r );$ 
11      |   end
12   | end
13 end
14 return  $matches$ 
```

Algorithm 2: SDRP Route Search Algorithm

3.4 SDRP - Flooding

SDRP-Flooding or SDRP-F is designed to investigate the viability of service discovery without assistance from an address-based routing protocol. SDRP-F uses a simple mechanism to facilitate service discovery based on the periodic flooding of service advertisements by service agents and incorporates the design elements described in the previous section. In the next section, SDRP-F's service advertisement dissemination mechanism will be described.

3.4.1 Service Advertisement

SDRP-F uses a naive flooding approach, similar to those used by reactive routing protocols for the distribution of route requests, to disseminate service advertisements. This approach generates unnecessary overheads due to redundant rebroadcasts by intermediate nodes, however it is a reliable method for achieving full coverage for disseminated messages.

SDRP-F service agents periodically broadcast service advertisements to neighbouring nodes. Advertisements include a packet type field, the node identifier of the service agent, a summary of offered services, a current message hop count, a time-to-live value and a sequence number that is incremented every time the service agent sends an advertisement. The time-to-live field is used to limit the propagation range of the advertisements if required so that larger

networks are not overwhelmed. The bloom filter containing a summary of the services offered by the service agent is referred to a *service filter*. The format of the SDRP-F service advertisement message is outlined in figure 3.2.

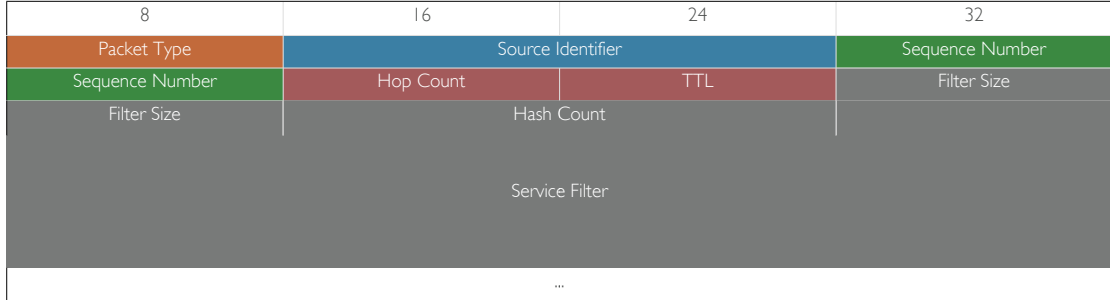


Fig. 3.2: An SDRP-F service advertisement contains a service filter describing the services offered by the source service agent, prefixed with an integer indicating the number of descriptors contained in the filter. The message also contains fields allowing the advertisement’s propagation range to be limited.

Nodes receiving a service advertisement check the last sequence number received from the message source. If the sequence number is greater than the last recorded sequence number, the advertisement is considered to contain fresh information. The contained sequence number is then recorded and the node service route table is updated.

If the message’s current hop count is less than the time-to-live value, the hop count is incremented and the advertisement is rebroadcasted. If the hop count value is greater than or equal to the time to live, the packet is instead dropped. The combination of hop count and time to live allows service agents to limit the propagation range of advertisements. This process is outlined in Algorithm 3.

This approach improves network scalability, as advertisements will be dropped past a certain distance instead of covering the entire network. The time-to-live can be dynamically updated if required in response to changing network conditions or available resources.

The identifier of the advertisement source, whether the message originator or the last node to relay it, is recorded as the next hop on the route to the services contained in the the advertisement. The distance to the source, described by the hop count field, is also recorded. Using this information, user agents can choose the shortest route to a service agent advertising a particular service. In the next section, an example will be used to illustrate how SDRP-F operates.

Input: Service Advertisement Packet

```
1  $p \leftarrow$  advertisement packet;
2  $sequenceNum \leftarrow p.sequence\_number$ ;
3  $tll \leftarrow p.ttl$ ;
4  $hops \leftarrow p.hops$ ;
5 // Construct and record a route to the source service agent
6  $route \leftarrow route$ ;
7  $route.src\_id \leftarrow p.src\_id$ ;
8  $route.dst\_id \leftarrow p.dst\_id$ ;
9  $route.last\_hop \leftarrow p.last\_hop$ ;
10  $route.services \leftarrow p.service\_filter$ ;
11  $route.hops \leftarrow p.hops$ ;
12  $route.refresh \leftarrow now$ ;
13  $record\_route( route )$ ;
14 // Get the last highest sequence number received from the source
15  $lastSeq \leftarrow get\_seq\_num( p.src\_id )$ ;
16  $tll \leftarrow tll - 1$ ;
17 /* If the hop count is less than the TTL and the sequence number is valid,
    rebroadcast the packet and record the new sequence number */
18 if  $hop$  greater than  $tll$  AND  $sequenceNumber$  less than  $lastSeq$  then
19   |  $set\_seq\_num( p.src\_id, sequenceNumber )$ ;
20   |  $rebroadcast\_packet( p )$ ;
21 end
```

Algorithm 3: SDRP-F Packet Handling Algorithm

3.4.2 Example

The operational cycle of SDRP-F will be illustrated using a simple example involving a network composed of three nodes, one of which has a single service that should be made available to the network.

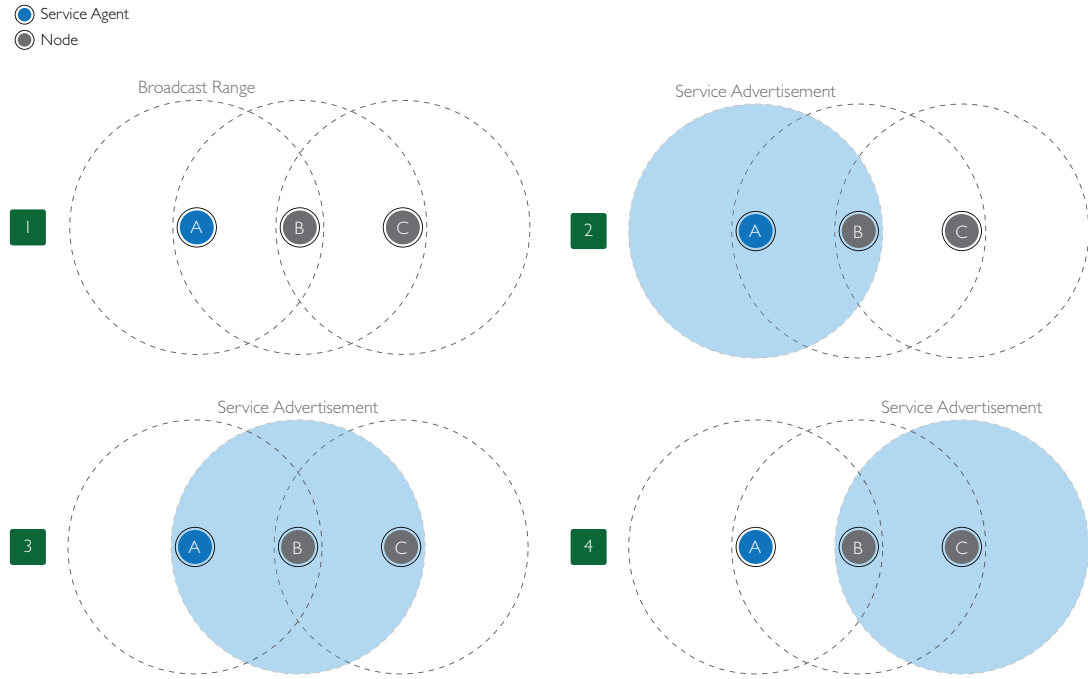


Fig. 3.3: Operation Cycle Diagram for SDRP-F Example

The operational steps taken to advertise the service are illustrated in figure 3.3. In step 1, the SDRP instance running on node A has a new service identifier inserted into its service filter.

Periodically, the SDRP instance running on all nodes checks whether any services have been inserted into the local service filter. If the filter contains no services, there is nothing to advertise and the SDRP instance will do nothing. Otherwise, if the filter contains one or more service identifiers, SDRP will construct a Service Advertisement message, providing the local node address, message sequence number and services to be advertised in the form of a bloom filter, before broadcasting the message to all neighbouring nodes.

In step 2, node B receives the service advertisement message from node A. The SDRP instance deserializes Service Advertisement message and stores a route record containing the last hop from which the advertisement was received, the time at which it was received, the source

service agent and the advertised service filter. Route records over a certain age are periodically purged from the route table by the SDRP instance.

The advertisement sequence number is compared to the last sequence number recorded from the advertisement's source service agent. If the sequence number is less than or equal to the last recorded sequence number, the advertisement is considered a duplicate and is discarded. If the advertisement sequence number is greater than the last recorded value, its hop count is then compared against the message's maximum TTL. If the hop count is less than the set maximum TTL, the advertisement can be rebroadcast. The message's hop count is incremented and rebroadcast to all neighbouring nodes. Node A, recognizing that the message originated from itself through inspection of the source field, drops the packet. In step 3, node C handles the message.

Step 3 is similar to step 2 in terms of the algorithm used. The packet received from node B is deserialized by the SDRP instance and a new route to the source, node A, is recorded with B acting as the next hop. If the sequence number and hop count are valid, which they are, the message is then rebroadcast. Node B deserializes the message and matches the contained sequence number against the one it recorded upon receiving the same message from node A and discards it. A route record is still recorded, setting node C as the next hop towards the destination, however that route will not be used as it is longer than the route with an edge between nodes A and B. Node C increments the message hop counter and rebroadcasts the message. At this point, no new nodes are available to receive the message and the advertisement cycle ends. Nodes B and C now have available routes to the services provided by node A and can send messages to those services through the next hop addresses recorded in their routing tables.

3.4.3 Summary - SDRP-F

SDRP-F is designed to test the viability of enabling service discovery without the use of an underlying routing protocol. SDRP-F service agents flood service advertisements through the network, allowing user agents to become aware of available services ahead of time. Service advertisement size is reduced through the use of bloom filters for service description, reducing control overheads with the cost of some probability that user agents will falsely determine that a service agent provides a required service.

SDRP-F compares favourably to existing service discovery approaches based on proactive routing protocols in scenarios involving few service agents. Unlike approaches such as Mercury, in which all nodes selected as MPRs act as sources for disseminated service advertisements and topology control data, and all nodes act as sources for broadcast HELLO messages, in SDRP-F only service agents act as sources for protocol messages. SDRP-F's compact service description

approach compares favourably with that of protocols such as SMF. The use of bloom filters for service description instead of the direct serialization of strings results in reduced message size.

The advantages for SDRP-F in terms of produced overheads decrease as more service agents are added to the network, as the naive flooding algorithm used to disseminate advertisements causes interference and medium contention. SDRP-Multi-Point Relay was developed to address this issue by incorporating a modified version of OLSR's MPR message dissemination algorithm. The following section discusses SDRP-MPR in greater detail.

3.5 SDRP - Multi-Point Relay

SDRP-MPR is designed to further reduce overheads by incorporating a revised service advertisement dissemination algorithm based on OLSR's MPR algorithm. The OLSR MPR algorithm allows topology control messages to be distributed efficiently when compared with naive flooding. Nodes periodically broadcast HELLO messages containing the addresses of their immediate neighbours. This information is then used to calculate the minimum set of neighbours through which all second-hop neighbours can be reached.

The selected neighbours are marked as multi-point relays using subsequent HELLO messages and will rebroadcast topology control messages, allowing second-hop neighbours to be reached using the minimum number of required nodes. A more detailed discussion of OLSR's MPR algorithm can be found in section 2.1.1.2.

The OLSR MPR algorithm requires that all nodes periodically broadcast HELLO messages containing the address and link-state of their immediate neighbours. Addresses and link-states are serialized directly into HELLO messages. As a result, HELLO message size tends to increase with increasing network density, as the number of nodes within direct contact with each other increases. With sufficiently high network densities, HELLO messages may span multiple packets, further increasing impact on application traffic due to interference. This relationship is depicted visually in figure 3.4.

SDRP is designed for deployment in high-density networks, thus an MPR selection approach that scales more efficiently was needed. In the next section we will describe our MPR algorithm.

3.5.1 Optimized MPR Selection

MPR selection requires that nodes periodically broadcast the identities of their first-hop neighbours. The direct serialization of neighbour addresses into HELLO messages results in an in-

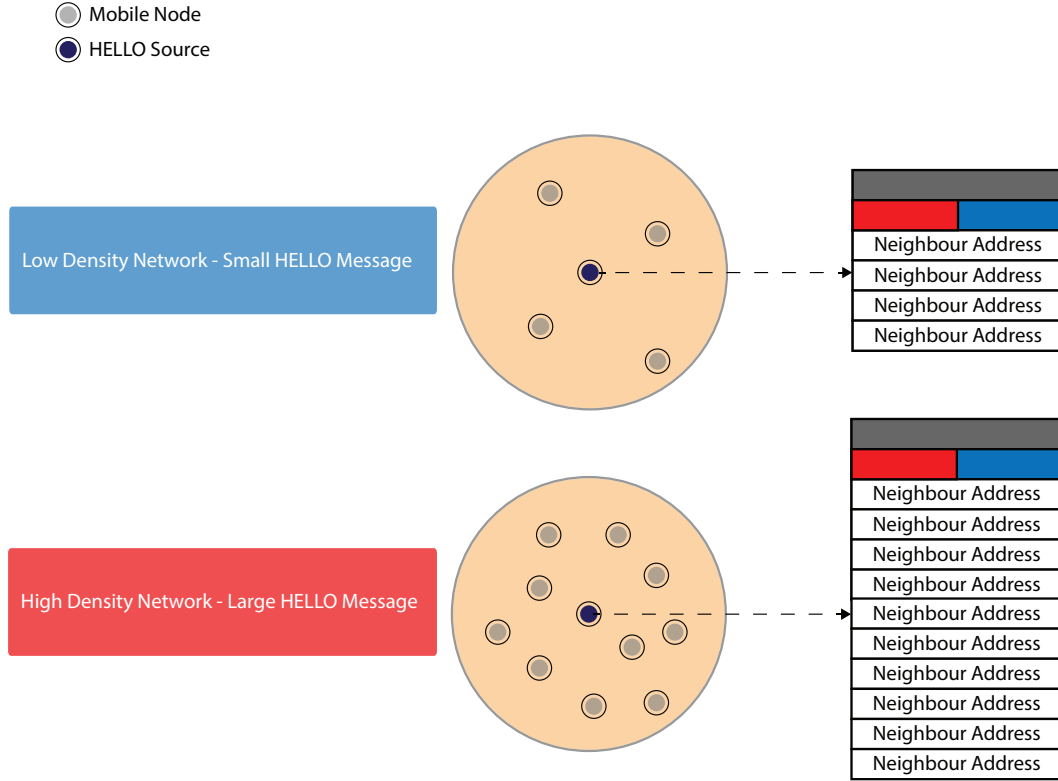


Fig. 3.4: The HELLO message format used by many routing protocols include summaries of the addresses of neighbouring nodes, the link-state of the source or distance-vector information. These messages can increase in size dramatically as network density increases.

crease in message size proportional to the increase in local network density; producing additional overheads. To reduce this cost, an alternative, space-efficient approach to neighbour address advertisement was required.

The use of bloom filters for service set description reduces advertisement message sizes when compared to direct serialization. However this approach is difficult to extend to the task of neighbour set advertisement. The selection of MPR nodes requires that neighbour sets included in the HELLO messages received from neighbouring nodes be compared.

Direct comparison of serialized lists of addresses may be accomplished using simple set operations. Due to the probabilistic nature of bloom filters and the fact that elements, once inserted, cannot be extracted; their use in MPR selection presents a difficult challenge. In the following sections, we will outline the set operations that can be performed on bloom filters and

explain how we applied these operations to MPR selection as part of SDRP-MPR.

3.5.2 Logical Operations on Bloom Filters

Operations on sets such as union and intersection can be approximated on bloom filters in order to facilitate comparisons. Direct comparison between bloom filters requires that filters share the same filter size, m , and the same k hash functions.

A Union is performed using bitwise OR operations on the input filters, resulting in a filter equivalent to one into which all constituent set elements had been inserted. This operation is visualized in figure 3.5.

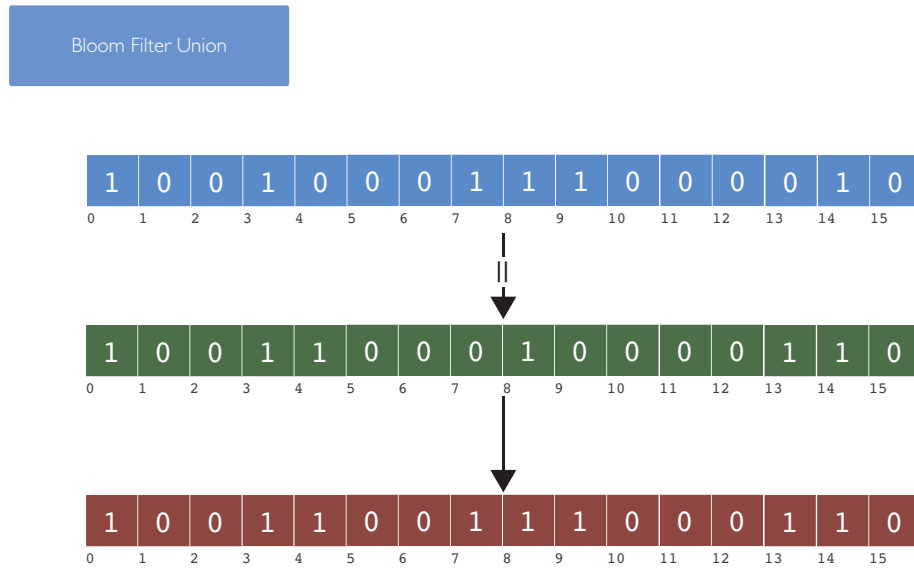


Fig. 3.5: A filter union is performed using bitwise OR, resulting in a filter with the same properties as one into which all source set elements had been inserted.

Intersection is performed using bitwise AND operations and results in a filter with a false-positive probability that is at most that of one of the input filters, but which may be higher than the probability resulting from insertion of the input set intersection. This property arises from the fact that intersecting bits in a pair of filters may not result from insertion of the same element. This operation is visualized in figure 3.6.

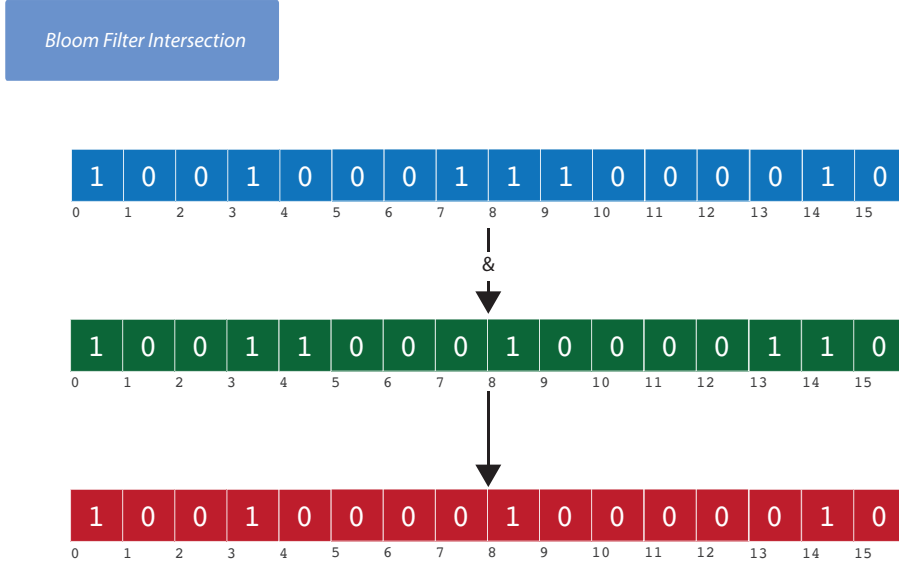


Fig. 3.6: A filter intersection is performed using bitwise AND.

3.5.3 Bloom Filter-Based MPR Selection

OLSR's MPR selection algorithm operates on the sets of neighbour link-states deserialized from HELLO messages. As network density increases, the size of these HELLO messages also increases and may even span multiple packets; increasing network overheads and reducing performance. SDRP-MPR attempts to reduce the overheads induced by neighbour link-state advertisement by broadcasting bloom filters containing the set of the source node's neighbour addresses, rather than serialized link-states.

3.5.3.1 HELLO Messages

Due to the constraint imposed by set operations on bloom filters, the *neighbour filter* used to describe link-state by all SDRP instances in the network must be the same size, m , and use the same k hash functions. Thus suitable bloom filter parameters for the target network must be configured before deployment in order to maximize space efficiency.

Figure 3.7 shows the packet format for an SDRP HELLO message. The message contains the identifier of the source, a packet type field indicating that it is a HELLO message and a bloom filter containing the identifiers for all of the source's neighbours. Nodes which receive HELLO

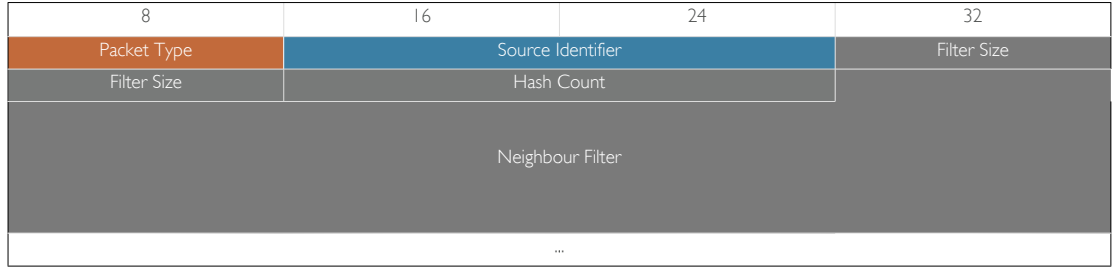


Fig. 3.7: SDRP HELLO messages contain the source node identifier and a bloom filter containing the identifiers of all of the source’s neighbours.

messages add the source identifier to their neighbour identifier cache and insert those neighbour identifiers into their a neighbour filter when a HELLO message is transmitted. If a node does not receive a HELLO message for a configured period of time, the neighbour’s identifier is removed from the neighbour identifier cache.

3.5.3.2 MPR Selection Algorithm

OLSR MPR selection is carried out using the following steps:

- Form a set of all 2-hop neighbours
- Select all neighbours with unique 2-hop connections
- Select neighbours in order of descending 2-hop connection count
- When all 2-Hop connections are included, selection is complete

SDRP-MPR replicates this algorithm using the bloom filter operations outlined above. A filter describing all 2-hop neighbours is formed by performing a union operation on all received neighbour filters. Neighbours with unique 2-hop connections are selected by iterating over all neighbour filters and performing a difference operation; effectively the opposite of an intersection. If the neighbour filter still has set bits after differencing against the other filters, it has a unique connection to a 2-hop neighbour, as illustrated in figure 3.8. This neighbour is then added to the MPR set. Nodes added to the MPR set have their neighbour filters added to an MPR bloom filter using a union operation. Once the MPR filter contains the same set bits as the union of all neighbour filters, MPR selection is complete. This approach is outlined in Algorithm 4.

It should be noted that this method for identifying neighbours with unique 2-hop connections carries a non-zero false negative probability, as the combination of bits set by a unique second-hop neighbour may be covered by the hashing of a combination of other node identifiers.

Input: Current Neighbour Node Data

Output: Multi-Point Relay Bloom Filter

