Integrated Intra-HAN and Inter-HAN Service Interoperability

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

Zohar Etzioni

Knowledge and Data Engineering,
Department of Computer Science,
Trinity College, University of Dublin.

September 2011

Declaration

I declare that this thesis has not been submitted as an exercise for a degree at this or any other university and it is entirely my own work.

Zohar Etzioni

_____

September 2011

Permission to Lend and/or Copy

I agree to deposit this thesis in the University's open access institutional repository or allow the library to do so on my behalf, subject to Irish Copyright Legislation and Trinity College Library conditions of use and acknowledgement.

Zohar Etzioni

_____

September 2011

Acknowledgments

Firstly, I would like to thank my supervisor, Dr. Dave Lewis, whose vast knowledge and experience, patience and encouragement has made this work possible. Special thanks are also reserved for Dr. John Keeney, my co-supervisor, whose insightful input was invaluable.

I would also like to extend my gratitude to the members of the Knowledge and Data Engineering Group (KDEG) in Trinity College Dublin. Their friendship and insightful contributions have had a significant impact on this research.

I would like to express my gratitude to the Science Foundation Ireland, for funding the research detailed in this thesis.

As importantly, I would like to thank my family. My parents, for their priceless encouragement, belief and guidance throughout my life, and my wife, Drorit, for her patience and unconditional love and support.

Special thanks to Alex K. for the inspiration, guidance, and support which made this research possible, and for teaching me the true meaning of *insanity*.

Most of all to my children, Nimrod, Noa, and Nadav for their patience, I promise I'll play with you now!

Abstract

In recent years the Home Area Network (HAN) has been going through a revolution. From enabling multiple desktops in the household to share an Internet connection, it evolved as a service-oriented platform, enabling devices to communicate with each other. The HAN hosts devices from multiple applications domains including entertainment, home automation, security and healthcare. A plethora of different technologies have been suggested for addressing the challenges of the HAN, have been widely deployed. These protocols and standards enable devices and services to be discovered and to interact with each other, however they lack native support for service composition, are incompatible with each other, and are limited by design to a single HAN. While service interoperability efforts have focused traditionally on a single household, the growing interest and user demand for content sharing has promoted efforts into supporting interoperability of the existing HAN service between HANs. While a number of systems in literature address the intra-HAN and inter-HAN service interoperability separately, the problem of integrated service interoperability across both remains unsolved.

This thesis therefore addresses the need for an integrated system to support both intra-HAN service interoperability enabling services from multiple service protocols to interact and be composed, and inter-HAN service interoperability, enabling services from multiple service protocols to be securely shared with remote HANs. This thesis synthesises the requirements for an integrated service interoperability system and introduces the integrated *Krox* system architecture and design that satisfy these requirements through a pluggable architecture with plug-ins per service protocol and an extensible event model that specifies the interaction between plug-in components in remote HANs. The *Krox* architecture supports inter-HAN service interoperability through service virtualisation. It supports intra-HAN service interoperability through representation of local and remote services as web services in the

local HAN enabling their composition using standard web service orchestration techniques. The *Krox* system architecture builds on the Instant Messaging & Presence (IM&P) user metaphor, to share devices and composite services between HANs in a secure and scalable manner. This thesis demonstrates the feasibility of the architecture and its generality across service technologies through the implementation of plug-in instances for UPnP and Jini.

The thesis presents an evaluation of the key *Krox* system performance parameters that affect intra-HAN and inter-HAN service interoperability. Finally the thesis includes a comprehensive security analysis of potential relevant attacks on a HAN and how they can use the system to spread to remote HANs.

Contents

List of Figures

List of Tables

Chapter 1
INTRODUCTION

Recent years have radically changed the concept of a Home Area Network (HAN) beyond enabling multiple desktops and laptops at home to share an Internet connection and exchange documents and files. Led by the increasing availability of high-speed Internet connectivity and the reduction in cost of high performance networked devices, the HAN now hosts many network-enabled devices in various application domains including digital entertainment, home automation, intelligent appliances, home security, and healthcare. With network-enabled devices including TV, media centres, Network-Attached Storage (NAS), tablets, cameras, printers, game consoles, picture frames, and others, gaining in popularity, it is predicted that the number of connected devices on HANs will keep rising to over 7 billion devices by 2015 [56]. In parallel, the number of HANs has grown from 100 million in 2007 to an expected 200 million households with a HAN in 2011 [44]. Home networking has the potential therefore to cut across the traditional boundaries between application domains and industries. Legacy home automation systems (e.g. for the control of lighting, heating and cooling) have the potential to be fused with state of the art entertainment systems, home security systems, and health monitoring systems amongst many others to yield new integrated applications. However, the persistence of this application-centred industry fragmentation means that no single domain-independent standard for exposing and invoking services in the HAN has emerged. With the assumption that no single technology for device internetworking is likely to become a de facto standard across these multiple application domains, interoperability between protocols becomes of paramount importance. Service Oriented Computing (SOC) [96] is a promising paradigm that can enable interoperability between different service protocols, potentially from different device vendors. While already used in several HAN service protocols in a different way, applying SOC concepts and principles more consistently in the HAN can open new opportunities for

both consumers and service providers to create new applications and new reusable services from existing HAN device services through service composition. In this way, service composition can motivate more application-driven intra-HAN service interoperability.

In addition, content sharing via the Internet has become popular in recent years. Peer-to-peer file sharing systems that broker sharing activities beyond immediate social cycle of family and friends, have often conflicted with the interests of copyright holders. However, in the recent years the commercial success of many Web 2.0 based enterprises such as Facebook[1], and Flickr[2], was strongly based on harnessing the compulsion for sharing within social networks. Users wish to share their media with family and friends, however, they may be wary of doing so while giving up their usage and administration rights to a third party [113]. There are several reasons for users concern, such as the potential exposure of their information to 3rd parties, or unintentional exposure beyond the expected audience. In the context of the HAN, sharing resources is not limited to just content, as other networked services available at home can also be usefully shared with trusted remote HANs. Home users can benefit from consuming various services in other networks, e.g. remote security or health monitoring, printing on a remote printer, using available space in a remote Digital Video Recorder (DVR), consuming remote content such as photos, streaming user generated content from remote HANs. Sharing of HAN services with remote HANs can enable seamless collaboration between devices across multiple HANs, which here is termed inter-HAN service interoperability. This, coupled with service composition, opens new application opportunities for both consumers and service providers.

In the following sections, the scope, goals, and objectives of this research is presented, addressing how the HAN can be extended for integrated service interoperability both inside the HAN and across multiple HANs, overcoming the barriers to integration that exist.

[1] www.facebook.com

[2] www.flickr.com

1.1. Motivation

The Service Oriented Computing (SOC) paradigm has emerged in the last few years as a key approach to building loosely coupled, low cost, interoperable, evolvable software systems [95]. The core of service oriented computing is the concept of a *service*. A service is a program or a piece of software that is autonomous, capable of completing a task, can be published, discovered, invoked, and composed [96]. Singh et al. argue that while there can be some value in accessing a single service, the greater value is achieved through enabling the composition of services, which leads to creation of new services from existing ones [120]. Typically in HANs, devices offer services to other devices or to end-users. For example, logically connecting a media server with a media renderer and speakers enables streaming multimedia directly from the source without having to physically move the media source (e.g. a DVD disc) to the media player. Connecting via a mobile phone for controlling home lighting or air-conditioning system can provide fine-grained one click home control from your bed. HAN networked devices should be able to use SOC to seamlessly integrate themselves into the network and be able to discover and communicate with each other. This must be done without manual administration and control, which is typically impractical for home users.

Several service protocols for addressing the HAN challenges have been widely deployed, including Universal Plug and Play (UPnP) [82, 130], ZeroConf [29], Jini [7], Zigbee [147], Device Profiles for Web Services (DPWS) [28], Service Location Protocol (SLP) [47], Bluetooth [15], Home Audio-Video Interoperability (HAVi) [75] and Open Service Gateway initiative (OSGi) [124]. Driven by the requirement for minimal to zero configuration, these protocols and standards define how devices and services connect to the network, and how they discover and interact with each other. Attempts have been made to promote interoperability amongst devices in the home network however they were limited to specific application domains, such as multimedia or home automation. For example, the Digital Living Network Alliance (DLNA) [34] is an industry consortium formed to promote interoperability of Internet, mobile and broadcast services through simple integration of consumer devices with home networks, and to operate by certifying compliant devices. However interoperability is promoted by DLNA by selecting “best of breed technologies” and certifying devices that support them, rather than defining interoperability interfaces between multiple service protocols, which co-exist in the HAN. To address interoperability

between service protocols, this thesis proposes a fuller application of SOC principles in the HAN. It uses service composition to support integration of services implemented using different protocols and for the generation of new, composite, services that in turn can be discovered and invoked via different service protocols.

HAN service interoperability efforts have focused on the single household, however the increased value placed by users in the content and services held by devices and their interplay, rather than the value of the device itself, makes the sharing of content and services an increasingly attractive option. At the same time, the widespread adoption of SOC for service discovery and invocation within the home provides a readily extensible architecture for sharing services between homes. Such sharing must, however, address the HAN users' wish to remain in control over what they share and with whom, as opposed to unrestricted P2P sharing. Privacy is also a major issue, so resources may need to be shared with multiple HANs with different access rights. Sharing must be easy to perform, easy to configure and must be performant to allow users to share and consume remote resources as easily as they do local ones. Allowing users to share HAN devices, services, and content with their friends and family in a controlled way could increase the potential value of the HAN for both consumers and service providers. Consumers would benefit from having more services available in their local HAN. Service providers could offer more innovative composite services leverage services already deployed in the local HAN.

While the intra-HAN and inter-HAN service interoperability have been addressed separately in literature, an integrated approach is still missing. Such integration between intra-HAN and inter-HAN service interoperability may enable services from multiple HANs to be seamlessly discovered, invoked and composed across multiple HANs. Being able to compose local and remote services seamlessly can enable a new platform for 3rd party vendors to offer new applications from existing services and thereby help HAN users to realise more of the network potential of their HAN devices.

1.2. Research question

Creating a system that integrates intra-HAN and inter-HAN service interoperability presents several challenges:

1. Extending HAN service protocols to *multiple HANs*: Most of the common HAN service protocols are not natively extensible to multiple HANs by design. The main challenge is how to extend a HAN service protocols beyond the scope of a single household while allowing local HAN client applications to seamlessly integrate with remote services.
2. Enabling services from *multiple service protocols* to be *composed together*: Service protocols for HANs (e.g. UPnP, Jini, DPWS) are not interoperable with each other, which hinders the ability to compose them for creating complex applications from existing services.
3. *Performance*: As a system that runs in the HAN, solutions must not have a notable impact on the HAN resources, such as CPU processing power, memory consumption, and bandwidth. Such a system must scale to simultaneously support sharing with a number of households that is representative of a typical domestic social network including a number of close family and friends.
4. *Security*: With the assumption that HAN devices work behind a home gateway firewall, vendors of HAN devices and service protocol designers often relax security requirements in favour of ease of use. However when sharing the same resources with remote HANs over an unsecured network such as the Internet another balance must be found between simplicity of configuration and the need to protect against unauthorised access or malicious attacks. From a user's point of view, sharing resources with remote HANs should be done with minimal additional potential vulnerability for the HAN.

Therefore the research question addressed by this thesis is: how to enable integrated sharing and composition of devices, services and content within the HAN and between multiple HANs in a secure and performant manner?

1.3. Research objectives

To address this research question the following research objectives are pursued:

1) A review of the state of the art in intra-HAN and inter-HAN service interoperability to establish the requirements for an integrated approach.
2) The design of an architecture that extends current intra-HAN systems with support for service composition that uses multiple service protocols, and inter-HAN sharing

of services scalable to a number of HANs consistent with sharing with a household's personal circle of family and friends.

3) Validate the architecture through implementation with two established HAN service protocols.
4) Evaluate the performance and security of the system implementation.

1.4. Research methodology

In his classification of research methodologies [67], Jarvinen distinguishes between approaches studying reality and mathematical approaches. The study of reality is further classified as "what is reality" and "utility of artefacts", also referred to as design science [49]. This research adopted the design science research methodology as a problem solving paradigm that leads to innovations that have utility in the problem domain. Design science requires research to produce an artefact in the form of a construct, a model, or a method. The objective of design science research is to develop a technology-based solution to a relevant business problem [57]. The business problem addressed by this thesis is how services from the HAN could be shared with remote HANs and composed with both local and remote HAN services to enable the introduction of complex composite services from existing services. Design science requires the design utility, quality and efficacy to be demonstrated via well-executed evaluation, which must be constructed and applied rigorously.

The design artefact of this thesis is the *Krox*[3] architecture and system design, which addresses the unsolved problem of integrated intra-HAN and inter-HAN service interoperability. *Krox* architecture defines a method for sharing devices and services from multiple service protocols, enabling their service composition in local and remote HANs. The architecture defines an extensible event model for the plug-ins for supporting inter-HAN service interoperability through automatic representation of remote devices and services as virtual devices and services in the local HAN. This facilitates the seamless integration with client applications in the local HAN and their interoperability and composability with other services in the local HAN. Intra-HAN service interoperability and

[3] http://starwars.wikia.com/wiki/Krox

service composition are integrated with the *Krox* system architecture through the use of a common service model to enable service interoperability and service composability of services from multiple service protocols, from local and remote HANs.

The utility of the design artefact is demonstrated with two service protocols (UPnP and Jini). The contributions of design science must be clear and verifiable in the area of the design artefact so a clear description of the design is given in chapter 4, and the corresponding implementation is described in detail in chapter 5. Two evaluation methods were used to assess the quality and efficacy of the design artefact, experimental evaluation and analytical evaluation. The performance evaluation used emulated devices designed for the purpose of evaluating the system in a controlled experimental environment (section 6.2, 6.3). The security analysis uses attack trees as a formal framework to identify potential attacks on the system and on HAN service protocols (section 6.4).

1.5. Contributions of the research

While a number of systems were suggested in literature for sharing services from HANs [10, 12, 31, 51, 55, 65, 76, 77, 84, 97, 119, 121, 137, 139], none of them aims to define a generic method for sharing services via multiple service protocols, that *combines* use of a standard application layer communication mechanism, simple configuration, and support for seamless integration of remote services with existing applications in the local HAN. In addition, they do not address the challenges of intra-HAN service interoperability and service composition. Similarly, while the subject of intra-HAN service interoperability has been addressed by a number of systems in literature [16, 17, 18, 52, 106, 126], those systems lack the support for inter-HAN service interoperability.

The key contribution of this thesis is therefore, the design, prototype implementation, and evaluation of the *Krox* architecture and system design for *integrated* intra-HAN and inter-HAN service interoperability. The approach is based on standard secure communication protocols enabling HAN services from remote HANs to be discovered in the local HAN and to seamlessly interact and be composed with local HAN services and client applications. The generality of the approach is demonstrated with implementations for two important service-oriented HAN protocols: UPnP and Jini.