```
1 mprNodes ← bloom filter;
2 mprCoverage ← bloom filter;
3 requiredCoverage ← bloom filter;
4 /* Construct a bloom filter containing the identifiers of all neighbours */
5 foreach Node n in Neighbour Node Data do
6   | requiredCoverage.insert( n.neighbours )
7 end
8 /* Determine neighbours with unique 2-hop connections */
9 foreach Node n in Neighbour Node Data do
10  | t ← requiredCoverage;
11  | rem ← requiredCoverage.remove( n.neighbourFilter );
12  | if t.num_set_bits != rem.num_set_bits then
13    |   mprNodes.insert( n.address );
14    |   mprCoverage.insert( n.neighbourFilter );
15  | end
16 end
17 /* Sort nodes by number of bits set in neighbour filters */
18 /* Select MPR nodes until their coverage encompasses the required coverage */
19 while !mprCoverage.contains( requiredCoverage ) do
20   | n ← next node mprNodes.insert( n.address );
21   | mprCoverage.insert( n.neighbourFilter );
22 end
23 return mprNodes
```

Algorithm 4: SDRP MPR Selection Algorithm

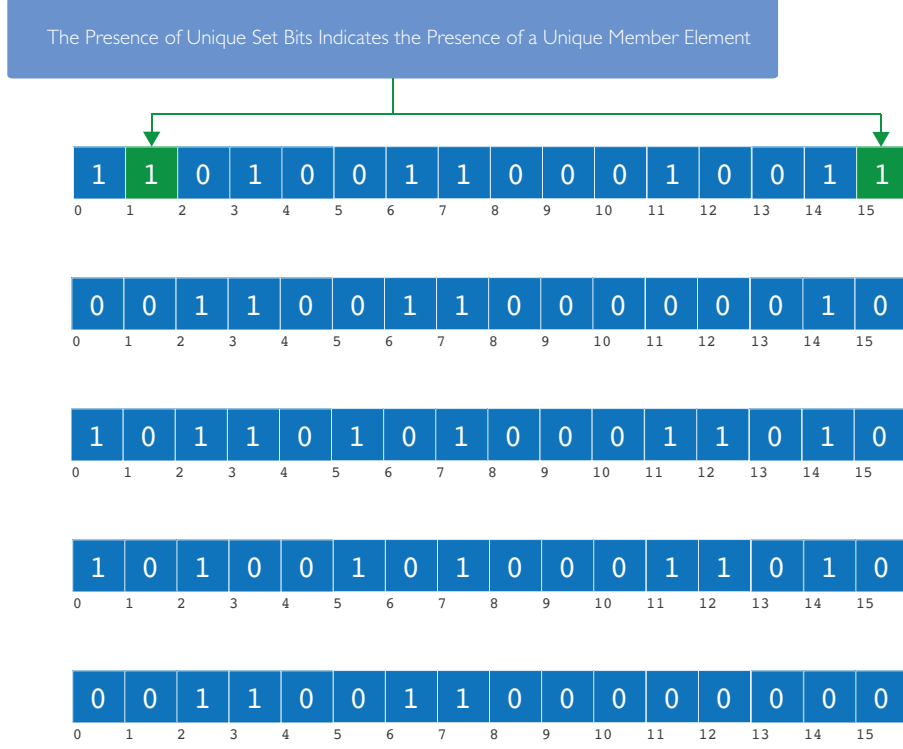


Fig. 3.8: Neighbour filters containing uniquely set bits indicate a unique connection to a 2-hop neighbour. Such neighbours are the first to be selected as MPRs.

As outlined in appendix A, this false positive probability can be reduced through an increase in filter size at the cost of increased control overheads.

The neighbour filter field of the HELLO message is prefixed with an integer indicating the number of identifiers in the filter. This allows nodes performing an MPR selection to iterate over all filters in order of their size. Neighbours have their neighbour filters added to the MPR filter in order of coverage area, as described, until the MPR filter matches the union of all neighbour filters; indicating that full coverage has been achieved. The source of each selected filter is inserted into an MPR set and is used for service advertisement.

It should be noted that the accuracy of MPR selection is highly dependent on the parameters used to generate neighbour filters. Larger filters result in a greater probability of correctly selecting 1-hop neighbours with unique connections to 2-hop neighbours and of correctly select-

ing MPR nodes due to the lower probability of bit intersection. Correctly configuring neighbour filter parameters at deployment time can therefore have a significant effect on the accuracy of MPR selection which in turn affects the successful distribution of service information. Incorrect MPR selection may result in a lack of coverage in some areas of the network, leaving user agents unable to contact service agents. The use of larger filters has a negative effect on control overheads, however, due to the resultant increase in message size.

3.5.4 Service Advertisement

SDRP-F service agents distribute service advertisements using a naive flooding algorithm, resulting in unnecessary re-broadcasts by intermediate receivers. SDRP-MPR adopts a variation on the OLSR MPR algorithm, ensuring that only the subset of neighbours necessary to reach all 2-hop neighbours rebroadcast advertisements. Once MPR nodes have been selected using the algorithm described in the previous section, a variable-length bloom filter is then generated with parameters based on the number of MPR nodes. MPR nodes identifiers are inserted into this filter. The filter is then serialized into the service advertisement message. Nodes receiving a service advertisement query the MPR filter for their own identifier and will rebroadcast it to their own MPR nodes if it is present.

This approach differs from that of OLSR in that MPR nodes are indicated in the service advertisement rather than in HELLO messages. SDRP-MPR's use of bloom filters for neighbour description makes associating attributes with particular nodes difficult; nodes attempting to ascertain their link-state with a HELLO source would need to query the filter for the combination of their identifier and every possible link-state; an approach which would be difficult to extend and would complicate comparison between neighbour filters during MPR selection.

SDRP-MPR retains the service descriptor encoding approach used in SDRP-F; service descriptors are encoded into a variable-length bloom filter in order to reduce message size. A received service filter can be queried by user agents for required services with a false-positive probability determined by the filter parameters. The packet format for an SDRP-MPR service advertisement is shown in figure 3.9.

3.5.5 Example

The operational cycle of SDRP-MPR will be illustrated using a simple example using a small network composed of 7 nodes, as illustrated in figure 3.10. This topology illustrates the problem presented by a pure-flooding mechanism; in order to reach all nodes with a service advertisement, only nodes E and C should rebroadcast node A's message. Redundant rebroadcasts by nodes

8	16	24	32
Packet Type	Source Identifier		Sequence Number
Sequence Number	Hop Count	TTL	Filter Size
Filter Size	Hash Count		
MPR Filter			
			Filter Size
Filter Size	Hash Count		
Service Filter			
...			

Fig. 3.9: SDRP-MPR Service Advertisements contain an MPR filter indicating which nodes should rebroadcast the advertisement upon receipt.

B, D, F and G, which would occur under SDRP-F, interfere with application traffic, reducing performance and usability.

Initially, the service agent A has no knowledge of its local neighbourhood. After a period defined by t_{hello} , all nodes broadcast a HELLO message containing their own identifier and a bloom filter containing the identifiers of their known neighbours as illustrated in figure 3.11.

The HELLO messages received by A from nodes B, C, D and E do not indicate the presence of nodes F, A and G, as they had not yet received HELLO messages from those nodes before sending their own. After a period defined by $t_{advertise}$, node A will attempt to send a service advertisement. However, as no neighbouring nodes have indicated that node A is in their neighbourhood, and thus no MPR nodes can be chosen, the advertisement will not be sent.

After an additional t_{hello} seconds, a second HELLO message will be broadcast, as shown in figure 3.12. Node A now recognizes that it's identifier is contained in the HELLO message neighbour filters of nodes B, C, D and E.

$t_{advertise}$ seconds after the initial advertisement attempt, node A now has sufficient knowledge of the network to select a set of MPR nodes for advertisement rebroadcast. It composes a full neighbour filter containing the identifiers of nodes B, C, D, E, F and G. Examination of node C and E's neighbour filters shows that they have unique connections to nodes F and G, due to the presence of unique set filter bits. The MPR filter composed of node E and C's neighbour filters, when compared with the full neighbour filter, shows no differences in terms of set bits, as these nodes are sufficient to cover all two-hop neighbours. Node E and C have their identifiers inserted in an MPR filter, which is then serialized into a service advertisement message.

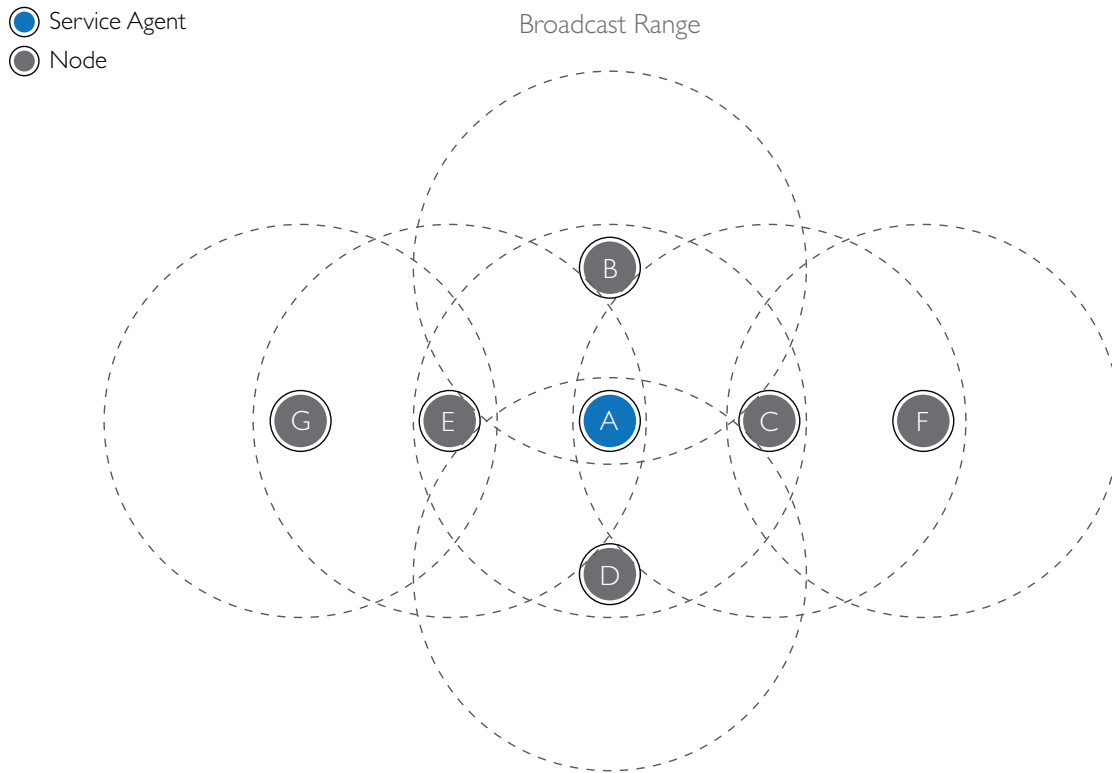


Fig. 3.10: Example network configuration running SDRP-MPR. MPR selection should result in nodes E and C being chosen to rebroadcast service advertisement messages, as they will be able to reach the second-hop neighbours F and G. Nodes B and D have no second-hop neighbours, therefore they should not be chosen as MPR nodes.

The advertisement is sent and rebroadcast by nodes C and E, as shown in figure 3.13. Nodes C and E will attempt to form an MPR set using their knowledge of the local area. Node A will be chosen as their MPR node and will have its identifier inserted into the MPR filter. When node A receives its own rebroadcasted service advertisement message from nodes C and E, it will drop the message; both due to the fact that it was the source of the message and because the included message sequence number is the same as the one it recorded when sending the message itself.

3.5.6 Summary - SDRP-MPR

SDRP-MPR is designed to solve the problem of redundant rebroadcasts of service advertisement messages that emerged from the design of SDRP-F. SDRP-MPR incorporates a modified version

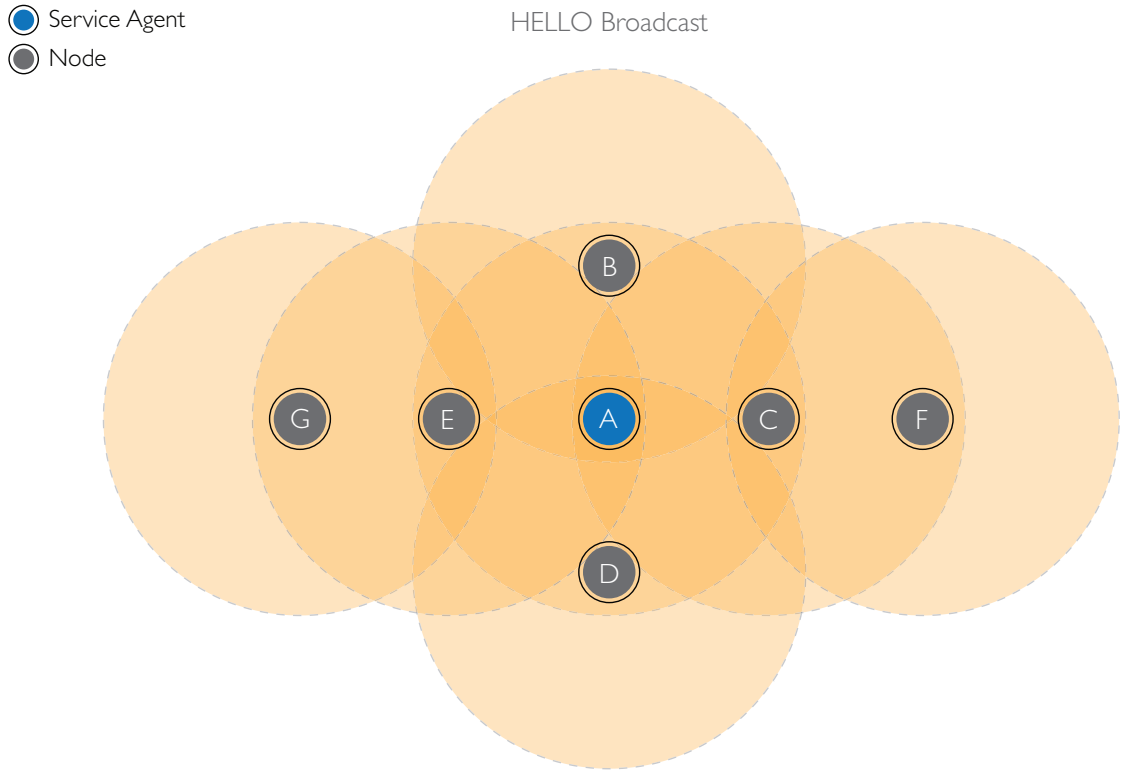


Fig. 3.11: All nodes periodically broadcast HELLO messages, allowing neighbouring nodes to become aware of their presence and service advertisement broadcasters to select MPR nodes.

of OLSR's MPR algorithm that operates using bloom filters instead of direct serialization of node identifiers and link-states. The OLSR MPR algorithm attempts to determine the subset of a node's neighbours through which all 2-hop neighbours may be reached. Those 1-hop neighbours are then used to rebroadcast messages, reducing control overheads due to the elimination of redundant rebroadcasts while still ensuring full coverage. The MPR algorithm requires knowledge of neighbour node link-states, thus all nodes must periodically broadcast HELLO messages containing their link-states with their immediate neighbours.

SDRP-MPR's use of bloom filters for neighbour link-state description results in significant control overhead reduction due to the space-efficiency of bloom filters, however it comes at the cost of a reduction in MPR selection accuracy depending on the bloom filter parameters used. SDRP-MPR also displays a lack of flexibility, as the bloom filter parameters cannot be adjusted at run-time without coordination between nodes, otherwise MPR selection cannot be performed due to failure of bloom filter comparison operations. If the configured parameters generate filters which cannot contain all neighbours without high false-positive probability, due to increasing

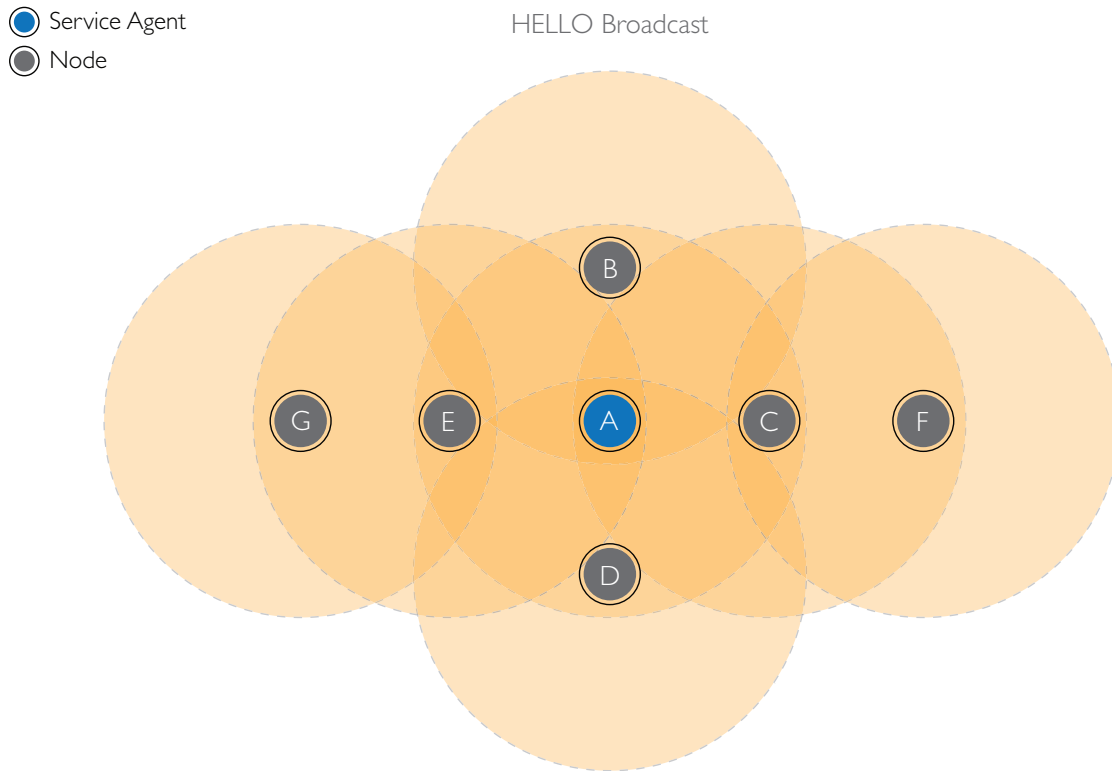


Fig. 3.12: After the initial HELLO broadcast, nodes have now become aware of their neighbours and include their identifiers in subsequent HELLO messages.

network size or fluctuations in topology, MPR selection accuracy and service advertisement coverage are then compromised.

Given optimal filter parameters, SDRP-MPR can be expected to produce lower control overheads than SDRP-F; particularly as the number of service agents in the network increases and the cost of sending HELLO messages becomes less significant relative to the cost of flooding service advertisements.

3.6 SDRP - Reduced MPR

SDRP-Reduced MPR (SDRP-RMPR) is designed to retain the performance benefit of MPR selection while eliminating the control overheads associated with HELLO message broadcasts and to more easily adapt to changes in network density and topology. SDRP-RMPR incorporates a modified version of the MPR algorithm used in SDRP-MPR that uses variable-length bloom

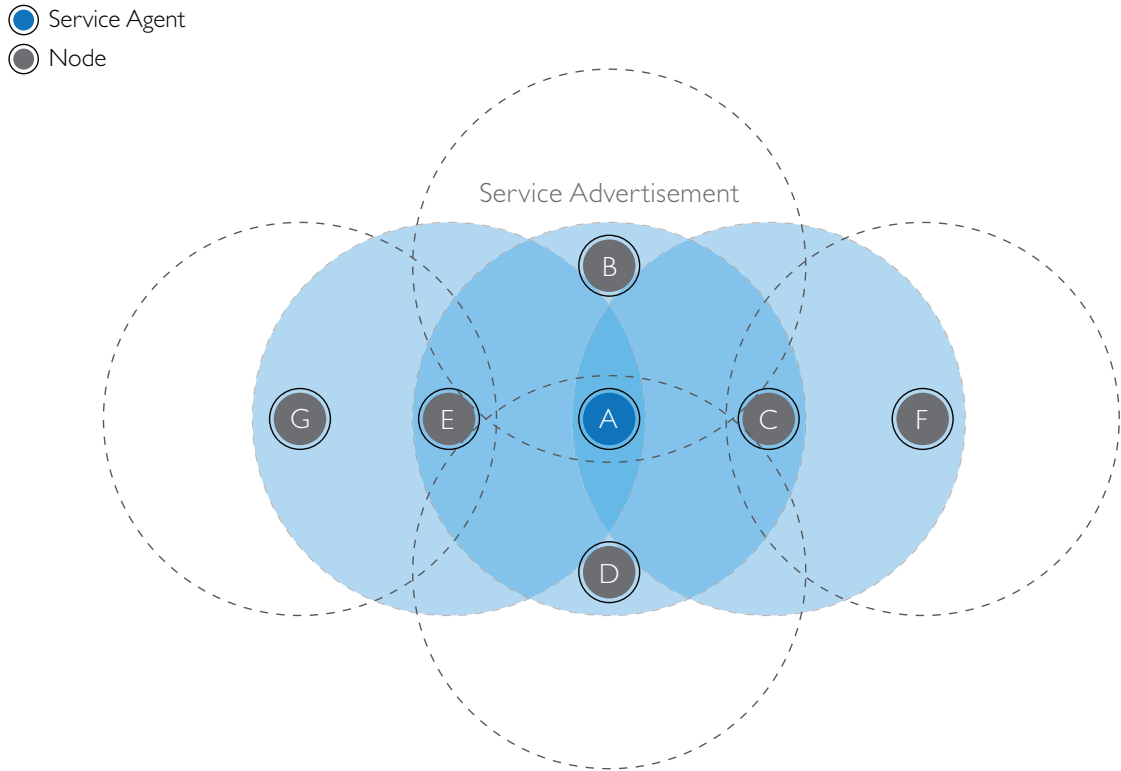


Fig. 3.13: Nodes C and E, upon receipt of the service advertisement from node A, examine the message's MPR filter and determine that it contains their identifiers. They then rebroadcast the message. As nodes B and D's identifiers are not included, they do not rebroadcast the message.

filters for neighbour link-state description and incorporates those filters directly into service advertisements, removing the reliance on HELLO messages. The following sections will describe the reduced MPR algorithm in greater detail.

3.6.1 MPR Selection using Variable-Length Bloom Filters

SDRP-MPR uses bloom filters with a preconfigured length to advertise neighbour link-state. This approach was used so that received filters could be compared using set operations, as described in section 3.5.2; operations that cannot be performed on filters of differing lengths.

This approach suffers from changes in network density, as the configured filter size may not be sufficient to describe link-state without a high false-positive probability. With high false-positive probabilities and high filter fill rates, accurate MPR selection cannot be performed accurately and advertisement coverage is then compromised. Similarly, low network density

may result in wasted message space as the neighbour filter size may be significantly larger than required.

To solve this problem, a method for selecting MPR nodes using neighbour filters of varying size was required. The OLSR MPR algorithm involves comparison of neighbour sets to find unique connections to 2-hop neighbours and achieve full coverage. However, without the ability to compare neighbour filters using set operations, this would no longer be possible.

Review of existing routing approaches lead to the design of a new algorithm based on the probabilistic approach used by Cartigny and Simplot (2003). In Cartigny and Simplot's approach, the distance between a node and its neighbours is estimated based on the degree of intersection between their neighbour sets. If the degree of intersection is high, the neighbouring node is probably near while a low degree of intersection indicates greater distance. Nodes with a distance metric greater than some limit are then chosen to rebroadcast packets as they are more likely to be on the edge of the source node's broadcast range.

By modifying this approach to use variable-length bloom filters, distance to neighbouring nodes can be estimated and more distant nodes can then be chosen to rebroadcast service advertisements. The use of variable-length bloom filters ensures a consistently low false-positive probability while reducing message sizes and control overheads.

3.6.2 HELLO Message Removal

The OLSR MPR algorithm requires that all nodes periodically broadcast HELLO messages containing the addresses and link-states of neighbouring nodes. In a highly dense network, the overheads generated by these messages can have a detrimental effect on application throughput.

SDRP-MPR attempts to reduce HELLO message size through the use of bloom filters for neighbour set description, however all nodes are still required to periodically broadcast HELLO messages regardless of whether the local network topology has changed, producing potentially unnecessary overheads. SDRP-MPR's use of static-length filters also reduces MPR selection accuracy when local network density exceeds the limits imposed by the configured filter parameters.

SDRP-RMPR solves this problem by removing the requirement that nodes periodically broadcast HELLO messages. Instead, service advertisement messages incorporate a variable-length bloom filter describing the local neighbour set. Nodes rebroadcast service advertisements under two circumstances; if their identifier is contained in the MPR filter or if their identifier is not contained in the neighbour filter. The SDRP-RMPR service advertisement format is illustrated in figure 3.14.

Thus highly mobile nodes will tend to rebroadcast advertisements frequently in order to

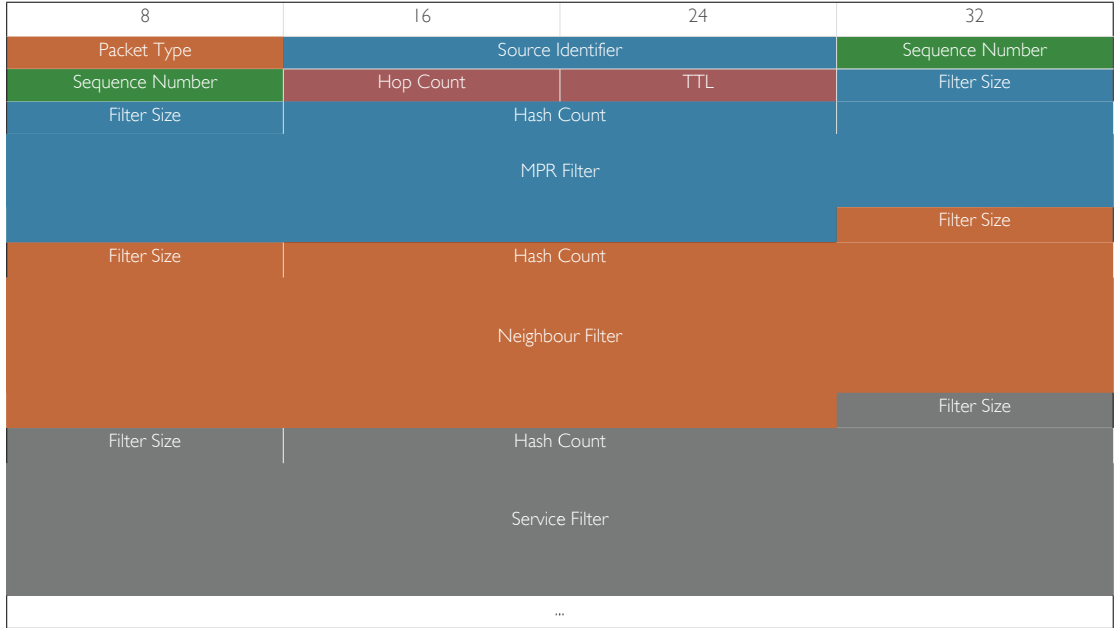


Fig. 3.14: SDRP-RMPR Service Advertisements contain a variable-length neighbour filter. This filter is used for border-node selection by receiving nodes.

inform their neighbours of their presence, as they will frequently receive advertisements that contain a neighbour filter that omits their identifier. Conversely, immobile nodes that are not selected as MPR nodes will broadcast infrequently. In order to ensure that more distant topology changes detected by silent nodes are broadcast, nodes that have not rebroadcast an advertisement for a pre-configured length of time will do so at the next opportunity.

Nodes that receive a service advertisement add the source identifier and contained neighbour filter to a cache. This cache is consulted during the border node selection process. If a message is not received from a neighbour for a configured length of time, its identifier and filter are removed from the cache.

This approach allows SDRP-RMPR to selectively rebroadcast packets without requiring periodic HELLO message transmissions. Service advertisements are rebroadcast more frequently by highly mobile nodes and less frequently by stable nodes outside the MPR set, allowing the protocol to adapt better to mobility while keeping overheads low. The next section will describe the border node selection algorithm used by SDRP-RMPR to select advertisement rebroadcasters.

3.6.3 Border Node Selection

Cartigny and Simplot’s border node selection algorithm involves comparison between a broadcast source’s neighbour set and those of its neighbours, with distance to each neighbour then being estimated based on the degree of intersection between sets. Replicating this algorithm using variable-length bloom filters allows probable border nodes to be selected for message rebroadcasts while reducing message size when compared to direct serialization of neighbour addresses.

Service advertisement rebroadcasts by neighbouring nodes allow an advertisement source to build a cache of neighbour identifiers and their neighbour filters. The received neighbour filters vary in length depending on the number of neighbours within range of the message source. In order to determine the distance between the selecting node and a neighbour, the selector constructs a bloom filter with the same parameters as that of the neighbour’s neighbour filter. The selector then inserts its own identifier and the identifiers of all of its own neighbours, except for the neighbour it’s comparing against, into the constructed filter.

The filters are then compared; the number of bits in the subject’s neighbour filter that are set, which do not have corresponding set bits in the selector’s filter, are counted. This count indicates the number of 1-hop neighbours within range of the subject that are not shared with the selector. The degree of difference is then defined as the ratio of unshared set bits to the size of the filter. A greater degree of difference makes a subject more likely to represent a border node, while a lower degree of difference indicates closer proximity to the selector. This process is illustrated in figure 3.15.

The degree of difference metrics for all neighbours are calculated and the mean of all differences is then used as the lower-limit for border node selection. Neighbours with a degree of difference that is greater than or equal to this limit are selected as MPRs and their identifiers are inserted into the service advertisement’s MPR filter, as described in Algorithm 5.

3.6.4 Example

An example network topology composed of 9 nodes will be used to illustrate the operation of SDRP-RMPR, as illustrated in figure 3.16. Node A is a service agent, with the remaining nodes in the network acting as user agents.

At periods of $t_{advertise}$ seconds, node A broadcasts service advertisement messages. Its neighbouring nodes will rebroadcast that message under two circumstances: if their identifier is not contained within the advertisement’s neighbour filter or if their identifier is contained in the MPR filter. Initially, none of the nodes have knowledge of the network topology, thus node A’s service advertisement will contain an empty neighbour filter. Upon receipt of this advertisement,

Input: Current Neighbour Node Data
Output: Multi-Point Relay Bloom Filter

```

1 differences  $\leftarrow$  map of floats to node identifiers;
2 mprNode  $\leftarrow$  bloom filter
3 foreach Node n in Neighbour Node Data do
4   /* Construct a filter with matching properties to those of the currently
      evaluated neighbour */
5   matchingFilter  $\leftarrow$  bloom filter;
6   matchingFilter.set_hash_count( n.neighbourFilter.hashCount );
7   matchingFilter.set_length( n.neighbourFilter.length );
8   /* Insert the identifiers of all of the local node's neighbours, apart from
      that of n */
9   foreach Node o in Neighbour Node Data do
10    if o.identifier  $\neq$  n.identifier then
11      | matchingFilter.insert( o.identifier );
12    end
13  end
14  difference  $\leftarrow$  num_unshared_bits( matchingFilter, n.neighbourFilter );
15  differences.insert( difference );
16 end
17 /* Get the mean of all calculated differences */
18 mean  $\leftarrow$  get_mean( differences );
19 foreach Node Identifier i, Difference d in differences do
20   if d greater than or equal to mean then
21     | mprNodes.insert( i );
22   end
23 end
24 return mprNodes

```

Algorithm 5: SDRP RMPR Border Node Selection Algorithm

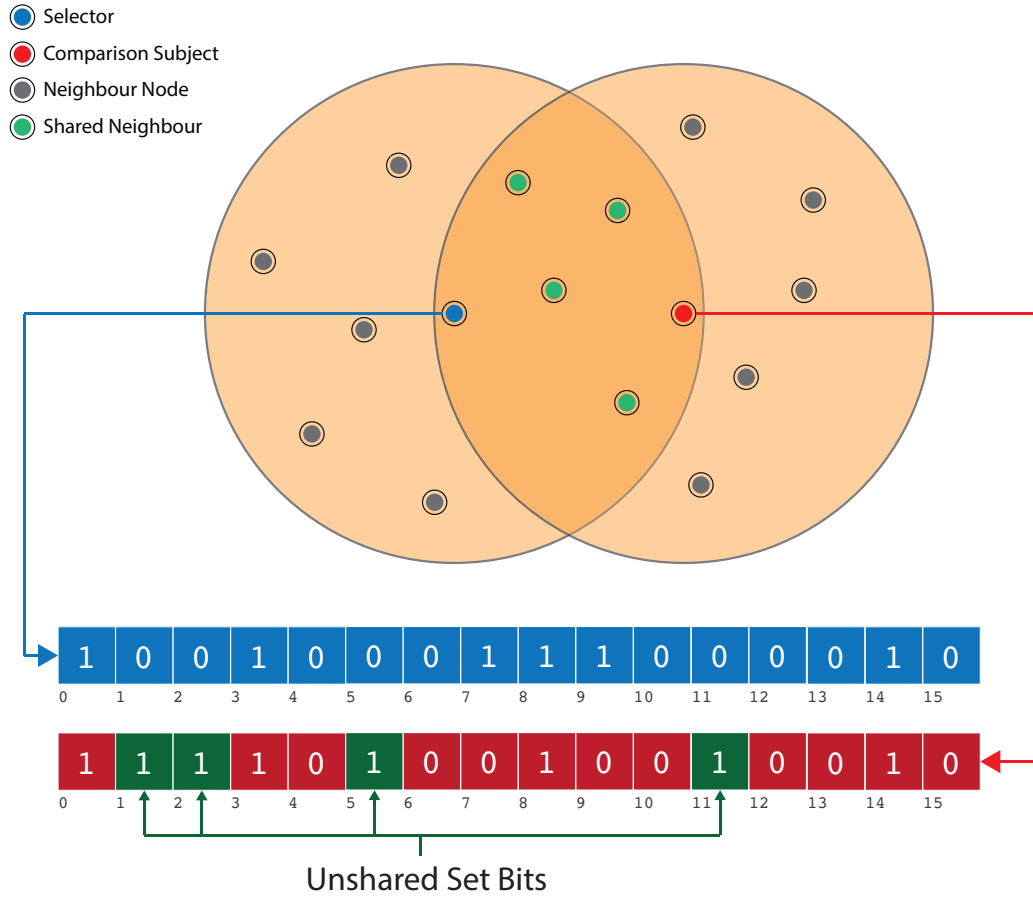


Fig. 3.15: Border nodes are selected through neighbour filter comparison. The selector constructs a bloom filter with identical parameters to those of the subject's neighbour filter, inserts its own neighbours' identifiers, then compares the filters. If the subject's filter has a large number of set bits that are not shared with the selector, this indicates a large number of unshared neighbours and, thus, greater distance from the selector.

all neighbouring nodes will rebroadcast the message. The message will be effectively flooded throughout the network, as shown in figure 3.17.

Node A now has awareness of its immediate neighbours, however their previous advertisement rebroadcast did not contain the identifiers of their own neighbours. When node A attempts to select border nodes, it will calculate the degree of difference between its neighbour filters and those of nodes B, C, F, G and H. As those nodes have empty neighbour filters, the degree of difference metric will always be the same and all nodes will subsequently be chosen as MPR

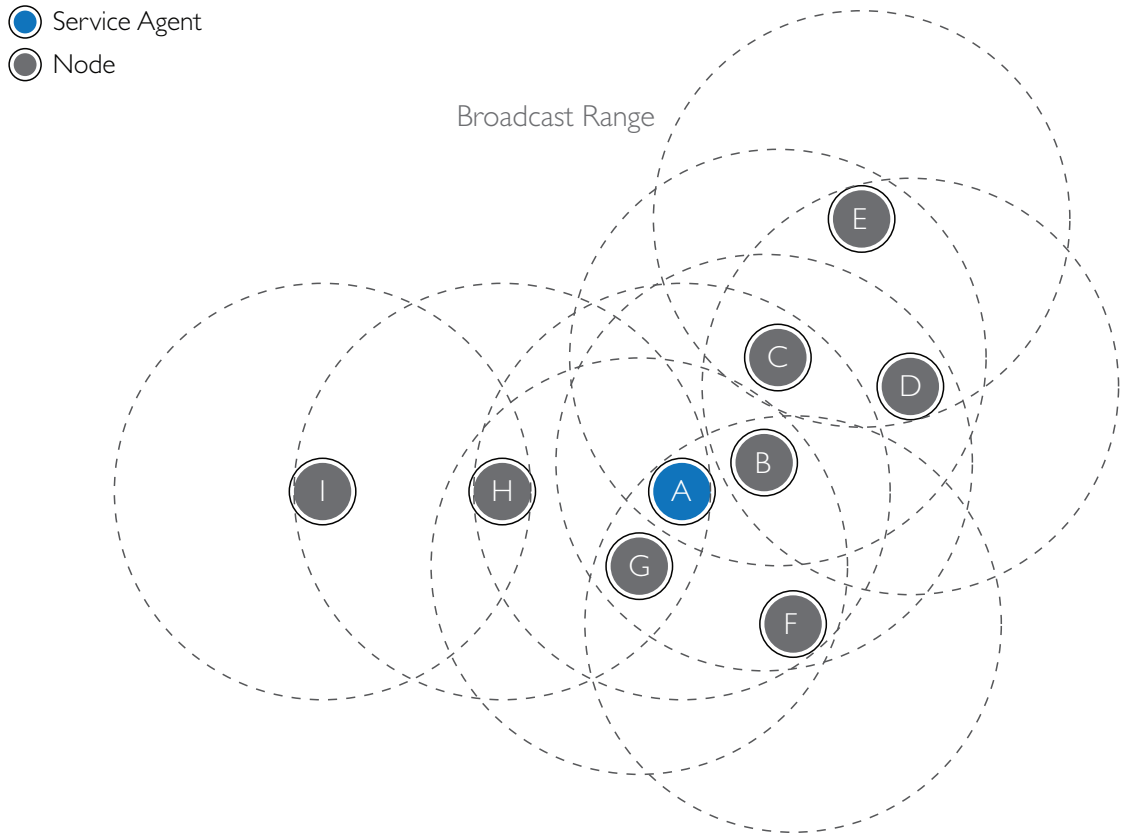


Fig. 3.16: In the example network topology, nodes B, C, F, G and H are first-hop neighbours of node A, with nodes D, E and I acting as second-hop neighbours.

nodes; their degree of difference being equal to the mean.

Node A broadcasts a service advertisement message with an MPR filter containing the identifiers of its immediate neighbours. The message is further rebroadcast, as shown in figure 3.18. Node A receives these rebroadcasts, which include updated neighbour filters for its neighbouring nodes. These filters contain the identifiers of node A's second-hop neighbours, allowing it to refine its border node selection when broadcasting the next service advertisement message.

Node A determines that the degree of difference between its neighbour filter and those of nodes H and C is greater than that of the other neighbouring nodes. Nodes H and C are then chosen as MPR nodes for the next service advertisement broadcast. Upon receipt of the message, node A's neighbours will be included in the neighbour filter but only the identifiers of nodes H and C will be in the MPR filter, thus only those nodes will rebroadcast the message as shown in

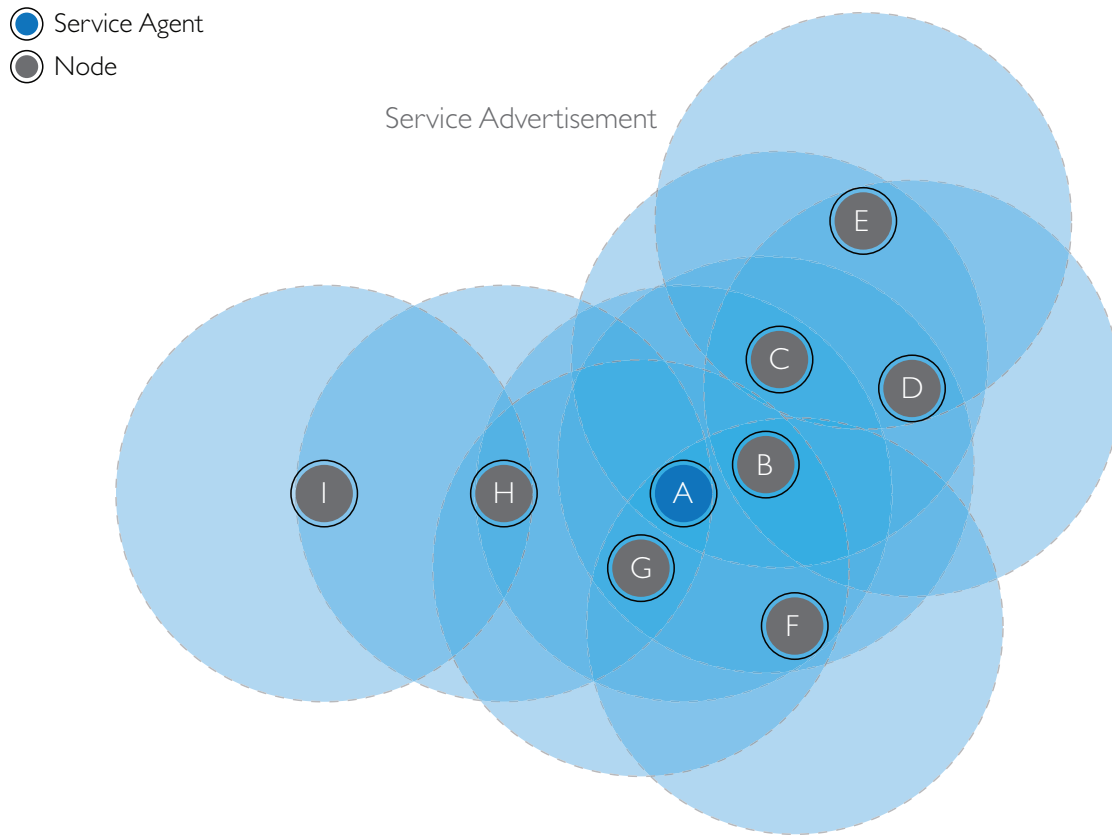


Fig. 3.17: After a period of $t_{advertise}$ seconds, node A will broadcast a service advertisement message containing an empty neighbour and MPR filter. Receiving nodes will rebroadcast this message, as their identifiers cannot be found in the neighbour filter. All nodes are now aware of their immediate neighbourhood.

figure 3.19.

Eventually, node A will remove the identifiers of nodes B, F and G from its neighbour filter, having not received a message from them for a configured period of time. The next service advertisement message will be rebroadcast by those nodes, as their identifiers cannot be found in the message's neighbour filter; allowing node A to refresh its knowledge of the local area.

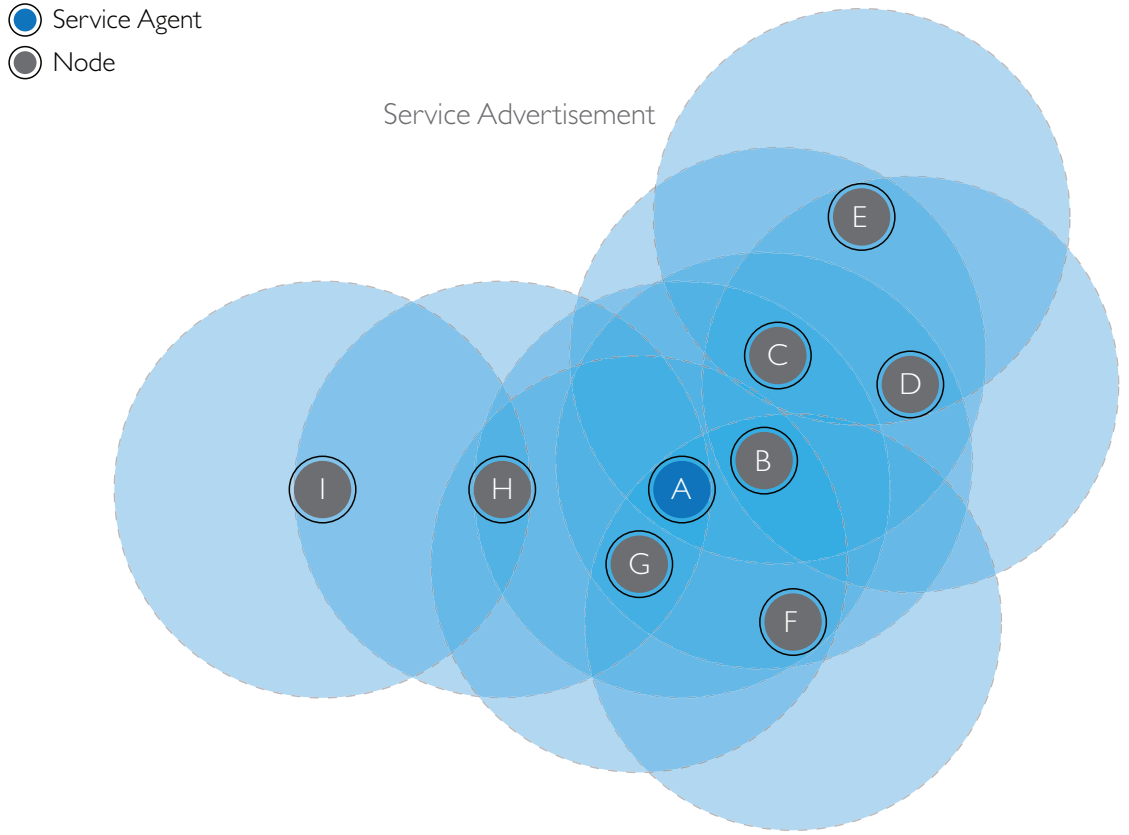


Fig. 3.18: After another $t_{advertise}$ seconds, node A will broadcast another service advertisement message. The neighbour filter for this message will contain the identifiers of nodes B, C, F, G and H. Node A will choose all of its immediate neighbours as MPR nodes, as the border node selection algorithm will calculate similar degrees of difference between node A's neighbour filter and those of its neighbours. When its neighbouring nodes rebroadcast the advertisement, node A will receive an updated neighbour filter, allowing it to refine its MPR selection.

3.6.5 Summary - SDRP-RMPR

SDRP-RMPR further reduces control overheads by removing the reliance on periodic HELLO message broadcasts by all nodes and incorporating the use of variable-length bloom filters for neighbour link-state description. Neighbour filters are inserted into service advertisement messages and rebroadcast by receiving nodes if their identifier is not present in the neighbour filter or if their identifier is present is the MPR filter. This approach results in more frequent rebroadcasts by highly mobile nodes and more infrequent broadcasts by immobile, non-MPR nodes; maintaining rebroadcaster selection reliability while keeping control overheads low.

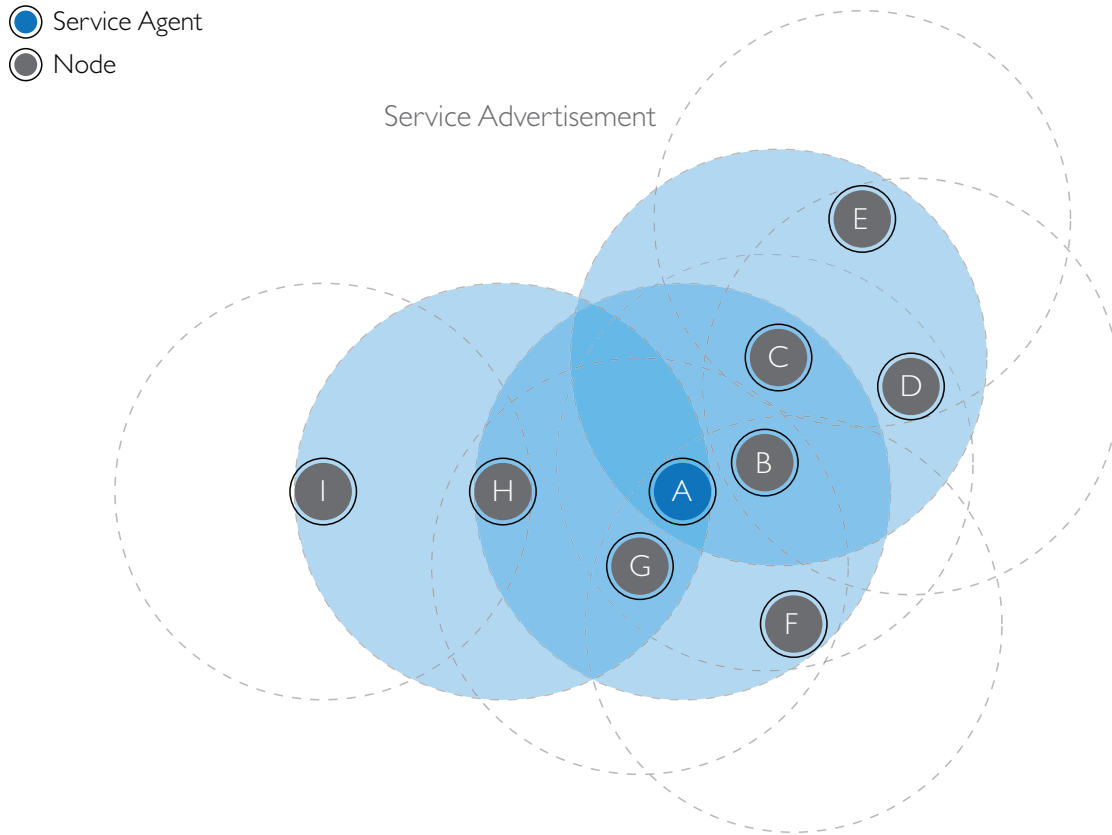


Fig. 3.19: Node A now has sufficient information to choose MPR nodes. The degree of difference between its neighbour filter and those of nodes C and H is greater than that of its remaining neighbours, as nodes C and H are within range of one or more two-hop neighbours. The identifiers of nodes C and H are inserted into the MPR filter and only they rebroadcast this service advertisement.

MPR selection diverges from the approach taken by SDRP-MPR and OLSR, instead more closely resembling the border node selection approach used in the protocol described by Cartigny and Simplot. This approach works by determining the distance between a message source and its neighbours and selecting those neighbours closer to the edge of the source's transmission range to rebroadcast messages under the assumption that rebroadcasts by nearby nodes would be redundant.

SDRP-RMPR advertisement broadcasters select the nodes that should rebroadcast their packets using neighbour filter comparison. More distant nodes tend to have a high degree of difference between their neighbour filter and that of the broadcaster, while nearby nodes have

a low degree of difference. By calculating this metric for all current neighbours, SDRP-RMPR broadcasters probabilistically select nodes nearer to the edge of their transmission range to rebroadcast service advertisements.

3.7 Summary

The three versions of SDRP described in this section are designed to explore particular aspects of the solution described by this thesis; service discovery and routing without reliance on an underlying routing protocol and low-overhead selective message rebroadcasting using bloom filters for link-state description for the purpose of improving network scalability when compared with existing approaches.

SDRP-Flooding attempts to provide basic service discovery and routing without reliance on an underlying routing protocol. Its service advertisement approach is simple and reliable when compared with the more complex approaches explored in section 2.1.1, however in networks containing many service agents it is likely to generate excessive overheads. SDRP-F adopts the service description approach used in the Mercury service discovery protocol and further refines it through the use of variable-length bloom filters, reducing control overheads when compared with protocols using verbose service descriptors.

SDRP Multi-Point Relay is designed to address the problem of redundant rebroadcasting inherent in SDRP-F's naive flooding mechanism by adopting a modified version of OLSR's multi-point relay message rebroadcast algorithm. The MPR algorithm selects only those neighbours required to reach all 2-hop neighbours as message rebroadcasters. This approach reduces control overheads by eliminating redundant rebroadcasts at the cost of requiring that all nodes periodically broadcast HELLO messages containing their link-state with their immediate neighbours.

SDRP-MPR modifies this approach to use a more compact link-state descriptor based on bloom filters in order to further reduce overheads. However, the requirement that static-length bloom filters be used in order to facilitate MPR selection using set operations reduces the ability for SDRP-MPR to adapt successfully to significant changes in network topology. Thus while SDRP-MPR may reduce control overheads when compared to OLSR's MPR approach under ideal conditions, it may perform badly if network conditions diverges significantly from those assumed prior to deployment.

SDRP-Reduced MPR includes a further refinement on the rebroadcaster selection algorithm used in SDRP-MPR that more closely resembles the border-casting algorithms seen in cluster-based and probabilistic routing protocols. SDRP-RMPR also eschews the use of periodic HELLO messages and static-length bloom filters for link-state description, instead describ-

ing neighbour link-state using variable-length filters added to service advertisement messages. SDRP-RMPR is designed to better adapt to changing network conditions than SDRP-MPR while further reducing control overheads.

Protocol	Advertisement Mechanism	Advantages	Disadvantages
SDRP Flooding	Naive Flooding	• Reduced control overheads generation through removal of address-based routing • Compact service description using bloom filters • Proactive advertisement reduces end-to-end delay compared with query and directory approaches	• Flooding mechanism causes rapidly increasing overheads with increasing service agent density • Reverse route approach relies on frequent application traffic • Service description using bloom filters introduces possibility for false positives
SDRP Multi-Point Relay	Selective Rebroadcasting	• Further reduction in control overheads through selective rebroadcasting, allowing for greater service agent density than flooding approach	• Potential for errors in MPR selection due to use of bloom filters for neighbour link-state description, particularly when number of neighbours exceeds neighbour filter capacity
SDRP Reduced MPR	Probabilistic Rebroadcasting	• Removal of HELLO messages further reduces control overheads • Use of variable-length neighbour filters allows for better adaptation to changes in network density	• Probabilistic approach increases probability that parts of the network will not be reached by service advertisements

Fig. 3.20: A summary of the different SDRP versions.

A summary of the described SDRP versions, their advantages and their disadvantages is illustrated in figure 3.20. In the next section, the implementation of the three SDRP versions will be described.

Chapter 4

Implementation of SDRP

This chapter describes the implementation of the Service Discovery and Routing Protocol. SDRP has been developed in three iterations; SDRP-F, SDRP-MPR and SDRP-RMPR, as outlined in the previous chapter. The working process of each version of SDRP can be broken down into three distinct parts: service registration, service advertisement and routing between user agents and service agents. SDRP has been developed as a shared library in order to facilitate installation both into simulators and onto devices such as laptops or mobile phones, should the opportunity arise.

The widespread use of NS-2 in the MANET research community means that a number of the service discovery and routing protocols described in Chapter 2 are available as NS-2 agents, allowing us to compare them with SDRP. However, not all of the protocols that we would like to compare SDRP against have been implemented in NS-2. SMF, for instance, an example of a cross-layer reactive protocol, was implemented in GloMoSim while LSD, a cross-layer directory-based protocol, was implemented in OPNET. As part of SDRP's evaluation, both of these protocols were ported to NS-2.

This chapter begins with a description of SDRP's implementation as a shared library and its interfaces. The specific implementation of key components of the protocol, including bloom filters, MPR selection algorithms, and routing management system will then be discussed. Next, the NS-2 network simulator, the simulator platform used to evaluate SDRP and related approaches, is described and details of how it was extended to allow for a more thorough evaluation will be given. The chapter will finish with a description of the implementation of a number of service discovery protocols in NS-2 for the purpose of comparison against SDRP and a summary of the chapter.

4.1 SDRP

As outlined in the introduction to this chapter, SDRP has been developed as a shared library rather than a directly integrated NS-2 Agent. This design choice allows SDRP to be deployed on device such as laptops and mobile phones in the future, should the opportunity arise, as well allowing it to be evaluated using simulators such as NS-2.

4.1.1 Core Classes

Figure 4.1 outlines the core classes that make up the SDRP library implementation. SDRP exposes a control interface to user classes through the *SDRP* class interface. Platform-specific functions such as packet sending and time acquisition are handled by the *SDRPDelegate* interface; this interface is implemented by user classes and a delegate object is passed to the node SDRP instance when it is constructed.

Node is a model class that describes a node in terms of its address, provided services and its neighbours. A single instance of the *Node* class is used to describe the local node and is shared among several of the core classes; most notably the SDRP object and Routing Manager.

RoutingManager is an important class used to handle incoming packets, send protocol messages and provide routes to services and hosts. Route information is stored in a *RouteTable* instance in the form of *Route* objects. Route objects describe an endpoint, its services, the next hop towards that endpoint, the distance to the endpoint in terms of number of hops and the last time the route was refreshed. The *RouteTable* can purge stale routes based on their last update timestamp and contains methods for searching the table for particular services and nodes.

LocalAreaMonitor tracks the local neighbourhood, providing the ability to purge stale neighbour entries and calculate an MPR set based on the current MPR selection mode; flooding, MPR and reduced MPR.

MPR set calculation is performed by classes implementing the *MPRCalculator* interface; the *MPRFactory* class is responsible for constructing a calculator based on the configured MPR selection mode. This calculator instance is used by the *LocalAreaMonitor* instance to calculate the MPR set.

Together with a number of helper classes, these core classes make up the SDRP library implementation. In the following sections, we will describe the functions and interactions between these classes in greater detail.

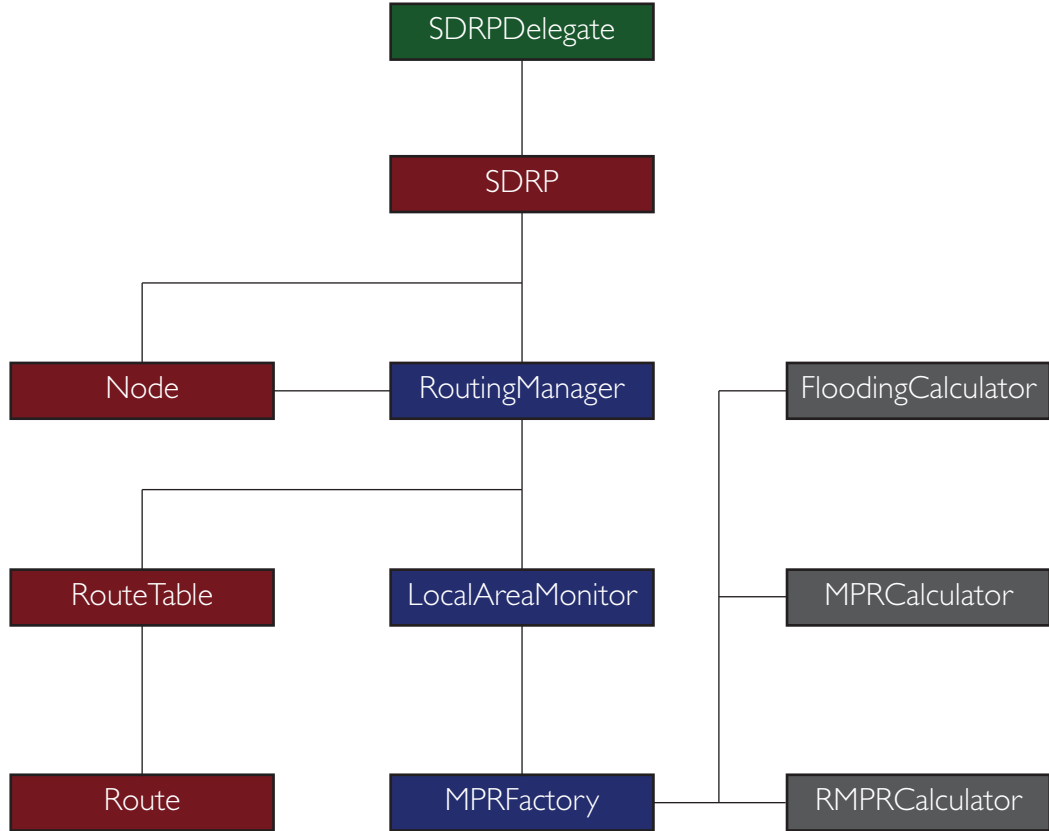


Fig. 4.1: SDRP Library Core Classes.

4.1.2 SDRP and Delegate Interface

SDRP exposes a simple interface that allows it to receive packets as well as provide information about available services and routes, as shown in figure 4.2. It should be noted that SDRP uses typedefs to ensure cross-platform compatibility. Thus, types such as *RDNetworkAddress*, *RDTimeStamp* and *RDTimeStamp* will often resolve to primitive types such as floats and doubles, depending on the deployment platform.

The **LocalNode** method is used to retrieve the Node object used by SDRP to represent the local node. A reference to this object is held by various helper classes within the SDRP library; any changes to the object are immediately propagated and used. Users can control the local node's address and its services through the local node object.

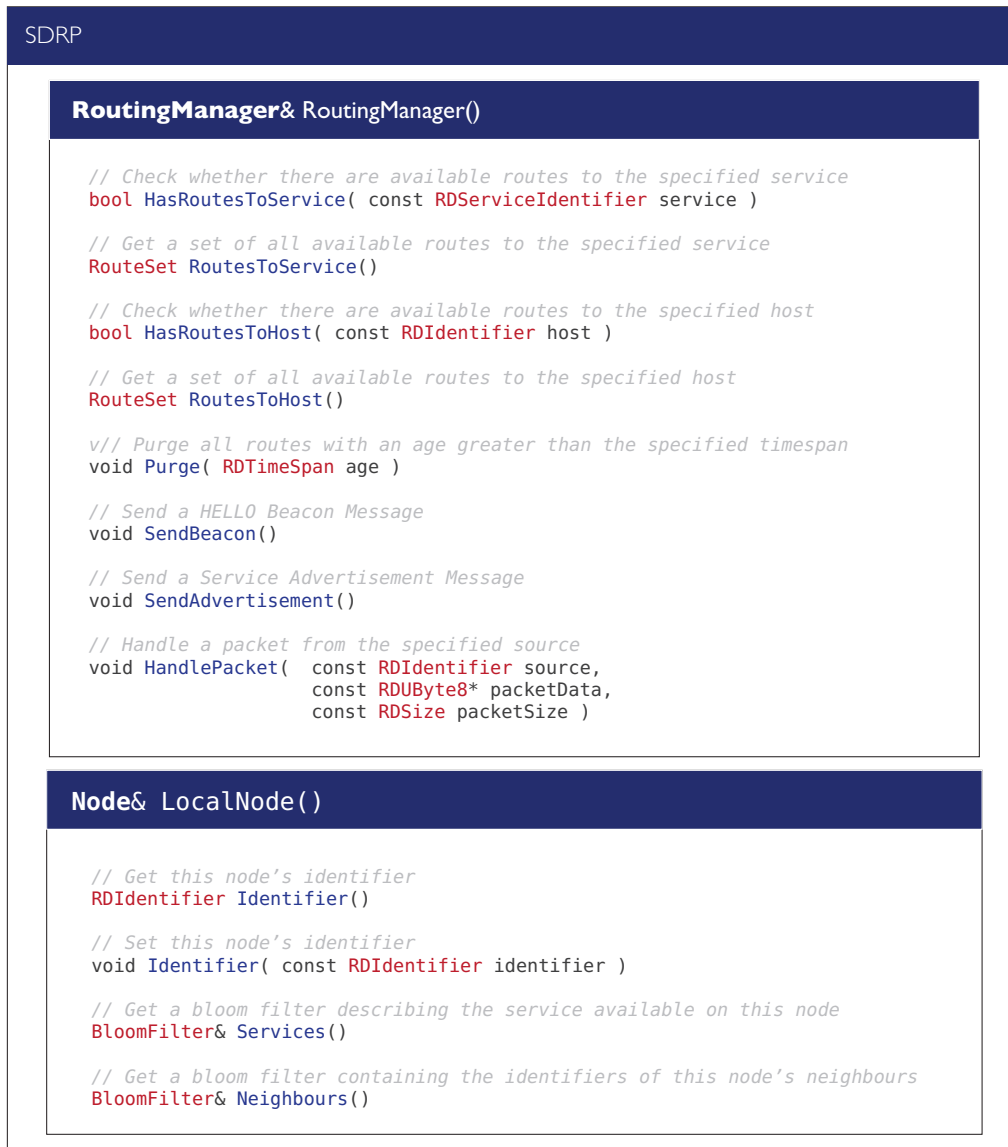


Fig. 4.2: SDRP's interface exposes two key objects; A Node object referring to the local node that allows its network address and services to be controlled and a routing manager that handles incoming packets and provides routes to services and hosts.

The **RoutingManager** method allows user classes to get a reference to the node's *RoutingManager* object. Through this object reference, routes to specific hosts or services can be requested and packets can be passed to SDRP for handling.

SDRP sends packets through a delegate interface that is implemented on a per-platform basis. In the case of the NS-2 simulator, the SDRP Agent class implements the interface and translates SDRP packet data into NS-2 *Packet* objects before sending them. As SDRP does not implement the functionality of a full address-based routing protocol, the network address specified by the *Send* method will typically indicate the identifier of an immediate neighbour of the local node that can be reached through a single message unicast.