While security is a major challenge and a clear requirement to systems for inter-HAN service sharing, very little attention has been given in the literature to the security aspect of resource sharing systems. A minor contribution of this thesis therefore, is the security analysis for sharing of HAN resources, specifically for UPnP and Jini networks, in the form of attack trees, which could be used for future HAN sharing research.

Very little information exists in the literature regarding the performance of systems for sharing services between remote HANs. A minor contribution of this thesis, therefore, is a comprehensive performance evaluation of inter-HAN interoperability, identifying the key parameters that affect inter-HAN service sharing and providing a baseline for future benchmarks with other systems.

1.6. Publications

The contributions of this thesis were published in the following peer reviewed papers:

I. David Lewis, Stephen Curran, Kevin Feeney, Zohar Etzioni, John Keeney, Andy Way, and Reinhard Schäler 2009. Web service integration for next generation localisation. In Proceedings of the Workshop on Software Engineering, Testing, and Quality Assurance for Natural Language Processing (SETQA-NLP '09). Association for Computational Linguistics, Stroudsburg, PA, USA, 47-55.
II. Rob Brennan, Dave Lewis, John Keeney, Zohar Etzioni, Kevin Feeney, Declan O'Sullivan, Jose A. Lozano, Brendan Jennings, Policy-based Integration of Multi-Provider Digital Home Services, IEEE Network Magazine, special issue on Digital Home Services, 23, (6), 2009, p50 - 55
III. Zohar Etzioni, John Keeney, Rob Brennan, David Lewis, Supporting Composite Smart Home Services with Semantic Fault Management, The 5th International Symposium on Smart Home (SH 2010) at FutureTech 2010,21-23 May 2010 in Busan, Korea, 21-23 May 2010, 2010

IV. Zohar Etzioni, Kevin Feeney, John Keeney, Declan O'Sullivan, Federated homes: Secure sharing of home services, IEEE Consumer Communications and Networking Conference (CCNC 2011), Las Vegas, USA, 9th-12th January, IEEE, 2011

V. Rob Brennan, Zohar Etzioni, Kevin Feeney, John Keeney, Declan O'Sullivan, William Fitzgerald, Simon Foley, Federated Autonomic Management of HAN Services, to appear in IFIP IEEE International Symposium on Integrated Network Management, IM2011 Technical Sessions, Dublin, Ireland, 23rd-27th May, 2011

The contribution of [I] to this thesis is mainly in the foundation of the concept for service composition with BPEL. My main contribution to this paper was the technique of composing linguistic services from multiple providers with BPEL service orchestration. While this work is not directly related to this thesis it helped in establishing the idea of using web services as a canonical service description and leverage the composition capabilities of BPEL for web services.

The contribution of [II] to this thesis is in setting the scene with the motivating scenario and home area network technologies. The paper looks at the problem of end-to-end service management for federating multi-provider services. My contribution to this paper was focused in service composition for home networks. At this point the idea of mapping UPnP services to web services and their composition with BPEL was developed.

The contribution of [III] was in the development of a prototype for plug-in based architecture that enabled cross service protocol service orchestration. The prototype demonstrated the architecture with UPnP and was extended to support composite service fault management based on semantic annotations and fault ontology. Though this semantic management approach was not pursued further in the thesis, the paper introduced the motivation for interoperability between HAN service protocols for service orchestration and the plug-in based approach for automatic mapping from a service protocol (demonstrated with UPnP) to web services.

The contribution of [IV] was the presentation of the design for the sharing system based on XMPP as a communication channel and the extension of UPnP for multiple home networks. Additionally, this paper presented the application of model-based capability sharing management and its application to management of resource sharing between HANs.

Finally, the contribution of [V] was in the presentation of partial evaluation of the sharing system with an earlier prototype of *Krox* system architecture and design.

1.7. Thesis outline

The rest of this dissertation is structured as follows: This chapter has briefly introduced the motivation for sharing and composition of HAN services, the main challenges associated with it and presents the goals and objectives for the rest of the work.

- Chapter 2 sets the scene of home area networking and the infrastructure, concepts and technologies involved in home networks and service architectures. The concept of service oriented computing is introduced and the various service protocols for HANs are assessed with regard to their service orientation.
- Chapter 3 describes the interoperability problem in the HAN and between multiple HANs and surveys existing approaches for solving this problem. The main focus of the chapter is a review of the research directions for sharing home devices, services and content between home networks that have been published in literature. Finally the requirements for a service-oriented architecture for sharing and composition of multiple HAN services are presented.
- Chapter 4 introduces the *Krox* service-oriented architecture for inter-HAN and intra-HAN service interoperability. The second part of the chapter describes the details of the design of plug-ins for UPnP and Jini service protocols, and illustrates the realisation of the *Krox* system architecture with these service protocols.
- Chapter 5 presents a prototype implementation for *Krox* architecture and design with plug-in implementations for UPnP and Jini service protocols, supporting service sharing between HANs and service composition. The prototype demonstrates the feasibility of *Krox* system architecture and system design.
- Chapter 6 presents an evaluation of the *Krox* system design and implementation. The evaluation focuses on two significant aspects of the system behaviour: performance, and security. The performance evaluation examines the system's behaviour under stress conditions. The security analysis discusses the potential security weaknesses of the system and ways to protect against them. Finally the chapter concludes how the objectives of the thesis were met and how contributions were achieved.

- The final chapter presents the conclusions of the work presented in this dissertation and summarises the contribution. Finally, areas for potential future work are identified.

Chapter 2
BACKGROUND

The widespread availability of network enabled electronic devices for a variety of different application domains are finally fulfilling the vision of Mark Weiser from 1991 [140] for ubiquitous computing environment where computing devices are gracefully integrating with human users. In recent years home networking has been going through a revolution, from a few desktops sharing Internet connections to a network of digital appliances, devices with embedded computing and networking that are capable to support IP level protocols. The home area network can host diverse devices from multiple application domains including entertainment (e.g. home theatre systems, gaming consoles), security (e.g. surveillance cameras), communication (IP telephony), and comfort (e.g. heating and cooling, lighting, health care). In reality, the availability of devices from multiple application domains did not result in full-networked collaboration between devices but only in "functionality islands" that are not interoperable with each other [19]. Moreover, even within an application domain, multiple competing protocols and standards were suggested. The combination of the diversity of devices protocols and their lack of interoperability with each other creates a real challenge for creating applications that connect devices from multiple protocols [18]. Mixing services from multiple service protocols and application domains can enable creation of richer value-add services for the home environment.

Nowadays, along with the social networking trend as well as the emergence of user generated content, users are interested in sharing their home resources and content with their family and friends as well as being able to consume content and services remotely. An ABI market research [1] argues that the next generation in the evolution of the HAN is centred on sharing media between connected devices from the HAN. However users wish to share

their services and content while remaining in control over what they are sharing, with whom and when without compromising their privacy [113].

The purpose of this chapter is to analyse the HAN from a service oriented computing point of view: i.e. assess the service orientation of the HAN device and service protocols and standards based on a criteria that is established in this chapter. Since this thesis is focused on extending the service oriented HAN beyond the scope of a single household for integrated intra-HAN and inter-HAN service interoperability, the analysis is necessary for understanding capabilities and gaps of existing service protocols used with HAN with regard to the objectives of this thesis to present an integrated system for composition and sharing of home services.

The next sections define the scope of the home area network, and specify the criteria for comparison between the home network technologies in relation to common service oriented concepts. The rest of the chapter describes and compares the relevant service protocols and standards according to the defined criteria.

2.1. Home area networking

With the widespread availability of always-on high speed broadband to an ever-growing percentage of households, the usage scenarios of home networking have evolved from web browsing and printer sharing to complex peer to peer communication and interaction between devices.

A Home Area Network (HAN) is defined by Oh et al. [93] as interconnecting electronic products and systems, enabling remote access to and control of those products and systems, and any available content such as music, video or data. Rose identified a number of requirements from this definition [110]: (i) devices need to be able to connect with each other; (ii) devices need to facilitate access to content that can come from the home network device or from external service providers; (iii) devices need to enable their control from either within the home or remotely. Another point of paramount importance in home area networking is the ease-of-use as users expect technology to be simple, transparent and working. Survey of key factors impeding home networking uptake [74] name complexity of installation and configuration as a key factor. In contrast with enterprise networks, the home

network lacks the function of the expert system administrator. Typical home users can be considered non-technical, therefore most HAN device features must be supported "out of the box" and not require additional user configuration, which is not appropriate for non-technical home users.

The HAN is connected to the Internet over cable, telephone, or electricity network through a residential gateway (also referred to as home gateway or HG) [26]. In addition to connecting the home network to the external network, the HG can provide additional services such as quality of service (for both upstream and downstream), firewall [41], Network Address Translation (NAT) [38], Dynamic Host Configuration Protocol (DHCP) [36] and Virtual Private Network (VPN) [72]. Network interfaces typically offered to the HAN by HG include Ethernet [81], IEEE802.11 [62], HomePNA (phone line) [58], power line [100], IEEE1394 [63], Bluetooth [15], and Universal Serial Bus (USB) [135]. Home gateways run three layers of software: the firmware layers which runs diagnostics, boot loader, and additional required support for the operating system. The second layer is the operating system, an embedded operating system (such as Embedded Linux, VxWorks[4], and Nucleous[5]) equipped with drivers for all the home gateway's physical interfaces. On top of the operating system runs the application layer of the home gateway, which supports the communication protocol stacks for routing, bridging, DNS, NAT, DHCP, VPN, Firewalls, and system management. Other application layer support often includes Voice over IP, OSGi support, and support for audio/video streaming protocols such as Real-time Transport Protocol (RTP) [116], Real-time Streaming Protocol (RTSP) [117], and Real-Time Transport Control Protocol (RTCP) [116]. HG components are based on open standards such as IEEE specifications, IETF RFCs, and other industry standards such as UPnP Forum, OSGi Alliance, and DSL Forum.

A home area network establishes connectivity between multiple devices at home and enables their access and control using one or more of several network interfaces. Additionally some device and service technologies leverage the physical connectivity for providing application level protocols that are used for inter device communication. The next section presents the existing technologies for physical medium connectivity for home area

[4] http://www.windriver.com/products/vxworks/

[5] http://www.mentor.com/embedded-software/nucleus/

networks.

2.1.1. Physical medium connectivity for HAN

The physical connectivity of the HAN defines the physical aspect of how devices communicate with each other. The purpose of physical connectivity technologies is to provide network level interoperability and enable higher-level service protocols to be established in the HAN. A common classification for HAN physical connectivity technologies refers to wired technologies and "no new wires" technologies [143].

The most common wired technology is Ethernet [81], which offers mature technology from years of use in enterprise networks, high-speed communication and no complex configuration. Its simplicity and low cost make it one of the major alternatives for the HAN. One drawback of Ethernet for home networks is that it does not have inherent support for QoS for isochronous streams, which is needed for multimedia scenarios [143]. IEEE1394 [63] offers another wired connectivity alternative with a digital interface that integrates entertainment, communication and computing into a single consumer multimedia network. IEEE1394 supports peer-to-peer high-speed communication and both asynchronous and isochronous data transfer. As IEEE1394 was designed mainly for consumer applications it is a very common choice for entertainment networks. Universal Serial Bus (USB) [135] is typically used to connect peripheral devices such as mass-storage devices directly to other devices.

The other approach for home networking is "no new wires" which indicates that connectivity should be provided either by using technologies that already exist at the home such as power distribution network, or based on the telephone line, or by using radio technologies. PowerLine [100] is based on connecting devices to the network through the mains power supply. PowerLine is considered an attractive wired connectivity alternative for the connected home due to its simplicity and since it does not require any "rewiring". A disadvantage of PowerLine is the lack of standardisation. Home Phone-line Network Alliance (HomePNA) [58] offers high-speed communication over coax and phone lines. Phone line networking has a number of challenges, such as the signal noise on phone lines from home appliances, and the signal attenuation in delivering data, voice and video throughout the HAN caused by the random topology of telephone wiring. IEEE802.11 [62]

became very popular for wireless LAN (WLAN) in recent years enabling wireless connectivity to the local HAN with the reduction of implementation costs. The IEEE802.11 addresses both the physical layer and the Media Access Control (MAC) layer. Ethernet is often used as wired Local Area Network (LAN) with IEEE802.11 interacting with Ethernet through a wireless access point (Wi-Fi), another option is Wi-Fi integrated in the home gateway. Bluetooth [15] is another industry standard for wireless communication based on short-ranged, low-cost radio. Bluetooth was designed to replace infrared and cabling technologies for communication between portable devices such as laptops, PDA and mobile phones. Bluetooth relies on IEEE802.15.1 [64] as its physical layer and supports a Service Discovery Protocol (SDP) [15]. Bluetooth is mainly used to connect between mobile devices and laptops, printers, tablets, keyboards, and gaming devices for replacing cumbersome wiring. ZigBee [147] is a specification for high level communication protocols for low power radios aimed to support low-cost, low power networks – mainly but not only sensor networks. Most of the applications for Zigbee are in the home network are related to home automation, such as energy management, including heating, lighting, and air-conditioning.

2.1.2. Summary

The home area network enables devices to be distributed around the household. Moreover, while some networked devices in the HAN are stationary, others are mobile and can be located in different places at different times, and can join and leave the network frequently. The distribution of devices in the HAN and the mobility of devices lead to required dynamism of the network configuration. The HAN hosts devices from multiple application domains such as home automation, entertainment, security, and healthcare. The heterogeneity of applications lead to heterogeneity of technologies and standards - each industry has its own hardware standards, software and protocol evolution. Interoperability at an industry level – such as between multimedia devices and home automation device, and at a vendor level – such as between multimedia devices from multiple vendors, is a great challenge.

With the heterogeneity of devices added to the dynamism of configuration, the result is increased management complexity. Typical management tasks involve configuration of network resources, assurance of quality of service, provision of dynamic network changes

and resources, and implementation of security measures and access rights. Ibrahim et al. [61] argue that these tasks and repeating them per network resource can be time consuming as well as complex and error prone. However this contradicts with the ease of use that is desired for the HAN where users are typically non-technical. This implies that systems and applications running in the HAN must not require deep technical understanding and must automate as much of its operations as possible. In addition, while a market exists for various price levels, the vast majority of consumers look for cheap solutions, devices, and applications, however they require these to have acceptable performance and security levels. Home users see the HAN as a private space. Users expect their home network to be secure against malicious attacks, against eavesdropping, unauthorised access, phishing, privacy violations, and others. However the management complexity resulting from the dynamism and device heterogeneity makes security a challenge.

This section presented the home network environment with the physical connectivity infrastructure, and its potential applications and challenges. The physical infrastructure by itself is not enough for devices to successfully interact. Higher-level protocols are needed to enable devices and services to be discovered, described, invoked, and reused by other devices or applications in the HAN. The SOC paradigm, presented in the following section, defines concepts that can be used for home networking applications to face the challenges presented above in this section.

2.2. Service-oriented computing

Service oriented computing (SOC) is a paradigm that defines principles and concepts that support the development of rapid, low cost, and easy composition of distributed applications through the concept of services. A *service*, in SOC, is merely a networked software with a fine-grained goal that can act as an autonomous unit with respect to other service-oriented programs, and can be described, published, discovered, invoked and assembled [96]. Papazoglou defined 3 principles for services in SOC [95]: (i) loose-coupling between service providers and service consumer - such that interaction with the service does not require knowledge of the internals of the service at the client or service side; (ii) technology-neutral - a service must be independent of a programming language or operating system; (iii) location transparency - such that services should have their definition in a repository

that is accessible by clients that can in turn find a service and invoke it, irrespective of the service location.

Service oriented computing defines distinct roles for service providers, services consumers, service brokers and a loose coupling between them. The service provider describes the service, implements it, and publishes it in a service registry, or elsewhere where service consumers can find it. The service broker operates the service registry, enabling service providers to register services and service consumers to find them. The service consumer searches for a service, finds it, binds to it, and then invokes it.

The concept of a service as an atomic unit of functionality promotes reusability; the loose coupling between service consumer and service provider, the technology abstraction, and the location transparency, enable to create composite services by assembling existing services, potentially from different service providers for achieving some goal.

2.2.1. Web services

Web services present a promising realisation of the service-oriented computing [138]. They offer loosely coupled networked autonomous units of functionality. Web services are published, discovered, composed, and are designed for ease of integration via a set of standard commonly used protocols across platforms, programming languages and enterprises. Web services can be dynamically discovered and invoked over the web. Web services are offered in two flavours: web service based on Simple Object Access Protocol (SOAP) [20]; and web services based on the Representational State Transfer (REST) approach [39].

Web services based on SOAP, are described in Web Service Description Language (WSDL) [32]. WSDL is a machine-readable specification using XML [22] syntax to describe and specify the functional capabilities of the web service. The WSDL specification includes information required for a consumer to interact with this service. SOAP is used as a message passing protocol that defines how messages are sent over the wire for interacting with the service. SOAP web services are invoked via SOAP messages typically but not exclusively over HTTP.

RESTful web services are based on Representational State Transfer (REST), an architecture style that was first referenced by Fielding [39] in the context of the massive scalability of HTTP. RESTful web services expose a set of resources through URI and are accessed using a small set of remote operations that describe the action to be applied on the resource. The services are stateless and use only universal operations: PUT, GET, POST, and DELETE. RESTful web services do not use SOAP, WSDL, or any other web service specifications. Instead they only leverage HTTP as an application protocol.

Semantic Web Services (SWS) [79] suggest a semantic based approach for interoperability. The semantic web [14] transforms the web into a repository of a machine-readable data, and web services provide the means for consuming this data. Semantic web services extend the definition of web services to support not only service syntax but also service semantics, to enable semantic service interoperability. SWS describe a service not only syntactically but also provide a semantic description of the service, thereby enabling the automation of dynamic discovery, selection, and composition of services.

2.2.2. Service composition

SOC defines two types of services: a *simple* service, and a *composite* service [95]. Simple services are the smallest unit of reuse in SOC that accomplishes some task without using additional services in its implementation. Composite services involve assembly of existing services potentially from multiple sources for accomplishing some task. Singh et al. define service composition as a form of putting services together to achieve some desired functionality [120].

There are several categories for service composition approaches. Static and dynamic service composition, are strategies that differ in the time concrete services are composed. Static service composition refers to a non-adaptive composition, where services are hard-wired to each other at composition design time. With dynamic service composition, only control flow is hard-wired, while services can be dynamically selected when the composition is executed.

Service orchestration [5] refers to an explicit description of the interactions through message

exchange between multiple web services along with control and execution flow that is controlled by a single party. Service orchestration describes the composite service from the point of view of a single participant in a hub/spoke model. Business Process Execution Language (WS-BPEL or BPEL in short) [5, 69] is a standard language for describing SOAP web services orchestrations. In addition to being able to compose services, in BPEL the service composition is also rendered as a SOAP web service described with WSDL. The language enables definition of both abstract and concrete processes. BPEL terminology includes a process concept, which is the service composition itself; partners, which define the services that are taking part in the composition; and activities, which correspond to an exchange of messages with a partner or some transformation on the messages. The business process logic is manifested in the process description as a set of interactions between constituent web services and structured activities facilitating sequential and parallel execution of process flows. BPEL defines several control flow structures that facilitate the interaction between web services. As it is targeted for running long-lived business processes, it supports transactions, fault handlers and compensation handlers. Working with BPEL typically involves using a tool for the design of a service orchestration, which would create the service orchestration template. This template then needs to be deployed into a BPEL engine to enable its execution and calls to constituent services. Typically the BPEL engine would provide some administration for deployed BPEL service compositions, such as suspend, resume, get state, and reconfigure, but no standard has been agreed.

Service choreography [101] is another alternative for service composition that relies on collaboration between services. Service choreography does not require nor depend on a central controller. Instead it is based on multi-party peer-to-peer collaboration. In choreography, the message exchange is not defined from the point of view of a single participant, but from the perspective of all parties.

Another loose form of composition is referred to as mash-up, where for mash-ups content from unrelated data sources (e.g. Google maps, Flickr, news sources, shopping sites) is composed in an innovative way to create useful new content made for human (rather than machine) consumption [144]. A mashup can be created using traditional web application server-side dynamic content generation technologies, such as PHP or using client-side scripting with JavaScript. Mashups can use both technologies (client and server) for implementing their business logic.

2.3. HAN service standards and protocols

The previous section presented the concepts of service orientated computing in general. In this section various service standards and protocols for enabling client applications to control devices and device-to-device communication are presented. The main purpose of HAN service protocols is to enable applications to discover and control devices and allow interoperability between devices supporting similar protocols in the HAN, for improved user experience or enabling new applications and services to be constructed from existing ones. While a large variety of service standards and protocols were suggested for the HAN, the focus of this survey is IP level service protocols and standards. The review includes the following service protocols: UPnP [82], DPWS [28], Jini [7], SLP [47], ZeroConf [29], OSGi [124], and HAVi [75]. Additional service protocols that do not work at IP layer were excluded from this survey: Bluetooth [15], Zigbee [147], LonWorks [85], X.10 [125], HomeRF [89].

Before the service protocols are described it is important to define the criteria for their comparison. Zhu et al. [146] suggest a classification of service protocols for pervasive environments based on several distinguishing attributes. To highlight the differences the attributes can be divided into several groups:

(i) How services are discovered - Search mechanism, communication method, support for query, support for advertise, scope of discovery

(ii) How services are used (how services are selected from the result of a search, how service interaction is facilitated, how can the service status be inquired)

(iii) Supported security mechanism – Authentication, authorisation, confidentiality, integrity, and privacy.

This classification is extended here to form a more comprehensive comparison criteria as the following:

- **Scope and market uptake**
    - **Application domain** – Services in the home network come from multiple different application domain and industries: entertainment devices (e.g. media

servers, speakers, TV), computing products (e.g. pc, laptop), telecommunication devices (e.g. IP phone), home automation (e.g. lighting, Heating Ventilating and Air Conditioning - HVAC, security systems), mobile devices (e.g. smartphones), and home appliances (e.g. white goods, brown goods). While some service technologies are generic, others specialise in specific application domain, such as entertainment multimedia systems, and home automation. The purpose of this criterion is to evaluate the generality of the protocol. While the scope of this thesis is not limited to a specific set of devices or application domains, the set of supported devices will be derived from the set of devices supporting the relevant service protocols. However, from the application domain it can be concluded how pervasive the service technology is, and the more generic its use is, the better it is as a case study for this thesis.

  - **Market acceptance** – It is important to understand the market uptake of a service protocol or standard. The relevance to the thesis is in order to avoid selecting service protocols with low market acceptance as case studies for the sharing and composition system.
  - **Standardisation body** – It is interesting to compare the various standardisation bodies and the segmentation of their support for the different standards and protocols.

- **Generality**
  - **Physical layer dependency** – While some service technologies are physical layer agnostic, some other depend on a specific physical layer.
  - **Programming language dependency** – Some protocols and standards are programming language agnostic while others depend on a specific programming language for facilitating the interaction with the service.

- **Service orientation**
  - **Service discovery** – The most basic aspect of service-oriented computing is the ability for services to be discovered. There are several approaches to service discovery, such as advertisement of services by their host when they become available in the network, responding to search requests for a specific service type, or using a lookup service as a broker between the service providers and service consumers. Service discovery will play an important role in the system design for service sharing and composition, therefore it is important to study the various approaches used by the different protocols and standards and consider their extensibility.

- **Service description** – A service description provides information about the actions supported by the service. The importance of this factor is in the ability to extract information about the service from its description. The more information about the service and its supported actions can assist in its composition with others. Services that cannot be described cannot be composed with other services in a generic fashion.
- **Service invocation** – Compares the level of support for service invocation. There are three levels of support for service invocation that can be provided by a service protocol [146]:
  (i) Service location – A network address representing an endpoint where the service can be communicated
  (ii) Communication mechanism – In addition to service location, a mechanism for invocation of service methods, e.g. Java RMI, SOAP.
  (iii) Application Programming Interface (API) – Domain specific application domain operation definition.
- **Service composition –** To what extent does the service protocol includes native support for composition of services.

- **Extensibility**
  - **Multi-home readiness** – To what extent is the service protocol supporting discovery and access to services from outside the HAN.
- **Non functional aspects**
  - **Security** – There are many aspects for security such as authentication, authorisation, confidentiality, privacy, and data integrity. The purpose of this parameter is to understand the extent of support for security that is embedded in the protocol.
  - **Performance** – Are there any known performance issues with the service protocol?

The following sections present a number of service protocols and standards for home area networks that can be considered service oriented protocols. Finally section 2.3.8 presents a comparison of the protocol and conclusions from the comparison.

2.3.1. Universal Plug and Play

Universal Plug and Play (UPnP) [82, 130] is a standard and an architecture initiated by the UPnP Forum, which was formed in 1999. The UPnP Forum [130] currently has members from 929 leading companies from various industries: consumer electronics, printing, networking, home appliances, automation, security, and mobile products. The UPnP architecture is a plug and play service oriented architecture for enabling seamless home networking. UPnP relies heavily on standard protocols such as IP, UDP, HTTP and XML. UPnP does not dictate any physical layer, which enables it to work with various physical layer architectures. UPnP Device Architecture (UDA) [133] specifies several protocols that allow devices to connect to the network, be discovered, describe their capabilities, invoke their capabilities and send events to interested parties. A UPnP network contains service providers (termed devices) and service clients (termed control points). Devices are typically hardware-based, however they can be either software or hardware, physical or logical. A device is merely a container for services and further embedded devices. A UPnP service defines an operational aspect of the device that is controlled through the service interface and can be invoked over the home network. Services define actions, state variables, and associated events. A common example of a device is a media server (either hardware or software) that offers services for browsing and searching for digital media. UPnP devices do not communicate directly with other devices but only through control points. Control points are controllers that are responsible for facilitating interaction with devices on behalf of the users. In some cases a physical device can serve as both a UPnP device and a UPnP control point in order to engage in peer to peer networking.

The Simple Service Discovery Protocol (SSDP) [133] is used by UPnP to enable devices to advertise their presence and enable clients (control points) to search for devices that support a certain service type. SSDP uses multicast over UDP to send search requests to the network and for devices to announce their presence. Device and service capabilities are described via an XML document with a standard schema – which conforms to the device and service profile defined by the UPnP forum. The device and service description document can be accessed over HTTP within the local HAN. The device description contains information about the device such as manufacturer, version, embedded devices and supported services. The service description contains information about supported actions and their parameters. SOAP [20] is used as a control protocol, allowing clients to invoke actions on the UPnP devices over an HTTP connection. UPnP does not dictate an operating system or programming language, which enables implementers to choose their preferred environment

as well as clients to interact with the device from their programming environment of choice.

In addition to the specification of UPnP communication protocols, the UPnP specification standardises a set of device profiles, which define a standard schema for common devices such as Internet Gateway Device (IGD)[6], media devices[7], and others. A device profile defines a device category and specifies the services expected to be supported by this type of device. Some of the services may be required while others may be optional within the device profile specification. Vendors can also extend the specification with their specific extensions such as additional services, actions and arguments. The purpose of the profile specification is to facilitate device standardisation. The UPnP organisation forms working groups that define these standards for an increasing number of device types however at the moment its application domain coverage is not very wide. Most effort is spent standardizing the entertainment application domain (A/V architecture) and home Internet gateway.

Security is not an integral part of UPnP and is only supported as an add-on. With the huge popularity of UPnP over the years, security has become the primary flaw hindering its proliferation. The rationale behind not implementing security (authentication, access control) in the protocol is to avoid complex administration and to support easy plug and play. However this advantage requires compromising security demands and has resulted over the years in several exploits of UPnP vulnerabilities.

Digital Living Network Alliance (DLNA) [34] is an industry organisation of consumer electronics, computers, and mobile communication manufacturers whose mission is to promote interoperability between products for home networks. While UPnP provides some level of interoperability, it is incomplete in some ways. For example, UPnP does not define formats of media that need to be supported by media players or digital rights management. The scope of DLNA is therefore broader than specifying how devices communicate in the home network. In order to tackle the interoperability problem, DLNA defines a set of standard recommendations to be used by device vendors, including connectivity standards, IP networking, discovery and control, media management, media transport, media formats, and link protection. UPnP is the DLNA recommendation for discovery and control, and for media management. DLNA defines a set of standard device classes such as Digital Media

[6] http://upnp.org/specs/gw/igd2

[7] http://upnp.org/specs/av/av3/

Player (DMP), Digital Media Server (DMS), and others to which device manufacturers can conform to and certify their devices to be in compliance with the DLNA guidelines. At the moment DLNA is focused at scenarios within a single network and does not address multi-home sharing scenarios.

Evaluation:

- **Scope and market uptake**
    - **Application domain** – UPnP is generic and is not specific to an application domain. However first standardised device profiles and popular implementations were focused on multimedia devices and networking devices. Nowadays, the standard includes profiles for additional application domains including printing, and home automation.
    - **Market acceptance** – UPnP is very popular in current home networks. UPnP is promoted by the Digital Living Network Alliance (DLNA) [34] as a recommendation for service protocol for home networking, which contributed to its popularity.
    - **Standardisation body** – UPnP is standardised by the UPnP Forum.
- **Service orientation**
    - **Service discovery –** Enabling service discovery through search request as well as device and service announcement. Search is restricted to service type. Service announcement includes the device/service type, the duration for which the device/service is expected to be available in the HAN, and a URL from which the device description can be retrieved.
    - **Service description –** XML document that conforms to a standardised schema. The device description specifies the services supported by the device and general information about the device, such as friendly name, and version. Service description includes information about the actions supported by the service and the state variables related to those actions.
    - **Service invocation –** Supports communication mechanism for service invocation with SOAP over HTTP.
    - **Service composition –** UPnP specification does not define how services can be composed.
- **Generality**
    - **Physical layer dependency** – UPnP is physical layer agnostic.
    - **Programming language dependency** – UPnP is programming language

agnostic.