```

SDRPDelegate

// Send the provided packet to the specified destination
virtual void Send( const RDByte8* packetData,
                  const RDSIZE packetSize,
                  const RDIdentifier destination )

// Get the current time
virtual RDTimeStamp Time();

// Get a human-readable name for this delegate, for logging purposes
virtual std::string& Name();

```

Fig. 4.3: The SDRP Delegate interface is used by SDRP to send packets and get the current time. Implementing this interface on a per-platform basis allows SDRP to be deployed on simulators, real nodes and a variety of operating systems.

The methods exposed by the SDRPDelegate interface are described in figure 4.3. The interface includes other methods that allow SDRP to get the current time so that stale route entries can be purged. This method is considered platform-dependent; a simulator will provide simulator time while a real-world device will provide the current time.

4.1.3 Routing Manager and Route Table

The *RoutingManager* acts as the core of SDRP, sending and receiving packets and acting as the interface for the service routing table. *RoutingManager* holds instances of two important classes; *RouteTable* and *LocalAreaMonitor*. *RouteTable* contains a set of *Route* objects; containers for route information. *Route* objects are constructed by the *RoutingManager* upon receipt of *ServiceAdvertisement* messages and stored in the *RouteTable* instance.

The *RoutingManager* instance receives incoming protocol messages through the **HandlePacket** method. SDRP expects packets to be provided in the form of a byte array with an additional integer parameter indicating the size of the packet and a network address indicating

the packet source. This format is typical for socket communications and the same data can be extracted from NS-2 packet objects for use by the library.

The initial type byte of a message is used to determine the message type; the two message types, Beacon and ServiceAdvertisement, use type values 0x00 and 0x01 respectively. Beacon is an alias for HELLO messages and will be used when describing messages of that type in this section. *RoutingManager* deserializes the initial type byte and then constructs a message object of the appropriate type. Message classes expose **Serialize** and **Deserialize** methods that allow message data to be inserted or extracted from packet buffers.

All serialization and deserialization methods use a utility class called *Serializer* to perform their operations. *Serializer* exposes a number of static methods that allow primitive types to be inserted into byte buffers. A number of specialized methods are used to perform serialization and deserialization on integer types. At runtime, the *Serializer* checks the endian-ness of the system and will perform endian-swapping operations on integers being serialized and deserialized if the system does not use network byte-order.

RoutingManager calls **Deserialize** on the message object, providing the packet byte array and its length. Message deserialization may fail if the packet length passed to the **deserialize** method is insufficient to contain all of the child objects expected. A *ServiceAdvertisement* object, for instance, may contain up to 3 bloom filters and a number of integers. Each child object is deserialized in turn, first checking that the packet data remaining in the buffer is of sufficient length to contain the object. If deserialization reaches the end of the buffer before all child objects have been read, the method will return false, indicating failure.

Deserialized Beacon and *ServiceAdvertisement* messages are used to update the *RoutingManager*'s *RouteTable* object entries as well as the information contained in the *LocalAreaMonitor*. Beacon messages contain a bloom filter indicating the link-state between the source and its neighbours, this data is passed to the *LocalAreaMonitor* for use in MPR calculations. *ServiceAdvertisement* content indicates the route to a service agent; this information is stored in the *RouteTable*.

The *RoutingManager* is responsible for rebroadcasting *ServiceAdvertisements* through the *SDRPDelegate* object's **Send** method. Received messages first have their sequence numbers checked against a map that associates integers with node identifier. The identifier is that of the advertisement source while the integer represents the last highest sequence number received from that source. If the advertisement sequence number is less than or equal to the sequence number associated with the source identifier, the message is discarded. Otherwise, the sequence number associated with the source is updated with the new value. The message is then rebroadcast if any of the following conditions are met:

- MPR Mode is set to Flooding
- MPR Mode is set to MPR and the advertisement MPR filter contains this node's identifier
- MPR mode is set to ReducedMPR, the advertisement MPR filter contains this node's identifier or the advertisement neighbour filter does not contain this node's identifier.

RoutingManager exposes the **RoutesToService** and **RoutesToHost** methods, allowing users to get a set of all available routes to service agents and hosts. These Route sets are retrieved from the RouteTable object's Route container. Routes indicate the next hop towards a node, the identifier for the route endpoint, the service filter advertised by that node and a timestamp indicating when the route was last updated. *RoutingManager* is periodically instructed by user classes to purge all routes with an age over some limit; ensuring that stale routes are not used. RouteTable is responsible for iterating over stored Route objects and removing them if their timestamps indicate they are stale.

RoutingManager also exposes the **RegisterConnection** method; this method is used to register the connection between a user agent and service agent so that a reverse route back to the user agent can be retrieved at a later time. The Route object constructed by the method is stored in a separate RouteTable instance used specifically for reverse route storage and is consulted when **RoutesToHost** is called.

The Route sets retrieved using **RoutesToService** and **RoutesToHost** are sorted in order of increasing distance to the Route endpoint. User classes may use whichever Route they deem suitable to send service request and response packets.

The **SendBeacon** and **SendAdvertisement** methods request that SDRP send a Beacon message or Service Advertisement. These methods are exposed so that user classes have full control over the frequency at which these messages are sent. Timer management is different for real nodes and simulators, as simulators have the concept of simulation time, thus it was decided that internal control over message sending frequency should not be retained. SDRP may or may not send Beacon messages and Service Advertisements, depending on its operational mode (Flooding, MPR, Reduced MPR) and whether the local node has services registered in its service filter.

Beacon messages are only sent in MPR mode; in this mode, Beacon messages contain the information necessary to calculate MPR sets. In Reduced-MPR mode, the same information is contained in a variable-length that's inserted into ServiceAdvertisement messages. Flooding mode involves no MPR calculation, thus a node in Flooding mode will not send Beacon messages.

As mentioned previously, message classes expose methods allowing them to be serialized and deserialized into and out of byte buffers. The **Serialize** method is used to insert

4.1.4 Local Area Monitor and MPR Calculation

The *LocalAreaMonitor* is responsible for storing a list of current neighbour nodes and calculating an MPR set based on the current *MPR Mode*. The neighbour set is an STL set (Stepanov, 1994) of Node objects that is updated when the *RoutingManager* receives Beacon or ServiceAdvertisement messages, or the *LocalAreaMonitor* is instructed to purge all Node objects with a last refresh time over a specified limit. The *LocalAreaMonitor* contains an MPR set calculator object, the true type of which is determined by the current MPR mode, as configured through the *RoutingManager*.

The MPR Mode can be set to *Flooding*, *MPR* or *Reduced MPR*, depending on which version of SDRP is to be used. The MPR mode setting is passed to an internal object, *MPRFactory*, which constructs a class implementing the *IMPRCalculator* interface and returns it to the *LocalAreaMonitor*.

The three available MPR calculator classes are *FloodingCalculator*, *MPRCalculator* and *ReducedMPRCalculator*; acting as MPR calculators for the three available MPR selection modes. This design pattern is known as the *Factory* pattern; a factory class can construct an object that conforms to a particular interface without the client class knowing the exact type being constructed. This allows SDRP to be extended to include new MPR selection algorithms without requiring that other sections of the codebase be edited.

```
IMPRCalculator

// Construct an MPR Calculator, providing a Node object representing the
// local node
IMPRCalculator( const Node& localNode )

// Get a set of all available routes to the specified service
virtual BloomFilter Calculate( const std::set<Node>& neighbours )
```

Fig. 4.4: The MPR Calculator interface is implemented by MPR calculator classes representing the three MPR selection algorithms; Flooding, MPR and Reduced MPR. Using the Factory design pattern, SDRP can be extended to use new MPR selection algorithms by adding a new calculator subclass.

The IMPRCalculator interface is described in figure 4.4. It contains only a constructor taking a Node instance representing the local node as a parameter, and a method that returns a bloom filter containing the identifiers of all nodes that should rebroadcast ServiceAdvertisement messages based on the set of neighbour Node instances passed to the method.

4.1.4.1 Flooding Calculator

The FloodingCalculator is a simple implementation of the calculator interface. It simply returns a bloom filter with m equal to 1 and all bits set to 1. A bloom filter with all of its bits set to 1 essentially represent a universe of elements; any query for membership against such a filter will succeed.

When this 'universal' bloom filter is received by peer nodes, it will appear as though all of them will have been selected as MPRs by the sender; evoking a message flooding pattern. Thus, a service advertisement message sent by SDRP in Flooding mode contains an MPR filter, albeit a small one. This approach was taken so that additional development effort would not be required to write separate advertisement message types for each MPR selection mode. The service advertisement message also includes a neighbour filter, as specified in the description of SDRP-RMPR in chapter 3; the filter is also set to a universal value and ignored by the *RoutingManager*.

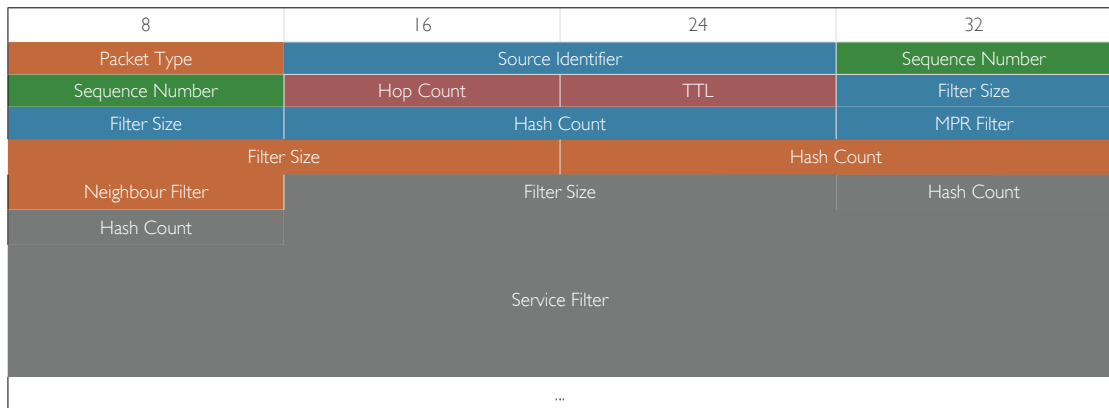


Fig. 4.5: The implemented packet layout for an SDRP-F service advertisement includes an MPR filter and neighbour filter set to the 'universal' value; adding 10 unused bytes to the packet length. All MPR selection modes share the same packet format, but differ in terms of the filter content at runtime.

This approach adds an additional 10 bytes to the service advertisement packet layout described in chapter 3; 2 bytes to specify the size of each additional filter for deserialization, 2 bytes for the filter hash count and a single byte for the filter itself. The actual packet layout for the SDRP-Flooding service advertisement is shown in figure 4.5.

4.1.4.2 MPR Calculator

The MPR calculator determines the identities of MPR nodes using an algorithm similar to that of OLSR. The **Calculate** method expects all of the Node objects passed to it to have neighbour filters with the same length, this is checked at runtime and calculation will fail with a thrown *BloomFilterSizeMismatch* exception. The neighbour filters contained in the Node objects are set by the *LocalAreaMonitor* upon receipt of a Beacon message by the *RoutingManager*.

The MPR calculator forms empty bloom filters representing the required coverage for the MPR filter and the current coverage based on the MPR nodes selected thus far. The required coverage filter is filled using the neighbour filters of all neighbouring nodes. This algorithm is illustrated in figure 4.6.

There is a non-zero probability, determined by the filter parameters configured at deployment time for Beacon messages and the number of identifiers inserted, that an identifier will be falsely determined to be contained in a node's neighbour filter. The worst-case result of this would be the potential removal of one or more second-hop neighbour identifiers from the neighbour filter, adversely affecting the algorithm's ability to accurately select MPR nodes. The false-positive probability can be reduced at deployment time by increasing the filter size used to indicate neighbour link-state in Beacon messages, however this comes at the cost of higher control overheads.

Required Coverage Algorithm

```
// Construct an empty bloom filter representing the required second-hop coverage area
BloomFilter requiredCoverage;

// Iterate over all neighbour nodes
for( std::set<Node>::iterator i = neighbours.begin(); i != neighbours.end(); i++ )
{
    // Insert the temporary filter, this is effectively a union operation
    requiredCoverage.Insert( i->Neighbours() );
}
```

Fig. 4.6: The required coverage in terms of second-hop neighbour identifiers is calculated by iterating over all nodes and filtering the identifiers of immediate neighbours from their neighbour filters before inserting the resulting filter into a coverage filter.

It should be noted that BloomFilter objects are, in actuality, counting filters. This means that filter positions can hold integer values. The required coverage filter calculated using the above algorithm is a counting filter, thus any second-hop neighbour identifiers that are contained in multiple neighbour filters will have their corresponding filter offset incremented a corresponding number of times.

This aspect of counting filters is used in the next step of the MPR algorithm to determine nodes with unique connections to second-hop neighbours. The BloomFilter class has a utility method, **RemovalResultsInDifference** which, given a value, will determined whether decrementing the offsets produced by hashing that value will result in one or more filter offsets being set to zero; indicating that the value had been inserted into the filter only once.

The temporary filters generated previously representing each node's coverage of second-hop neighbours are tested against the required coverage filter. Any filter whose removal results in a zero filter index is determined to contain an identifier that is not contained in any other filter. The Node object to which the coverage filter belongs is designated as an MPR due to its unique connection.

The remaining Node objects are then sorted in order of their coverage area - determined by counting the number of set bits in their neighbour filters - and added to the MPR set. Their neighbour filters are also added to the coverage filter. Once the coverage filter matches the required coverage in terms of set filter offsets, MPR selection is complete. A bloom filter of sufficient size to contain the identifiers of the selected MPR nodes' identifiers is constructed, with the filter target false-positive rate being pre-configured, and the identifiers of all MPR nodes are inserted into it before being returned to the *LocalAreaMonitor*.

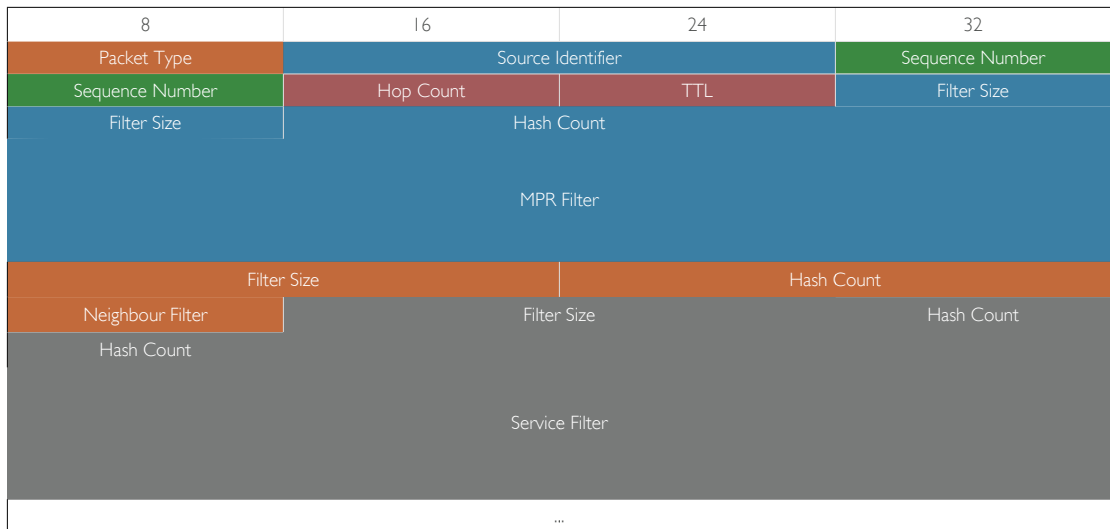


Fig. 4.7: The implemented packet layout for an SDRP-MPR service advertisement includes an MPR filter containing the identifiers of selected MPR nodes. It also contains an unused neighbour filter, adding 5 unused bytes to the packet's length.

The MPR filter is passed to the *RoutingManager* which then inserts it into a ServiceAd-

vertisement message. The serialized form of the ServiceAdvertisement is shown in figure 4.7. As mentioned previously, the ServiceAdvertisement message contains redundant elements in some modes; in this case, the neighbour filter is unused as this is instead broadcast using Beacon messages when SDRP is in MPR mode.

4.1.4.3 Reduced MPR Calculator

The reduced MPR calculator selects a subset of the neighbour nodes passed to it as MPR nodes based on their probable distance from the selector and from each other. The algorithm works over a number of stages; the first involves elimination of nodes from the possibility set based on their similarity to each other. This is accomplished by iterating over all neighbour nodes and comparing their neighbour filters; if their contents are similar, based on a limit configured at deployment time, the filter with the greater number of set bits is retained and the match is eliminated.

The reduced possibility set is iterated over. For each Node in the set, a bloom filter with the same parameters as the Node's neighbour filter is constructed. That filter then has the identifiers of all of the local node's neighbour nodes inserted into it. At this point, we now have two neighbour filters that can be compared for similarity due to the fact that they have the same hash count and length; the one belonging to the neighbour against which we are comparing the local node, and the filter constructed with similar parameters containing all one-hop neighbours of the local node.

The two filters are compared by counting the number of filter indices that do not match, as described in Chapter 3, figure 3.15 and illustrated in figure 4.8. This difference is then associated with the subject neighbour Node. Nodes are selected based on their calculated difference metric, with a greater difference metric indicating greater distance from the selector and/or a larger number of uncommon neighbours. The limit used to further filter nodes based on difference is configurable at runtime. In the current implementation of SDRP-RMPR, the mean of all of the difference is calculated and all nodes with a difference greater than the mean are chosen as MPR nodes.

Similarly to the MPR calculator, a variable-length bloom filter of sufficient length to contain the selected MPR nodes is constructed, and the identifiers of the selected nodes are inserted into it. The MPR filter is returned to the *LocalAreaMonitor*, which in turn passes it to the *RoutingManager* for insertion into the outgoing ServiceAdvertisement message. The *LocalAreaMonitor* also composes the identifiers of all current one-hop neighbour nodes into a bloom filter for insertion into the ServiceAdvertisement message. Unlike in MPR mode, the neighbour filter length used by RMPR nodes can be constructed to fit the number of neighbours

Difference Calculation Algorithm

```
// Construct a map associating filter differences with nodes
std::map<int, Node> differences;

// Iterate over all neighbour nodes
for( std::set<Node>::iterator i = neighbours.begin(); i != neighbours.end(); i++ )
{
    // Construct a filter with identical properties to the current neighbours'
    BloomFilter comparisonFilter( i->Neighbours().FilterLength(),
                                i->Neighbours().HashCount() );

    // Fill the comparison filter with the local node's neighbour identifiers
    for( std::set<Node>::iterator j = neighbours.begin(); j != neighbours.end(); j++ )
    {
        // Don't insert the identifier of the subject node
        if( i->Identifier() == j->Identifier() )
        {
            continue;
        }
        comparisonFilter.Insert( j->Identifier() );
    }