- **Non functional aspects**
  - **Security** – UPnP lacks embedded security mechanism, which results in several serious security vulnerabilities (see section 4.3.3.1.7 and section 6.4.2)
  - **Performance** – UPnP requires a small memory footprint on the device, however XML and SOAP require significant amount of memory and processing power for parsing as opposed to less verbose protocols such as Java Remote Method Invocation [123] for example. Another problem of UPnP is the vulnerability of its discovery protocol. UPnP service discovery does not embed mechanisms for automatic shutdown or control, therefore in some conditions the network may be overwhelmed with discovery messages. For example, a control point that sends many frequent search requests can cause the devices in the network to crash because devices must respond to search requests if they support the service type in the request.
- **Extensibility**
  - **Multi-home readiness** – UPnP is limited by design to a single household due to its reliance on local multicast for discovery. Another problem is that devices are assigned private addresses that are not accessible remotely; therefore HTTP access, which is required for description, control, and eventing protocols of UPnP is problematic.

2.3.2. Device Profile for Web Services

Device Profile for Web Services (DPWS) [28], submitted for standardisation to Organisation for the Advancement of Structured Information Standards (OASIS) in 2008, is proposed as a lightweight service oriented architecture targeted at home network devices. The proliferation of web services in enterprise markets and their success in promoting interoperability encouraged their adoption to devices and home networking. Device profiles constrain the web service standards to guarantee that implementations remain interoperable by defining a set of specifications that are mandatory for implementations described below. DPWS shares goals with UPnP, however by using web services it inherits native interoperability with enterprise web services, which can be used for its composition with other services, including external web services. The DPWS architecture specifies how DPWS-enabled devices can be discovered in the local HAN, how messages can be exchanged with a DPWS service, how services are described, how the service is invoked,

and how to subscribe for event notifications from a service. The DPWS terminology defines a hosting service, which corresponds to a device, and hosted services, which are services hosted on a device. All messaging in DPWS is based on SOAP and WS-Addressing. WS-Discovery [8] is used for discovering available devices. WS-Discovery is agnostic to the transport layer and can work over UDP, HTTP or other protocols. Services are described via Web Service Description Language (WSDL) [32] and service description and metadata is retrieved using WS-Transfer [3]. SOAP is used to invoke actions on devices and WS-Eventing [21] is used for managing subscriptions for device events. DPWS defines a recommendation for security, which is optional for device implementation. A baseline is defined such that if a device supports components of security (such as integrity, confidentiality, authentication), it is assumed to conform to the baseline specification.

Evaluation:

- **Scope and market uptake**
    - **Application domain** – DPWS is generic and is not specific to an application domain.
    - **Market acceptance** – Little market acceptance at the time of writing of this thesis, however the recent appearance of commercial automation products such as ConnectedLife.Home[8] equipped with DPWS interface indicates that this might change in the near future.
    - **Standardisation body** - OASIS
- **Service orientation**
    - **Service discovery –** Supported through WS-Discovery, enables to search a service by its type. Devices must support presence announcement when they join and leave the HAN.
    - **Service description** – Service description provides information about the available actions represented as WSDL 1.1.
    - **Service invocation –** Supports communication mechanism for service invocation with SOAP over HTTP.
    - **Service composition –** DPWS services can be composed using standard web service composition techniques.
- **Generality**

[8] ConnectedLife.Home - http://www.bestbuybusiness.com/bbfb/en/US/adirect/bestbuy?cmd=catProductDetail&showAddButton=true&productID=BB10722723

- **Physical layer dependency –** DPWS is physical layer agnostic.
- **Programming language dependency –** DPWS, through its reliance on web services is programming language agnostic.

- **Non functional aspects**
  - **Security –** DPWS has built-in mechanisms for security including authentication, encryption and integrity however they are not mandatory.
  - **Performance –** Small memory footprint
- **Extensibility**
  - **Multi-home readiness –** DPWS is designed for a single HAN. It suffers from similar limitations to UPnP, including the multicast based discovery and the HTTP based invocation.

2.3.3. Jini

Jini [7] is a Java based service-oriented architecture. Jini can be seen as a successor of earlier distributed architectures such as CORBA [92], Microsoft Distributed Component Object Model (DCOM) [25], and Java Remote Method Invocation (RMI) [123]. These technologies share the common goal of facilitating networked distributed applications. Jini proposed a new approach for reducing the burden of administration by making "plug and play" networked services and abstracting the networking and communication aspects from its users. Jini relies on mobile code for facilitating the interaction between clients and services. Service discovery is supported through the use of look up services that can provide clients with catalogues of available services. Jini services register a service proxy with one or more lookup services. A lookup service is a meta Jini service that acts as a broker that connects clients with services and either has a well-known IP address or listens to a well know multicast address. TCP/IP is used for communication between clients, lookup services and services. Clients discover services through searching in lookup services. Once they find a service, they download the service proxy and can invoke methods on it. The service proxy communicates with the service implementation and returns the result to the calling client. The protocol for communication between the service proxy and service implementation is not defined by Jini and is left for the implementation.

Evaluation:

- **Scope and market uptake**

- **Application domain** – Jini is generic and is not targeted for a specific application domain.
- **Market acceptance** – At the time of writing, Java, (and therefore Jini) is not very common yet in the embedded hardware industry.
- **Standardisation body** – There is no general standardisation body for defining Jini service, however some conform to Application Programming Interfaces (API) defined in Java (by Sun/Oracle), such as Java Printing API.

- **Service orientation**
  - **Service discovery –** Service discovery is facilitated through a look up service, which enabled the registration and discovery of services. Service clients can search services by the service interface and additional characteristics that were registered with the service in the lookup service.
  - **Service description** – Services are described via Java interfaces.
  - **Service invocation –** Jini supports communication mechanism for service invocation using Java RMI
  - **Service composition –** Jini does not define a specific mechanism for composition of Jini services. A composite Jini service can be written (with Java programming language) to use other Jini services with some business logic, and then registered with a lookup service as an ordinary Jini service.
- **Generality**
  - **Physical layer dependency –** Jini is physical layer agnostic.
  - **Programming language dependency –** Jini is a Java based protocol.
- **Extensibility**
  - **Multi-home readiness –** Jini discovery depends on broadcast and multicast and does not natively support discovery beyond the scope of a single home.
- **Non functional aspects**
  - **Security –** Java based security mechanisms for authentication, authorisation, confidentiality, and integrity.
  - **Performance –** There are no known inherent performance issues with Jini. In general the use of Java Remote Method Invocation is considered more efficient than SOAP because of the verbose nature of SOAP.

2.3.4. Service Location Protocol

Service Location Protocol (SLP) version 2 [47] is an Internet Engineering Task Force

(IETF) protocol for service discovery and advertisement (RFC 2608). Unlike other protocols, which aim to abstract the transport layer from the service discovery protocol, SLP is restricted to allowing applications to discover the existence of networked resources and provide a service location where the service can be interacted. Services can represent physical or logical devices. SLP enables clients to find services according to their type and associated attributes rather than their host name. SLP defines 3 types of entities: service agents, user agents, and directory agents. Service agents advertise one or more services enabling user agents to discover them. Directory agents cache service information and enable look up. Directory agents are optional, however they can improve performance by reducing the number of searches that user agents initiate to the network. Directory agents and user agents listen to periodic device multicast presence messages. For active discovery, service agents and user agents can send multicast messages and locate directory agents. Once they find a directory agent, they can register a service/look up services respectively. SLP has limited support for security. In fact only authentication is supported while access control, and the security of the communication channel is not defined.

Evaluation:

- **Scope and market uptake**
    - **Application domain –** SLP is generic and is not targeted for a specific application domain.
    - **Market acceptance** – SLP is used in printers and earlier versions of Mac OS X.
    - **Standardisation body** – Internet Engineering Task Force (IETF)
- **Service orientation**
    - **Service discovery –** SLP supports discovery through user agents that can search for services in directory agents, where services register information about them, and through service agents' announcements.
    - **Service description** – SLP does not support service description.
    - **Service invocation –** SLP service invocation support is limited to defining the URL where the service is located.
    - **Service composition –** SLP services support invocation by service location only, therefore their composition requires tight coupling between the consumer and the provider of the service. The inputs, outputs and operations on the service are not expressed explicitly in a service interface, which can be used for service composition.

- **Generality**
  - **Physical layer dependency –** SLP is physical layer agnostic.
  - **Programming language dependency –** SLP is programming language agnostic.
- **Non functional aspects**
  - **Security –** SLP supports only authentication
  - **Performance –** SLP is considered a scalable protocol, and since it does not dictate a specific interface other than the network characteristics of the service and the way information is sent or received from the service is left for the specific service implementation, enables efficient implementation.
- **Extensibility**
  - **Multi-home readiness –** SLP relies on multicast for service discovery, therefore it is not natively extensible beyond the scope of a single network.

2.3.5. ZeroConf

ZeroConf [29] is a technology for zero-configuration networking aimed for HANs. ZeroConf has multiple implementations including Apple Rendezvous (formerly Bonjour), Avahi for Linux and Link Local Multicast Name Resolution (LLMNR) by Microsoft for Windows CE. ZeroConf involves 3 main protocols: address auto-configuration, name-to-address translation, and service discovery. Devices obtain their IP address from a DHCP if such is available or use IPv4 link local addressing if DHCP is not available. The name-to-address protocol enables mapping from host name to host address in the absence of a DNS server. This is done using Link-local multicast DNS (mDNS). The multicast DNS protocol allows client applications to resolve a name or an IP address by sending the request to a well-known multicast address. The host whose name matches the one in the request must respond if it has a match to the request. ZeroConf supports two types of service discovery protocols: Service Location Protocol (SLP) version 2 [47], and Domain Name System Resource Record (DNS RR) [46]. SLP v2 uses an administrative-scope multicast address as opposed to other protocols of ZeroConf, which are based on link local multicast. DNS RR enables a client to lookup a service by the transport protocol, and the domain name, and type of service. A comprehensive list of supported services types can be found in [35, 59]. ZeroConf has another protocol for allocating multicast address for applications. The ZeroConf multicast address allocation protocol (ZMAAP) is required to allow applications obtain and maintain unique multicast addresses. The scope of ZeroConf is limited to service

discovery and does not dictate an application layer protocol for invocation. The last step of ZeroConf is a binding of a host and a port number to a specific service and protocol type (e.g. UDP, TCP), however the interaction with the service remains as a contract between the client and the service.

Evaluation:

- **Scope and market uptake**
    - **Application domain** – ZeroConf is generic and is not targeted for a specific application domain. It is targeted for IP networks and was implemented for various services in the HAN.
    - **Market acceptance** – ZeroConf is very common as a service discovery protocol with several implementations from Apple, Microsoft and Linux.
    - **Standardisation body** – Internet Engineering Task Force (IETF)
- **Service orientation**
    - **Service discovery –** Search is supported through multicast DNS. Devices/services can announce their presence in the network.
    - **Service description** – ZeroConf does not support service description.
    - **Service invocation –** ZeroConf service invocation support is limited to defining the URL where the service is located.
    - **Service composition –** ZeroConf does not support service composition, and since service invocation is limited to service location, it is inherently limited in its potential for service composition.
- **Generality**
    - **Physical layer dependency –** ZeroConf is physical layer agnostic.
    - **Programming language dependency –** ZeroConf is programming language agnostic.
- **Non functional aspects**
    - **Security –** ZeroConf supports only authentication.
    - **Performance –** ZeroConf employs a number of optimisations for the service discovery such as exponential reduction mechanism, and same query suppression.
- **Extensibility**
    - **Multi-home readiness –** ZeroConf has limited support for multi-home networks with its use of DNS. Through modifying a DNS server, services can become available over the Internet (i.e. can be discovered remotely). This

requires both admin rights to the DNS server, and technical know-how for modifying the DNS server allowing services to be discovered by remote ZeroConf clients.

2.3.6. Open Service Gateway initiative

Open Service Gateway initiative (OSGi) [124] defines a component model and service oriented architecture for Java, initially targeted at limited resource devices such as home gateways, but now more generally used in both desktop applications and enterprise application servers. Key features of the framework are that it manages the life cycle of Java-based software components and supports loose coupling of these components through a common service model. OSGi has a dynamic module system, which can load and unload modules dynamically during runtime, which makes it efficient to install, start, stop, update, and uninstall modules on an as-needed basis. OSGi is restricted to a single Java process, therefore does not support distributed computing between multiple devices. In recent years some work has enabled distributed OSGi with OSGi version 4.2[9] and remote access to OSGi services with R-OSGi [107] so enables the use of OSGi beyond the scope of a single system. While the Distributed OSGi intends to extend the OSGi platform for distributed computing, R-OSGi is an OSGi bundle that can run on any OSGi platform and enables remote access to services.

OSGi has wide penetration in the embedded systems marketplace with large deployments worldwide; thus OSGi is a mature technology with many attractive features for HAN equipment vendors, especially software life cycle management.

An OSGi service is an implementation of a Java interface and a bundle is the deployment packaging mechanism that can contain service implementations. During the deployment of a bundle, its services are registered with the service registry, thereby allowing it to be found by clients. Once a service has been found, a Java interface is returned to the client and the client can then interact with it directly. Due to the various facets of OSGi, it can be found in different contexts in the home network including as a device management interface, home interoperability environment [19, 136], and service composition for home devices [106, 126].

[9] http://www.osgi.org/download/r4v42/r4.cmpn.pdf

Evaluation:

- **Scope and market uptake**
  - **Application domain** – OSGi is generic and is not targeted at a specific application domain, however in the HAN it is common in set top box and home gateway devices.
  - **Market acceptance** – OSGi is common for device management for specific devices such as set top box, but it is not common for device-to-device interoperability.
  - **Standardisation body** – OSGi alliance
- **Service orientation**
  - **Service discovery –** Services are registered and can be found in a service registry.
  - **Service description** – Services are described in a service interface.
  - **Service invocation –** OSGi supports service invocation on service objects.
  - **Service composition –** With OSGi services can be composed with programming language code and registered as an OSGi service with the OSGi framework.
- **Generality**
  - **Physical layer dependency –** OSGi is physical layer agnostic.
  - **Programming language dependency –** OSGi is Java based.
- **Non functional aspects**
  - **Security –** As a Java based framework OSGi supports Java security.
  - **Performance –** OSGi has a significant overhead and its minimum requirements are for at least 8MB of RAM and a CPU of 50MHz. According to performance measurements, R-OSGi outperformed both UPnP and Java RMI service invocation by two orders of magnitude when evaluated with the Javaparty/KaRMI performance benchmark as shown in [108]. The benchmark tested the same services implemented as a UPnP service, Java service invoked with RMI, and R-OSGi service.
- **Extensibility**
  - **Multi-home readiness –** OSGi can support multi-home networking through the use of R-OSGi, however it does not have a multi-home service discovery protocol as its main use case is as a centralised system. R-OSGi is relatively

new and at the moment is targeted for distributed application rather than for HAN.

2.3.7. Home Audio-Video interoperability

Home Audio–Video Interoperability (HAVi) [75] first introduced in 1998, aims to allow consumer electronics and networked appliances such as television, speakers, and cameras to communicate with each other through a set of API, services and on wire protocols. HAVi builds on IEEE1394 as a communication medium. As with UPnP, HAVi has support for plug and play, enabling devices to join the network, configure themselves, and find other devices in the local HAN in a peer-to-peer fashion without requiring user intervention. The HAVi architecture defines controlled devices and controllers. A controlled device is a device that provides some functionality, e.g. a DVD, or a set top box. A controller enables control of a controlled device and can be either part of the device or an external entity. Devices are controlled through Device Control Modules (DCM) – a set of compiled Java objects that can be downloaded from the controller. A DCM contains a set of Function Control Module (FCM) such that each FCM corresponds to a single service supported by a device. HAVi devices can be divided into several categories: Full AV Devices (FAV), Intermediate AV Devices (IAV), Base AV Devices (BAV), and Legacy AV Devices. The FAV and IAV are controlling devices while the latter are controlled devices in the HAVi network. FAV has the richest HAVi capabilities enabling it to control other devices. Typical HAVi Full AV devices are set top boxes, and residential gateways. IAV have more limited control capabilities, such devices are typically amplifiers or DVD players. BAV are devices that can be controlled but have no ability to control other devices, for example camcorders, and portable audio players. Legacy devices do not directly support HAVi however they can be connected to a HAVi network through an FAV or IAV acting as a gateway. Discovery in HAVi is facilitated through registries. Applications can query the registry with the set of required attributes. Once a service is found in the registry, applications can then download required device or function control module from the registry.

Evaluation:

- **Scope and market uptake**
  - **Application domain** – HAVi is targeted to multimedia devices in the HAN.
  - **Market acceptance** – HAVi is not widely used
  - **Standardisation body** – HAVi consortium.

- **Service orientation**
  - **Service discovery –** Service discovery is made via a service registry.
  - **Service description** – Services are described with a java interface.
  - **Service invocation –** HAVi supports invocation on Java objects.
  - **Service composition –** HAVi does not support service composition.
- **Generality**
  - **Physical layer dependency –** HAVi depends on IEEE1394, typically wired connections.
  - **Programming language dependency –** HAVi is Java based.
- **Non functional aspects**
  - **Security –** The HAVi service discovery protocol is vulnerable to internal and external attacks, e.g. due to a faulty device. For example a single device update contains a virus and it uploads it to all other devices in the network.
  - **Performance –** HAVi is not designed for use in large networks due to its reliance on IEEE1394, which is limited to 63 devices in the same network, however since it is designed for consumer electronics and multimedia networks, this limit is reasonable.
- **Extensibility**
  - **Multi-home readiness –** HAVi is not designed for multiple HANs.

2.3.8. Summary

The above sections reviewed a number of service-oriented architectures for home area networking. The protocols and standards were compared according to the criteria defined in section 2.3 and the results are summarised in table 1. The following section summarise the comparison with respect to the criteria defined in section 2.3.

2.3.8.1. Scope and market uptake

From market acceptance point of view, UPnP and ZeroConf are the most dominant protocols in the HAN. Many standardisation bodies are involved in HAN service protocol standardisation efforts: UPnP Forum, OASIS, IETF, HAVi consortium, OSGi, and Sun.

2.3.8.2. Service orientation

From a service oriented standpoint the protocols and standards can be divided into two main groups: those supporting full service orientation including service discovery, service description, and service invocation with a communication mechanism for service invocation; and those supporting partial service orientation with service discovery and URL for service location as service invocation mechanism.

A closer examination of discovery strategies reveals two main patterns for finding services: (i) multicast based; (ii) via lookup service; Multicast based service discovery indicates that a search request is sent to a multicast address that devices listen on and they must respond (via unicast) to the initiator of the search in case they support the service type indicated in the search request. The lookup service based discovery implies that services register themselves with a lookup service and client applications use the lookup service to search for services of interest. Another aspect of service discovery is the availability of service announcements when a device or service becomes available/unavailable in the HAN. Announcements are an important complementary mechanism to discovery that enables a reduction on the number of search requests by letting applications learn about the existence of new devices and services through their presence announcements.

With regard to service invocation, the reviewed protocols can be divided between those supporting service location only (SLP, and ZeroConf), and the rest of the protocols (UPnP, DPWS, Jini, HAVi, and OSGi) that describe a communication mechanism to invoke service methods. While the API is not part of the protocol itself for those protocols supporting a communication mechanism for invocation, it is part of the standardisation effort and profiles or device interfaces are defined and used by these protocols. The protocols supporting a communication mechanism for service invocation can be further divided between Java based protocols, SOAP based protocols. Java based protocols are more limited in their ability to integrate with other non-java services, however they benefit of a potentially more optimal on-wire data representation than SOAP XML which is relatively verbose.

Service composition is not the focus of any of the reviewed service protocols. UPnP specification does not define how services can be composed. DPWS do not define how services are composed however, since DPWS services are compatible with web services

they can be composed with web service composition techniques. Jini does not define how services are composed. SLP and ZeroConf are limited in their invocation support therefore they are limited in their ability to be composed. HAVi specification does not define how HAVi services can be composed. OSGi does not define how services are composed.

2.3.8.3. Generality

Several of the reviewed protocols and standards are programming language agnostic (UPnP, DPWS, SLP, and ZeroConf) while the rest depend on Java (Jini, HAVi, OSGi). The dependency on Java is more restricting in that it requires client applications to be written in Java. It also requires the service provider and the service client to agree on a Java interface – which implies a Java service is the product of the device or service standardisation, while in language agnostic protocols the standardisation can be defined as an XML document. This observation applies only to those protocols that support a communication mechanism for service invocation.

All of the reviewed protocols except HAVi, which relies on IEEE1394, are physical layer independent.

2.3.8.4. Non functional aspects

Most of the reviewed protocols support some level of built in security except for UPnP, which has security as an add-on. For others the level of security varies between authentication only (SLP) and full support for authentication, authorisation, integrity, and encryption (Jini, DPWS, and HAVi). The relevance of the levels of security supported by the protocol is for evaluating the impact of sharing on the level of security.

With regard to performance, UPnP and DPWS have small memory footprint on the device. The use of SOAP for remote invocation in these service protocols is considered less efficient than Java RMI because of its verbosity as opposed to a binary representation, as used by Jini, OSGi and HAVi. SLP and ZeroConf are considered scalable because of their abstraction of invocation mechanism. In addition ZeroConf has optimised discovery mechanisms for reducing the traffic needed for discovery.

2.3.8.5. Extensibility

With regard to multi-home readiness it can be seen in the table that ZeroConf is the only protocol with some support for multi-home networking. It should be noted though, that this support is not native and requires manual administration and configuration of a DNS server, which is not appropriate for typical home users. To some extent OSGi can also support multi-home setting with distributed OSGi and R-OSGi, however this is not common and requires administration. The rest of the protocols and standards (UPnP, DPWS, Jini, SLP,

Property/Protocol		UPnP	DPWS	Jini	SLP	ZeroConf	OSGi	HAVi
Scope and Market Uptake	Application Domain	Generic, currently focused in consumer electronics	Generic	Generic	Broad TCP/IP services	Broad TCP/IP services	Generic, common for set top box	Consumer electronics
	Standardization Body	UPnP Forum	OASIS	Sun (Oracle)	IETF	IETF	OSGi Alliance	HAVi consortium
	Market Acceptance	Very common, promoted by DLNA	Not yet common in HAN	Not common in HAN	Used as part of ZeroConf	Very common	Not common for inter-device interoperability	Not widely used
Service Orientation	Service Adverts	Multicast	Multicast	Via lookup service	Multicast	Multicast	Via service registry	Via lookup service
	Service Discovery	Multicast	Multicast	Via lookup service	Multicast	Multicast	Via service registry	Via lookup service
	Service Registry	Not supported	Not supported	Supported	Supported	Supported	Supported	Supported
	Service Description	XML	WSDL	Java API	Not supported	Not supported	Java API	Java API
	Service Invocation	Communication mechanism (SOAP)	Communication mechanism (SOAP)	Communication mechanism (Java RMI)	Service location	Service location	Communication mechanism (Java API)	Communication mechanism (Java RMI)
	Service Composition	Not supported	Compatible with web service composition	Not supported	Not supported	Not supported	Not Supported	Not supported
Generality	Programming Language	Any	Any	Java	Any	Any	Java	Java
	Physical Layer	Any	Any	Any	Any	Any	Any	IEEE1394
Non Functional Aspects	Security	Available as an add-on	Authentication, integrity, encryption	Authentication, authorization, integrity, encryption	Authentication	Authentication, integrity	Java based security	Authentication, authorization, integrity
	Performance	Small memory footprint	Small memory footprint	RMI is considered to have better performance than SOAP	Considered scalable because of the abstraction of invocation protocol	Optimised discovery protocol with shutdown mechanisms. Considered scalable.	No specific issues	Designed for small networks
Extensibility	Multi-home Readiness	None	None	None	None	Via DNS-SD	R-OSGi/ Distributed OSGi	None

Table 1 Service Protocols and Standards Comparison

and HAVi), do not support natively service discovery or access to services beyond the scope of a single network.

2.4. Service oriented HAN

As demonstrated in the previous section, service oriented computing has been adopted by multiple standards and service protocols for the HAN. Bottaro et al. claim that SOC is a promising paradigm for addressing the challenges of pervasive computing within the HAN for enabling seamless integration between heterogeneous devices and services [18]. A "home service", hosted by a home device, in this terminology can define an action or operation that can be invoked over the network by clients in the HAN. For example, a clock service may have a get time operation, which returns the current time, a media server may have a search action, enabling client to search for media. Devices may have additional embedded devices, for example a mobile phone device may have a digital camera embedded in it.

While multiple service-oriented protocols have been suggested and deployed in the HAN, the incompatibility between these service protocols, as demonstrated in the previous section, resulted in formation of islands of interoperability [88]. Redondo et al. [106] argue that the availability of different devices forms a pool of resources that is committed to various activities in the HAN and can be leveraged by combining and reusing existing resources through service composition. Devices can shift from providing content only over the network, to providing services, such that innovative composite services can reuse these services for building custom applications for the HAN. However, while service composition is a key premise for SOC for enabling creation of reusable complex services from multiple simple ones, it is limited by the lack of interoperability between multiple service protocols. Applications can still compose services in the traditional style, by writing code in a programming language that interacts with multiple services. For example a UPnP control point can interact with multiple UPnP services, and can even include code to interact with a service that in DPWS. However this overlooks the SOC advantages of loose coupling and technology neutrality, and does not result in reusable services. Support for nested composition of atomic services means that service providers can expand their service offerings by re-using their own atomic resources and services as parts of composite or

specialised services that they then offer directly to consumers. Finally the ability to compose local services with other service providers facilitates the building and maintenance of strategic partnerships with other service providers in order to maximise the value (and hence demand) for the atomic services.

2.5. Summary

This chapter introduced the main HAN challenges, applications, and technologies. The main focus of this chapter was a review and assessment of the various service-oriented protocols and standards that leverage the home physical connectivity for enabling services to be discovered and invoked locally by other devices, services, or applications. The outcome of this assessment is a comparison of the protocols and standards based on the criteria defined in section 2.3. The purpose of the review was to assess their support for service orientation in terms of service discovery, description, invocation, and composition, and in addition to assess their suitability for extension beyond the scope of a single HAN.

While significant efforts were made in enabling device-to-device interoperability resulting in the diversity of service protocols reviewed above, the plethora of service protocols and standards that were suggested for the HAN presents new challenges from a service-oriented point of view. HAN service protocols and standards enable, to a greater or lesser extent, interconnectivity between heterogeneous devices and services. While they generally provide useful abstractions for interaction with a particular device, these protocols and standards are not interoperable with each other. This leaves vendors and consumers locked into spot solutions and ultimately not delivering on the promise of pervasive HAN device and service integration. As indicated in this chapter, service composition is not an inherent part of the reviewed service protocols, except for DPWS services, which are compatible with web services and can be composed with web service composition techniques. Finally, only ZeroConf and distributed OSGi and R-OSGi support access of HAN resources from outside the HAN.

The next chapter presents a review and analysis of the state of the art in the area of service oriented architectures for home area networks with emphasis on intra-HAN service interoperability and service composition, and on inter-HAN service interoperability.

Chapter 3
STATE OF THE ART

The previous chapter discussed the service oriented HAN, presenting the various service protocols and standards for home networking, and assessing their service orientation. The previous chapter also presented the fundamental concepts of service orientation and emphasised the role of service composition and its potential value for HANs both for consumers as well as service providers. Being able to compose services available in the HAN can realise the full potential of the HAN through enabling the innovative creation of new complex services from those already existing in the home network. Moreover, service oriented computing can also benefit from the availability of services from multiple HANs. With increasing broadband availability, consumers can leverage remote resources shared with them, thereby extending the scope of the service-oriented HAN. However this extension of scope must be made carefully with regard to security, privacy and performance.

This chapter presents the state of the art in these two aspects of interoperability related to SOC in the HAN: Intra-HAN service interoperability, which allows services from different service protocols to discover and interact with each and to be composed in the scope of a single network; and inter-HAN service interoperability, which allows services in different HANs to discover and interact with each other.

The purpose of this chapter is to identify the shortcomings of existing approaches for both intra-HAN with respect to service composition (section 3.2) and inter-HAN service interoperability (section 3.3) and derive the requirements for an integrated service oriented system that supports both intra-HAN and inter-HAN service interoperability (section 3.4).

3.1. HAN service interoperability

The previous chapter showed the diversity of devices, services, and the service protocols and standards used for their interconnection. These service protocols and standards are incompatible with each other and therefore are not interoperable in their syntax, and the semantics of the interfaces they expose. It is unlikely in the near term to expect that a single technology will dominate the home network. There are several reasons for the coexistence of multiple service technologies at the home network. Due to the wide spectrum of devices from low-cost, resource-constrained appliances to powerful desktops, a single service protocol may not be suitable for all devices. Additionally, the traditional segmentation of industries has led to the development of multiple service protocols and standards, each with its own application domain. Moreover, this segmentation has shown little sign of convergence, which could promote interoperability in recent year. While on more powerful devices multiple protocols could be supported, on low-cost devices, it is unrealistic to expect to find more than one service protocol implemented on a single low cost device.

HAN service interoperability is defined as the ability of devices and services to discover, configure and control other devices and services in the network [83]. There are various levels of interoperability from the lower network level interoperability, which specifies how messages can be exchanged between systems, through syntactic interoperability, which defines the structure of messages that can be sent between heterogeneous systems in the home networks. Finally there is semantic interoperability, which is required to understand the format and content of the actual data that is exchanged between systems.