    // Calculate the divergence between the comparison filter and the subject's
    // neighbour filter and store it in the map
    int difference = comparisonFilter.SetDifference( i->Neighbours() );
    differences[ difference ] = ( *i );
}
}
```

Fig. 4.8: For each neighbouring node, the Reduced MPR calculator constructs a bloom filter with identical properties and fills it with the local node's neighbour identifiers. The two filters can then be compared in order to determine the degree of intersection between the local node's neighbourhood and that of the subject node.

that a node currently has, rather than having a length that is homogenous across all nodes. The neighbour and MPR filters are inserted into the ServiceAdvertisement message, which has the format illustrated in figure 4.9.

4.1.5 Bloom Filter

Bloom filters play an important role in the SDRP protocol, acting as descriptors for provided services as well as node neighbour sets and selected MPRs. The class *BloomFilter* is used to represent filter instances. The class exposes methods for initializing filters inserting and removing elements and for serializing and deserializing filters into and out of packets.

A BloomFilter object is constructed either by copying another BloomFilter, providing a desired false-positive probability rate and expected element count or by directly specifying m , the filter size, and k , the number of hash functions that should be used. When the desired

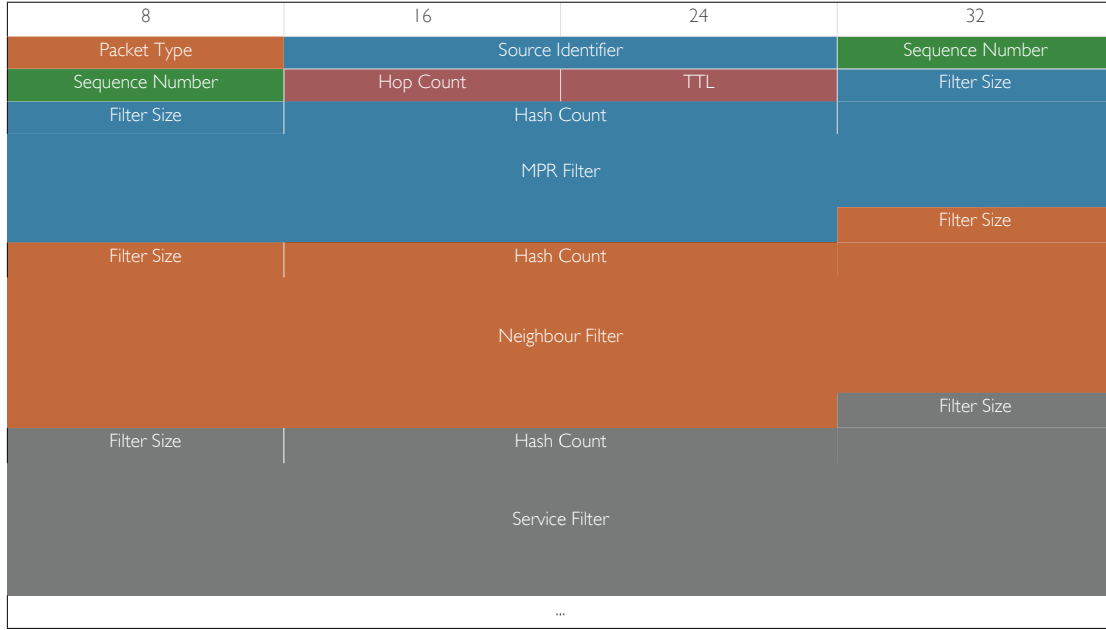


Fig. 4.9: The implemented packet layout for an SDRP-RMPR service advertisement includes an MPR filter containing the identifiers of selected MPR nodes and also includes a neighbour filter describing the node’s link-state with its neighbours.

false-positive rate and expected element count are provided, m is first calculated according to the equation:

$$m = -\frac{n \ln p}{(\ln 2)^2} \quad (4.1)$$

A C++ version of this equation was implemented using the *cmath* headers provided with the platform’s C installation. Similarly, k is determined by substituting the calculated value of m into the equation:

$$k = \frac{m}{n} \ln 2 \quad (4.2)$$

As outlined in section 4.1.2, the SDRP LocalNode object exposes BloomFilter objects describing the services offered by the local node, as well as the node’s current neighbours. Manipulating the filter describing services changes the content of ServiceAdvertisement messages sent to peers. While adding service descriptors to the service filter is a simple operation, removal involves unsetting bits which are potentially also part of the hash output for other descriptors. This characteristic of bloom filters is described in detail in Appendix A.

This limitation was overcome by representing filters internally as counting filters; allowing elements to be removed without interfering with other elements' bit offsets. When a filter is serialized into a packet, it is represented as a bit array, optimizing space usage. When a bloom filter is deserialized, it is not possible to ascertain the integer value of each filter offset as set by the source node; this each set filter offset is given a value of 1 in the counting filter.

An array of unsigned integers of size m is allocated to act as storage for the counting filter. The murmurhash hashing algorithm was chosen as the hash function for the filter implementation due to its speed and low collision rate (Appleby, 2008).

4.1.6 Summary

This section outlined the core classes, interfaces and algorithms used to implement SDRP. SDRP is designed to operate across multiple platforms, including simulators, by having platform-specific functionality implemented through a simple interface, *SDRPDelegate*. SDRP uses implementations of the *SDRPDelegate* interface to communicate with the network and get the current time for the purpose of stale table entry purging and internal algorithms.

The *RoutingManager* class exposed through SDRP's interface is responsible for handling incoming packet data, sending protocol messages and supplying routes to hosts and services as required. *RoutingManager* relies on the *LocalAreaMonitor* to store information about the local network topology and calculate MPR sets for service advertisement dissemination.

The *LocalAreaMonitor*, in turn, uses a subclass of the *IMPRCalculator* interface to calculate MPR sets. The different SDRP operational modes, Flooding, MPR and Reduced MPR, are implemented as *IMPRCalculator* subclasses. *IMPRCalculator* subclasses are instantiated by the *MPRFactor* class based on the current operational mode. This factory pattern allows SDRP to be extended with alternative MPR selection algorithms as required.

The *RoutingManager* records routes to service agents and reverse routes, upon receipt of messages, in *RouteTable* instances. These *RouteTable* instances can be instructed to purge stale routes based on a pre-configured Route age limit and provide sets of *Route* objects sorted by increasing distance to the Route's endpoint.

The *BloomFilter* class is used extensively within the protocol to represent sets of node and service identifiers in Beacon and ServiceAdvertisement messages, as well as during MPR set calculation. The *BloomFilter* class is a flexible implementation that represents sets as a counting filter internally, allowing for the removal of elements, and serializes and deserializes sets as actual bloom filters; reducing their size when inserted into messages.

In the next section, we will describe the NS-2 network simulator, discuss some of its

limitations and strengths and outline the adaptations made to it and the evaluated protocols in order to obtain the results required for this thesis.

4.2 The NS-2 Network Simulator

The NS-2 network simulator is a widely-used platform for research in the field of MANETs (Kurkowski et al., 2005). NS-2 is an extensible discrete-event simulator designed to simulate both wireless and wired networks. The core of NS-2 is written in C++, with shadow classes and variables replicated in OTcL to facilitate scripting. NS-2 simulations are described using OTcL scripts containing simulation setup, network topology and event information. The use of OTcL scripts means that many complex simulation tasks do not require a full recompile of NS-2.

NS-2 interprets input OTcL scripts, using them to instantiate a network topology, applications and protocols, simulates interactions between those entities and outputs a trace file containing event information. Events are generated by agents and applications, scheduled for handling some point in simulation time, and then handled in order until the simulation ends. Some events generate trace output, such as packet sending, dropping and forwarding. These are written to a trace file which can then be interpreted after the simulation has completed.

NS-2 can be run on the Linux, Mac and Windows (through Cygwin) operating systems. NS-3, the next version of NS, was first released in 2008 and is in active development. It uses Python for simulation scripting, allowing greater flexibility and ease of use when compared to OTcL. However, due to the fact that many MANET routing protocols and service discovery agents have yet to be implemented in NS-3, we chose NS-2 for our evaluation instead.

4.2.1 Limitations

Routing in a simulated NS-2 network is handled by classes derived from the Agent base class. The basic Agent interface accepts commands from OTcL scripts as an array of strings, similar to the standard *main* function in C and C++ programs, and receives packets from upper and lower layers. Agents act as endpoints where network packets are created or consumed.

Implementation of a routing protocol in NS-2 is accomplished by first inheriting from the Agent class and implementing the *command* and *recv* methods. Variables used for agent configuration can be shadowed in OTcL if required. Any packets used by the protocol must be defined as structs and must include a static member, *offset*, which is used by NS-2 to position that particular packet type in Packet data buffers.

Agents receive packets either from the network or from an attached Application subclass. Applications can range from simple constant bit-rate traffic generators to complex systems such as an FTP or Web application. An assumption made in the design of NS-2's suite of applications is that the connection between two application instances in a simulated network be static. Application connections are typically defined in OTcL simulation scripts rather than established dynamically at runtime.

To understand why this is a problem, one must understand how NS-2 routes packets within a node. A node instantiated by NS-2 contains one or more Classifier instances. Classifiers are responsible for routing packets between objects within a node. This is necessary given that the various network layer components for NS-2 nodes are defined at runtime through OTcL scripting; some method for passing packets and events between those components is required. Classifiers route packets and events based on their *slot number*. This number, defined in the packet header or event attributes, determines which object should receive a particular event or packet.

When a connection between Application instances is described in OTcL, NS-2 defines a random set of slot numbers that are used to route packets to those Application instances within the node itself. Applications in the NS-2 suite insert those slot numbers into their packet headers, allowing the receiving node to route the received packet to the appropriate internal sink object.

This assumption poses a problem when evaluating application throughput in a service-oriented MANET context, as a connection between a client and server may not be statically-defined, particularly when there are multiple servers in the network providing the required service.

Arbitrarily routing an Application packet to a node to which it is not connected results in runtime errors, as the receiving node's Application instance has not registered a connection with the sender and the node's Classifier objects cannot locate a receiver object with the slot number contained within the packet.

This limitation in NS-2's design may explain why none of the service discovery protocols discussed in Chapter 2 that are implemented using the simulator include any sort of application throughput evaluation. Instead, protocols are evaluated in terms of their discovery success rates and control overheads; characteristics which can be tested without the use of an Application instance.

In order to properly evaluate SDRP, this limitation in NS-2's design had to be overcome and our solution had to be applied to the other protocols we planned to evaluate as well in order to obtain a proper throughput comparison.

4.2.2 Extending NS-2

Our solution to the statically-defined connection problem was to bypass use of the NS-2 Application component entirely. We decided to design a generic service client and server class that would be integrated into the inheritance hierarchy of each evaluated protocol. This class, called *ServiceAgent*, could then be used to evaluate application throughput, service discovery success rate and other statistics as needed.

Typical routing and service discovery protocols tend to inherit directly from the *Agent* base class, allowing them to receive packets and OTcL commands. Some, however, inherit from extensions of *Agent* such as *UDPAgent*, giving them extra capabilities through those classes' methods. To facilitate easy insertion of *ServiceAgent* into the inheritance hierarchy, it is implemented as a template class that in turn inherits from whatever *Agent* subclass the target protocol is based on. This allows the modified protocol to retain access to any required base class methods while allowing integration of *ServiceAgent* without modification. The integration process is illustrated visually in figure 4.11. The interface signature for *ServiceAgent* is illustrated in figure 4.10.

The virtual methods described in the *ServiceAgent* interface allow it to request that the implementing class execute a service discovery operation, send a packet to a server once one has been found and inform the agent when the packet is sent.

The advantage to our approach is that application throughput measurement can be integrated easily into all of the protocols we plan to evaluate. In addition, by integrating *ServiceAgent* directly into the inheritance hierarchy, the class is able to receive OTcL commands and packets as if it were an *Agent* itself. Thus, the *ServiceAgent* instance of each node can be configured using OTcL simulation scripts and the *ServiceAgent* can receive packets sent by other *ServiceAgent* instances; all without relying on the NS-2 Classifier.

The commands accepted by *ServiceAgent* through the **command** method and their purpose are detailed in figure 4.12.

4.3 SDRPAgent

The *SDRPAgent* class acts as a delegate for SDRP, sending and receiving packets through the functions made available by NS-2 and providing time information through the NS-2 simulation scheduler. *SDRPAgent* is also a subclass of *ServiceAgent*, allowing it to respond to service requests and route messages to and from service agent nodes.

SDRPAgent sends Beacon and ServiceAdvertisement messages by calling the appropriate

ServiceAgent<T>

```
// Process an OTcL command. Inherited from T, where T is a subclass of Agent
int command( int argc, char** argv )

// Receive a packet from NS2 for processing
void recv( Packet* packet )

// Request that the Service Agent search for a server providing the
// service with the specified ID
virtual void RequestService( const ServiceID service )

// Send the provided packet to a server providing the specified service
virtual void SendToServer( Packet* p, const ServiceID service )

// Attempt to send a response to the client with the specified address
virtual void RespondToClient( Packet* p, const nsaddr_t address )

// Called on the implementing class when the service agent locates
// a server that provides the service requested by RequestService()
virtual void DidFindService( const ServiceID service )