Since the HAN service protocols and standards differ in their network, syntax, and semantic layers, they are not designed for discovering, and interacting with services supporting other service protocols. In addition, these protocols are designed to run within the scope of a single HAN and are not able to interoperate with other services in remote HANs (i.e. discover, interact). Interoperability between the different service protocols is also a building block in enabling HAN service composition. Service interoperability, both inter-HAN and intra-HAN has various advantages to several players including consumers, service providers and service developers. Service interoperability can enable service consumers to enjoy a

richer choice and experience from being able to connect multiple services, in multiple HANs, augment the functionality of a single device or service with available extensions. Service providers can profit from the potential of their devices and services being used in ways that were not necessarily intended through combination and collaboration with other services. Interoperability enables rapid service development and makes it easier to introduce new applications from existing devices and services.

The lack of service interoperability indicates that devices supporting one protocol cannot discover and interact with devices supporting other protocols and with devices supporting the same service protocol in other networks. This poses a challenge to the home service oriented environment and impedes the realisation of its potential, which can be achieved through connecting and composing services from multiple protocols and networks.

The following sections therefore present the state of the art in intra-HAN and inter-HAN service interoperability. The importance of intra-HAN service interoperability for this thesis is in that HAN service composition requires broader intra-HAN service interoperability.

3.2. Intra-HAN service interoperability

The objective of intra-HAN service interoperability is to enable devices supporting different service protocols and standards to discover and interact with each other. The approaches proposed in the literature can be classified into two main groups: bridge-based service interoperability, and middleware-based interoperability. The bridge-based interoperability attempt to connect two specific service protocols and enable their unidirectional or bidirectional discovery and interaction. The middleware-based approach to intra-HAN service interoperability takes a platform centric standpoint in positioning a middleware as a hub between multiple spoke service protocols.

3.2.1. Bridge-based interoperability

A number of bridge-based solutions were suggested for enabling the unidirectional or bidirectional bridge between service protocols, enabling clients of one protocol to access

services of the bridged service protocol. In [91] Newmarch presents a lightweight approach for a unidirectional bridge connecting Jini clients with UPnP services. Jini's abstraction of the way a service proxy communicates with a service implementation is leveraged to create Jini service proxies for UPnP devices such that SOAP is used to facilitate the interaction. This requires the UPnP device side to register the proxy with the lookup such that the proxy should implement all the methods defined in the service description of the UPnP service. This requires an extension of the UPnP device stack to make the proxy classes available to be registered with the lookup service.

Allard et al. present an approach for a bidirectional bridge between Jini and UPnP service discovery architectures [4]. This approach is external to the devices and enables bidirectional communication. While it does not require clients to be rewritten, it does require new code to be added for each additional service introduced for bidirectional mapping between UPnP and Jini to be deployed as part of the system. The design suggests creating a virtual proxy for each UPnP service that would register with a Jini look up service and bridge Jini client calls to the UPnP device, and similarly for a Jini service, a UPnP virtual proxy would advertise the service using SSDP and bridge SOAP calls to the Jini implementation. The architecture is meant to minimise the amount of code needed and to facilitate rapid development of bridged services. While the bridge presents a solution to the syntactic interoperability problem, the semantic interoperability remains a problem, e.g. a Jini printer service may differ from a UPnP printer in the functionality offered, therefore mapping between them is imperfect. Another problem refers to the argument types in Jini services that tend not to be restricted to primitive types and Strings, which is more problematic to map to UPnP. Another problem for bridging service discovery is that while UPnP search is limited to the service type, in Jini lookup, several attributes can be used for locating a required service (in addition to the service interface), including manufacturer and version. These attributes if available in UPnP are part of the service description, which is not available before the service is located.

Guttman et al. [48] present another bridge between Jini and Service Location Protocol (SLP) service discovery. Their work is targeted for devices that support SLP discovery however they do not host a Java Virtual Machine (JVM) and do not have a Jini interface. In order to allow Jini clients to use SLP devices, the Jini-SLP bridge acts as an SLP User Agent (SA) and discovers Jini-enabled SLP services in the network. While these devices are expected

not to have a JVM, they are required to have a Jar file that contains a driver factory class that can be used to register the service with a Jini lookup service. When a client wants to invoke the Jini service, it downloads the driver factory object from the lookup service and the driver factory can then interact with the SLP service agent. The advantage of this approach over direct interaction with the SLP service agent is that it abstracts the networking code from the client and enables thinner, simpler, and more flexible clients.

3.2.2. Middleware-based interoperability

A more generic approach for establishing service interoperability between home service protocols is the platform centric approach where interoperability is achieved through central middleware software. Such a middleware can abstract service discovery and service interaction differences between different protocols and technologies and allows uniform access to services for clients.

Several service oriented middleware systems for home networks were suggested in literature. These middleware systems can be classified into 3 groups:

1) OSGi middleware - OSGi provides useful flexibility in its module system for service middleware systems. The service technology can be mapped to an OSGi service interface, then loaded into the platform runtime, and then discovered and invoked by other bundles [136].
2) Web services middleware – SOAP or RESTful web services can provide syntactic interoperability between services in addition to the network interoperability and basic connectivity provided by lower layers (e.g. IP, UDP, wireless, Ethernet) [102]. Web services are used as the service representation, such that the web service serves as the proxy between various service protocols. The benefit of using web services for home networking interoperability is their reuse of open standards, proven scalability in enterprise applications given their loose coupling, and their independence of platforms, programming languages, and operating systems [2].
3) Proprietary canonical service representation – proprietary frameworks that define or reuse a service format to which service protocols are transformed, in addition to discovery mechanisms and invocation protocol. Universal Service Description Language (USDL) [88] for example is an XML language used by uMiddle to describe semantics of devices. The main drawback of this approach is that it is

incompatible with any existing software in the home network. It also requires application developers to be trained with new API rather than well known industry standards.

- Middleware systems using OSGi service representation:
    - Home SOA [19] is an attempt to tackle the home interoperability problem via open pluggable component architecture. Based on an OSGi platform, Home SOA offers a service oriented device control framework. The main concept in this architecture is the use of several types of drivers in conjunction with an OSGi platform to provide a rich service oriented interoperable environment. Base drivers are used for handling a specific technology and hiding its protocol details through device reification on the platform. Refined drivers react to the discovery of a base driver and perform mapping between multiple technologies. Service composition is supported through chaining syntactic functions via reactive adapters.
- Middleware systems using web service representation:
    - FedNet [70] is an intermediary based solution for transparent integration of applications with diverse smart objects by wrapping the interaction with the smart object and placing the FedNet intermediary between the applications and the smart devices. Smart devices are described via proprietary description documents. The interaction between the application and the smart device representation is made through a RESTful interface.
    - Perumal et al. [102] tackle the problem of interoperability and heterogeneity in smart homes, and discuss the different layers of interoperability and for each layer suggest their preferred solution. For connectivity interoperability they suggest an Ethernet cloud; for network interoperability – TCP; and for syntactic interoperability - XML/SOAP web services.
- Middleware systems using proprietary service representation:
    - uMiddle [88] is a universal and extensible middleware for interoperability in pervasive environments. uMiddle uses a proprietary XML universal description language (USDL) as a canonical service description thereby enabling developers to write applications that interact with devices with a technology-neutral interface.

3.2.3. Home service composition

As mentioned earlier in this chapter an important aspect of a service-oriented architecture is service composition – the ability to construct complex services from atomic simple ones available in the network. This section reviews several middleware-based solution for intra-HAN interoperability that focus on service composition.

Several service composition platforms for home services are focused on a specific technology while others aim at a more holistic mediation solution for multiple service technologies. Bobek et al. [16] suggest a framework for enabling workflows in UPnP networks. The architecture supports two different modes: workflow as a UPnP device, and workflow as a UPnP control point. By adding a UPnP interface to a workflow engine, it can theoretically interact in a seamless manner with other UPnP devices in the network. Workflows are modelled as embedded devices in the workflow engine device enabling control points to discover and invoke them via UPnP protocols. Taking the role of a control point allows the workflow engine to interact directly with UPnP devices. The combination of both capabilities enables workflows modelled as UPnP devices to be further composable for creating composite workflows, which are again modelled as UPnP devices for further composition. Wrapping the workflow management system as a UPnP device enables uniform workflow management via a UPnP service. This approach has the advantage of enabling control point applications in the home network to discover and invoke composite services. For service composition, the disadvantage of [16] is that it only supports UPnP services.

In [126] Timm et al. present MORE, a middleware system, which supports dynamic service orchestration on DPWS-compliant embedded devices. Their main application domain is for hierarchical sensor networks. The middleware runs DPWS on an OSGi platform. WS-BPEL is too heavy for a sensor network deployment – for example, the open source Apache ODE BPEL engine requires Java runtime as well as a Tomcat servlet container, which requires 10-15 megabytes of RAM. The static nature of BPEL is not flexible enough for the application domain – this is because the sensor network is dynamic, sensors are added, network conditions change in such way that the workflow has to be modified dynamically throughout its lifetime. Instead MORE suggests a simplified workflow technique called service chaining, which is merely a description of the order of execution of a set of web

services such that the WSDL for the service chaining is generated on the fly. Through the web service interface for the service chaining, the system achieves seamless integration with BPEL, which may be applicable at higher layers of the sensor network. While the simplistic approach of dynamically generated service chains may suffice for sensor networks, however for home service composition it lacks concurrency (in service chaining services are executed sequentially), and structural activities such as conditional branching, and sophisticated event handling.

An OSGi based service composition for smart services with BPEL is presented in [106]. In this architecture a composite service is expressed as a BPEL process that orchestrates a set of OSGi services. The composite service is deployed in the OSGi platform as a virtual bundle and then can be accessed by other services as an ordinary OSGi service. When a composite service is located and invoked, the implementation bundle calls the BPEL engine, which interprets the process specification and executes the process and can call back OSGi services. Smart home services are discovered and registered with the OSGi platform as services via technology related bundles (e.g. a UPnP bundle) and thereby can be invoked from composite services running on the OSGi platform. An important aspect of the proposed approach is an improvement of the OSGi service registry to support semantic description of services and thereby enable the automatic semantic OSGi service composition, which is based on an understanding of what the service does rather than what is the syntax of the service. This allows replacement or load balancing of equivalent services. The main drawback of this approach is the lock-in to Java indicated by the use of OSGi. In addition, OSGi is typically centralised and while it allows clients to discover and invoke services, these clients must run in the same process as the OSGi platform. In this sense, seamless integration is achieved only through embedding of client applications into the OSGi process space, which is not likely for the dynamic environment of the home network. The main advantage of this approach is that the modularity of OSGi enables support for multiple service protocols through the OSGi service interface by registering a service protocol bundle with the OSGi framework.

The authors of [17] suggest an extension of BPEL for dynamic device and service discovery. The purpose of this framework is to adjust BPEL to the more dynamic environment of mobile devices. The discovery extension supports DPWS enabled devices. The benefit of this approach is in its flexibility such that the services invoked during the

execution of the process do not need to be known at design time. Another contribution is a code generator that generates Java code from BPEL service description. The advantage of this approach is that the generated byte code can be executed on a Java virtual machine without requiring a heavy BPEL engine, therefore it may be appropriate for running on devices.

Sliver [52] is an extension for BPEL targeted for mobile devices. While the disk space and memory size required for running BPEL engine are reasonable for modern computers, they are inappropriate for most mobile devices. Another problem is that most mobile devices are not equipped with full Java runtime but with a more limited version (JavaME[10]). Mobile devices that do not support 3G typically lack support for TCP/IP sockets and UDP/IP datagrams, while typically BPEL relies on HTTP for communication with hosts. Through the abstraction of communication with partner links, use of dedicated lightweight parsers and small set of Java API available on mobile device Java environment the goals described above are met.

In [18] Bottaro et al. suggest an OSGi service-oriented middleware supporting ontology-based home service composition, specifically in the audio/video domain. By using OSGi, the system can support multiple different service protocols, for dynamic service composition based on semantic technology. Service composition is accomplished by matching required services with provided services. They define device and capability ontology, which are used to assist with service composition to enable determining which device can perform which semantic task.

3.2.4. Analysis

The sections above presented a number of bridge-based and middleware-based approaches for enabling intra-HAN service interoperability. Bridge-based solutions provide a direct mapping between two service protocols, enabling services of one (or both) protocols to be discovered by the other service protocol's clients. Middleware approaches take a platform centric approach and suggest mapping each service protocol to a canonical service protocol

[10] http://www.oracle.com/technetwork/java/javame/overview/index.html

and service format. For home service composition, BPEL is commonly used for static service composition, where dynamic service composition is supported through either extension of BPEL or extension of OSGi. As was shown no service-oriented middleware for intra-HAN service interoperability supports inter-HAN service interoperability. The following sections analyse the bridge-based, middleware-based and service composition existing solutions for intra-HAN service interoperability.

3.2.4.1. Analysis of bridge-based approaches

The main problem with bridge-based solutions to intra-HAN service interoperability is that they are not extensible to additional protocols. Supporting another protocol requires a bridge between the new protocol and all existing protocols. With regard to service composition, bridge-based solutions to intra-HAN service interoperability do not support or enable service composition. They focus on making more services available to application clients – by enabling one protocol's client to interact with another protocol's services - however they do not address the problem of being able to compose services, either from a similar or different service protocol in a reusable manner.

3.2.4.2. Analysis of middleware-based approaches

Middleware systems aim to hide the heterogeneity of networks behind a middleware by mapping between service protocols and some other service protocol selected as a canonical form. The main advantage of the middleware approach over the bridge-based approach is its extensibility for supporting new protocols and service formats – where all service protocols are mapped to a single canonical protocol and service format. Therefore, mapping a new protocol requires only mapping to the canonical for, rather than mapping to all of the other protocols as in the bridge approach. A major drawback of the middleware approach is that due to mapping to a canonical protocol, client applications supporting a specific protocol cannot benefit from the service interoperability unless they are modified to use the canonical protocol. With bridge-based approaches, e.g. UPnP to Jini bridge, UPnP control points can discover Jini services, and Jini client applications can seamlessly discover UPnP devices and services through the bridge. Similarly to protocol bridges, a problem of middleware approaches is that while they solve the syntactic interoperability between protocols, the semantic interoperability problem remains. If a printer service in UPnP has different operations than a printer service in Jini, their representations in the middleware will differ. Through the abstraction of protocol syntax, a middleware approach enables applications to

be written in a protocol agnostic manner, however the mismatches and data format differences between the service interfaces of semantically equivalent services limit this ability.

Several service representations were suggested above: OSGi service format, web services, and proprietary service format. The advantage of OSGi services and web services over proprietary service formats is that a proprietary middleware supporting a non standard service format is required to re-implement a complete service oriented environment for this service format, i.e. service discovery, service selection, service invocation, and service composition, while all of the above are already supported for web services and OSGi services through standard and commonly known protocols. One strong disadvantage of OSGi based solutions is that it is centralised and requires client applications to run as bundles on the OSGi platform. Another drawback of OSGi is that it is Java specific and requires Java-based interaction, on the other hand web service based middleware enables programming language agnostic interaction. A common drawback to all middleware solutions is that they are incompatible with existing client applications supporting the service protocol and require new applications to be developed to interact with the middleware-based services.

3.2.4.3. Analysis of service composition

Table 2 summarises the approaches presented above for home service composition. As can be seen home service composition is supported mainly with BPEL or OSGi service composition. BPEL is considered a common solution for service composition for enterprise applications. The advantage of applying the same solution to home service composition is in the simplicity that it offers and the readiness of development tools. There are two main drawbacks for using BPEL for home service composition. BPEL is not lightweight and requires significant amounts of RAM. While it is still reasonable for deployment on more powerful home computers, it may not be appropriate for smaller devices such as mobile phones. A potential solution to this problem is proposed in [17] with the mapping of a BPEL composite service to Java such that the executable composite service is compiled into Java. Another drawback of BPEL is its lack of support for dynamic composition. In a BPEL composite service, the service bindings are hard wired at composition design time. In dynamic environments this may be insufficient, as services are added and removed from the

network and better options may exist for the selection of a constituent service within a composite service.

Dynamic service composition allows service selection to be done during runtime rather than design time as in static service composition. While it makes dynamic service composition more flexible, it comes at the cost of complexity. Mechanisms are needed to extend service definitions with semantic information, and to support just-in-time selection. Additionally, semantic information needs to be available for services to provide the service composition contextual information.

Finally, it is interesting to note that none of these systems discussed above supports inter-HAN service interoperability.

3.3. Inter-HAN service interoperability

This thesis argues that the service oriented HAN could benefit from the availability of remote services. By extending the boundaries of the HAN and making more services from remote HANs available, home users could realise even more of the potential of their own devices. More services available in the local HAN (through sharing from remote HANs) can lead to more potential for service compositions that could leverage these devices and services and provide more added value complex services to the home user. In addition, with the emergence of social networks such as Facebook, YouTube[11], Flickr and others, along

	Middleware type	Service composition technique	Service composition level	Adaptability	Supported service protocol	Inter-network interoperability
Bobek et al. [16]	Proprietary	Workflow	Syntactic	Static	UPnP	Not supported
Timm et al. [126]	OSGi	Service chaining	Syntactic	Dynamic	DPWS	Not supported
Redondo et al. [106]	OSGi	BPEL	Syntactic/ semantic	Dynamic	Generic	Not supported
Bohn et al. [17]	BPEL extension	BPEL/Java	Syntactic	Static	DPWS	Not supported
Hackmann et al. [52]	BPEL extension	BPEL	Syntactic	Static	SOAP	Not supported
Botaro et al. [18]	OSGi	OSGi	Semantic	Dynamic	Generic	Not supported

Table 2 Home Service Composition Approaches

[11] www.youtube.com

with the growing popularity of high speed broadband at households, and the growing, yet limited standardisation of home devices interoperability with DLNA, users are ready to share their HAN resources and content in a controlled and managed manner with their friends and family as shown in an ABI market research survey [1]. The focus of this thesis is on integrated inter-HAN and intra-HAN service interoperability and composition. Inter-HAN service interoperability refers to the ability to share services between multiple HANs.

Remote access to HAN resources is a closely related scenario to sharing, however it has a number of subtle but important differences. In the remote-access scenario, the sharing is performed with the identity of a person. In HAN-to-HAN scenario, the sharing is performed with the identity of the network. Hence if a service is shared in the latter scenario, it is shared (and will be discoverable) in the target network, while in the prior scenario, a shared service will not be discoverable in the target network but only by the user identity. The differences in the scenarios imply differences in the set of problems that need to be handled by potential solutions, therefore given the scope of this thesis is restricted to home-to-home scenario, this section focuses on architectures that support inter-HAN service interoperability.

In order to be able to identify shortcomings and advantages of the various approaches and compare between them, a common comparison criteria has to be defined first. The following parameters are defined as the assessment criteria for inter-HAN service interoperability systems:

- **Seamless integration** – A major challenge for inter-HAN service interoperability is to enable the seamless integration of remote resources into local HANs without requiring the extension of the device, the service or network protocols, or of client applications. Another aspect to this is the ability to make local services available to multiple remote HANs simultaneously.
- **Private networks** – The introduction of Network Address Translation (NAT) and private networks solves the problem of the explosion in IP addresses, however it poses a serious challenge for enabling sharing of resources and content between remote HANs. With NAT, the private IP addresses assigned to devices in the local HAN are meaningless outside this scope. The same IP address can be used in two HANs. Moreover, the private IP address of a device is not addressable from the outside world.

There are two different challenges with private IP addresses in the context of sharing services and content outside private networks:

  - Service discovery – How to announce a service with a private IP address in remote HANs?
  - Communication – How to enable point-to-point communication between private IP addresses in remote HANs?

- **Security** – Interconnecting remote HANs for service interoperability presents several security related challenges:
  - Authentication – Needs to guarantee that communication is made only between authenticated parties.
  - Access control – The main challenge is to enable maximal flexibility for the home users to control what they wish to share, with whom, and when. More fine-grained sharing specification, leads to higher level of flexibility.
  - Encryption – Prevents eavesdropping and revealing of private data while being transferred between remote HANs.
- **Extensibility** – With the coexistence of many service protocols in the HAN supporting multiple service protocols for service sharing.
- **Manageability** – Each system is deployed in the HAN and interacts with non-technical home users. It is important to assess what is required from the home user in order to be able to configure the system, e.g. add friends to share services with, control which resources are shared.
- **Intra-HAN service interoperability** – The extent of support for intra-HAN service interoperability, i.e. enable services from multiple service protocols to interact.
- **Support for service composition** – The extent of support offered by the architecture for service composition, in the same HAN or across multiple HANs.

In the following sections the various approaches and implementations are assessed with regard to the above criteria.

3.3.1. IP addressing in a HAN

The most basic approach for enabling remote access to a service or content within the home network is based on configuration of the home gateway [13]. Typically home networks are configured as private networks behind a NAT such that the home network has a single

external IP address and devices in the home network are assigned private addresses. To enable remote traffic to a specific device the NAT router needs to be configured to forward such traffic to the specific private address IP/port. Given the correct NAT configuration, devices within the home network could be accessed from outside the home. The same configuration needs to be applied for each service or device that the user wishes to share. The configuration is made on IP level and means that incoming traffic to the external IP with a given port will be internally redirected to the home device or service. For example, to enable remote access to a media server that is locally advertised with the address: 192.168.1.3:56000, the home gateway needs to be configured to map incoming traffic to the external IP address of the HAN: 89.100.49.96:56000 to the internal address of the device. It only means that a device can be seen as having an external address, and therefore a remote host can interact with it. It does not mean that a remote host can see device announcements made to the local HAN. Therefore it does not support seamless integration and remote services cannot be discovered. A remote user needs to know the IP address of the home network sharing a device and the service details in order to communicate with it. Additional security settings are required to the firewall to make sure access can only be made from a specific source IP addresses. The main drawback of this approach is the requirement for complex manual configuration to home gateway and firewall system, which is not appropriate for most non-technical users. Misconfiguration or lack of security configuration can put the network at risk of unrestricted access to the local HAN. Finally it requires remote users to be aware of the IP address of the home network, which again may not be reasonable to ask of home users.

3.3.2. Web-based content sharing

It is envisaged that for inter-HAN interoperability sharing content is one of the main drivers, e.g. for sharing multimedia content with friends, it is relevant to refer to it in order to better understand the motivation behind sharing of resources, its evolution, and its limitations.

Early approaches for web based sharing were based on emailing media from one user to another [11]. These approaches however had a few problems. Due to the nature of email, size was a problem. High definition movies, high-resolution pictures require large size files, which are not appropriate for email. Another problem with email is that once the media is shared, it cannot be un-shared as it is always available to the recipient of the email. The next

generation of web based sharing of content came with the introduction of 3rd party services such as YouTube, Flickr, and Facebook. These services enable users to share content they generate with other users. As part of the social networking proliferation, these web-based services have become widely used. Content is uploaded to 3rd party hosting sites and is managed beyond the scope and out of control of the home network. Most of these services also provide the users with some mechanism of access control allowing them to control what they share with whom. While these services provide very convenient access for users they have an inherent drawback in the way users give up full control over administration of their content to a 3rd party. This could also lead to invasion to user's privacy by data mining of the content the user uploads. In [113] Rosenblum discuss the implicit loss of control that takes place when users upload pictures and movies to such a 3rd party service, which can have severe consequences, such as unauthorised use by a 3rd party, or having the content exposed to broader than expected audience. Another drawback of web based sharing is the need to conform to some format required by the hosting services, e.g. flash or mp4. Another aspect is that due to the hosting service being public (and typically free) it puts some limitations on the content's quality, length, and space, therefore high quality media can only be stored if it is short enough according to the provider's guidelines. Another aspect of web based sharing is that at the moment the integration between TV and Internet content is limited. In order to watch shared content stored on the web on a TV, users need to connect the TV to the computer, however with the migration of traditional services over IP (IPTV) [66] this may be changing.

TiVo [127] is a popular Digital Video Recorder enabling consumers to record television programs onto an internal storage for watching it at their preferred time. In addition, TiVo enables users to share their movies and photos with friends by creating a personalised channel. Movies are uploaded to a server and friends with whom the channel is shared can access them through a TiVo device. Friends can access all content placed on the channel as soon as they are given the channel key. They can either watch the content or download it to their TiVo device similarly to a regular TV channel. The main purpose of this channels are for photos and short video clips.

Spotify [73] is a peer-to-peer streaming service that enables users to search and stream music on demand. Spotify uses a proprietary client and network protocol. Spotify has two

modes – stream from dedicated servers and peer-to-peer streaming. The Spotify client has local access to music files and enables other peers to discover them.

3.3.2.1. Analysis

Web based sharing is easy-to-use, and the skills required are easily accessible to home users, however it has a number of significant limitations and drawbacks.

- Size limitations – When uploading files to a 3rd party hosting service there are typically restrictions on the maximum file size, preventing sharing of large files, e.g. high definition videos.
- Loss of control – When files are uploaded to a 3rd party hosting site, users have no longer full control over the content.
- Privacy – When users upload files to a 3rd party hosting service they surrender their administration rights and in some cases their copyright over the shared files.
- Limited to file sharing – The approach is only applicable to file sharing, rather than resources or services in general therefore insufficient to address the full range of inter-HAN service interoperability.

The main advantage of this approach is its simplicity and familiarity of home users with it, however as mentioned earlier, web based sharing enables content sharing, it does not support services and seamless integration as only content sharing available through the web browser. Authentication and authorisation depend on the level of security and privacy supported by the service provider. No manual configuration is required and web-based sharing does not support intra-service or service composition. Private networks do not present a challenge to web based sharing since sharing not made directly from the HAN. Extensibility is not applicable to web based sharing because sharing is supported only for content and only by uploading the content to a hosting web site, therefore it does not support a specific service protocol, but the content that is hosted by the devices that use the service protocol.

3.3.3. Peer-to-peer sharing

Peer-to-peer (P2P) networking is a distributed computing paradigm in which there is no

hierarchy between the peers. Unlike conventional client-server architectures where client and server are well defined and have different functional roles, in P2P peers can act as both clients and servers at the same time. Peers expose to other peers' shared services, resources or content, which can then be accessed without the orchestration of a server. Communication between the nodes is based on hop-by-hop messaging from each peer to its neighbour until the message is delivered to the destination. In addition to file sharing, P2P has been very popular infrastructure for Internet telephony applications like Skype[12], and IPTV broadcasting applications like Joost[13].

JXTA [68] is an attempt to standardise P2P architectures and API. JXTA creates a virtual network on top of the physical network hiding the complexities of the physical network from interacting peers. JXTA enables peers to exchange messages with other peers regardless of their network location including firewalls, NAT and non-IP devices. In [97] Park et al. present a JXTA based P2P collaboration platform for connecting multiple home networks. The platform supports a number of use case scenarios: remote control of devices and services, P2P content sharing, home-to-home multimedia content sharing, P2P multicasting. The architecture connects personal computers and diverse devices including legacy devices, non-IP devices and various service protocols to a P2P network through P2P middleware. For each of these service protocols a proxy is defined that acts as a gateway between the service protocols and the P2P protocols. Special peers called relay peers are responsible for relaying messages to peers with NAT thereby supporting private networks. Only authenticated devices can join the P2P network, however no access control is defined which means all services are available to all authenticated users. The communication between the nodes is made over P2P pipes that define virtual communication channel that enables remote devices to exchange messages. The architecture does not support seamless services integration in the sense that service protocol clients for services running in the home network are not able to discover remote services, as they are not advertised locally. Sharing is not made with a specific user but with the P2P network, this enables applications to search for services in the P2P "cloud" and use them, however it requires them to use the P2P middleware. The system does not require manual configuration, however it does require installing a P2P proxy for each "legacy" device in the network to relay the discovery and

[12] http://www.skype.com/

[13] http://www.joost.com/

interaction with the P2P middleware. The architecture is extensible and can support multiple service protocols, however it does not address specifically intra-HAN service interoperability and service composition.

Loeser et al. [77] suggest architecture based on integrating JXTA with OSGi for creating Virtual Home Environment (VHE) enabling users to remotely control their home devices. The concept behind the virtual environment was to provide users with unified access to all of their resources regardless of their location. The architecture supports two modes: in-home and inter-home. The in-home mode supports intra-HAN service interoperability by making services from multiple protocols available to each other over the P2P network following the middleware approach described in 3.2.2, however it does not support service composition. The inter-home mode supports connecting multiple home networks over the P2P network such that the communication with the local devices is facilitated through a centralised OSGi server running the local HAN, which then communicates with other peer OSGi servers running in remote HANs. Several aspects of security are supported: authentication, encryption, and message fingerprints for data integrity, however access control is not supported. The architecture does not support seamless integration with existing service protocol client applications. Interaction with local and remote devices requires a P2P client application that connects to the P2P network. Interaction with the local devices is made from the OSGi platform via device drivers that are loaded as bundled to the OSGi platform, which makes the approach extensible. There is no information in literature regarding how this approach handles NAT and private IP addresses assigned to devices.

In [137] Venkitaraman proposed an approach for secure sharing and control of DLNA home resources based on P2P middleware. The middleware is responsible for the creation and maintenance of groups, which form a logical union of devices based on some criteria. For example a group of devices can correspond to the set of devices in a certain home network, or a group of mobile devices can represent a set of friends. Groups could also be based on context information such as location or interests. Groups are implemented and maintained as an overlay network that is not dependent on an underlying particular protocol or service logic. In order to enable extension of DLNA to multiple home networks [137] suggested xUPnP as a UPnP extension that runs over the P2P middleware. xUPnP understands the UPnP discovery protocol and maps it to messages that can be sent between the device node and friends with which it is shared by sending the messages to the members of the groups –

where the group can be defined as all the friends of the home network sharing the device. In the remote HAN messages are advertised, thereby achieving seamless integration with client applications. Once advertised in the remote HAN, direct communication can be made between local control points and remote devices and services. The P2P overlay is responsible for providing the local xUPnP application with presence information about group members joining or leaving the network, which can trigger local updates, e.g. removing remote devices originating in the leaving peer, or sending joining peer information about local services. Private networks are supported in a limited way through tunnels as long as there are no colliding IP addresses i.e. devices with identical addresses in different NAT-ed domains. While this approach may be applicable for connecting mobile devices connected to a public network to a home network, it is not reasonable to expect no IP collision between two home networks. The IP addresses assigned by home gateways are typically in the same range therefore collisions are highly likely. The system is designed for UPnP/DLNA services and is not extensible to support other service protocols. The home user is required to configure and administer the membership of the groups, which determines the groups to which discovery messages are advertised. Communication is allowed between authenticated peers and is encrypted. The system does not include support for intra-HAN service interoperability.