// Called by the implementing class successfully sends a packet
// to a server, used for statistics gathering
virtual void DidSendPacket( Packet* p, const ServiceID service )
```

Fig. 4.10: The ServiceAgent interface exposes methods for statistics gathering and other methods that should be implemented by the child class for sending packets to servers.

methods on the SDRP instance's *RoutingManager* object. The Beacon and ServiceAdvertisement messages are sent at intervals determined by OTcL simulation script parameters passed to the Agent through the **command** method when the simulation begins. Figure 4.13 describes the available commands in detail.

SDRPAgent handles requests for services by interrogating the SDRP instance's *RoutingManager* through the **HasRoutesToService** method. If no route is available to the specified service, the request is cached for a configured period of time. If a route to the service is subsequently found, the ServiceAgent is informed through its **DidFindService** method. If the caching time limit passes, the request is purged and the ServiceAgent will record the request as having failed at the end of the simulation.

SDRPAgent received packets from upper and lower layers through the **recv** method, inherited from the NS-2 Agent interface. NS-2 Packet objects include a number of headers, including a mandatory common header containing a packet type field that is used by Agent instances to distinguish their packet types from those of other Agents. SDRPAgent interrogates the common

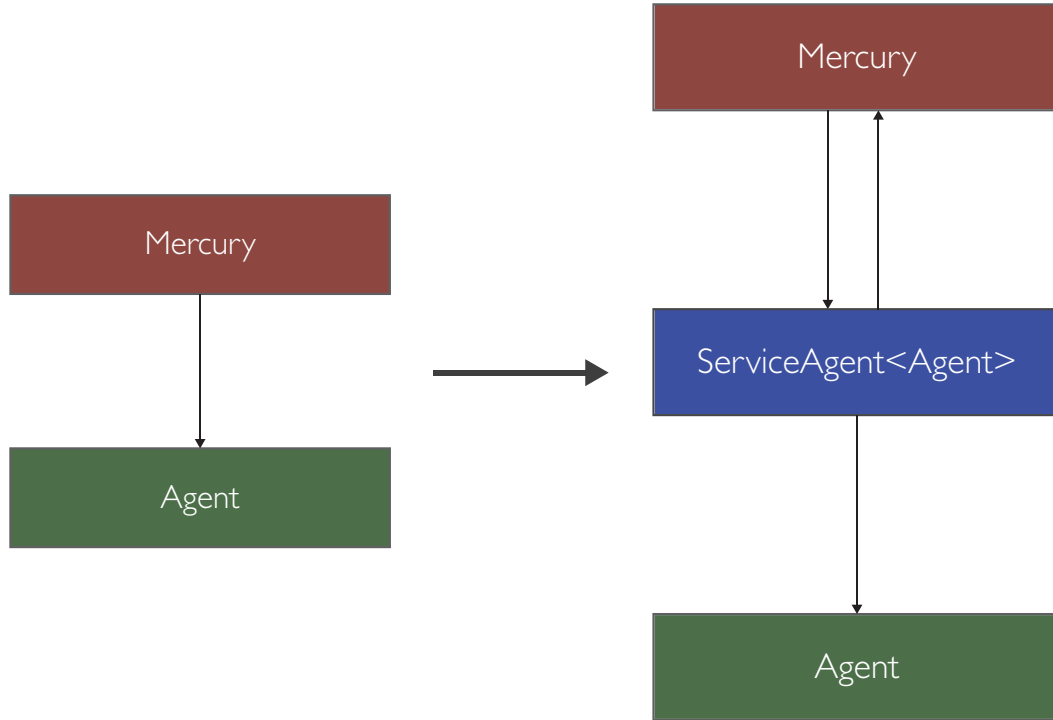


Fig. 4.11: The ServiceAgent class is inserted directly into the inheritance hierarchy of each evaluated protocol. In addition, each protocol is extended to implement the ServiceAgentDelegate interface. Through this interface, ServiceAgent instances can request services and send packets, allowing application throughput and other statistics to be measured.

header packet type and, if the packet type matches the SDRPAgent packet type, the packet is passed to an internal method, **HandleSDRPPacket**. Otherwise, the packet's IP header TTL is decremented and, if the packet's TTL is greater than zero, it is forwarded. Otherwise it is dropped.

HandleSDRPPacket extracts the packet body and its length and passes it to the SDRP instance through **RoutingManager::HandlePacket**. If the packet is a service advertisement, the local node's service routing tables may have been updated. SDRPAgent iterates over its cached ServiceAgent requests and calls **HasRoutesToService** to check if a route to any requested services have now become available. If one or more routes are now available, SDRPAgent informs the ServiceAgent through the **DidFindService** method and removes the request from the cache.

Command	Valid Values / Formats	Description
send-start	-	Starts the timer responsible for triggering service requests
server-start	-	Allows the Service Agent to begin accepting service request packets
send-interval	Floating Point	Defines the interval at which service request packets are sent. It should be noted that a service response received from a service agent will immediately trigger another request
expire-time	Floating Point	Defines the maximum time allowed between a request being made to the service discovery protocol and a positive response being received before the request is considered to have failed
packet-size	Unsigned Integer	Defines the size of the service request packet data payload sent to the service agent when a request is made. This value influences application throughput results.
register-service	Unsigned Integer	Register a service identifier with the Service Agent instance. When a request is received, the requested service identifier will be checked against those registered to determine whether the request can be satisfied by this Service Agent.
target-service	Unsigned Integer	Defines the service identifier that should be requested by this service agent when sending service requests.

Fig. 4.12: ServiceAgent receives commands from OTcL simulation scripts through the **command** method. This table describes the commands that ServiceAgent acknowledges and their purpose.

Packets are forwarded by consulting the SDRP instance's routing tables. The destination service identifier, contained either in a custom header, the TCP header or the UDP header, depending on the packet type, is extracted and checked against the *RoutingManager*'s route tables using **RoutingManager::HasRoutesToService**. If routes are available, the shortest route to the destination service is chosen and the packet is forwarded to the next hop along that route.

Command	Valid Values / Formats	Description
start	-	Starts the timers responsible for purging stale routes and sending Beacon and Advertisement messages
routing-mode	"flooding", "mpr", "reduced-mpr"	Sets the SDRP MPR selection mode. This will cause the SDRP LocalAreaMonitor to request a new IMPRCalculator instance from the MPRFactory object.
beacon-interval	Floating Point	The interval between Beacon message broadcasts, expressed in seconds
advertisement-interval	Floating Point	The interval between Service Advertisement message broadcasts, expressed in seconds
purge-interval	Floating Point	The interval between route table stale entry purge commands, expressed as seconds
register-service	Unsigned Integer	Register a service identifier with the SDRP instance. The service identifier will be inserted into the SDRP Local Node's service filter.

Fig. 4.13: SDRPAgent receives commands from OTcL simulation scripts through the **command** method. This table describes the commands that SDRPAgent acknowledges and their purpose.

If no route to that destination service is available, the destination node identifier is used to look up routes in the *RoutingManager*'s reverse route tables. These routes connect service agents back to user agents. If a route is found, this indicates that the received packet is a service response and should be forwarded to the service agent. The shortest route to the user agent is chosen and the packet is forwarded to the next hop along that route. If no route can be found to the destination service identifier or node identifier, the packet is dropped.

4.4 SCDP

In the next chapter, we will describe the evaluation of SDRP, including its comparison to other service discovery protocols. The NS-2 source code for a number of protocols was obtained from their respective authors for the purpose of evaluation. However, in a two cases, the protocol implementation was unavailable or incompatible with NS-2, having been written for use with alternative simulation software.

The Lightweight Service Discovery protocol (Li and Lamont, 2005), described in Chapter 2, is a cross-layer directory-based service discovery solution integrated with the Optimized Link-State Routing protocol (Jacquet et al., 2001). LSD is an ideal protocol for comparison against SDRP, as it combines directory-based discovery with peer-to-peer fallback functionality; providing both performance and redundancy.

However, LSD was implemented as an OPNET simulator agent, which is incompatible with NS-2's interfaces and functions. Repeated attempts to contact the authors of LSD in order to obtain the source code also failed. As such, the choice was made to partially re-implement LSD in NS-2 based on the description of the protocol provided in Li and Lamont's paper, with additional revisions made in order to expand the overall evaluation. This revised version of LSD is called the Service Coordination and Discovery Protocol, or SCDP.

LSD is described as a cross-layer protocol based on top of OLSR, however for the purpose of evaluation, we decided to implement SCDP as an application-layer protocol that could be deployed on a routing protocol of our choice. This was done in order to increase the number of application-layer protocols evaluated, as the only protocol of this type we had obtained thus far was the Pervasive Discovery Protocol (PDP). This would also allow us to expand the evaluation of protocols based on reactive routing beyond the Service Magnetic Field (SMF) protocol; the only service-discovery approach we had obtained based on AODV.

SCDP is otherwise designed to include the same features as LSD; SCDP nodes may be designated as directories which periodically broadcast their location throughout the network. Service Agents that receive directory broadcasts periodically register their available services

with all known directory nodes. User agents that receive directory broadcasts can send queries to those directories in order to obtain the address of service agents. In the absence of directory nodes, user agents can broadcast queries throughout the network. Service agents receiving a query check the required service identifier against their own available services and respond if a match is found.

SCDP's directory and query broadcast approach differs from that of LSD due to the lack of cross-layer integration. LSD appends its directory and broadcast messages to OLSR topology control messages, allowing them to be disseminated efficiently throughout the network. SCDP, by contrast, floods these messages directly as it cannot make assumptions regarding the capabilities of the underlying routing protocol beyond its ability routing unicast messages. SCDP can be expected to produce higher control overheads than LSD due to this naive flooding pattern, however it has the advantage of flexibility in terms of the underlying routing protocol used.

The formats of the various SCDP message types are illustrated in figure 4.14. During our evaluation, we instantiate SCDP agents on top of both OLSR and AODV routing layers. This allows us to compare OLSR and AODV in terms of their impact on the network while testing the performance of a directory-based discovery approach in terms of discovery success rates and throughput. SCDP instances that receive a directory broadcast will store that directory's IP address and the time at which the broadcast was received in their directory cache. Similarly, directory nodes store the IP addresses, time of registration and registered services of service agents in their service agent cache. Both caches have stale entries purged at intervals configured through OTcL.

If an SCDP instance has been designated as an active user agent through the underlying ServiceAgent instance's **command** method, it will attempt to unicast a service request message to each registered directory node. If no directory is available, or no positive response is received after a configured request expire period, the SCDP instance will flood a service request message. SCDP nodes receiving a service request message will consult their registered services table and send a directory service response message containing their own IP address if the service is found. Otherwise, they will rebroadcast the request message to surrounding nodes. Sequence numbers are used to ensure that request messages are not rebroadcast needlessly by receiving nodes.

SCDP service agents reference the underlying ServiceAgent class' registered service set when checking for the availability of specific service upon receipt of a query. Thus services are registered with the SCDP instance through commands sent to the ServiceAgent. SCDP nodes are designated as directories through the OTcL simulation script. The full set of commands acknowledged by the SCDP Agent and their purpose are described in figure 4.15.

Unicasting of protocol messages and data packets is handled by the underlying routing protocol through a direct call to its **recv** method. In the case of both OLSR and AODV, the



Fig. 4.14: SCDP uses 4 protocol message types, allowing directories to advertise their presence, service agents to register their services and user agents to request services either by directly querying directory nodes or by broadcasting.

protocol will interpret the packet as a data packet and will attempt to forward it to its destination, as detailed in the packet IP header.

Command	Valid Values / Formats	Description
start	-	Starts the timers responsible for registering with directory agents, purging stale service agent registrations and broadcasting directory beacons.
directory	-	Indicates that the SCDP instance should set itself to Directory mode. In this mode, the directory broadcast timer will trigger a directory beacon flood.
cache-max	Floating Point	Defines the maximum time for which a service agent registration will be considered valid by directory nodes.
directory-max	Floating Point	Defines the maximum time for which a directory registration, triggered by the receipt of a directory beacon flood, is considered valid by receiving service and user agents.

Fig. 4.15: SCDP receives commands from OTcL simulation scripts through the **command** method. This table describes the commands that SCDP acknowledges and their purpose.

4.5 SMF

As described in the previous section, the protocol implementations for two of the protocols compared against SDRP during our evaluation were incompatible with NS-2's interfaces and functions and had to be reimplemented. SMF, a cross-layer discovery approach based on AODV, was implemented by its authors as a GloMoSim (Zeng et al., 1998) agent. The source code for the SMF agent was obtained from the authors in January 2012 and rewritten as an NS-2 agent for use in our evaluation.

The new SMF Agent was written as a subclass of NS-2's built-in AODV Agent class, allowing it to utilize AODV's request flooding mechanism and routing tables. SMF relies on AODV HELLO messages to advertise services within a service agent's immediate neighbourhood and flooded service query messages by user agents to locate services. An extension was made to the built-in AODV HELLO message format, appending information on services available from the source service agent.

Command	Valid Values / Formats	Description
start	-	Start the timers responsible for sending HELLO messages and purging stale route entries
hello-interval	Floating Point	Defines the interval between HELLO message broadcasts, in seconds.
purge-interval	Floating Point	Defines the maximum age of a route in terms of time since its last refresh, in seconds. Routes with an age greater than this limit will be purged when the purge timer is triggered.

Fig. 4.16: SMF receives commands from OTcL simulation scripts through the **command** method. This table describes the commands that SMF acknowledges and their purpose.

Additional message formats were written, mirroring those defined in the GloMoSim source code obtained from the protocol authors, such as messages for service querying and response. The commands accepted by the SMF agent through OTcL are detailed in figure 4.16. The HELLO broadcast interval and route purge interval represent the important variables for simulations.

SMF's service querying is controlled by the underlying ServiceAgent. ServiceAgent will call the virtual **RequestService** and **SendToServer** methods, which are implemented by the SMF agent, in order to send requests to servers. SMF consults its route tables and HELLO message cache when a service request is made by ServiceAgent. If the requested service identifier is found within the cache, the request message is sent to the service agent directly using the methods provided by the AODV agent class. Otherwise, SMF will flood a service request throughout the network.

Like SCDP, SMF relies on the underlying ServiceAgent class to provide a list of services registered through OTcL. When a service request broadcast is received by an SMF node, it checks the ServiceAgent's registered services to determine whether it can satisfy the request. If a matching service identifier is found, SMF will unicast a service response message to the sender using the methods provided by NS-2's AODV agent class. Otherwise, it will rebroadcast the message to its own neighbours.

4.6 Summary

In this section we described the implementation of SDRP. SDRP is implemented as a library which may be linked against a simulator or integrated into a real node as required. An overview of SDRP's core classes and interfaces was given in section 4.1.1, before being elaborated in detail in the following sections.

The NS-2 network simulator used to conduct the evaluation of SDRP was described in section 4.2. NS-2's limitations in terms of spontaneous connections between nodes and this limitation's effect on previous evaluations of service discovery protocols was also described. The steps taken to overcome those limitations such as our implementation of a generic service agent class that could be integrated into the evaluated protocols, allowing us to measure application throughput and discovery success rates, was detailed in the following sections.

Finally, our implementation of the SCDP and SMF service discovery protocols was detailed in sections 4.4 and 4.5. SCDP is a modified version of the Lightweight Service Discovery protocol, designed to work as an application-layer discovery solution on top of AODV and OLSR. SMF was rewritten as an NS-2 agent based on GloMoSim source code obtained by the protocol's authors.

In the next chapter, we will describe our evaluation of SDRP and discuss its performance when compared with other discovery approaches.

Chapter 5

Evaluation and Results

In this chapter, we will describe our evaluation of SDRP and its comparison with existing service discovery approaches. The results of the evaluation will also be detailed and discussed. As outlined in chapter 4, SDRP and related protocols were evaluated using the NS-2 network simulator. Due to deficiencies in NS-2 application model, a new Agent class was developed that would allow application throughput and discovery success rates to be measured using a similar mechanism for all protocols.

In the next section, we will describe the experimental setup used to conduct our evaluation. The measured outputs from experiments and the parameters used to vary simulations will be detailed. The utilities used to compose and run experiments will also be described. Finally, the results of our experiments will be discussed before the chapter is summarized.

5.1 Experimental Setup

The NS-2 discrete-event network simulator was used to conduct our evaluation. However, other tools were required in order to formulate full experiments, such as tools for generating mobility patterns, connection patterns and for generating the experimental formats themselves. In this section, we will describe these tools before moving on to a description of the objectives for our evaluation and the experimental parameters used to measure the required outputs.

5.1.1 Target Network Configuration and Objectives

SDRP is specifically designed to perform well in highly-dense networks in which the number of user agents is greater than the number of service agents. This is the network configuration that we will primarily focus on during our evaluation. Due to limitations in NS-2, simulating a network of more than 100 nodes is extremely time-consuming, with a single simulation taking several hours even when run on a modern high-powered machine. As such, few of our experiments could support more than 100-200 nodes without consuming more time than was available or compromising statistical accuracy by executing too few experimental runs. In spite of this limitation, we will demonstrate scenarios in which current approaches to service discovery experience significant reductions in performance.

Our objective in this evaluation is to determine SDRP’s ability to successfully facilitate service discovery, both in terms of measuring user agents’ ability to locate services and the protocol’s ability to route messages between user agents and service agents. We aim to determine SDRP’s limitations in terms of its ability to adapt to varying network properties such as node velocity, density and the number of deployed user agents and service agents. For all scenarios evaluated, we wish to compare SDRP with a number of existing approaches and determine whether our approach exhibits advantageous characteristics.

5.1.2 Mobility and Connection Pattern Generation

NS-2’s OTcL script interface includes commands that move instantiated nodes from point to point within the defined topology area. NS-2 is distributed with a tool, *setdest*, that allows random-waypoint mobility patterns to be generated on the command-line. The parameters used to vary the produced mobility pattern are illustrated in figure 5.1.

Our evaluation only utilizes the random waypoint model, other mobility patterns are considered to be outside the scope of this thesis. The *setdest* utility will be fed various parameter values over the course of this evaluation, resulting in varying mobility patterns that can be used to test the mobility adaptation of the evaluated protocols. The *setdest* utility outputs mobility patterns in the form of OTcL commands which can be integrated into a simulation script using the TcL **source** function.

A connection pattern, for the purposes of our simulation, designates a subset of the nodes in the simulated network as service agents or user agents, defines when user agents should begin making service requests, the interval between those requests and defines the number of services that should be distributed among service agents. An OTcL script, *connngen*, was written specifically for connection pattern generation as existing connection pattern generators for NS-2

Command	Valid Values / Formats	Description
speed	Floating Point	Speed at which nodes should move, in meters per second
pause-time	Floating Point	Length of time for which nodes should pause when between changes in direction
node-count	Unsigned Integer	Indicates the number of nodes for which a mobility pattern should be generated. For our evaluation, this always matches the number of nodes deployed in the simulated network
sim-time	Floating Point	Indicates the length of the simulation, in seconds. No movement events will be generated that trigger beyond this time.
x	Floating Point	The extent of the simulated network topology defined as meters along the X axis. The mobility pattern generator will not instruct nodes to move beyond this point in the X direction.
y	Floating Point	The extent of the simulated network topology defined as meters along the Y axis. The mobility pattern generator will not instruct nodes to move beyond this point in the Y direction.

Fig. 5.1: The setdest mobility pattern generator utility is packaged with the standard NS-2 distribution. Given a set of parameters indicating the network size, simulation length, topology dimensions, desired speed and pause time, it will output OTcL commands that cause nodes to move using the random-waypoint model. Those commands can then be integrated into the OTcL simulation script.

were tailored specifically for its built-in application agents and defined static connections between nodes. During our simulations, user agents attempt to connect to whichever service agents are chosen by their selected discovery protocol, making static connections between nodes unsuitable.

Command	Valid Values / Formats	Description
servers	Unsigned Integer	The number of nodes to be designated as servers or service agents
clients	Unsigned Integer	The number of nodes to be designated as clients or user agents
services	Unsigned Integer	The number of unique service identifiers to be distributed among service agents.
settle	Floating Point	Defines an initial time period, in seconds, after the simulation begins when no service requests should be made. This period should give the protocol enough time to disseminate advertisements and discover routes.
stop	Floating Point	Defines the time after which no connections should be attempted, typically aligned with the simulation length.
node-count	Unsigned Integer	The overall number of nodes in the simulated network. Used by the random number generator when selecting nodes as service or user agents.
interval	Floating Point	The time interval between request attempts. It should be noted that user agents will immediately send a new service request upon receipt of a service response from a service agent. This value defines the period for which a user agent should wait before sending a new request, if no response was received.

Fig. 5.2: The conngen connection pattern generator script designates a subset of the simulated nodes as user or service agents, based on the parameters passed to it.

The `conngen` script accepts the set of parameters described in figure 5.2. By varying the parameter values, the number of service agents, user agents and services deployed in the network can be controlled, along with the interval between service requests. Additionally, `conngen` accepts a parameter, the settle time, indicating an initial time period within which no service requests should be made. This is included to give the various discovery protocols, particularly those based on service advertisement or which rely on a proactive routing protocol, time to establish routes to services. If requests are made as soon as the simulation begins, some protocols may exhibit low initial success rates which skew their overall results.

Similarly to the `setdest` utility, `conngen` outputs the generated connection pattern as OTcL commands which can be directly integrated into the OTcL simulation script using the `source` function. The generated OTcL code configures the `ServiceAgent` instances on the selected user and service agents and schedules the start of the selected user agents' request cycles at a random point between the end of the settle period and the end of the simulation.

Once a request cycle begins, `ServiceAgent` will send service requests to the underlying discovery protocol through its virtual methods. The target service identifier will be chosen at random within the range from zero to the value of the `services` parameter passed to `conngen`. Designated service agents have service identifiers registered with them at random within the same range.

5.1.3 Experiment Distribution

A significant degree of random variation in experimental results obtained from simulated networks can be expected, due to the effects of node mobility and unpredictable interference patterns. This random variation reduces the accuracy of statistical tests and can only be overcome by running each experiment multiple times. However, as mentioned previously, a single simulation can take several hours to complete; with simulation length highly dependent on the number of simulated nodes and the amount of traffic generated. In order to carry out a sufficient number of simulations to satisfy the requirements of statistical comparison within a reasonable timeframe, experiments would have to be distributed across multiple machines.

To facilitate this, an automated system was developed which, given a description of an experiment in the form of a comma-separated-values (CSV) file, could distribute experimental runs among a number of worker machines. This system was written in Python and was comprised of a client application and a server application, called `SimulationClient` and `SimulationServer` respectively.

The `SimulationServer` presents a user interface based on *ncurses*, a library allowing graphical interfaces to be presented in text-based shells, which describes the current experiment, the

number of client machines currently running simulations and counters indicating the number of running, pending and completed experimental runs. The `SimulationServer` application binds to a TCP port and listens for connections from `SimulationClient` instances. `SimulationClients` connect in order to request new experimental runs, submit results or report errors.

When a client requests a new experimental run, the server responds with a JSON-encoded key-value set describing the experimental parameters that should be used to execute the run. Submitted results are encoded in CSV format and written to an experiment output file for later analysis. When an error is reported by a client, the failed experimental run is re-listed for later execution, reported in the `ncurses` UI and details of the error are logged to a file.

The `SimulationServer` can be configured in a number of ways at startup using command-line arguments. The TCP port used to listen for incoming client connection can be set, the input file describing the experiment, the AWK script used to analyze experiment results and the OTcL simulation script which should be used by clients to execute each run can all be specified.

The `SimulationClient` operates by connecting to a server specified by the user, requesting simulation runs, executing them and then analyzing the result using an AWK script specified by the `SimulationServer`'s configuration. Similarly, an OTcL simulation script is specified by the `SimulationServer` for use when NS-2 is run. As such, each `SimulationClient` must have access to the required scripts, as well as an identical build of NS-2 to all of the other clients.

This was facilitated by creating a SAMBA network file share containing the required scripts and the current NS-2 build. When the `SimulationClient` is deployed onto a worker machine, the SAMBA share containing the required scripts is mounted and a script is run which copies NS-2 onto the worker's filesystem and rebuilds it.

As mentioned in previous sections, a number of utilities are required to execute a full NS-2 simulation, such as mobility and connection pattern generators. `SimulationClient` instances parse experimental parameters, searching for specific sets of keys that are applicable to those utilities. For instance, if speed and pause time parameters are specified, `SimulationClient` will run the *setdest* mobility pattern generator script. Similarly, if the parameter set contains keys specifying the number of clients and servers to be deployed in the network, the *connngen* connection pattern generator will be run.

The output from these scripts, along with the experimental parameters retrieved from the server, are then passed to NS-2. `SimulationClient` parses *stdout* output generated by NS-2 in order to gauge the simulation's current progress. The OTcL scripts used in our simulations include a special function that instructs NS-2 to output the current simulation's progress periodically based on the simulation length specified by the script's parameters. The simulation progress is reported to the simulation server, which then displays it in the user interface.

NS-2 simulations output trace files detailing events that occurred during the simulation, including packet send and receive events. The AWK text parsing tool is used to parse these traces and calculate results such as protocol overheads and the throughput of client-server traffic. The AWK script to be used when analyzing the current simulation's trace is also specified in the experimental parameters retrieved from the server and SimulationClient will run this script against the trace file once NS-2 completes its execution. SimulationClient parses the AWK file's output, searching for results. Results are printed as name-value pairs separated by a colon. These results are returned to the SimulationServer when the AWK script has finished analyzing the trace file and the SimulationClient then requests a new run, starting the process over again.

SimulationClient can be configured at startup using command-line parameters. In order to take advantage of multi-core worker machines, the number of SimulationClient instances that should be spawned in parallel can be specified. SimulationClient also expects the host name or IP address of the server and a port to which it should connect in order to retrieve new simulations.

Using the SimulationClient and SimulationServer, we were able to execute thousands of simulations per experiment in a reasonable time-frame; allowing us to produce more meaningful experimental results and comparisons for the evaluated protocols.

5.1.4 Evaluated Protocols

As outlined in Chapter 4, the SMF protocol was reimplemented as an NS-2 agent for the purposes of this evaluation, as well as a modified version of LSD called SCDP that operates at the application-layer rather than the cross-layer; allowing it to be deployed using a choice of address-based routing protocols. During our evaluation, we deployed SCDP on top of the AODV and OLSR routing protocols. SCDP with AODV will be referred to as SCDP-Reactive or SCDP-RA, while SCDP with OLSR will be referred to as SCDP-Proactive or SCDP-PA. In addition, NS-2 versions of Mercury and the Pervasive Discovery Protocol were acquired from their respective authors.

This selection of service discovery protocols allows SDRP to be evaluated against a wide range of approaches. Cross-layer and application-layer discovery are represented, as are peer-to-peer and directory based discovery. A summary of the evaluated protocols along with a short description of each is given in figure 5.3.

Mercury was chosen to represent cross-layer P2P discovery based on proactive address-based routing. Similarly to SDRP, Mercury incorporates the use of bloom filters for service description. Combined with its cross-layer integration with NS-2, we expect Mercury to perform relatively well. However, the control overheads generated by OLSR at high network densities are expected to adversely affect Mercury's performance in such scenarios.

Service Discovery Protocol	Routing Protocol and Integration Method	Description
Mercury	OLSR (Cross-Layer)	Cross-layer advertisement-based protocol that uses static-length bloom filters for service description. Advertisement messages are appended to OLSR TC broadcasts
PDP	OLSR (Application-Layer)	Application-Layer query/advertisement protocol. Service agents defer advertisement until they receive a query for their services, restricting advertisement traffic to those services that are in demand.
SCDP-RA	AODV (Application-Layer)	Hybrid directory/peer-to-peer protocol. Directory nodes broadcast their presence, allowing service agents to locate them and register their services and user agents to perform queries. In the absence of directories, user agents directory flood service queries to service agents.
SCDP-PA	OLSR (Application-Layer)	Hybrid directory/peer-to-peer protocol. Directory nodes broadcast their presence, allowing service agents to locate them and register their services and user agents to perform queries. In the absence of directories, user agents directory flood service queries to service agents.
SMF	AODV (Cross-Layer)	Cross-layer query-based protocol. Uses the AODV flooding mechanism to distribute service queries and AODV route tables to route traffic between user and service agents.
SDRP-Flooding	N/A	Service discovery protocol that does not rely on address-based routing. Uses flooding of variable-length bloom filters containing service descriptors by service agents to advertise services.
SDRP-MPR	N/A	Service discovery protocol that does not rely on address-based routing. Periodic broadcasts of HELLO messages containing bloom filter describing neighbour link states by all nodes allow service advertisements to be disseminated efficiently.
SDRP-Reduced MPR	N/A	Service discovery protocol that does not rely on address-based routing. Service advertisement is optimized using a border-node selection algorithm based on the use of variable-length bloom filters for neighbour link-state description.

Fig. 5.3: The suite of service discovery protocols evaluated in this chapter, along with a short overview of their features.

PDP represents P2P, application-layer service discovery. The NS-2 source for the PDP Agent and instructions for its use were obtained from the authors and integrated into our own NS-2 installation, with the additional integration of ServiceAgent and its required methods to facilitate output measurements. For the purpose of our simulation, PDP was paired with OLSR as the underlying routing protocol. Due to its lack of cross-layer integration, we would expect PDP to perform more poorly than Mercury in terms of its ability to adapt to high density and network scale.

PDP's hybrid advertisement/query approach is well-suited to networks in which we would expect service request rates to be low, as the protocol is designed to have service agents play a passive role until they receive a request from a user agent, after which they begin to advertise their services. Our evaluation will focus on scenarios involving high request frequencies, which is outside the deployment scenario described by PDP's authors.

SCDP is an implementation of the Lightweight Service Discovery protocol at the application-layer. SCDP-RA is the designation given to SCDP running on top of a reactive layer; specifically, the AODV routing protocol. All experiments on SCDP will involve the designation of one or more nodes as directories, a step which should improve SCDP's performance as it avoids query

flooding by user agents. We expect SCDP-RA to perform well, as AODV's route request flooding mechanism is not dissimilar to SCDP's directory and query broadcast approach; we would not expect to see a significant decrease in overheads through cross-layer integration in any case.

In scenarios involving small numbers of clients, SCDP-RA may perform better than protocols based on proactive advertisement, as the overheads produced by clients sending route request floods would likely be less than the overheads produced by all network nodes periodically broadcasting. However, in scenarios involving many clients, the naive flooding mechanism employed by AODV may prove to be a disadvantage, as frequent route request floods by multiple clients saturate the medium.

SCDP-PA is the designation given to SCDP running on top of a proactive routing protocol, in this case OLSR. We expect SCDP-PA to perform on par with PDP and worse than Mercury, as it does not integrate with OLSR in order to take advantage of that protocol's efficient message dissemination algorithm. SCDP-PA should perform well in scenarios involving few servers and many clients, as OLSR's proactive route establishment allows clients to avoid flooding route requests in order to send messages to directories and service agents.

SMF is a cross-layer P2P service discovery protocol integrated with AODV. SMF takes advantage of AODV's HELLO messages to allow service agents to advertise their services in the local area, negating the need for service query floods when service agents are nearby. We would expect SMF to perform similarly to SCDP-RA in spite of its cross-layer design, as AODV does not provide an efficient request dissemination algorithm for SMF to take advantage of; however its HELLO message integration should allow it to outperform SCDP-RA in scenarios involving greater network densities as service agents are likely to be near to user agents.

5.1.5 Measured Outputs

The key measurements we aim to gather during our experiments relate to service discovery and utilization. A key metric measured during the evaluation of existing approaches is the discovery success rate; the ratio of successful service discovery attempts to the total number of attempts made. We add to this, through the addition of the ServiceAgent system described in section 4.2.2, a measurement of the ratio of service requests successfully delivered to the total number sent. In this case, a service request is simply a packet sent by a ServiceAgent instance running on a user agent to the ServiceAgent instance running on a discovered server. ServiceAgent also facilitates measurement of response delivery ratios; ServiceAgent instances running on servers that receive a service request will immediately send a response packet to the source of that request. We will measure the ratio of successfully delivered service responses to the total number sent by service agents.

Request and response delivery ratios are a key metric in determining the effectiveness of a discovery protocol, particularly cross-layer approaches, as a system that cannot route messages between user and service agents is effectively useless. We expect service discovery success rate, request delivery ratios and response delivery ratios to be affected by the control overheads generated by the service discovery protocol deployed in the network. As more overheads are generated, the probability that collisions and errors will occur increases, which will negatively affect the delivery of protocol and application packets. Thus, control overheads generated by the deployed service discovery and, if applicable, routing protocol are measured for each simulation.

Finally, the total data throughput achieved between user agents and service agents will be measured. Throughput will be heavily affected by the success rates and delivery ratios mentioned previously and will describe the overall performance of each protocol. Protocols that exhibit high application throughput can be expected to have high discovery success rates, high delivery ratios and low overheads.

5.1.6 Experimental Parameters

NS-2 allows nodes to be configured with varying wireless interface parameters. For the purpose of our simulation, we chose interface parameters that conform to those of an ORiNOCO 802.11b PC Card, retrieved from the NS-2 user mailing list. 802.11b is limited in terms of maximum bandwidth, however the IEEE 802.11 standard does not include ad hoc mode support for any protocols above 802.11b; thus it was considered to be the most suitable protocol choice for emulating a realistic deployment scenario. A table of the interface configuration parameters and their values is given in figure 5.4.

Parameter	Value	Description
Antenna Transmit Gain	1.0	Gain applied by the omnidirectional antenna to the radio transmission power
Antenna Receive Gain	1.0	Gain applied by the omnidirectional antenna to the radio reception power
Interface Type	802.11	Wireless interface family type
Frequency	2.472GHz	Frequency band used for communications, conforms to 802.11b standard, channel 13.
Signal Propagation Model	TwoRayGround	Model used to calculate the degradation in signal power at a distance from the source
Bandwidth	11Mb	Maximum raw data rate of the interface, conforms to 802.11b standard
Control Frame Rate	1Mb	Data rate used for control frame and broadcast transmission
Receive Range	160m	The maximum receive range for a transmission, based on the transmission power, carrier sense threshold and receive threshold

Fig. 5.4: Wireless interface configuration used during this evaluation.

SDRP is designed to overcome the performance limitations of existing approaches in high-

density and large scale networks. By deploying a large number of simulated nodes into topologies of varying sizes, density and scale can be controlled. Our simulations will involve network densities varying from as few as ten nodes up to hundreds. For protocols incorporating periodic message transmission, such as HELLO messages, topology control messages and service advertisement messages; the interval between message transmissions will be set to similar values for all protocols.

Mobility adaptation will be measured by varying maximum node speed, the pause time used to generate mobility patterns will be kept static at 5 seconds as it is not considered to be a significant parameter. As mentioned previously, the random waypoint model is used to generate mobility patterns; nodes will move to a random waypoint within the simulated topology at a speed randomly selected between 1m/s and the maximum speed, pause for 5 seconds upon reach the waypoint and then repeat the process by choosing a new waypoint.

Parameters specific to service discovery that will be varied during our experiments include the number of user and service agents deployed in the network. The number of user agents is expected to have a greater impact on approaches based on reactive route discovery or service querying, as additional message traffic will be generated as the user agent count increases. Protocols based on proactive routing or service advertisement are not expected to be significantly affected by changes in user agent count. Conversely, such protocols are expected to suffer performance drops as additional service agents are deployed in the network, as service advertisement traffic can be expected to increase, thus increasing overall control overheads.

Some simulation parameters are not expected to have a significant effect on our measurements and will not be varied. The simulation time for all experiments is set to 5 minutes. This gives all protocols adequate time to establish routes and disseminate advertisements and for traffic to be measured. As more time is added to simulations, the required time for experiments increases; thus simulation time is set to a length adequate to obtain measurements while still completing as many simulation runs as possible. Similarly, the settle time before user agents begin requesting services is set to 30 seconds; enough time for all protocols to establish routes.

The interval between service requests by user agents is always set to 1 second. This describes the time that a user agent should wait before requesting a service, assuming a service response was not received for the previous request. A low interval increases each protocol's opportunity to attempt discovery and for us to measure their success rates. It should be noted that a user agent will immediately send a new service request if it receives a response to its last request; thus protocols that are able to provide stable high-throughput links between user and service agents will perform best.

The data payload size for both service requests and responses is set to 512 bytes; this value influences the measured results for application throughput, as throughput is based on the

Category	Parameter	Description
Network	Node Count	The number of nodes deployed in the simulated network
	Topology Width	The width of the simulated topology boundary in meters
	Topology Length	The length of the simulated topology boundary in meters
Node & Wireless	Protocol	The routing and service discovery protocol combination installed on simulated nodes
	Node Speed	The speed at which all nodes will move
Routing Protocol	HELLO Interval	For routing protocols such as OLSR, SDRP-MPR and (when using SMF) AODV, this is the interval between HELLO message broadcasts
	Advertisement Interval	For routing protocols such as OLSR, this is the interval between topology control message transmissions. For protocols such as SDRP, this will be interpreted as the interval between service advertisement broadcasts. SCDP will interpret this parameter as the interval between directory advertisement messages.
Service Discovery	User Agent Count	The number of user agents deployed into the network
	Service Agent Count	The number of service agents deployed into the network
	Service Count	The number of unique service identifiers distributed among deployed service agents

Fig. 5.5: Experiment parameters varied during this evaluation.

amount of data consumed by ServiceAgent instances over the course of a simulation. Variations in this value are not expected to affect particular protocols, thus it remained constant for this evaluation.

A description of the experimental parameters varied during our evaluation along with a short description is shown in figure 5.5.

5.2 Network Scale and Density

Our first experiment is designed to test the ability of the evaluated protocols to adapt to increasing network scale and density. The topology area will remain static while the number of nodes in the network is increased from 10 to 200 in increments of 10. A subset of the deployed nodes will be designated as either user agents or service agents with the number of each remaining static for all experiment runs. Node speed will remain static at 2m/s. At this speed, protocols will have to adjust routes over time but their ability to perform should not be significantly affected when compared with simulations involving nodes moving at high speed.

As the node count will start at 10, the number of user agents and service agents deployed must be less than or equal this number. Thus, for all experiments, there will be 3 deployed service agents and 5 deployed user agents. 3 services, one unique service per service agent,

will be distributed among the service agents. For SCDP-RA and SCDP-PA, 2 nodes will be designated as directories. The user and service agent count will remain static so that the effect of varying node count can be separated from the effect of varying user and service agent count. Later experiments will vary these parameters in order to determine their effect on performance. A summary of parameter values is given in figure 5.6.

Category	Parameter	Value
Network	Node Count	Varying between 10 and 200 in increments of 10
	Topology Width	1000m
	Topology Length	1000m
Node & Wireless	Protocol	SDRP-F, SDRP-MPR, SDRP-RMPR, SMF, SCDP-RA, SCDP-PA, Mercury, PDP
	Node Speed	2m/s
Routing Protocol	HELLO Interval	2 Seconds
	Advertisement Interval	5 Seconds
Service Discovery	User Agent Count	5
	Service Agent Count	3
	Service Count	3

Fig. 5.6: Parameter values used for the scale and density experiment.

A total of 3,200 simulations were executed; for each combination of the 20 node count values and 8 service discovery protocols, 20 replicates were used in order to reduce the effects of random variation. In the next section, we will describe the results from our measurements of protocol control overheads for this experiment.

5.2.1 Control Overheads

The control overheads generated by each protocol are expected to have a significant effect on their ability to facilitate high-throughput communications between user and service agents. As control overheads increase, so too does medium contention and the amount of clear transmission time for application traffic decreases.

Figure 5.7 describes the overheads generated by each protocol for each network scale. The degree of random variation in the results is low for most protocols with the exception of SCDP-RA, where the 95% confidence interval is relatively large for some design points.

It is observed that all protocols start with low overheads at low node counts, however as

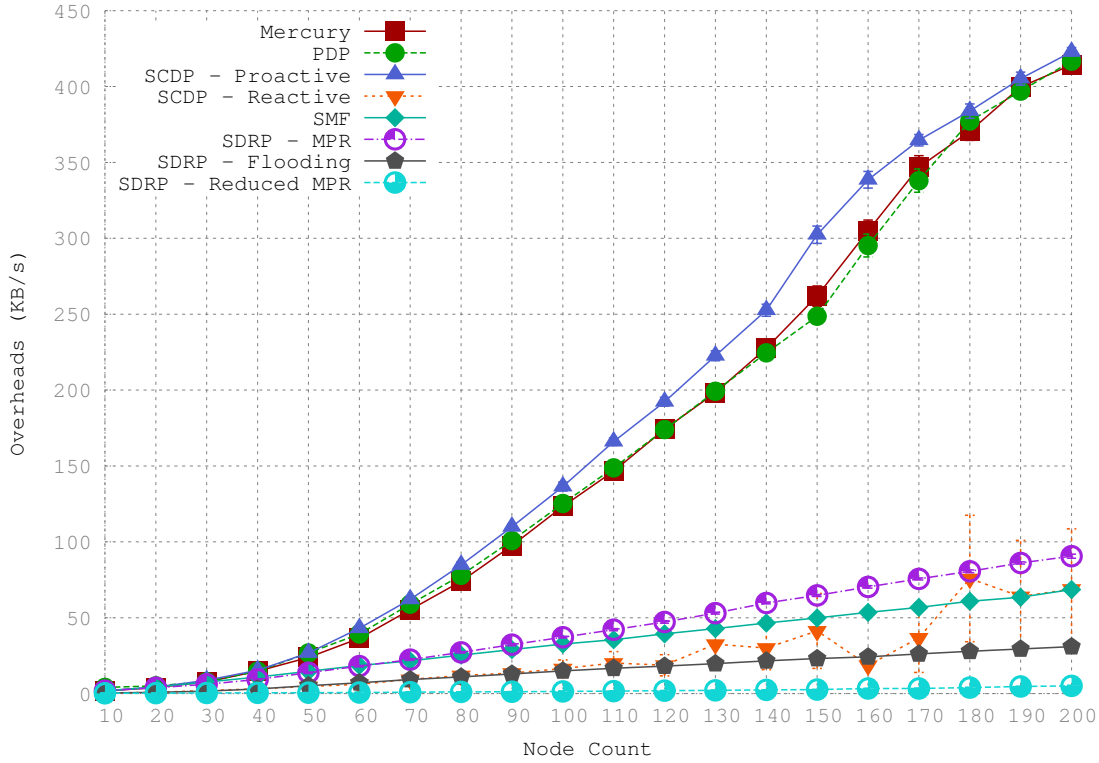


Fig. 5.7: Protocol control overheads with varying node count. Control overheads are defined as the total packet traffic sent by both the routing protocol and service discovery protocol. Error bars describe 95% confidence intervals for each design point.

the node count increases approaches based on address-based routing protocols tend to exhibit significantly increased overheads; particularly those based on OLSR. As more nodes are added to the network, the number of sources of HELLO and TC message traffic increases, thus increasing overall control overheads.

SMF and SCDP-RA do not exhibit a similar increase in overheads. AODV floods route requests in order to locate hosts and in this network; only clients are expected to generate such requests, with other nodes taking a passive role as request rebroadcasters. Overheads for these protocols can be seen to increase relatively slowly with increasing node count, as new nodes act only as rebroadcasters of route requests rather than flood sources.

The SDRP family of protocols exhibits favourable overhead generation when compared with protocols based on OLSR, as only service agents act as sources for disseminated messages. Thus, with increasing node count, additional control messages only result from service adver-

tisement rebroadcasts and HELLO message broadcasts. SDRP-F exhibits a slow increase in overheads with increasing node counts as expected; as additional nodes are added to the network, the number of service advertisement broadcasts also increases. SDRP-MPR exhibits higher overheads than the other SDRP algorithms and both of the reactive protocols. This is likely due to the fact that the savings resulting from the use of an MPR selection algorithm are cancelled out by the control overheads required for periodic broadcast of HELLO messages when the service agent count is low. This hypothesis will be tested in a later experiment, when the service agent count is varied in a network with a static node count.

SDRP-RMPR exhibits the lowest overheads of all of the evaluated approaches. This is expected, as the presence of a small number of service agents results in fewer service advertisement sources while the border-node selection algorithm results in further overhead reduction. In the next section, we will describe the measurement of application throughput obtained during this experiment.

5.2.2 Service Availability

Service availability is defined as the ratio of successful service discovery attempts to the total number of service requests made. It is a key performance metric for any protocol, as the ability for a user agent to utilize a service depends on the successful discovery of that service. Later measurements of application throughput and the delivery ratios for request and response packets will depend upon service availability.

Figure 5.8 describes the service availability exhibited by each protocol during this experiment. It is important to note that NS-2 prioritizes routing protocol traffic during simulations, making it more likely that those packets will be delivered successfully to their destinations. Cross-layer protocols such as Mercury and SMF, as well as the SDRP protocol family, will tend to exhibit high availability despite high control overheads. Application-layer protocols experience a greater degenerative effect from increasing control overheads, as application-layer traffic is prioritized lower than routing-layer traffic.

This is reflected in the results obtained for service availability. The increased control overheads observed in the previous section do not have an adverse effect on cross-layer approaches, as their advertisement and query messages are more likely to be delivered. Conversely, application-layer approaches have difficulty delivering messages. The effect of high control overheads will be more obvious in the measurements of throughput and service message delivery ratios described in later sections.

Application-layer protocols based on OLSR experience the greatest drops in service availability with increasing node count, correlated strongly with high control overheads. PDP exhibit

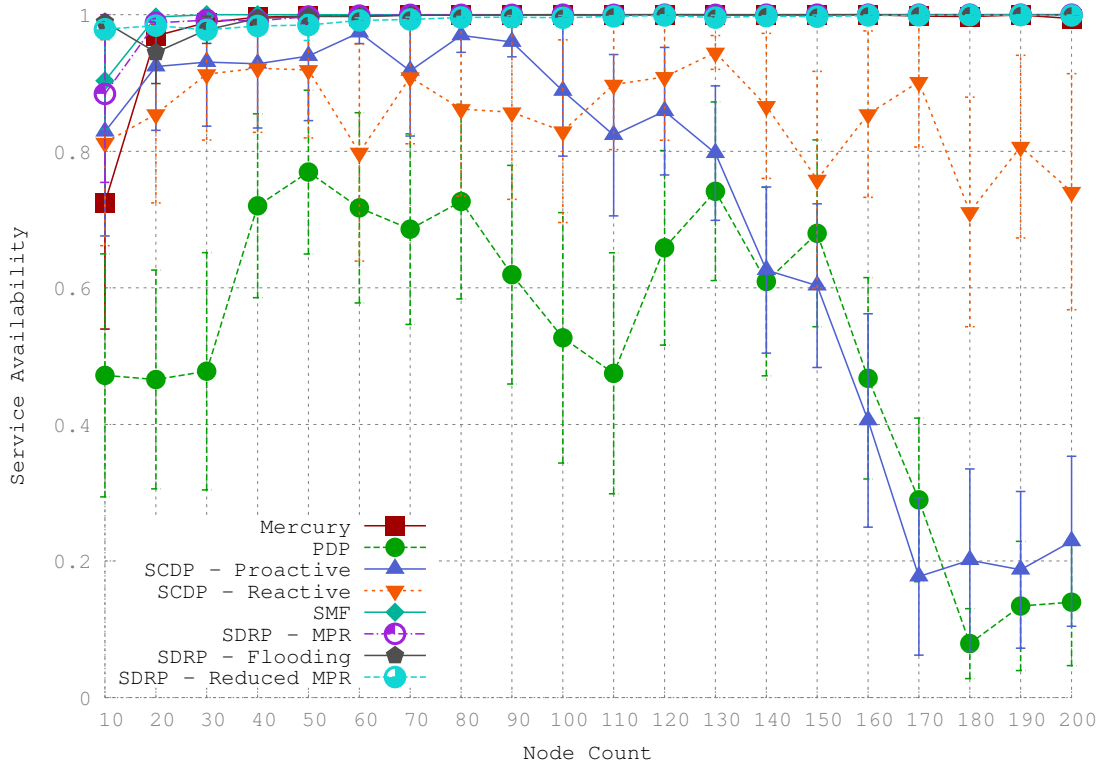


Fig. 5.8: Service availability with varying node count. Service availability is defined as the ratio of successful service discoveries to the total number of attempted discoveries. Error bars describe 95% confidence intervals for each design point.

relatively low discovery success rates throughout the evaluation, contributing to lower message delivery ratios and overall application throughput. PDP is primarily designed to reduced energy consumption and includes features that delay message sending for that purpose. ServiceAgent is configured to queue request messages for 5 seconds; if a positive discovery response is not received from the discovery protocol within that time, the request is considered a failure. Thus, PDP is more likely to exhibit low performance when compared with the other evaluated protocols.

SCDP-RA does not experience a significant drop in service availability, this agrees with our hypothesis that there is a correlation between high overheads and low service discovery message delivery success rates for application-layer approaches. SCDP-RA's control overheads remained low relative to PDP's and SCDP-PAs during this experiment, thus its ability to deliver service discover messages was not compromised.

The SDRP protocol family exhibits favourable service availability throughout the exper-

iment, indicating that each of the employed service advertisement dissemination approaches functioned correctly. In the next section, we will describe the measurements of service request message delivery ratio obtained from this experiment.

5.2.3 Request Delivery Ratio

The ServiceAgent installed on each deployed user agent is configured to make a service request to the node's service discovery protocol every second. If the discovery protocol successfully locates a service agent, ServiceAgent will then attempt to send a service request message to that service agent. The request delivery ratio is the ratio of the service requests successfully received by service agents to the number sent by user agents. It indicates the routing protocol's ability to successfully route packets from a user agent to a service agent, as well a service discovery protocol's ability to successfully identify a service agent.

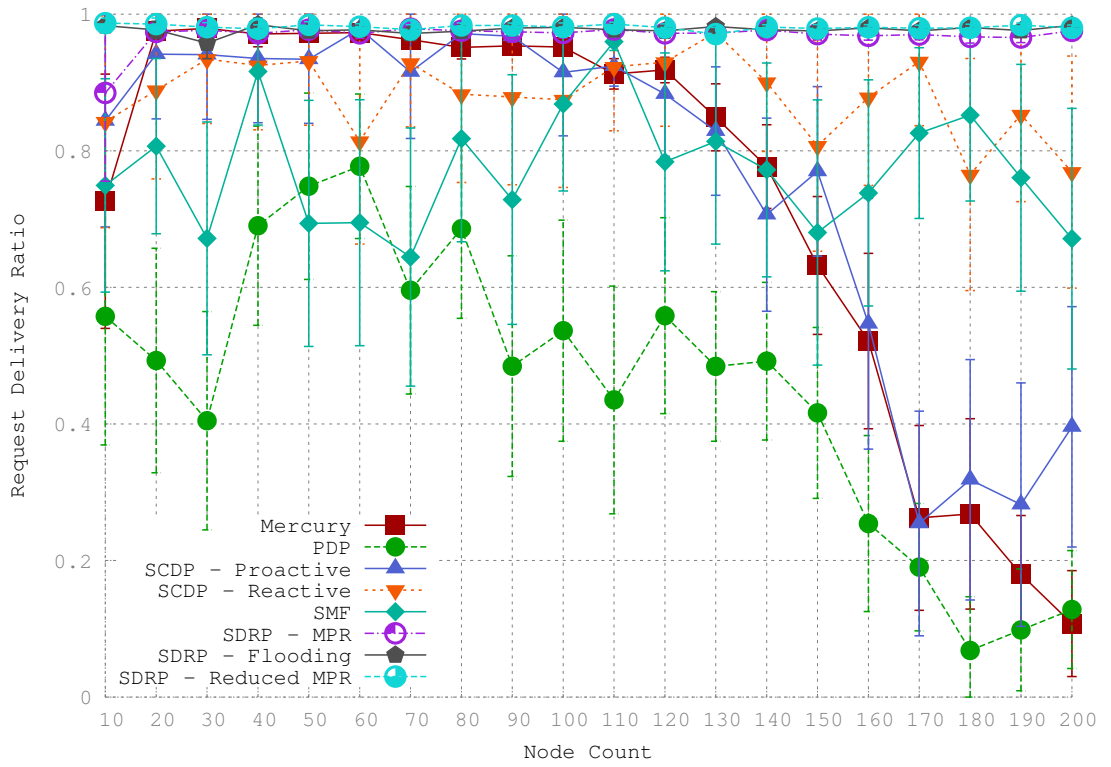


Fig. 5.9: Request delivery ratio with varying node count. Defined as the ratio of successfully delivered service request messages to the total number sent. Error bars describe 95% confidence intervals for each design point.

Figure 5.9 shows the request delivery ratios measured during this experiment. Here we can see that the bias introduced by NS-2's prioritization of routing packets is no longer apparent, as packets sent by the ServiceAgent are considered to be application-layer with a normal priority, regardless of the discovery protocol used.

The three protocols based on OLSR; PDP, Mercury and SCDP-PA, all experience significant reductions in their ability to successfully deliver request packets as the node count increases. Conversely, SMF and SCDP-RA experience no significant decrease; correlating with the relatively slow overhead increase exhibited by those protocols as the node count increases. However, the confidence intervals around their measured means are relatively large, indicating significant variation in those protocols' ability to route messages to service agents. Their initial performance is also lower than those of protocols based on proactive routing, indicating that routes discovered by AODV are more prone to packet loss than those discovered by OLSR or formed by SDRP service advertisements. AODV caches discovered routes for 10 seconds by default, with link breakage more likely to occur with increasing time from the point at which a route is formed due to node mobility. Conversely, OLSR updates routes using disseminated TC messages at 5 second intervals; leading to improved route stability, as observed in this experiment.

Mercury and SCDP-PA show consistent performance with node counts in the range of 10-120 before dropping off significantly at higher counts. Similarly, the SDRP protocol family shows little variation in delivery ratio, indicating that approaches in which routes are established proactively tend to exhibit more stable behaviour. PDP's performance remains low here as well, displaying a compromised ability to map service identifiers to node addresses which may indicate a problem with the protocol's implementation.

At the highest node counts, it is observed that the SDRP protocol family maintains consistently high request delivery ratios, with no significant decreases observed with increasing node counts. When compared with protocols based on address-based routing, SDRP performs favourably; combining the stability of proactive approaches with low control overheads. In the next section, the response delivery ratio measurements will be discussed.

5.2.4 Response Delivery Ratio

ServiceAgent instances running on nodes that are designated as service agents, upon receipt of service request messages, will return a response message to the source user agent. This reflects the request-response pattern employed by many services in infrastructure networks and allows us to measure a protocol's ability to successfully route messages between service agents and user agents. Approaches using address-based routing protocols have the ability to discover routes between service agents and user agents as required. However, SDRP establishes reverse-routes

to user agents when intermediate nodes route messages between user agents and service agents. The results of these measurements will indicate the robustness of this approach.

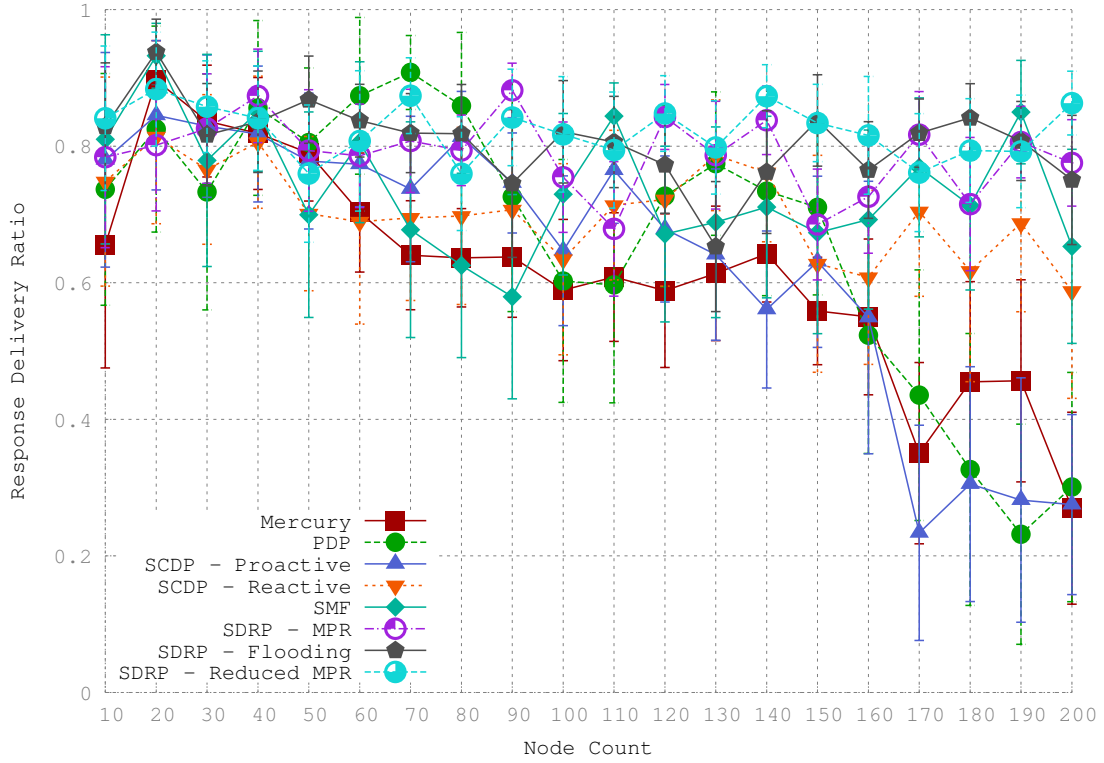


Fig. 5.10: Response delivery ratio with varying node count. Defined as the number of successfully delivered service response messages to the total number sent. Error bars describe 95% confidence intervals for each design point.

Figure 5.10 shows the response delivery ratios measured during this experiment. Greater overall stability is observed here when compared with the request delivery ratio measurements discussed in the previous section. The response delivery ratio is a good indicator of a protocol's ability to route packets and is unaffected by the discovery protocol's ability to accurately discover services.

All protocols exhibit high delivery ratios with low node count, with drop-offs being experienced by proactive approaches as the node count increases, similar to the drop-off in request delivery ratios observed in the previous section. This correlates with the rapid increase in control overheads exhibited by those protocols and indicates that failures to deliver response messages are likely due to higher medium contention. The results for SDRP and reactive approaches agree with this conclusion as they experience minimal drop-off in delivery success rates with increasing

node count, reflecting their lower control overheads.

The SDRP family of protocols maintains consistently high response delivery ratios during this experiment, indicating that its reverse-route approach functions adequately to facilitate communication in both directions between user and service agents. In the next section, we will describe the application throughput measurements obtained during this experiment.

5.2.5 Application Throughput

Application throughput is defined as the average traffic throughput for communications between user agents and service agents over time. The total amount of application traffic received by user and service agents is divided by the number of user agents to get per-agent traffic totals and then divided by the time span between the first packet receive event and the last, resulting in a measurement of average user agent throughput per second.

Figure 5.11 shows the application throughput recorded during this experiment. Here a strong correlation is observed between high overheads and low throughput. The family of protocols based on OLSR experience a decrease in throughput with increasing node count, as control message overheads come to dominate the medium and the delays increase. The drops in throughput exhibited by SCDP-PA and Mercury are similar, with Mercury performing better due to its cross-layer design. PDP performs poorly compared to both Mercury and SCDP-PA due its inability to successfully discover services and deliver service requests.

Both SMF and SCDP-RA are outperformed by proactive approaches with lower node counts. This reflects the higher end-to-end delay that is experienced by reactive approaches, as they must initiate route requests before packets can be forwarded to their destination. Proactive approaches can immediately forward packets as they discover routes ahead of time. Both SCDP-RA and SMF also exhibit greater degrees of variation in their results which is likely to reflect variation in route discovery delays. As the node count increases, SCDP-RA and SMF outperform Mercury, PDP and SCDP-PA due to their lower control overheads. SCDP-RA achieves lower throughput than SMF, both due to the fact that SMF is a cross-layer approach and because SMF is able to maintain high service availability at high node counts due to the bias introduced by NS-2's packet prioritization.

The SDRP family of protocols exhibits consistently high application throughput, reflecting low overheads, high service availability and high delivery ratios as described in previous sections. The difference in performance between SDRP-F, SDRP-MPR and SDRP-RMPR strongly reflects the control overheads produced by each protocol. SDRP-RMPR produces the lowest control overheads and thus achieves the highest throughput. SDRP-MPR exhibits the highest overheads

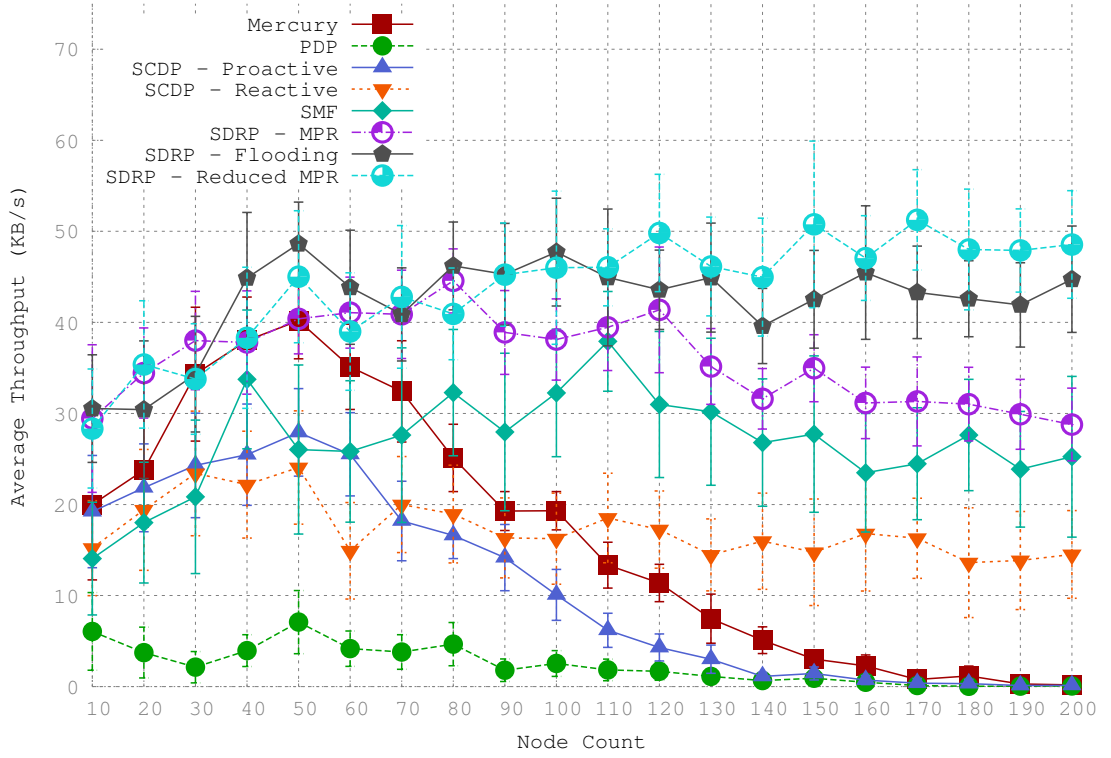


Fig. 5.11: Application throughput with varying node count. Application throughput is defined as the average amount of application traffic received per user agent. Error bars describe 95% confidence intervals for each design point.

of the three and thus has the lowest throughput, with SDRP-F falling between the two protocols in both metrics.

5.2.6 Summary

This experiment varied the number of nodes deployed in a topography of static size, with a static number of service and user agents, with an aim to determine the ability for the evaluated protocols to perform with increasing network density. A total of 3,200 simulations were run, with the node count increasing from 10 to 200 in increments of 10; resulting in 20 replicates per unique design point.

The control message overheads generated by the evaluated protocols closely reflected the nature of their underlying routing protocol. Those protocols based on proactive approaches such

as SCDP-PA, Mercury and PDP exhibit rapidly increasing overheads as the number of deployed nodes increased. SCDP-RA and SMF, two protocols based on reactive routing, were affected less significantly, as additional nodes act only as rebroadcasters rather than message sources.

The SDRP family of protocols displayed favourable results, with SDRP-RMPR performing well due to its border-casting approach and lack of periodic HELLO message broadcasts. SDRP-F displayed a steady increase in control overheads with increasing node count, similar to that of reactive approaches, as additional nodes rebroadcast service advertisements, thus contributing to control message traffic.

SDRP-MPR failed to outperform SDRP-F in terms of overhead reduction. This is likely due to the fact that SDRP-MPR nodes broadcast HELLO messages at 2 second intervals, while SDRP-F service advertisement messages are disseminated every 5 seconds, reflecting the HELLO and Advertisement interval settings used for this experiment. Thus, SDRP-MPR's HELLO traffic exceeded the bandwidth savings resulting from the use of the MPR algorithm. Later experiments with variable numbers of service agents can be expected to better test the control overhead savings afforded by the MPR algorithm.

Service availability measurements exhibited a bias that was attributed to NS-2's prioritization of routing protocol messages over application-layer messages. Application-layer discovery approaches exhibited drops in service availability with increasing node count that were not observed in cross-layer approaches. Here PDP was shown to perform poorly when compared with other approaches; this reflects its design goal of reduced energy consumption rather than performance. SCDP-PA exhibited high service availability at low-moderate node counts, with availability dropping at higher node counts. This correlated strongly with the increased overheads generated by that protocol. Cross-layer protocols and the SDRP protocol family all displayed high service availability throughout the experiment.

Measurements of request and response delivery ratios were unbiased by NS-2's packet prioritization system and showed a strong inverse correlation between protocol control overheads and delivery success rates. Those protocols generating low overheads, such as the reactive protocols and SDRP, tended to perform well while proactive high-overhead protocols performed relatively poorly. SDRP's reverse route approach performed well in this experiment, with high response delivery ratios measured at all design points.

The ability for each protocol to discover services and deliver messages was summarized by our application throughput measurements. Here, proactive approaches tended to suffer significant drops in throughput that correlated strongly with control overhead increases; drops which were not experienced by reactive protocols and SDRP. The throughput of the SDRP family of protocols was consistently higher than that of the other measured protocols, reflecting its low control overheads, high service availability and high delivery success rates.

5.3 Service Agent Density

Our second experiment measures the effect of service agent density on the evaluated protocols. In the previous experiment, the service agent density remained static as the node count increased; this tended to favour reactive approaches, such as SCDP-RA and SMF, and SDRP due to the fact that service agents act as sources for service advertisement messages. This experiment will measure SDRP's ability to cope with increasing numbers of service agents and will test the control overhead reduction approaches of SDRP-MPR and SDRP-RMPR in particular, with SDRP-F acting as a useful baseline for comparison.

Reactive approaches are expected to perform well, as increasing service agent density can be expected to reduce the delay associated with route discovery, with service agents being more likely to be near to query sources. SMF service agents periodically send HELLO messages in order to facilitate discovery without query flooding; with a low number of user agents this feature may work against SMF in terms of generated control overheads.

Category	Parameter	Value
Network	Node Count	100
	Topology Width	1000m
	Topology Length	1000m
Node & Wireless	Protocol	SDRP-F, SDRP-MPR, SDRP-RMPR, SMF, SCDP-RA, SCDP-PA, Mercury, PDP
	Node Speed	2m/s
Routing Protocol	HELLO Interval	2 Seconds
	Advertisement Interval	5 Seconds
Service Discovery	User Agent Count	10
	Service Agent Count	Varying between 10 and 90 in increments of 10
	Service Count	10

Fig. 5.12: Parameter values used for the service agent density experiment.

Figure 5.12 shows the parameter values used during this experiment. The node count, user agent count and service count remain static as the number of service agents increases. A total of 720 experimental runs were carried out; 8 routing protocols with 9 user agent count configurations replicated 10 times in order to account for random variation. In the next section we will describe the control overhead measurements obtained from this experiment.

5.3.1 Control Overheads

Figure 5.13 shows the control overheads measured during this experiment. Here SDRP-F can be seen to exhibit rapidly increasing overheads as the service agent count increases. With increasing service agent count, the number of service advertisement flood sources increases. This experiment illustrates the advantages of the selective rebroadcast strategies used by SDRP-MPR and SDRP-RMPR; the control overheads for both protocols remain low and increase slowly in comparison to the pure flooding approach, staying below even the OLSR-based protocols throughout the experiment.

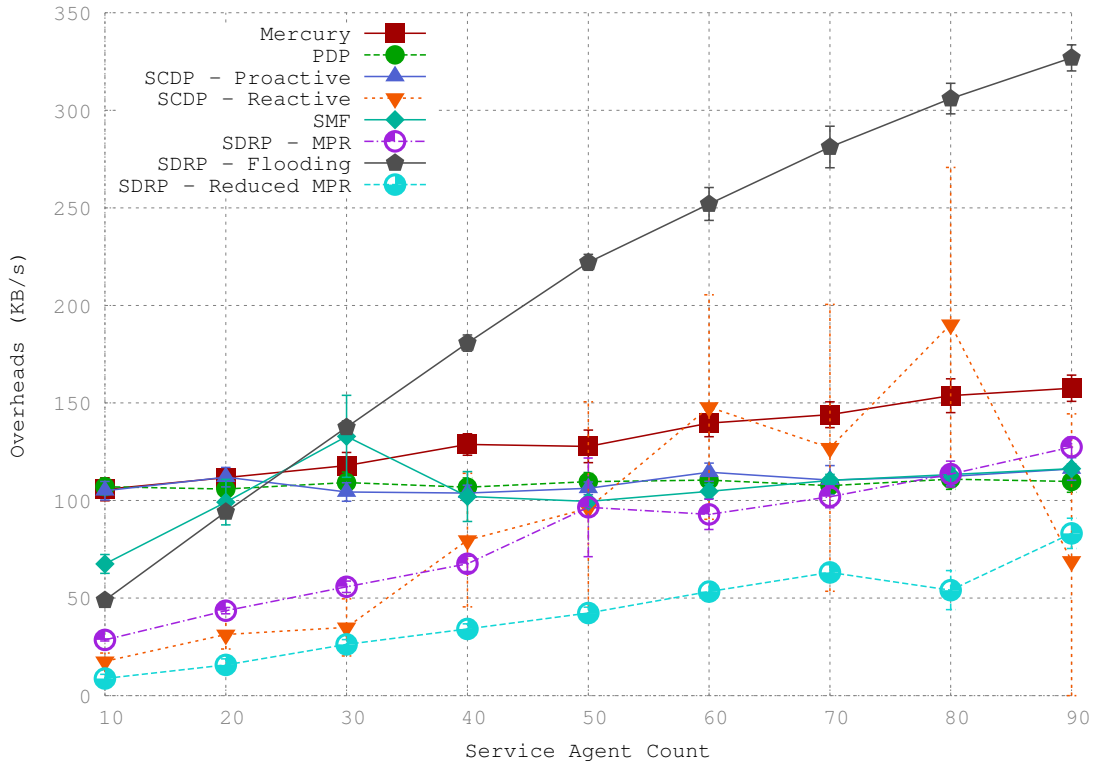


Fig. 5.13: Control overheads with varying service agent count. Control overheads are defined as the total packet traffic sent by both the routing protocol and service discovery protocol. Error bars describe 95% confidence intervals for each design point.

SCDP-RA displays a large degree of variation in control overheads during this experiment that is not replicated by SCDP-PA or SMF. This may be attributed to SCDP's periodic directory registration approach. SCDP service agents are required to re-register their services with known directories at set periods. This does not result additional overheads for SCDP-PA

due to OLSR's proactive route discovery. However, SCDP-RA's service registration messages will prompt request floods from AODV if a route to the target directory is not cached. As the number of service agents increases, the potential for flooded route requests increases as well, with the variation correlating with the probability that a cached route is available at the time of the request.

Mercury exhibits a control overhead increase of approximately 50% over the course of the experiment. Each Mercury service agent appends a fixed-length service filter to OLSR topology control messages, increasing message length and, thus, control overheads. Rebroadcasts of larger TC messages by MPR nodes also contributes to the increase in control overheads. SMF's control overheads remain stable throughout the experiment and are initially higher than those of SCDP-RA, likely due to SMF's incorporation of periodic HELLO messages by service agents. However, with increasing service agent count, SCDP-RA's control overheads exceed those of SMF; this may also be attributed to the requirement for periodic directory service registration, which SMF lacks. In the next section, service availability measurement will be described.

5.3.2 Service Availability

Figure 5.14 shows the service availability measurements obtained from this experiment. With increasing service agent density, service availability increases for all protocols. This can be attributed to the increased likelihood that a service agent will be closer to an active user agent as the service agent density increases.

Application-layer approaches suffer from low availability and high variation with few service agents in the network, with the bias likely due to NS-2's packet prioritization system. In the next section, request message delivery ratio measurements will be described.

5.3.3 Request Delivery Ratio

Figure 5.15 shows the request delivery ratio measurements recorded during this experiment. Similar to service availability measurements, all protocols exhibit an increased delivery ratio with increasing service agent density which can be attributed to the increased probability that service agents and active user agents will be in close proximity.

SCDP-RA and SCDP-PA both display a large degree of random variation and lower performance with lower numbers of service agents. The directory-based discovery approach used by both protocols introduces a disconnect between service availability and route availability measurements. SCDP satisfies service discovery requests by querying available directories, however

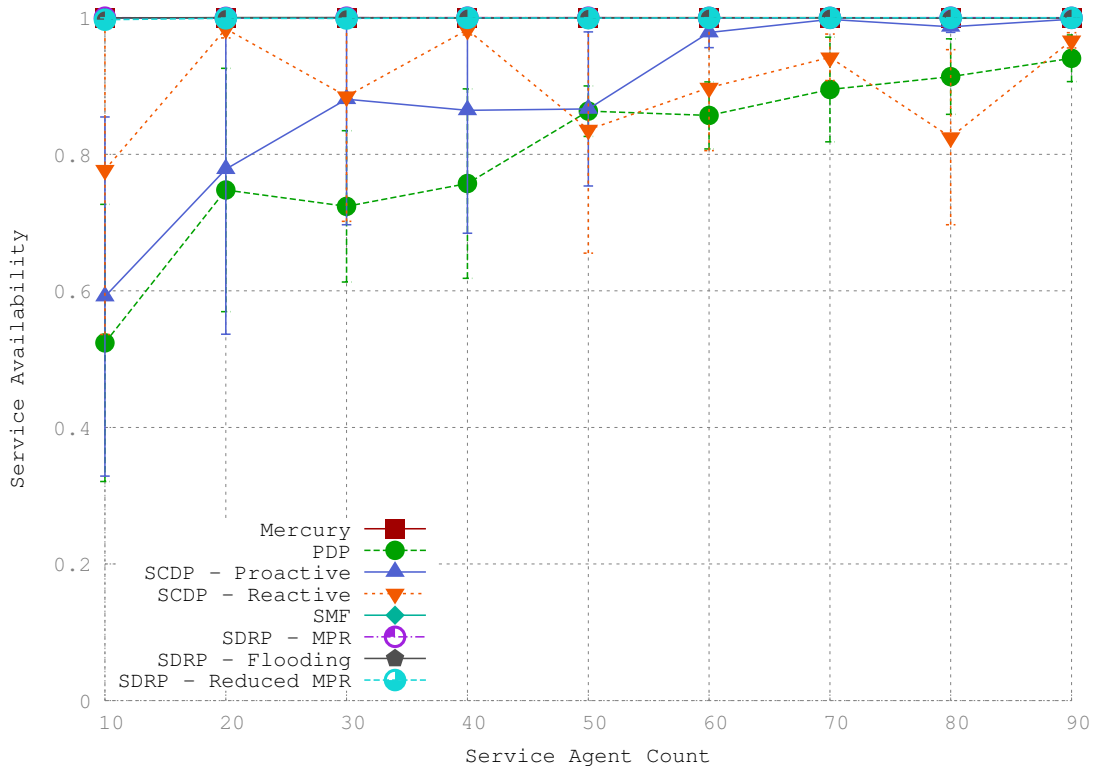


Fig. 5.14: Service availability with varying service agent count. Service availability is defined as the ratio of successful service discoveries to the total number of attempted discoveries. Error bars describe 95% confidence intervals for each design point.

this does not mean that a route to the target service agent is available, only that a route existed between the service agent and the directory. With peer-to-peer protocols, there is a higher likelihood that a route exists between the user agent and service agent if the discovery attempt succeeded, as advertisements or query responses were received directly from the target service agent.

If a requested service is reported as available, ServiceAgent will call **SendPacket** on the parent service discovery protocol, which will then attempt to route the request message to the target service agent. The other disadvantage of the directory-based approach is that the true distance, in hops, to a target service agent cannot be determined based on the query response received from a directory; thus SCDP may attempt to a route to service agents that are further away than necessary, resulting in increased variation in delivery ratios and low performance as route lengths increase. In the next section, response delivery ration measurements will be described.

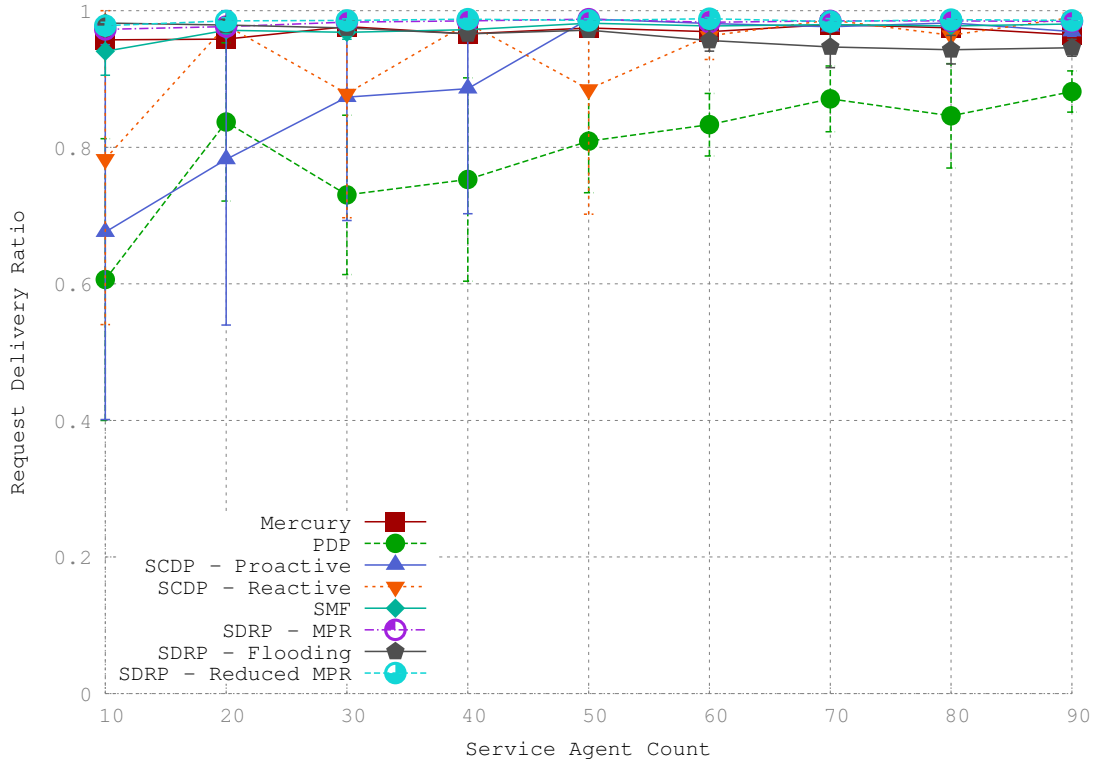


Fig. 5.15: Request delivery ratio with varying service agent count. Defined as the ratio of successfully delivered service request messages to the total number sent. Error bars describe 95% confidence intervals for each design point.

5.3.4 Response Delivery Ratio

Figure 5.16 shows the response delivery ratios recorded during this experiment. Here the divide between directory-based and peer-to-peer approaches is reduced, with less significant differences seen in recorded results at low service agent density. The disconnect between service availability and request delivery does not apply in this context, as a response sent from a service agent does not require consultation with a directory for discovery.

Large variations are observed in the results for directory-based approaches that may be attributed to less efficient service agent selection resulting in longer routes and an increased likelihood of collisions and errors when compared with the peer-to-peer approaches. This implies that peer-to-peer approaches are better-suited to reliable service discovery and client-server communication in multi-hop networks, as the distance to service agents can be determined during discovery and used to select the closest match.

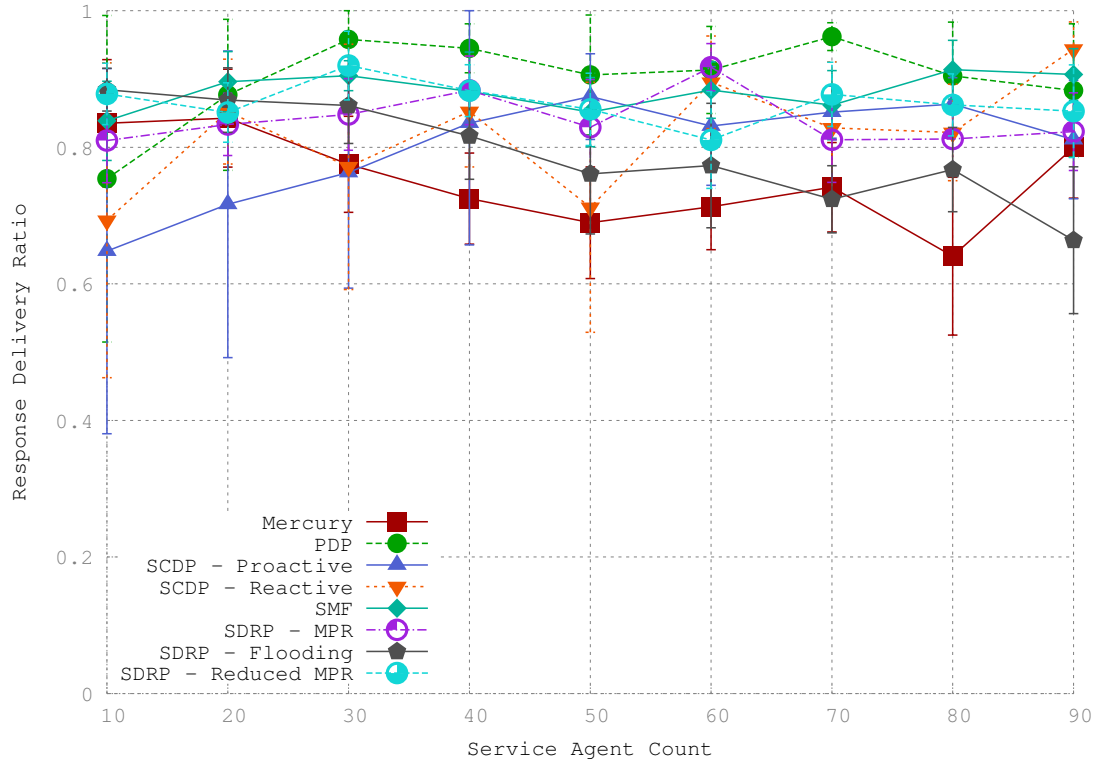


Fig. 5.16: Response delivery ratio with varying service agent count. Defined as the number of successfully delivered service response messages to the total number sent. Error bars describe 95% confidence intervals for each design point.

SDRP-F exhibits a significant drop in delivery rate as the number of service agents increases when compared to SDRP-MPR and SDRP-RMPR; this correlates with the more rapid increase in control overheads observed in section 5.3.1 for that protocol. A similar but less significant decrease in request delivery ratios was seen in the previous section as well. In the next section, application throughput measurements will be discussed.

5.3.5 Application Throughput

Figure 5.17 describes the application throughput measurements obtained from this experiment. A significant decrease in throughput of approximately 70% is observed for SDRP-F in this experiment. Given the less significant decreases seen in SDRP-MPR and SDRP-RMPR, this strongly indicates that a correlation exists between SDRP-F's rapid control overhead increase and the observed decrease in application throughput.

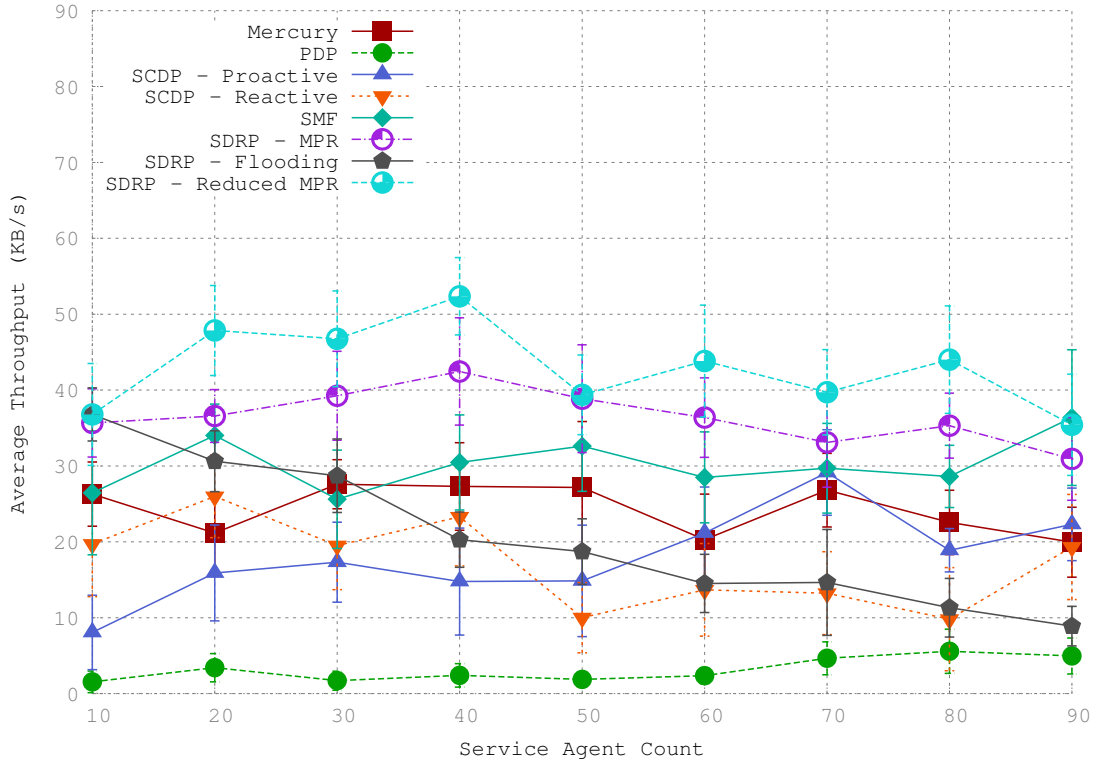


Fig. 5.17: Application throughput with varying service agent count. Application throughput is defined as the average amount of application traffic received per user agent. Error bars describe 95% confidence intervals for each design point.

This same correlation is observed for the other evaluated protocols; those with the highest control overheads tend to have the lowest application throughput, with SDRP-RMPR's low overheads resulting in the highest recorded throughput measurements.

5.3.6 Summary

In this experiment, the number of service agents deployed in the simulated network was varied between 10 and 90 in increments of 10, with other factors remaining static, in order to ascertain the effect of service agent density on protocol performance.

The contrast between the control overhead generation of the various versions of SDRP was significant during this experiment, with SDRP-F's advertisement flooding algorithm causing a rapid increase in overheads with increasing service agent density. Conversely, SDRP-MPR and

SDRP-RMPR exhibited a slower decrease due to their selective rebroadcasting mechanisms, with SDRP-RMPR displaying the lowest control overhead generation

Significant variations were observed in measurements of message delivery ratios for SCDP. It is hypothesized that the directory-based discovery approach results in an increased probability that no route exists between a user and service agent, despite a successful query response from directory nodes. The inability of SCDP to discern the distance, in hops, to service agents due to its indirect discovery approach is hypothesized to be a contributing factor here as well.

This implies that directory-based approaches are not well-suited to deployments in which high discovery and delivery reliability are required. However, the directory-based approach used by SCDP resulted in lower control overheads when compared with the peer-to-peer service advertisement approach used by Mercury, particularly in the case of SCDP-PA, which does not require new route discovery to facilitate service agent registrations with directory nodes. For greater service agent density, however, SCDP-RA exhibited rapidly increasing control overheads attributable to increasing incidence of AODV route request floods by service agents attempting to register with directory nodes.

In the next experiment, the effect of increasing user agent density on the evaluated protocols will be assessed.

5.4 User Agent Density

Our third experiment measures the effect of varying user agent density on the evaluated protocols. User agent density should not significantly affect the control overheads generated by SDRP and the proactive protocols, as protocol messages are primarily generated by service agents. However, reactive protocols that rely on service querying such as SCDP-RA and SMF are expected to exhibit an increase in control overheads as the incidence of query flooding increases with increasing user agent count.

Figure 5.18 shows the parameters used for this experiment. The node count, service agent count and service count are kept static as the number of user agents is increased. For SCDP-PA and SCDP-RA, 2 nodes are designated as directories. A total of 720 experimental runs were carried out; 8 routing protocols with 9 user agent count configurations replicated 10 times in order to account for random variation. In the next section, the measured control overheads for this experiment will be described.

Category	Parameter	Value
Network	Node Count	100
	Topology Width	1000m
	Topology Length	1000m
Node & Wireless	Protocol	SDRP-F, SDRP-MPR, SDRP-RMPR, SMF, SCDP-RA, SCDP-PA, Mercury, PDP
	Node Speed	2m/s
Routing Protocol	HELLO Interval	2 Seconds
	Advertisement Interval	5 Seconds
Service Discovery	User Agent Count	Varying between 10 and 90 in increments of 10
	Service Agent Count	10
	Service Count	10

Fig. 5.18: Parameter values used for the user agent density experiment.

5.4.1 Control Overheads

Figure 5.19 shows the control overheads measured for each protocol during this experiment. As expected, SDRP and the proactive approaches maintain consistent overhead generation throughout the experiment while the reactive approaches exhibit significantly increasing overheads with increasing user agent count.

SCDP-RA shows lower control overheads than SMF during this experiment. SCDP, unlike SMF, does not send periodic HELLO messages and benefits from directory-based discovery as user agents do not need to flood service queries when a service is requested. Both SCDP-RA and SMF exhibit increased over time, attributable to an increased incidence of route request floods by AODV.

SDRP-RMPR generates the lowest control overheads throughout the experiment, with SDRP-MPR and SDRP-F generating significantly lower overheads than SCDP-RA from 40 user agents onwards. In the next section, service availability measurements will be discussed.

5.4.2 Service Availability

Figure 5.20 shows the service availability measured during this experiment. Availability conforms to measurements obtained during previous experiments, with SDRP and cross-layer protocols maintaining high availability due to NS-2's prioritization of routing-layer packets.

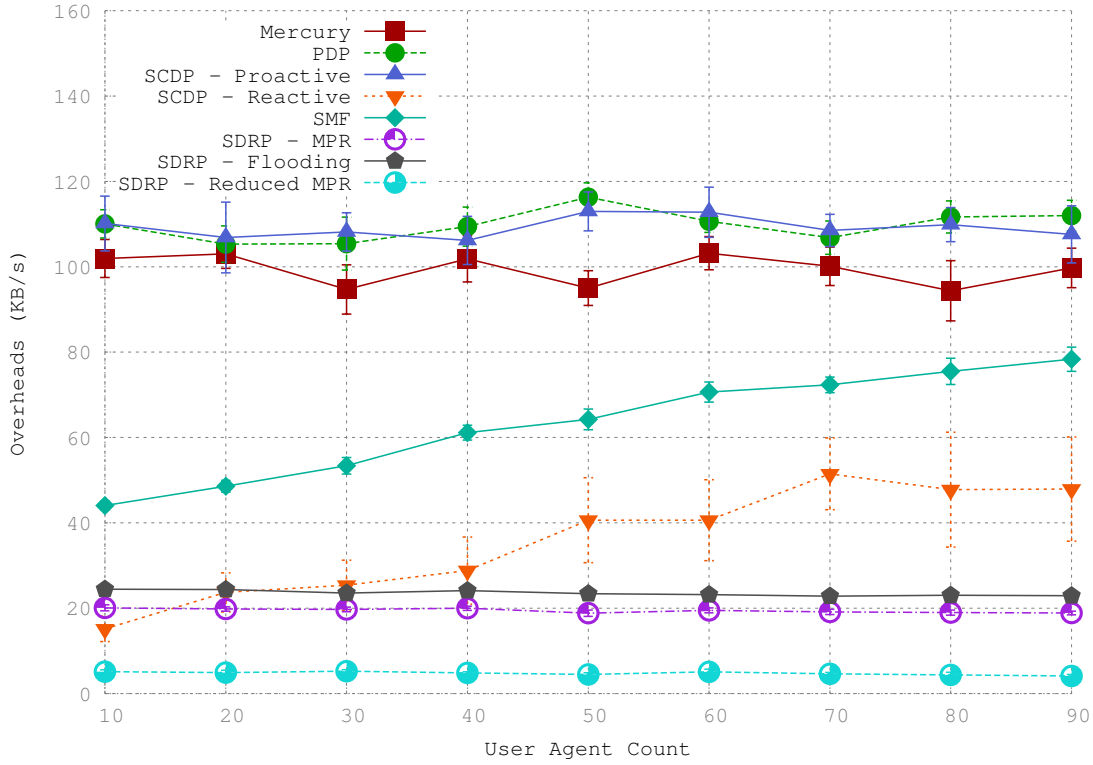


Fig. 5.19: Control overheads with varying user agent count. Application throughput is defined as the total amount of application traffic consumed over time. Error bars describe 95% confidence intervals for each design point.

PDP display a low initial availability before increasing inline with other protocols at 30 user onwards, the reason for this is unclear. Little significant variation is seen in SCDP-RA or SCDP-PA, likely due to their similar design and use of directories for service querying. In the next section, measurements of service request delivery ratios will be described.

5.4.3 Request Delivery Ratio

Measurements of service request message delivery ratios are unaffected by the bias introduced by NS-2's packet prioritization. Figure 5.21 shows the measurements obtained during this experiment. PDP's low delivery ratio is maintained here at low user agent counts before increasing later in the experiment, similar to measurements observed in previous experiments.

All protocols exhibit significantly decreasing request delivery ratios with increasing user

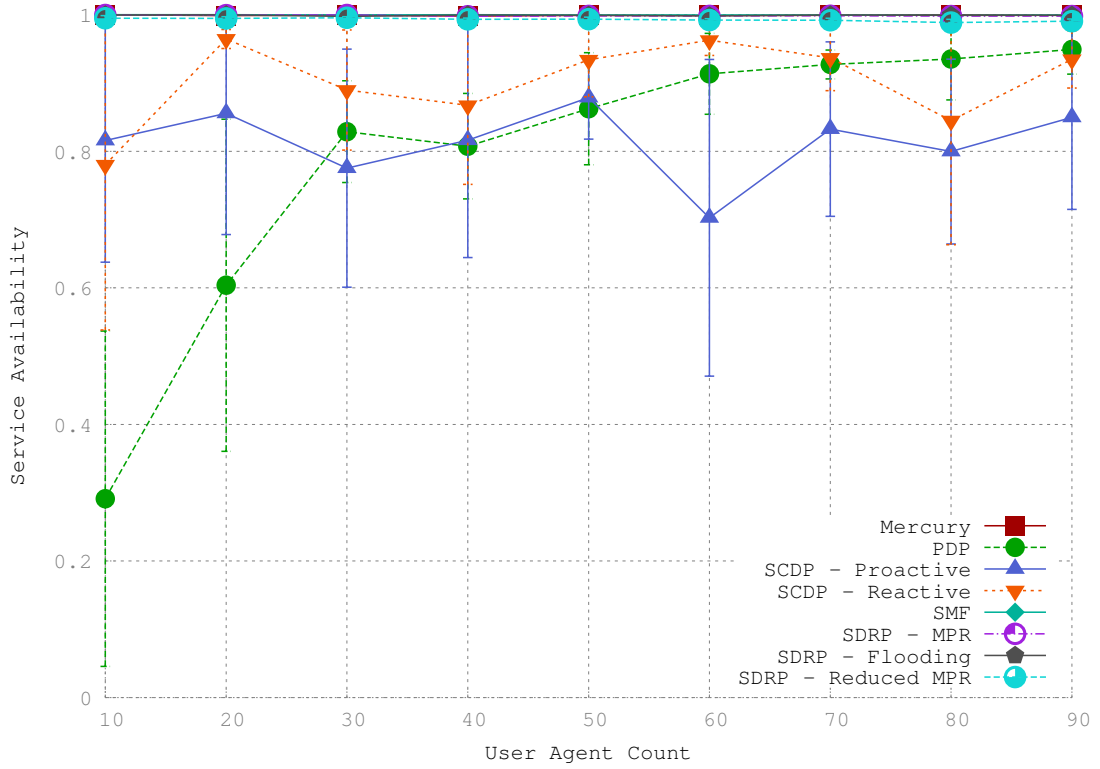


Fig. 5.20: Service availability with varying user agent count. Service availability is defined as the ratio of successful service discoveries to the total number of attempted discoveries. Error bars describe 95% confidence intervals for each design point.

agent count. As the number of user agents increases, the number of distinct unicast packet flows also increases. The 802.11 specification's MAC layer operates using CSMA/CA with an exponential back-off. As the number of unicast packet sources within a given area, so too does the potential for RTS collisions, back-offs and eventual drops as the packet send retry limit is exceeded. With increasing number of user agents, therefore, the request delivery ratio can be expected to drop for all protocols, regardless of their routing or discovery mechanism.

This is an interesting result that illustrates a problem with client-server communication in MANETs; when the number of user agents is high and the number of service agents is low, the medium in the area of highly utilized service agents will become increasingly saturated, reducing communication throughput. The next section will describe the response delivery ratios measured during this experiment.

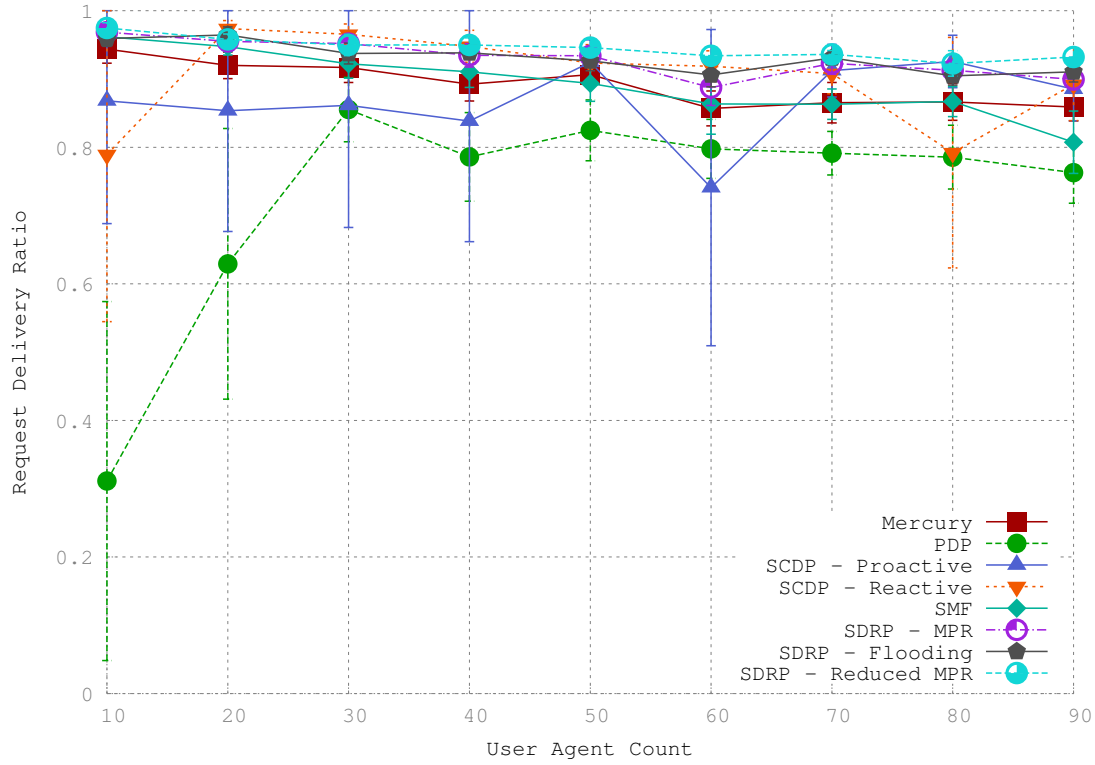


Fig. 5.21: Request delivery ratio with varying user agent count. Defined as the ratio of successfully delivered service request messages to the total number sent. Error bars describe 95% confidence intervals for each design point.

5.4.4 Response Delivery Ratio

The response delivery ratios observed during this experiment are shown in figure 5.22. The fall in request delivery ratio observed in the previous section is exacerbated here, with response delivery ratio falling even further with increasing user agent density.

This may be attributed to the same MAC-layer dropping effect hypothesized as the cause for request packet loss in the previous section. Service requests are routed towards service agents, thus areas with a higher service agent density will experience higher medium contention as more packets enter. This phenomenon is illustrated in figure 5.23.

Response messages are sent by service agents from the center of these contention focal points, increasing the probability that requests will be dropped as the number of incoming messages increases with the user agent count. The effect of this phenomenon can be observed in the response delivery ratios of all protocols. A service discovery solution which distributed

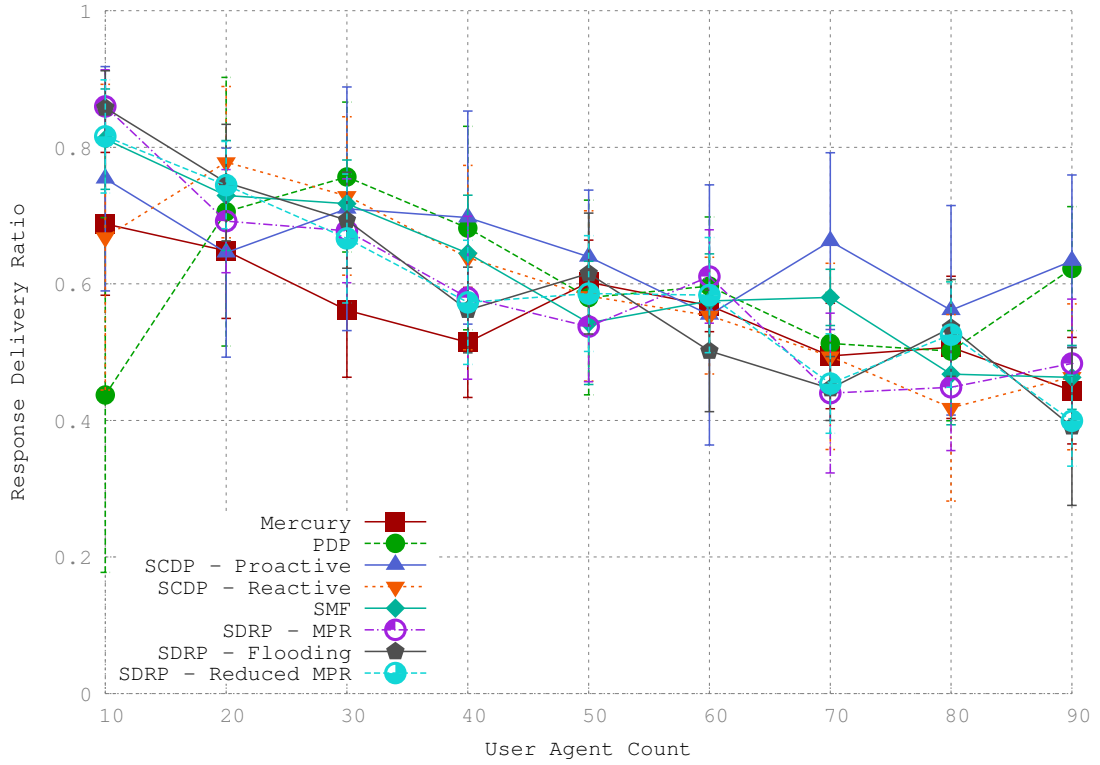


Fig. 5.22: Response delivery ratio with varying user agent count. Defined as the number of successfully delivered service response messages to the total number sent. Error bars describe 95% confidence intervals for each design point.

requests among multiple available service agents, regardless of distance from the user agent, might be better suited to some network configurations as it would help to alleviate this focal effect. In the next section, application throughput measurements will be described.

5.4.5 Application Throughput

Figure 5.24 shows the average throughput per-user agent recorded during this experiment. Here a strong correlation is seen between increasing user agent count and decreasing throughput; as more user agents generate application traffic, medium contention increases and medium access time decreases for all traffic flows.

Mercury shows poor initial performance which correlates with its low response delivery ratio relative to the other evaluated protocols. ServiceAgent is configured to immediately send

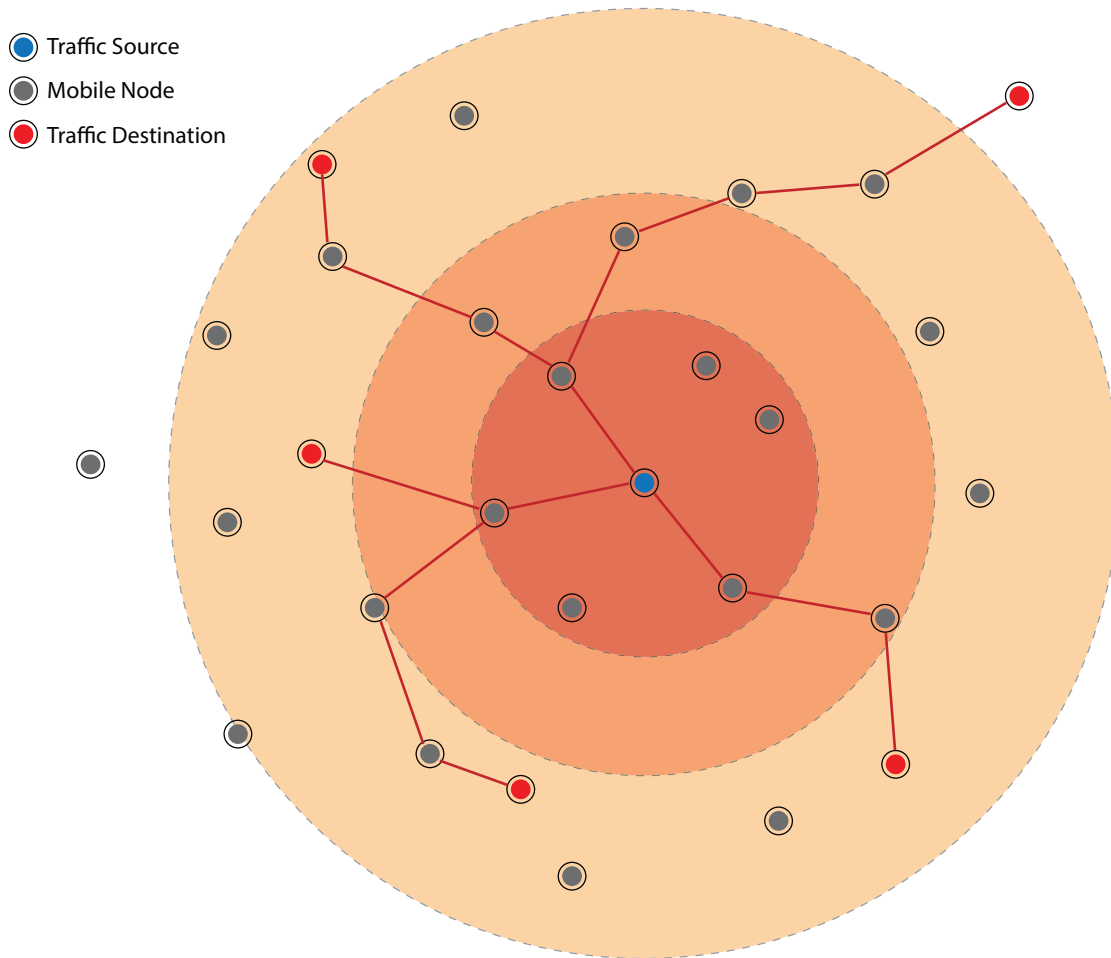


Fig. 5.23: Service agents act as focal points for application traffic flows, resulting in an increased probability of packet drops within their local area.

a request message upon receipt of a response message. If no response is received, ServiceAgent will wait 1 second before sending a new request, thus response delivery has a significant effect on overall throughput. The lower service availability displayed by SCDP-RA and SCDP-PA correlates with their decreased overall throughput, as no application traffic is sent until services are discovered.

The bias in service availability caused by NS-2's packet prioritization results in a decrease in service availability for application-layer approaches which would be unlikely to occur in a real network. SDRP's low control overheads, high availability, high delivery ratios and low end-to-end delay relative to reactive approaches contribute to its high throughput in this experiment.

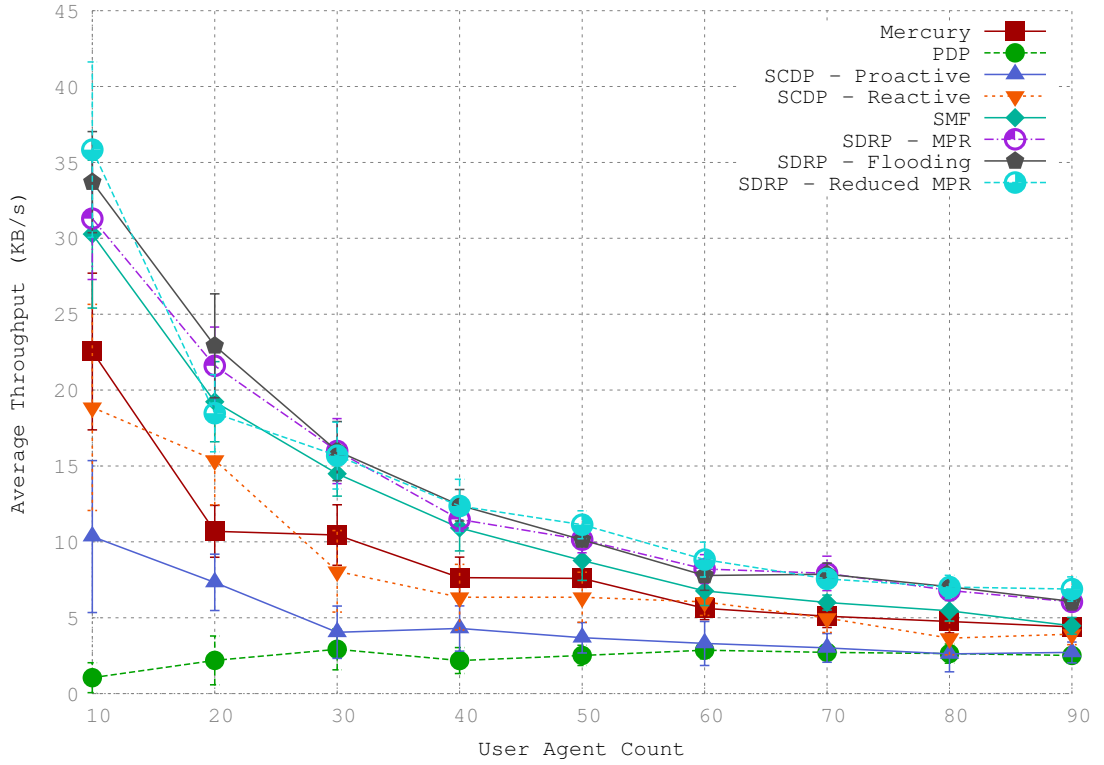


Fig. 5.24: Application throughput with varying user agent count. Application throughput is defined as the average amount of application traffic received per user agent. Error bars describe 95% confidence intervals for each design point.

5.4.6 Summary

In this experiment, the effect of varying user agent density was measured by varying the number of user agents in a network with a fixed boundary area, node count and service agent count. An interesting phenomenon was observed in which increasing user agent count lead to a drop in response delivery ratios. This was attributed to the increased numbers of traffic flows directed towards the limited number of service agents in the network, leading to network congestion in their local area. This indicates that MANETs in which the number of user agents heavily outweighs the number of service agents are likely to experience performance problems.

Significant increases were observed in the control overheads generated by protocols based on reactive routing, correlated with an increasing number of user agent route requests. Control overheads for advertisement-based approaches and approaches based on proactive routing remained stable, likely due to the fact that the service agent count remained the same throughout

the experiment.

The SDRP protocol family maintained low control overhead generation and high application throughput when compared with other approaches, however it too suffered from the service agent focal point effect discussed previously. A service discovery approach that selected service agents based on their current traffic load rather than distance may exhibit improved performance in this scenario, however as observed in SCDP-PA and SCDP-RA in previous experiments, lengthened routes between user and service agents can lead to an increased probability of packet drops.

In the next experiment, node mobility will be varied in order to ascertain its effect on the performance of the evaluated protocols.

5.5 Mobility

Previous experiments have involved nodes moving at moderate speeds of between 1m/s and 2m/s. In this experiment, the performance of the evaluated protocols will be tested with varying mobility. In particular, this experiment will test SDRP's reverse route approach; as mobility increases, the routes between service agents and user agents become invalid with increasing frequency. This is expected to reduce SDRP's response delivery ratio relative to that of approaches utilizing on address-based routing, as SDRP service agents must wait for new reverse routes to be formed once a route has become invalid.

Category	Parameter	Value
Network	Node Count	50
	Topology Width	1000m
	Topology Length	1000m
Node & Wireless	Protocol	SDRP-F, SDRP-MPR, SDRP-RMPR, SMF, SCDP-RA, SCDP-PA, Mercury, PDP
	Node Speed	Varying between 1m/s and 10m/s in increments of 1m/s
Routing Protocol	HELLO Interval	2 Seconds
	Advertisement Interval	5 Seconds
Service Discovery	User Agent Count	5
	Service Agent Count	3
	Service Count	2

Fig. 5.25: Parameter values used for the mobility experiment.

Figure 5.25 shows the parameter values used for this experiment. The number of nodes, service agents, user agents and services will remain static as the maximum node speed is increased from 1m/s to 10m/s. The number of nodes was decreased from 100 to 50 in this experiment in order to increase the incidence of partitioning and link breakage; this is expected to generate more meaningful results when compared with a denser network topology. A total of 1,600 experimental runs were executed; 8 protocols evaluated at 10 mobility design points with 20 replicates per design point. In the next section, the effect of mobility on protocol control overheads will be described.

5.5.1 Control Overheads

The control overheads measured during this experiment are shown in figure 5.26. The largest variations are seen in protocols based on OLSR, which exhibit an increase in control overheads of approximately 50% as the node speed increases. With increasing mobility, the makeup of each OLSR node's local topology changes more frequently, resulting in the addition of more node addresses and link states to HELLO messages. OLSR designates neighbours as stale if no HELLO messages are received from that neighbour for a set period of time. In the case of the NS-2 OLSR agent used for this experiment, that time limit is set to 6 seconds.

With increasing mobility, the number of new nodes added to the neighbour table increases while the rate at which stale nodes are removed remains stable. This results in HELLO messages increasing in size as mobility increases, resulting in an increase in control overheads. As neighbour tables increase in size, so too do calculated MPR sets, as more MPRs are required to reach the greater number of perceived 2-hop neighbours. Thus, TC message rebroadcasts will tend to increase as well with increasing mobility.

A smaller increase in control overheads is observed in SDRP-MPR. SDRP-MPR shares OLSR's neighbour caching algorithm and purges stale neighbours periodically, however HELLO message size remains static regardless of the number of neighbours a node may have. The MPR set calculated by SDRP-MPR, however, will be larger with increasing neighbour count. With only a subset of network nodes acting as service advertisement sources, the effect of larger MPR sets is reduced when compared with OLSR.

With lower node speeds, the probability that network partitions will persist for longer periods of time also tends to increase, which leads to a drop in control overheads. This effect is observed in SCDP-RA, SMF and SDRP-F; all of which exhibit an increase in control overheads which then stabilizes at a speed of 3m/s and above. In the next section, the effect of mobility on service availability will be described and discussed.

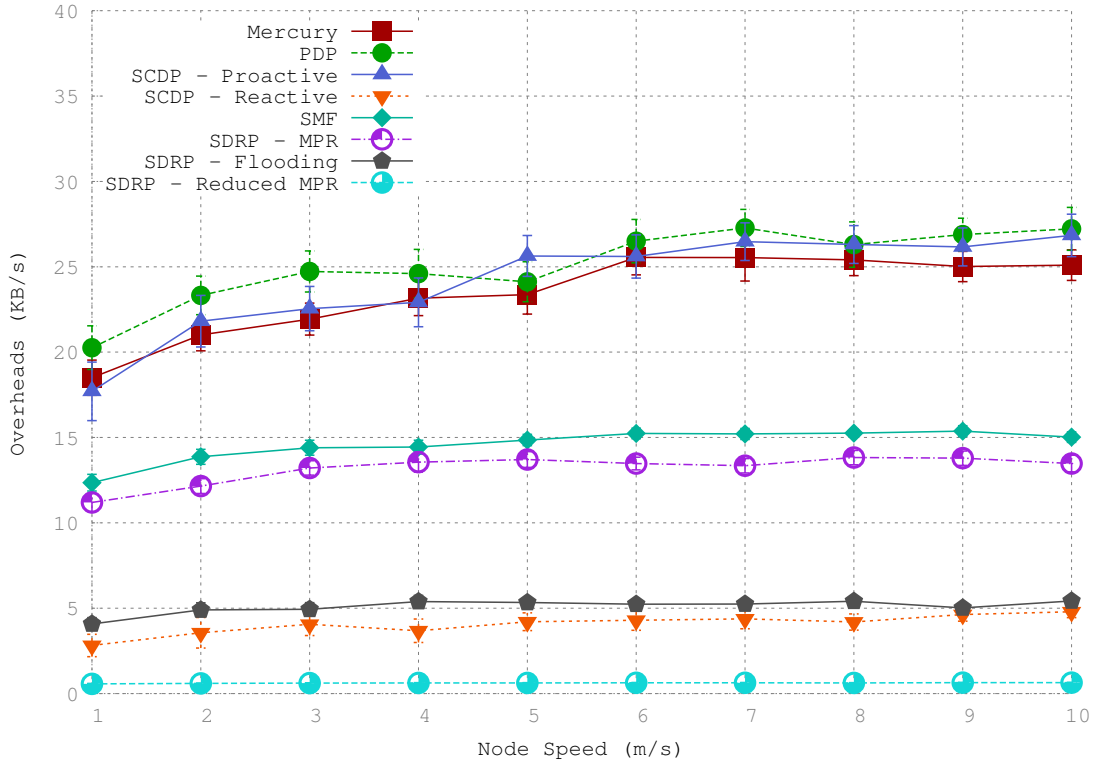


Fig. 5.26: Control overheads with varying node speed. Application throughput is defined as the total amount of application traffic consumed over time. Error bars describe 95% confidence intervals for each design point.

5.5.2 Service Availability

Figure 5.27 illustrates the measurements of service availability obtained from this experiment. SCDP-RA and SCDP-PA exhibit large degrees of variation in their results at low node speeds. This may reflect the extended network partitioning effect hypothesized in the previous section.

Service availability for both protocols is also initially reliant upon access to directory nodes for service queries by user agents. If an initial attempt to contact a directory node fails, user agents will then flood a direct request to service agents. This two-stage approach increases the probability of discovery failure, leading to large variations in availability.

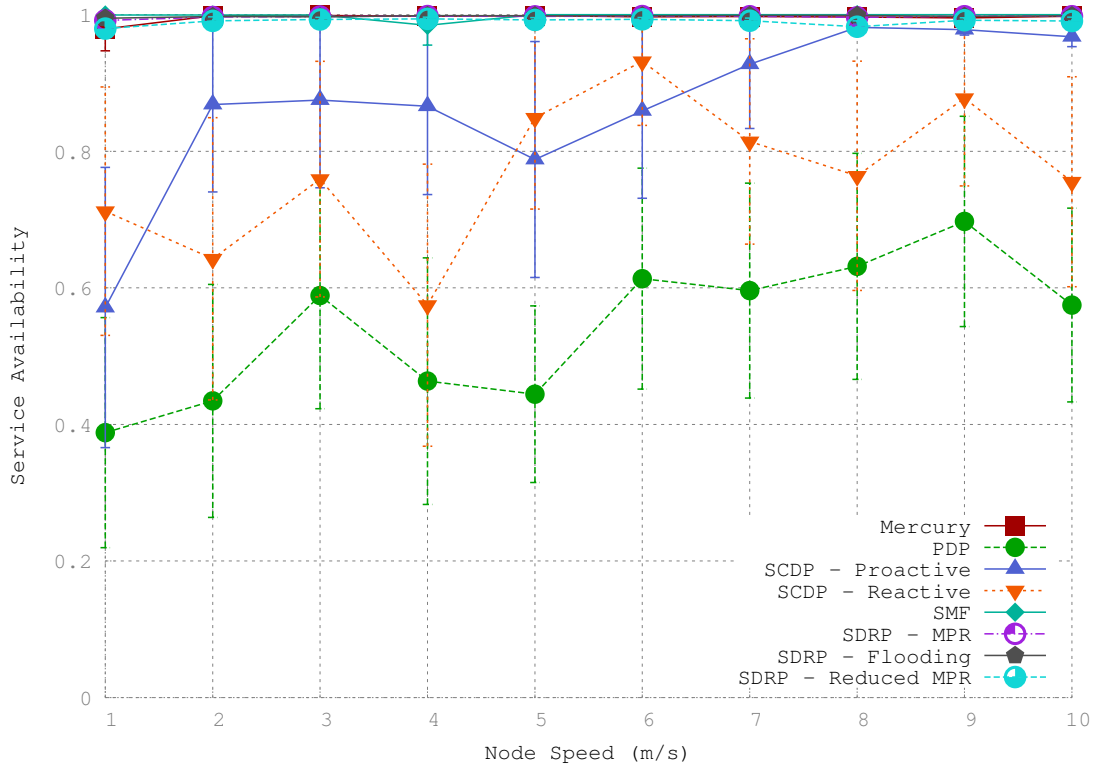


Fig. 5.27: Service availability with varying node speed. Service availability is defined as the ratio of successful service discoveries to the total number of attempted discoveries. Error bars describe 95% confidence intervals for each design point.

5.5.3 Request Delivery Ratio

The large variations seen in service availability for application-layer directory approaches are observed in request delivery ratios, as illustrated in figure 5.28. SMF's request delivery ratio is also exhibits large degrees of variation and is significantly lower than those displayed by service advertisement-based approaches.

These results are similar to those observed in section 5.2.3 and were attributed to route instability caused by AODV's route caching mechanism. This assertion is reinforced by observation of SCDP-PA's request delivery ratio, which exhibits greater stability than that of SCDP-RA. Both protocols, however, exhibit greater variation and lower performance than peer-to-peer approaches based on OLSR, or that of SDRP. This indicates that route unavailability and non-optimal route selection due to a lack of knowledge regarding distance to service agents; a problem common to both version of SCDP, causes a reduction in their ability to delivery service

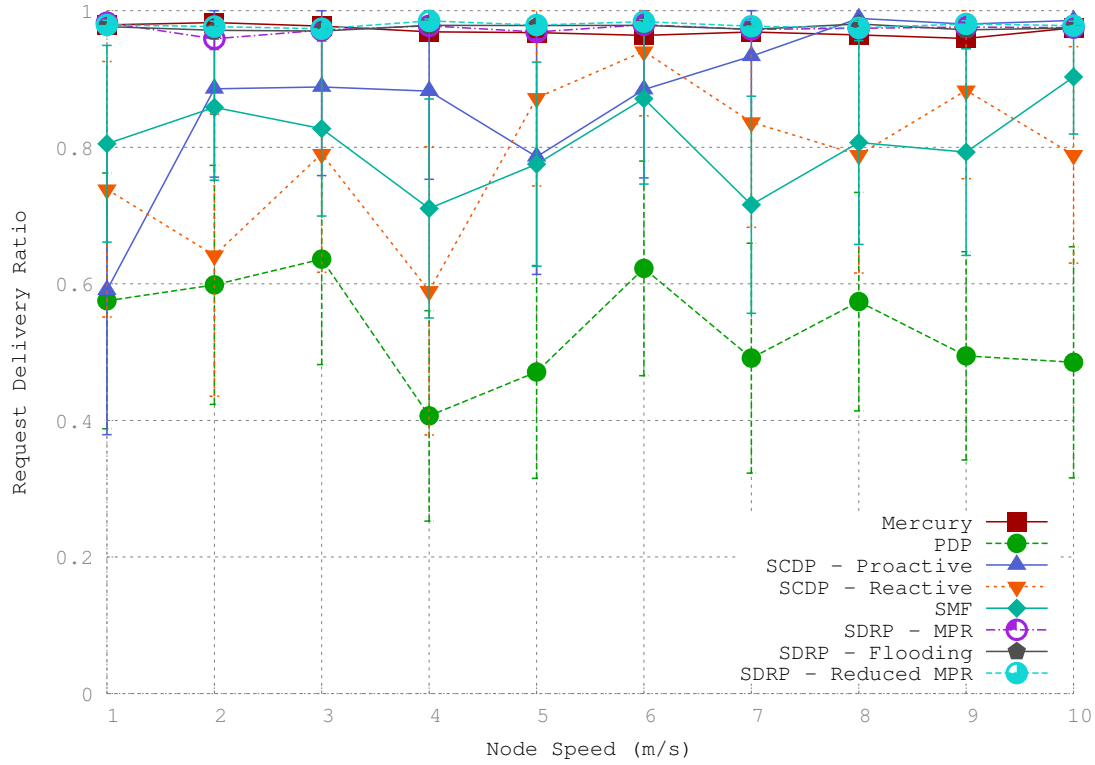


Fig. 5.28: Request delivery ratio with varying node speed. Defined as the ratio of successfully delivered service request messages to the total number sent. Error bars describe 95% confidence intervals for each design point.

requests. In the next section, the response delivery ratios recorded during this experiment will be discussed.

5.5.4 Response Delivery Ratio

Significant random variation in measured response delivery ratios can be observed, as shown in figure 5.29. Variation in the measured results is similar to those seen during the network scale and density experiment with 50 nodes, with that experiment involving the same number of user agents, service agents and services.

Most protocols show decreasing delivery ratios with increasing speed, which may be due to increased incidence of link breakage. SDRP does not display a significant decrease in response delivery relative to address-based protocols; indicating that for this experimental setup,

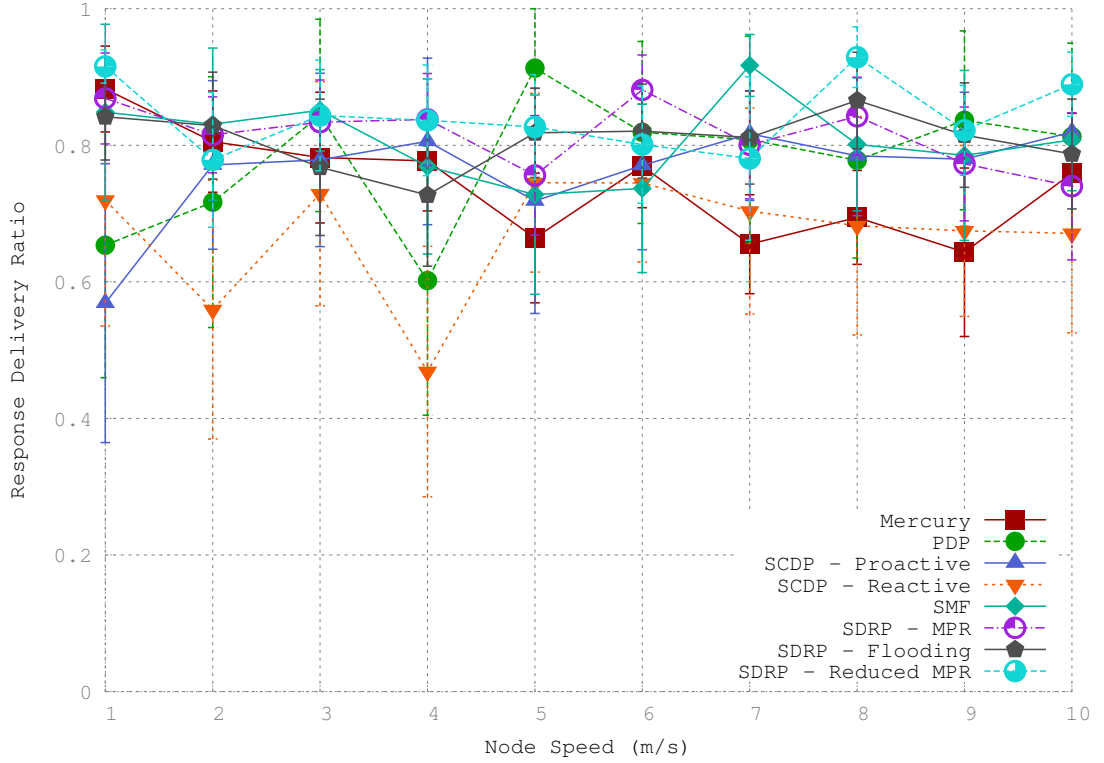


Fig. 5.29: Response delivery ratio with varying node speed. Defined as the number of successfully delivered service response messages to the total number sent. Error bars describe 95% confidence intervals for each design point.

the reverse-route algorithm used by SDRP exhibits stable performance. In the next section, application throughput measurements will be described.

5.5.5 Application Throughput

Application throughput appears to decrease with increasing node speed, which is correlated with the decreased response delivery ratios observed in the previous section, as seen in figure 5.30. Service-advertisement approaches exhibit a throughput decrease of between 20% and 40% with increasing mobility, correlated with the control overhead increases observed in previous sections associated with increases in perceived neighbour coverage.

SDRP maintains a throughput advantage over Mercury and SCDP-PA due its low control overheads, high service availability and high delivery ratios. SDRP's higher route stability,

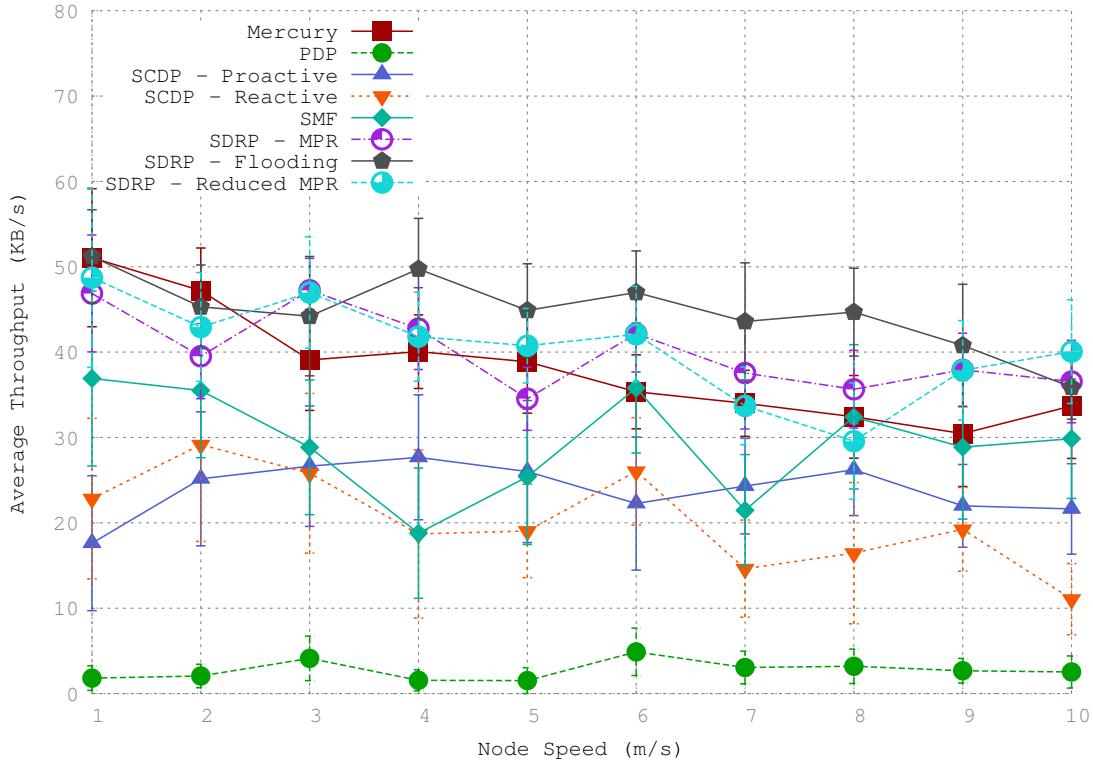


Fig. 5.30: Application throughput with varying node speed. Application throughput is defined as the average amount of application traffic received per user agent. Error bars describe 95% confidence intervals for each design point.

relative to reactive approaches, appears to result in decreased variation in throughput

5.5.6 Summary

In this experiment, the effect of increasing node speed on the evaluated service discovery approaches was measured. Significant variations in performance were observed, with control overheads tending to increase for approaches based on proactive address-based routing and service advertisement, and route instability observed for approaches based on reactive routing and directory-based discovery.

Increased control overheads were observed in SDRP-MPR and protocols based on OLSR. This was attributed to increasing HELLO message size and a resulting increase in the number of neighbours selected as MPR nodes. Both SDRP-MPR and OLSR purge stale neighbour entries

if no HELLO messages are received from those neighbours after a certain period of time.

With increasing mobility, the makeup of a node's neighbourhood changes with increasing frequency; resulting in the neighbour table also increasing in size as HELLO messages from new neighbours are received. The stale neighbour time limit, however, remains static and thus neighbour tables tend to become larger with increasing mobility. This increases HELLO message size as well, as OLSR serializes more neighbour link-states.

For both OLSR and SDRP-MPR, this results in an increase in the number of nodes selected as MPRs, as both protocols determine that a greater number of 2-hop neighbours must be reached. For OLSR, the increase in control overheads is observed to be higher relative to that of SDRP. This can be attributed to SDRP's static HELLO message length; the bloom filter used to describe a node's neighbour set is of static length. In addition, only service agents act as service advertisement sources in an SDRP network, whereas all MPR nodes are sources of TC messages in an OLSR network. This results in a greater proportional increase in control overheads with node speed when compared to SDRP.

SDRP's ability to route service response messages to user agents was similar to that of protocols based on address-based routing, demonstrating the robustness of the reverse-route algorithm used by SDRP even at high node speeds.

5.6 Results

Comparison of SDRP against existing service discovery and routing approaches has yielded interesting results during this evaluation. Approaches based on proactive routing exhibit a significant increase in control overheads with increasing network density. This control overhead increase correlates with an observed drop in the ability for those protocols to deliver application traffic. This was attributed to increased medium contention due to an increase in HELLO message sizes and topology control message broadcasts. This assertion was corroborated through observation of SDRP and those protocols based on reactive routing; stable control overheads observed for those protocols resulted in stable application throughput during the experiment.

Significant variations in request and response delivery ratios were observed for all protocols, with reactive protocols exhibiting the greatest degree of variation. AODV was configured to cache routes for 10 seconds, while both SDRP and OLSR were configured to disseminate service advertisements and topology control messages at 5 second intervals; thus link breakage resulting from node mobility has a greater effect on AODV as it is more likely to use invalid routes.

Significant variation in request and response delivery ratios were also observed in both SCDP-PA and SCDP-RA. It is hypothesized that this effect can be attributed to a lack of

available routes to service agents whose addresses have been retrieved from a directory node. In peer-to-peer approaches, service advertisements and query responses are received directly from service agents; indicating that a route to the source is available. Conversely, directory based approaches involve user agents querying directory nodes rather than service agents. A directory may return a positive response to a querying user agent, however a route to the service agent selected by the user agent may not be available.

In addition, the distance between a user agent and a selected service agent cannot be determined based on the response from a directory node; only the distance between the user agent and directory, and the distance between the directory and service agent can be determined. Thus, user agents may select service agents that are more distant than those selected by a peer-to-peer approach that has accurate distance information. The use of longer routes results in an increased probability of packet loss, leading to increased variation in message delivery success rates.

Increasing service agent density in a network of fixed size resulted in SDRP-F's naive flooding mechanism producing rapidly increasing control overheads. The increase in control overheads for SDRP-MPR and SDRP-RMPR was observed to be lower, indicating that their selective broadcast mechanism reduces control overheads relative to an approach based on flooding. Observation of application throughput and message delivery success rates indicates that both protocols' selective rebroadcast mechanisms do not compromise advertisement coverage and route formation.

Increasing user agent density lead to an interesting phenomenon whereby the increased number of packet flows directed at the static number of deployed service agents created a *focal point* effect. Converging packet flows increase medium contention, leading in particular to an inability for service agents to reliably send response messages to user agents. This phenomenon was observed in simulations involving all of the evaluated protocols. It is hypothesized that a service discovery approach that selects service agents based on their current traffic load may be well-suited to networks in which the number of user agents is significantly greater than the number of service agents. However, the resulting increase in route length can have a detrimental effect on message delivery success rates, as mentioned previously.

Increasing node mobility was shown to have a significant effect on the overheads generated by protocols based on OLSR. The version of OLSR used during this experiment includes no mobility adaptation mechanism, thus the increase in overheads was attributed to OLSR's caching of neighbour addresses. OLSR is configured to remove neighbour addresses from its cache if no HELLO message is received from those neighbours after a set time period. With increasing node mobility, the composition of an OLSR node's neighbour set changes frequently as HELLO messages are received from nodes entering the receiver's local area. The neighbour purge time

limit is static, thus a node's neighbour set will tend to become larger as node mobility increases.

This increase in neighbour set size produces a corresponding increase in HELLO message size, as OLSR serializes the link-state of all neighbours into those messages. Similarly, the number of nodes designated as MPRs will tend to increase, as OLSR will determine that a more two-hop neighbours must be reached by topology control messages. This results in a further increase in control overheads as more nodes rebroadcast topology control messages. A similar increase in control overheads was observed in SDRP-MPR, however this increase was less significant than that of OLSR's. SDRP-MPR's HELLO message size is static due to its use of static-length bloom filter for neighbour set description. In addition, only service agents act as sources for disseminated messages in an SDRP network, thus the increase in selected MPR set size has less of an effect on control overheads.

The SDRP protocol family was shown to perform favourably in all experiments; with consistently high application throughput and message delivery success rates in all scenarios. The reverse-route algorithm was shown to facilitate favourable response delivery success rates comparable to those of address-based routing protocols. Control overheads generated by SDRP remained low in most scenarios, with only SDRP-F exhibiting a rapid increase with increasing service agent density. Variation in message delivery ratios also remained low for the SDRP protocol family; demonstrating that proactive advertisement-based approaches reduce end-to-end delay and result in more reliable routes.

5.7 Summary

In this chapter, we evaluated SDRP and compared it with a number of existing service discovery approaches. The experimental setup and varied parameters were described and four experiments were conducted in which network density, service agent density, user agent density and node mobility were varied in order to determine their effect on the evaluated protocols.

The SDRP protocol family was shown to perform favourably, with consistently high application throughput and low variation in message delivery success rates in all scenarios when compared with existing approaches. Address-based approaches, particularly those based on proactive routing, were shown to suffer from significantly increasing control overheads with increasing network density which resulted in significant drops in application throughput. Reactive routing approaches were associated with high variability in delivery success rates due to caching of invalid routes. Conversely, the low control overheads generated by SDRP's service-only routing approach correlated with high application throughput, while its periodic advertisement approach was associated with high message delivery success rates.

Comparison of SDRP-F, SDRP-MPR and SDRP-RMPR has shown that SDRP-F's naive flooding approach can result in high control overheads as service agent density increases. Conversely, SDRP-MPR and SDRP-RMPR's overheads show a less significant increase due to their selective rebroadcast approach to service advertisement while maintaining favourable throughput and message delivery success rates. SDRP's reverse-route mechanism was not observed to cause a decrease in service response delivery success rate during these experiments.

Chapter 6

Conclusions and Future Work

This thesis presented SDRP, an alternative approach to service discovery and routing designed for high-density mobile ad hoc networks. This chapter summarises the most significant achievements of the work described in this thesis and assesses its contribution to the research area.

6.1 Achievements

This work was motivated by the recent and rapid rise in ownership of high-powered connected devices such as smartphones and tablet PCs. Such devices present a challenging deployment scenario for mobile ad hoc networks due to the potential for network densities and scales that exceed the capacity for current service discovery and routing approaches to facilitate while maintaining acceptable performance.

A review of existing approaches to service discovery and routing in mobile ad hoc networks has shown that none of the existing work is suitable for addressing the challenges of such network configurations. In particular, it was shown that existing service discovery approaches rely on address-based routing to facilitate communications between service agents and user agents; resulting in a two-tier approach to discovery that produces unnecessary control overheads, reducing performance.

We defined an alternative approach, the Service Discovery and Routing Protocol, designed to facilitate service-oriented communication in mobile ad hoc networks. SDRP improves network scalability by allowing routing only between service agents and user agents; reducing control overheads when compared with proactive address-based routing protocols while maintaining low

end-to-end delay. In addition, we defined a probabilistic approach to service advertisement rebroadcasting based on the use of bloom filters for neighbour link-state description.

SDRP was compared with a number of existing approaches in simulated scenarios involving varying network density, service agent density, user agent density and node mobility. SDRP was shown to consistently outperform current discovery and routing approaches; particularly in scenarios involving high network densities.

SDRP was developed in 3 stages, each incorporating a particular enhancement designed to reduce control overheads while facilitating service discovery. This approach was used so that the effect of each enhancement could be discerned through comparison with the baseline approach. Each enhancement was shown to produce a significant decrease in control overheads without compromising application throughput and message delivery success rates, with the probabilistic rebroadcasting mechanism used by the final version of SDRP, SDRP-RMPR, exhibiting consistently high performance throughout the evaluation.

6.2 Future Work

The development and evaluation of SDRP raised many interesting questions that are worthy of further investigation. This work presents a quantitative analysis of SDRP in simulation, however SDRP's performance in an actual mobile ad hoc network could not be tested due to a lack of resources. A significant extension to this work would involve the deployment of SDRP onto commodity smartphones in large enough numbers that its ability to perform in a large-scale network could be more thoroughly evaluated. A more formal analysis of SDRP would also constitute a significant addition to this work.

Limitations in the NS-2 network simulator made simulation of networks comprised of more than 200 nodes infeasible due to time constraints. Alternative simulators, such as OPNET, GloMoSim, JIST, OMNET++ or SWANS may be more suitable to the task of simulating very large-scale networks. It would be interesting to port SDRP to these simulators and experiment with greater simulated network scales with an aim to testing the protocol's limitations. This would also allow us to compare SDRP with protocols developed for those simulators, such as the original version of SMF and LSD.

SDRP facilitates routing between service agents and user agents through an algorithm based on reversal of the route taken by service advertisement messages. This approach could be more thoroughly tested using experiments in which service agents' responses to user agents were delayed for varying lengths of time. Such an experiment would require extensive modifications to the ServiceAgent framework described in section 4.2.2. Making these modifications would allow

weaknesses in the reverse-route algorithm to be measured more thoroughly.

Were SDRP's route reversal algorithm to be shown to perform poorly with increasing service agent response delays, it would be interesting to extend the protocol to allow route discovery between service agents and user agents. For SDRP-F, such an approach would likely involve query flooding. However, SDRP-MPR and SDRP-RMPR allow nodes to gain knowledge of their neighbour link-states; this information could be used to facilitate efficient route query dissemination. This combination of proactive service advertisement and on-demand route discovery would make SDRP more suitable for deployment in networks with high service agent response delays.

Our evaluation used the random waypoint model in order to generate mobility patterns for simulated nodes. However, our protocol is primarily targeted at deployment scenarios involving smart connected devices held by users in urban cores. Our evaluation could be expanded using patterns derived from studies of pedestrian movement such as those by Walsh et al. (2008).

6.3 Summary

This chapter summarized the motivations for, and the most significant achievements of, the work described in this thesis. In addition, suggestions for future work resulting from this thesis were presented.

Appendix A

Bloom Filters

A Bloom Filter (Bloom, 1970) is a space-efficient probabilistic data structure that supports queries against its contents. As outlined in section 2.4.1’s review of the Mercury service discovery protocol, bloom filters are represented by an array of m bits, initially set to 0.

The process for inserting an element into a bloom filter is outlined in figure A.1. Queries against a filter are performed using similar steps; the element is hashed using the same k hash functions and the resulting bit indices are checked. If any of the bits at the resulting indices are set to 0, the element is definitely not in the filter. Otherwise, if all of the bits at the resulting indices are set to 1, the element is probably in the filter.

A key concern when using bloom filters is the probability that a query against the filter will produce a false-positive; a positive query result for an element that isn’t actually in the filter. False-positive queries occur when previous insertions set the bits indicating the presence of a particular element, without that element actually having been inserted. The probability of this occurring is linked to the number of elements inserted into the filter, the size of the filter and the number and nature of the hash functions used for insertion.

Assuming that the hash functions used to populate the filter do not produce biased output, the probability that a given filter bit in a filter of size m bits is not set is given by equation A.1:

$$\left(1 - \frac{1}{m}\right) \tag{A.1}$$

For a filter using k hash functions per element insertion, the probability that a given bit is not set after an element has been inserted is given by equation A.2:

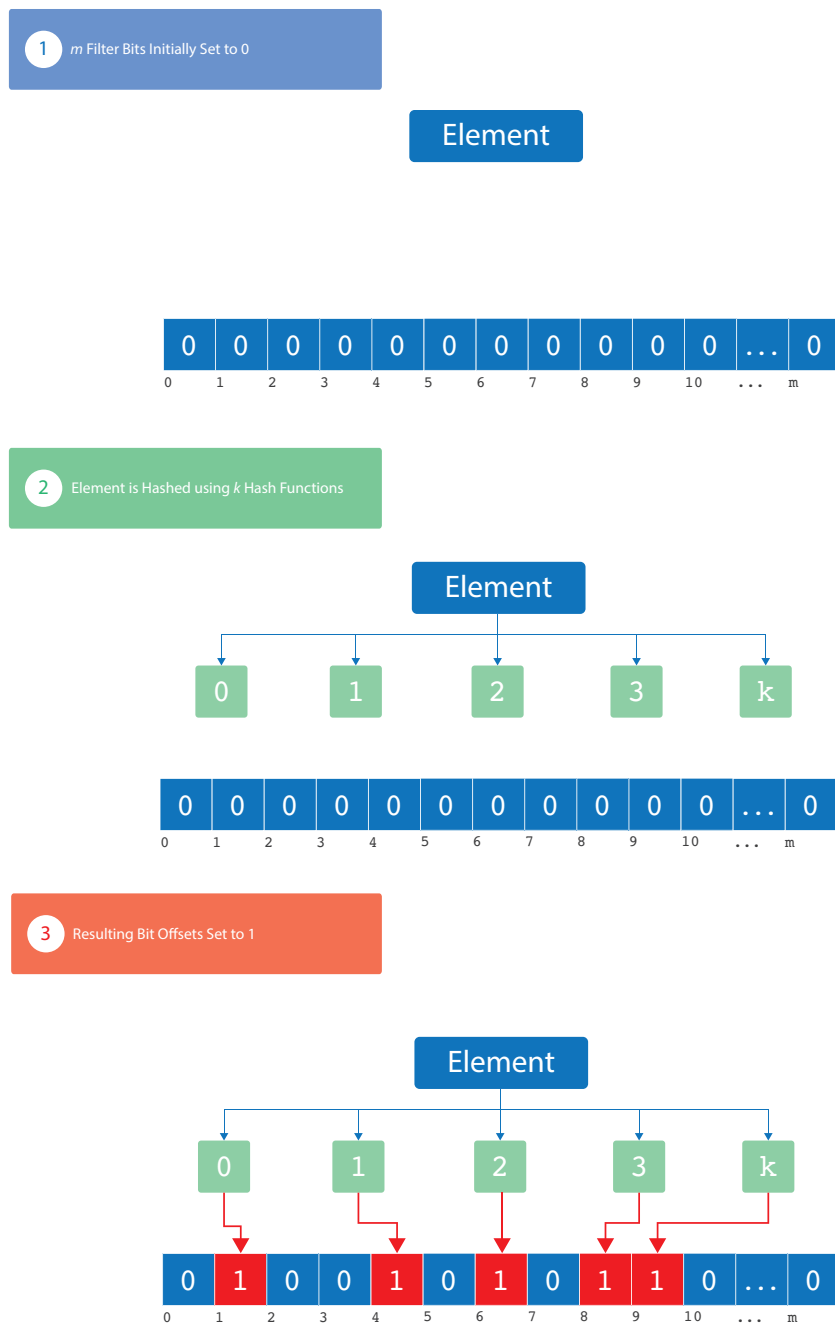


Fig. A.1: Bloom Filter Element Insertion Process.

$$(1 - \frac{1}{m})^k \quad (\text{A.2})$$

For a filter with n prior insertions, the probability that a given bit is not set is given by equation A.3:

$$(1 - \frac{1}{m})^{kn} \quad (\text{A.3})$$

Thus, the probability that a given bit is set after n element insertions is given by equation A.4:

$$1 - (1 - \frac{1}{m})^{kn} \quad (\text{A.4})$$

A false-positive membership query will occur if the k hash functions compute bit offsets that are already set to 1, with the probability stated above. Therefore, the probability that a membership query will result in a false-positive is given by equation A.5:

$$(1 - [1 - \frac{1}{m}]^{kn})^k \quad (\text{A.5})$$

Assuming independence for the probabilities of each bit being set, this equation may be simplified to the following form:

$$(1 - e^{-\frac{kn}{m}})^k \quad (\text{A.6})$$

Thus it may be observed that the probability of a false-positive increases with increasing n (number of filtered elements) and decreases as m increases. An optimal value for k that minimizes the false probability rate may be computed based the expected number of filtered elements and required filter size.

$$k = \frac{m}{n} \ln 2 \quad (\text{A.7})$$

By substituting this optimal k equivalence into our previous false probability equation, it is possible to formulate an expression that produces the optimal filter size m based on a desired false probability rate and expected number of filtered items n . First, the substitution produces the following equation:

$$p = (1 - e^{-\frac{(m/n \ln 2)n}{m}})^{(m/n \ln 2)} \quad (\text{A.8})$$

This can be simplified to the following form:

$$\ln p = -\frac{m}{n}(\ln 2)^2 \quad (\text{A.9})$$

Thus, the filter size for a given false-positive rate and expected element count may be given as:

$$m = -\frac{n \ln p}{(\ln 2)^2} \quad (\text{A.10})$$

Using these equations, it is possible to generate a bloom filter with the minimum size for a given purpose. With an optimal k value and a false probability of 1%, a bloom filter needs only 9.6 bits to represent a given element.

It is important to note that elements cannot be removed from a bloom filter, as setting one or more bits to 0 may result in membership queries for other filter elements failing even though they have not been removed. This is due to the fact that several elements may set the same bit(s) during insertion.

An alternative form of bloom filter, a counting filter, allows elements to be removed. Element insertion and querying using a counting filter is similar to the operations used in bloom filters. However, a counting filter is represented by an array of m integers instead of individual bits. Integers at offsets produced by hash functions are incremented rather than set to 1. In order to remove an element, it is first hashed and the resulting offsets are then decremented.

Bibliography

- Adam N., Ismail M., and Abdullah J. Effect of node density on performances of three MANET routing protocols. In *Electronic Devices, Systems and Applications (ICEDSA), 2010 Intl Conf on*, pages 321–325, April. 2010.
- Amis A. and Prakash R. Load-balancing clusters in wireless ad hoc networks. In *In Proceedings of the 3rd IEEE Symposium on Application-Specific Systems and Software Engineering Technology, 2000.*, pages 25–32. IEEE, 2000.
- Ansari I. An Implementation of Traffic Light System Using Multi-hop Ad hoc Networks. In *2009 International Conference on Network-Based Information Systems. NBIS '09.*, pages 177–181, August. 2009.
- Apple Inc. . Bonjour, August. 2002. URL <https://developer.apple.com/opensource/>. [Accessed on 14/5/2012].
- Appleby A. MurmurHash, January 2008. URL <https://sites.google.com/site/murmurhash/>. [Accessed on 12/06/2012].
- Arora D., Millman E., and Neville S. Assessing the Expected Performance of the OLSR Routing Protocol for Denser Urban Core Ad Hoc Network Deployments. In *Advanced Information Networking and Applications (AINA), 2012 IEEE 26th International Conference on*, pages 406–414, March. 2012.
- Aschenbruck N., Frank M., Martini P., and Tolle J. Human mobility in MANET disaster area simulation - a realistic approach. In *29th Annual IEEE International Conference on Local Computer Networks, 2004.*, pages 668–675, November. 2004.
- Basagni S., Chlamtac I., Syrotiuk V., and Woodward B. A distance routing effect algorithm for mobility (DREAM). In *In Proceedings of the 4th annual ACM/IEEE international conference on Mobile computing and networking*, pages 76–84. ACM, 1998.
- Bloom B. Space/time trade-offs in hash coding with allowable errors. *Communications of the ACM*, 13(7):422–426, 1970.

- Broch J., Maltz D., Johnson D., Hu Y., and Jetcheva J. A performance comparison of multi-hop wireless ad hoc network routing protocols. In *In Proceedings of the 4th annual ACM/IEEE international conference on Mobile computing and networking*, pages 85–97. ACM, June. 1998.
- Campo C., Garcia-Rubio C., Lopez A. M., and Almenarez F. PDP: A Lightweight Discovery Protocol for Local-Scope Interactions in Wireless ad hoc Networks. *Computer Networks*, 50 (17):3264–3283, 2006. ISSN 1389-1286. doi: DOI:10.1016/j.comnet.2005.12.007.
- Canourgues L., Lephay J., Soyer L., and Beylot A.-L. An Efficient Scalable Structure-Free Inter-Cluster Multicast Routing Protocol for Large Tactical Mobile Ad hoc Networks. In *IEEE Military Communications Conference, 2007. MILCOM 2007.*, pages 1–7, October. 2007.
- Cartigny J. and Simplot D. Border node retransmission based probabilistic broadcast protocols in ad-hoc networks. In *In Proceedings of the 36th Annual Hawaii International Conference on System Sciences, 2003.*, page 10 pp., January. 2003.
- Chen T. and Gerla M. Global state routing: A new routing scheme for ad-hoc wireless networks. In *IEEE International Conference on Communications, 1998. (ICC 1998)*, volume 1, pages 171–175. IEEE, 1998.
- Chen Y. P. and Liestman A. L. Approximating minimum size weakly-connected dominating sets for clustering mobile ad hoc networks. In *In Proceedings of the International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc '02)*, pages 165–172, 2002.
- Chiang C., Wu H., Liu W., and Gerla M. Routing in clustered multihop, mobile wireless networks with fading channel. In *In Proceedings of IEEE SICON '97*, volume 97, pages 197–211, April. 1997.
- Conti M., Maselli G., Turi G., and Giordano S. Cross-layering in mobile ad hoc network design. *Computer*, 37(2):48–51, February. 2004.
- Corson M. S. and Ephremides A. A distributed routing algorithm for mobile wireless networks. *Wireless Networks*, 1(1):61–81, February. 1995.
- Corson S. and Macker J. Mobile Ad hoc Networking (MANET): Routing Protocol Performance Issues and Evaluation Considerations, 1999.
- Dijkstra E. A note on two problems in connexion with graphs. *Numerische mathematik*, 1(1): 269–271, 1959.
- Flathagen J. and Øvsthus K. Service Discovery using OLSR and Bloom Filters. In *Proceedings of the 4th OLSR Interop & Workshop*, October. 2008.
- Garcia-Macias J. and Torres D. Service discovery in mobile ad-hoc networks: better at the

- network layer? In *International Conference Workshops on Parallel Processing, 2005.*, pages 452–457. IEEE, 2005.
- Giordano S. Mobile Ad hoc Networks. *Handbook of Wireless Networks and Mobile Computing*, pages 325–346, 2002.
- Gupta A., Sadawarti H., and Verma A. Performance Analysis of AODV, DSR & TORA Routing Protocols. *IACSIT International Journal of Engineering and Technology*, 2(2), 2010.
- Haas Z. A new routing protocol for the reconfigurable wireless networks. In *IEEE 6th International Conference on Universal Personal Communications, 1997.*, volume 2, pages 562–566, October. 1997.
- Haas Z., Halpern J., and Li L. Gossip-based ad hoc routing. In *In Proceedings of the Twenty-First Annual Joint Conference of the IEEE Computer and Communications Societies. INFOCOM 2002*, volume 3, pages 1707–1716, 2002.
- Hekmat R. and Van Mieghem P. Interference in Wireless Multi-Hop Ad-Hoc Networks and Its Effect on Network Capacity. *Wireless Networks*, 10:389–399, 2004.
- Helal S., Desai N., Verma V., and Lee C. Konark - a service discovery and delivery protocol for ad-hoc networks. In *Wireless Communications and Networking, 2003. WCNC 2003.*, volume 3, pages 2107–2113, March. 2003.
- IEEE . IEEE Standard for Information technology– Local and metropolitan area networks– Specific requirements– Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications Amendment 8: IEEE 802.11 Wireless Network Management. *IEEE Std 802.11v-2011*, pages 1–433, 9 2011.
- Jacquet P., Muhlethaler P., Clausen T., Laouiti A., Qayyum A., and Viennot L. Optimized Link State Routing Protocol for ad hoc networks. In *In Proceedings of Multi Topic Conference on Technology for the 21st Century, 2001 (IEEE INMIC 2001)*, pages 62–68. IEEE, 2001.
- Johansson P., Larsson T., Hedman N., Mielczarek B., and Degermark M. Scenario-based performance analysis of routing protocols for mobile ad-hoc networks. In *Proceedings of the 5th annual ACM/IEEE international conference on Mobile computing and networking*, pages 195–206. ACM, 1999.
- Johnson D. B. and Maltz D. A. Dynamic Source Routing in Ad Hoc Wireless Networks. In Imielinski T. and Korth H. F., editors, *Mobile Computing*, volume 353 of *The Kluwer International Series in Engineering and Computer Science*, pages 153–181. 1996.
- Jones C. E., Sivalingam K. M., Agrawal P., and Chen J. C. A Survey of Energy Efficient Network Protocols for Wireless Networks. *Wireless Networks.*, pages 343–358, September. 2001.

- Klein M., Konig-Ries B., and Obreiter P. Service rings - a Semantic Overlay for Service Discovery in ad hoc Networks. In *In Proceedings of the 14th International Workshop on Database and Expert Systems Applications, 2003.*, pages 180–185, September. 2003.
- Kozat U. and Tassiulas L. 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, 2004.
- Kurkowski S., Camp T., and Colagrosso M. MANET simulation studies: the incredibles. *SIG-MOBILE Mobile Computing and Communications Review*, 9(4):50–61, October. 2005.
- Lee S., Belding-Royer E., and Perkins C. Scalability study of the ad hoc on-demand distance vector routing protocol. *International Journal of Network Management*, 13(2):97–114, 2003.
- Li L. and Lamont L. A Lightweight Service Discovery Mechanism for Mobile Ad hoc Pervasive Environment using Cross-Layer Design. In *Third IEEE International Conference on Pervasive Computing and Communications Workshops, 2005. PerCom 2005 Workshops.*, pages 55–59, March. 2005.
- Lye P. G. and McEachen J. C. A Comparison of Optimized Link State Routing with Traditional Routing Protocols in Marine Wireless Ad-hoc and Sensor Networks. In *40th Annual Hawaii International Conference on System Sciences, 2007. HICSS 2007.*, pages 293–303, January. 2007.
- Marwaha S., Indulska J., and Portmann M. Challenges and Recent Advances in QoS Provisioning, Signaling, Routing and MAC protocols for MANETs. In *Australasian Telecommunication Networks and Applications Conference, 2008. ATNAC 2008.*, pages 97 –102, December. 2008.
- Mettala R. *SDP Specification*. Bluetooth Special Interest Group, August. 1999.
- Microsoft . ISO/IEC 29341-1:2008 - UPnP Device Architecture – Part 1: UPnP Device Architecture Version 1.0, December. 2008. URL http://www.iso.org/iso/catalogue_detail?csnumber=52674. [Accessed on 14/5/2012].
- Microsoft, IBM . Universal Description Discovery and Integration, September. 2000. URL http://www.uddi.org/pubs/Iru_UDDI_Technical_White_Paper.pdf. [Accessed on 17/5/2012].
- Murthy S. and Garcia-Luna-Aceves J. A routing protocol for packet radio networks. In *In Proceedings of the 1st annual international conference on Mobile computing and networking*, pages 86–95. ACM, 1995.
- Ohta T., Inoue S., and Kakuda Y. An adaptive multihop clustering scheme for highly mobile ad hoc networks. In *The Sixth International Symposium on Autonomous Decentralized Systems, 2003. ISADS 2003.*, pages 293–300, April. 2003.

- Park V. and Corson M. A highly adaptive distributed routing algorithm for mobile wireless networks. In *In Proceedings of the Sixteenth Annual Joint Conference of the IEEE Computer and Communications Societies. (INFOCOM'97)*, volume 3, pages 1405–1413. IEEE, 1997.
- Pearlman M. and Haas Z. Determining the optimal configuration for the zone routing protocol. *IEEE Journal on Selected Areas in Communications*, 17(8):1395–1414, August. 1999.
- Perkins C. and Bhagwat P. Highly dynamic destination-sequenced distance-vector routing (DSDV) for mobile computers. *ACM SIGCOMM Computer Communication Review*, 24(4): 234–244, 1994.
- Perkins C., Belding-Rowyer E., and S.Das . Ad hoc On-Demand Distance Vector (AODV) Routing. In *Proceedings of the 2nd IEEE Workshop on Mobile Computing Systems and Applications (WMCSA 1999)*, pages 90–100, February 1999.
- Peysakhov M., Mroczkowski A., Urbano L., Warren J., Cicirello V., Regli W., and Kam K. Agent survivability through power awareness. In *2005 IEEE 2nd Symposium on Multi-Agent Security and Survivability*, pages 31–38, August. 2005.
- Pullin A. and Pattinson C. A Realistic Battlefield Model for the Evaluation of MANETs. In *Fifth Annual Conference on Wireless on Demand Network Systems and Services, 2008. WONS 2008.*, pages 81–84, January. 2008.
- Rahman M., Anwar F., Naeem J., and Abedin M. A Simulation-Based Performance Comparison of Routing Protocols in Mobile Ad-hoc Networks (proactive, reactive and hybrid). In *2010 International Conference on Computer and Communication Engineering (ICCCE)*, pages 1–5, May. 2010.
- Rajaraman R. Topology control and routing in ad hoc networks: a survey. *SIGACT News*, 33 (2):60–73, June. 2002.
- Royer E. and Toh C. A review of current routing protocols for ad hoc mobile wireless networks. *Personal Communications, IEEE*, 6(2):46–55, 1999.
- Sailhan F. and Issarny V. Scalable Service Discovery for MANET. In *Third IEEE International Conference on Pervasive Computing and Communications, 2005. PerCom 2005.*, pages 235–244, March. 2005.
- Samar P., Pearlman M. R., and Haas Z. J. Independent zone routing: an adaptive hybrid routing framework for ad hoc wireless networks. *IEEE/ACM Transactions on Networking*, 12 (4):595–608, August. 2004.
- Sari R., Syarif A., Ramli K., and Budiardjo B. Performance evaluation AODV routing protocol

- on ad hoc hybrid network testbed using PDAs. In *2005 13th IEEE International Conference on Networks, 2005.*, volume 1, page 6 pp, November. 2005.
- Schiele G., Becker C., and Rothermel K. Energy-efficient cluster-based service discovery for Ubiquitous Computing. In *Proceedings of the 11th workshop on ACM SIGOPS European workshop*, pages 70–84, September. 2004.
- Shibata Y., Yuze H., Hoshikawa T., Takahata K., and Sawano N. Large Scale Distributed Disaster Information System based on MANET and Overlay Network. In *27th International Conference on Distributed Computing Systems Workshops, 2007. ICDCSW '07.*, pages 33–40, June. 2007.
- Stepanov A. Standard Template Library, March 1994. URL <http://www.sgi.com/tech/stl/>. [Accessed on 12/06/2012].
- Sun Microsystems . JINI, July. 1998. URL <http://river.apache.org/>. [Accessed on 17/5/2012].
- Sung K., Srinivasan S., and Schulzrinne H. BBS-ONE: Bulletin board and forum system for mobile opportunistic networks. In *2010 IEEE International Conference on Wireless Communications, Networking and Information Security (WCNIS)*, pages 371 –376, June. 2010.
- Toh C. A novel distributed routing protocol to support ad-hoc mobile computing. In *In Proceedings of the 1996 IEEE Fifteenth Annual International Phoenix Conference on Computers and Communications, 1996.*, pages 480–486. IEEE, 1996.
- Tonguz O., Wisitpongphan N., Parikh J., Bai F., Mudalige P., and Sadekar V. On the Broadcast Storm Problem in Ad hoc Wireless Networks. In *3rd International Conference on Broadband Communications, Networks and Systems, 2006. BROADNETS 2006.*, pages 1–11, October. 2006.
- Tseng Y., Ni S., Chen Y., and Sheu J. The broadcast storm problem in a mobile ad hoc network. *Wireless networks*, 8(2):153–167, 2002.
- Veizades J., Guttman E., Perkins C., and Kaplan S. Service Location Protocol. RFC 2165 (Proposed Standard), 1997. URL <http://www.ietf.org/rfc/rfc2165.txt>. Updated by RFCs 2608, 2609.
- Ververidis C. and Polyzos G. Service discovery for mobile ad hoc networks: A survey of issues and techniques. *Communications Surveys & Tutorials, IEEE*, 10(3):30–45, 2008.
- Ververidis C. N. and Polyzos G. C. A routing layer based approach for energy efficient service discovery in mobile ad hoc networks. *Wirel. Commun. Mob. Comput.*, 9(5):655–672, May. 2009.

- Walsh C., Doci A., and Camp T. A call to arms: it's time for REAL mobility models. *ACM SIGMOBILE Mobile Computing and Communications Review*, 12(1):34–36, January. 2008.
- Wu J., Gao M., and Stojmenovic I. On calculating power-aware connected dominating sets for efficient routing in ad hoc wireless networks. In *International Conference on Parallel Processing.*, pages 346–354. IEEE, 2001.
- Yu J. and Chong P. A survey of clustering schemes for mobile ad hoc networks. *Communications Surveys & Tutorials, IEEE*, 7(1):32–48, 2005.
- Zeng X., Bagrodia R., and Gerla M. GloMoSim: a library for parallel simulation of large-scale wireless networks. In *In Proceedings of the Twelfth Workshop on Parallel and Distributed Simulation, 1998. PADS 98.*, pages 154 –161, May. 1998.
- Zhou X., Ge Y., Chen X., Jing Y., and Sun W. SMF: A Novel Lightweight Reliable Service Discovery Approach in MANET. In *Wireless Communications, Networking and Mobile Computing (WiCOM), 2011 7th International Conference on*, pages 1–5, September. 2011.