3.3.3.1. Analysis

Peer to peer is a well established distributed computing paradigm. Using P2P for sharing home resources relies on connecting the home network, through a P2P middleware to a P2P network; thereby allowing searching and downloading content, or discovering services from participating networks. A peer that runs the middleware and sends the information to remote peers collects the information about the local HAN. P2P is an established mechanism for searching and downloading content. P2P file sharing applications have been shown to scale to millions of nodes. A major drawback of such systems with regard to controlled inter-HAN service interoperability is the lack of mechanisms for access control. The typical notion of sharing in P2P is that a resource, e.g. a file, is either shared with all, or not shared at all. Another drawback is that access to services is typically made through a P2P client [97, 77] rather than through a service protocol client, thereby implying the need for service-P2P bridging solution. None of the reviewed protocols support service composition. Loeser et al. [77] support intra HAN service interoperability by bridging between multiple service protocols and the P2P network. Park et al. [97] support only UPnP devices. While Park et al.

use relay nodes to circumvent NAT, it is unknown if and how private IP address are handled by [77].

3.3.4. Distributed OSGi-based sharing

OSGi platform (introduced in section 2.3.6) is a mature technology with many attractive features for HAN equipment vendors, especially software life cycle management. However it is typically used as a central server rather than a node in a distributed architecture.

Wegner presented an architecture for interconnecting UPnP networks using OSGi technology [139]. The architecture presented is based on 4 main concepts: (i) OSGi platform, (ii) proxying remote devices in the local HAN, (iii) communication through a single secure channel, and (iv) device/service filtering. OSGi provides the basic infrastructure for a proxy that is split between the local HAN and remote HAN. In each network a proxy runs and collects information about the local HAN UPnP services and reports to the remote proxy about shared services. When the local proxy learns about remote devices it generates a local device proxy in the local HAN to represent the remote device, thereby supporting seamless integration with existing UPnP applications. The communication between the remote HANs is based on R-OSGi [107] which also allows for a secure connection to be established, e.g. with SSL/TLS encryption. Moreover R-OSGi does not dictate a transport layer; therefore various secure communication channels could be used including TCP, HTTP, and HTTPS. An important aspect of the architecture is the ability to control and modify invocation requests before they are sent to the remote HAN. Filtering is supported through configuration of which users a device or a service should be shared with using LDAP syntax. The actual streaming of content is performed through the same communication channel. The system does not support intra-HAN service interoperability. This system provides a benchmark that demonstrates the invocation delay for remote invocation with an increased argument size for a SOAP request. The benchmark was made on Intel Core 2 Duo @ 2 GHz and showed that with small argument size (<8 killobytes) the delay is roughly 100 milliseconds, and when the argument size grows to 65 killobytes, the delay grows to 1400 milliseconds. The system was evaluated with one device and did not include support for UPnP eventing.

One drawback of this architecture is that the administration of the relationships with friend's

networks requires configuration of R-OSGi systems, which require knowledge of IP addresses or some other pre-shared key. In addition, it is not clear how the dynamics of adding and removing remote HANs with which the local HAN is agreeing to share devices is handled. While R-OSGi may be very efficient in supporting remote invocation as shown in [108], however it is not clear if it is suitable as a system that connects HANs with dynamic relations between them. Finally, the architecture is designed specifically for UPnP services and does not claim extensibility to other service technologies.

3.3.4.1. Analysis

OSGi architecture provides a flexible infrastructure for intra-HAN service interoperability as well as to inter-HAN service interoperability through its module management system. OSGi has many advantages for home networks, especially its flexible and resilient module system with existing support for mapping UPnP services to the OSGi platform. Additionally OSGi is positioned as a solution for intra-HAN service interoperability as well as device management, e.g. for set top box. However, as a distributed architecture, especially in the context of home network, OSGi is not mature yet. The home user is not an enterprise architect and the assembly and configuration of OSGi servers for establishing relationship between remote HANs is beyond the scope of a non-technical user. While the low-level mechanisms for enabling secure remote communication exist in R-OSGi, high-level and standard mechanisms for identifying users and therefore establishing trusted communication with remote HANs is missing. R-OSGi could potentially enable composition of services from multiple HANs, however this is not part of the architecture described in [107]. NAT is handled by using local addresses that are mapped to remote addresses. When a proxy advertises a remote device locally, it assigned it a local address. When a result from a UPnP action on a remote device contains URLs, they are replaced with local URLs that would be forwarded to the remote ones over the tunnel between the HANs.

3.3.5. SIP-based sharing

The Session Initiation Protocol (SIP) [112] was first introduced in 1996 as an application layer control protocol to establish, modify and terminate networking sessions. The protocol supports point to point, and point to multi-point sessions. Typical applications of SIP include instant messaging, presence, file transfer, video conferencing, phone/voice, and streaming of media [118]. In SIP terminology a SIP client is an end point identified by a SIP

URL that wishes to participate in a communication session. A client can initiate a SIP request, which is answered by a SIP server (User Agent Server). The SIP request is passed between SIP servers until it arrives to the destination. Connecting multiple networks with SIP requires additional support for NAT traversal because SIP uses UDP for both signalling and media. There are several solutions to this problem, such as SIP-aware NAT, STUN or TURN [78]. Another technique is Interactive Connectivity Establishment (ICE) [111], which enables a public rendezvous point for devices with private addresses.

In [51], a Service Virtualizer based on the SIP Service Discovery Gateway [50], facilitates service discovery through a presence extension of SIP. Communication between the networks is made over a SIP session between the users which both have a SIP identity. The local Service Virtualizer learns about remote services by subscribing to a remote service discovery gateway or an intermediary presence server. Once the local Service Virtualizer learns about a remote service it creates a local instance of it with proxies that handle control requests and event notifications. The HAN user has no control over which services are shared, remote HAN users can select which services they wish to virtualise in their HANs from all the services that are shared with them. Users control the virtualisation configuration either on demand when a new device is discovered or by defining a managed list of devices that should be virtualised. However this only defines which remote devices are shown locally, it does not define access control (i.e. which local devices are shared with remote HANs), which is not handled in this architecture. Once the device has been announced in the local HAN, control applications can interact with it directly, thereby supporting the desired seamless integration. Control requests and event subscription requests require establishing a remote service usage session with the remote service discovery gateway. As a result, the remote service discovery gateway configures the firewall to grant remote access to the requester. Authors of [51] claim that the service virtualisation is generic and can be applied to additional service protocols, however they only described a design and implementation for UPnP services. The architecture does not address intra-HAN service interoperability. In order to handle private networks and possible collisions between identical private addresses used by multiple networks, Home DNS [9] is used. Home DNS has a number of advantages, the most important of which is seamless addressing, such that the device can be accessed from within the HAN and from remote HANs with the same address. However one significant drawback is that it requires users to configure the address of a dynamic DNS provider, which is a service provided over the Internet. This is a one-time

configuration, however non-technical users are not typically familiar with this kind of configuration. In addition, some dynamic DNS providers charge for this service

W-DLNA [84] describes a SIP based architecture for connecting remote media devices with DLNA based home network. The main concept of this architecture is the virtualisation of Digital Media Player (DMP)/Digital Media Server (DMS) devices. The virtual devices are installed in the home gateway and communicate through a SIP server with remote counterparts. The SIP server is installed in some public domain, which is accessible over the Internet. W-DLNA supports various remote access scenarios for DLNA content including home-to-home, and remote mobile access to local home content. In each participating network, a gateway running the W-DLNA system is running. In a mobile device, the gateway is installed locally in the mobile device, in a home network it can be installed on the home gateway device. The SIP server is used for interconnecting the various gateways. Interaction with remote devices is made through dedicated proxies tailored for media access, which are generated in the gateway. In a network hosting a media server, a virtual media player will be created, and similarly in a network hosting a media player, a virtual media server will be created and hosted in the gateway. The architecture supports seamless integration between remote devices and client applications in the local HAN only partially - remote devices are not announced in the local HAN; instead their content is available through the local proxy that facilitates the access from the local player to a remote media server. This approach is therefore specific to media and does not extend to UPnP services or other service protocols in general. The virtual devices are also responsible for IP mapping between internal and external addresses. This is done by mapping the internal IP address and port of the local device to a port on the external IP address. When the local device sends a packet to a remote address, the virtual device (which is part of the home gateway) replaces the headers with a source IP address that corresponds to the external IP address of the HAN and a port that is mapped to the local device. The same procedure is repeated in reverse in the remote HAN when a message is received. This approach however relies on dynamic configuration for opening ports in the firewall and allowing only source based access. Authentication and access control are both supported. Access control is specified per content per user and is enforced by the virtual device. The implementation does not include support for intra-HAN service interoperability. The specific technique used by this architecture for NAT traversal for SIP is not described in the literature.

3.3.5.1. Analysis

SIP based solutions use SIP sessions to communicate between remote homes. Each home networks (or mobile device) additionally runs a server that collects information about the local HAN and interchanges this information with the server in peer remote HANs over the authenticated SIP session. The concept of service virtualisation was introduced, such that virtual services represent remote services in the local HAN. Discovery information is relayed over the SIP connection. SIP is problematic with regard to NAT traversal, however there are several solutions such as STUN, TURN, and ICE to work around these problems. There are several drawbacks to these approaches. TURN, which is an IETF standard has scalability issues and assumes clients have trust relations with the TURN server. In addition, TURN complicates the configuration of the SIP user agent. STUN is limited and is not appropriate for all NAT configurations, and finally ICE is complex to configure. None of the reviewed SIP based systems support intra-HAN service interoperability, and specifically none supports service composition. They are focused on a single HAN service protocol (UPnP) and do not address extensibility to additional service protocols.

3.3.6. VPN-based sharing

Virtual private networks (VPNs) have emerged as an economic alternative to leased lines for building private networks [72]. The private network is said to be virtual as it is constructed over another network (typically a public carrier network such as the Internet). A VPN enables users to tunnel their traffic securely through public networks as if they are connected to the same private network. A tunnel employs cryptographic techniques to prevent access to VPN packets for non-VPN members while sent on the public network. There are many variations of VPN tunnels, some are based on layer 2 tunnelling such as Layer 2 Tunnelling Protocol (L2TP) [128], while others are based on layer 3 (IP) tunnelling such as IPSec [71]. VPN was mainly targeted for enterprises aiming at reducing costs while providing security and scalability. While in enterprises, this could be established and maintained by system administrators and security experts, applying the same approach for home users requires special consideration to aspects such as the simplicity of setup, which may require more technical skills, the performance overhead and the bandwidth limitations. Some home gateways provide VPN functionality, allowing clients to establish end-to-end secure tunnels to the home network.

A VPN based solution for communication between home networks is presented in [10] with a special solution for discovery based on the ATOM [90] web syndication protocol. The architecture involves an IP tunnelling mechanism such as a VPN, which is used as a communication channel for all traffic except for discovery between the remote home networks. A UPnP aggregator in the local HAN collects information on local UPnP devices and publishes this information through an ATOM feed on a local HTTP server. An extension of a control point application can access remote HTTP servers and request the feed containing information about available devices and services. The aggregator can also accept HTTP POST message updates from remote HANs with information about remote devices. It is suggested that a future architecture of UPnP devices and control points will have the embedded support for out-of-band discovery mechanisms such as ATOM feeds. One weakness of this solution is that it rules out many of existing control applications using standard UPnP in their implementation because standard UPnP clients need to be enhanced with an ATOM reader to receive discovery notifications. Another drawback of this approach is the lack of a uniform approach to service discovery (of local and remote devices), which can lead to additional complexity of control points. Since the approach suggested builds on ATOM for discovery and IP tunnelling for the rest, a direct connection to the remote service endpoint is still required such that an IP packet can be routed to the remote IP address. With private networks in place such that IP addresses collision are very likely, this approach is not applicable for connecting two networks. The system architecture does not address access control for HAN resources. In addition user management (being able to add and remove networks with which services are shared) is not supported. Finally, the system architecture does not address intra-HAN service interoperability.

DLNA Agents for SNS (DAS) [121] is an architecture based on using social network services (SNS) and VPN connections for interconnecting remote HANs. The DAS architecture is composed of an SNS server, DLNA agent and DLNA devices. The user has an account on the SNS server and at configuration time the SNS authentication details are entered. When two users wish to connect with each other, the local DAS instance will collect information from the available media servers in the local HAN and upload it to the host user's SNS page. In addition the local VPN server in the user's DAS instance will upload connection details to the user's SNS page enabling the VPN client of the remote user to connect to it. The SNS server enables the configuration of what should be shared with which friend via editing of a sharing management configuration page. Once content has

been shared, a page is be created for the remote user (in their SNS account) with the list of shared content from which the remote user will be able to choose what they wish to access, and the VPN server details are downloaded to the remote DAS instance. A virtual media server is established in the remote HAN, which advertises itself as having the content list that was downloaded over the VPN connection, thereby enabling seamless integration. Finally a reverse proxy is used to transfer traffic between DAS instances. The reverse proxy is used to transfer traffic that is sent to the local address of the virtual DMS to the VPN IP of the remote the DAS in the remote HAN, where it is forwarded to the local device. Private IP addresses are not published beyond the local HAN, instead the local DAS instance maintains a mapping between content and the device, which owns it. The advantage of DAS is that it uses common social networking practice to hide the details of secure communication setup from the users. intra-HAN service interoperability is not addressed by this architecture.

The Dial-to-Connect (D2C) VPN System [55] was designed as a simplified version of an on-demand VPN that works over a SIP network to enable sharing of DLNA devices and services. Phone numbers are used for user identification. In order to establish a connection between users, they need to enter the phone number of the other user via a SIP client. Once the remote user accepts the request, the two-way VPN connection can be established. In order to meet the quality of service required for media streaming, D2C uses the QoS service provided by the SIP network. D2C uses an IPSec [71] tunnel in order to establish secure communication between the remote HANs. IPSec packets are encapsulated in UDP packets for NAT traversal. In addition, in order to handle duplicate private IP addresses (assigned by the NAT in the two networks) bi-directional NAT [122] is used, such that both the source and destination addresses are modified consistently to identify the devices in the local and remote HAN. Since DLNA discovery is based on UDP multicast, which cannot be sent over the VPN, an additional mechanism is required to relay messages from the local to the remote HAN. Once the VPN is established, local multicast discovery announcements are translated into unicast messages and are relayed over the VPN connection to the remote HAN, enabling seamless integration with local DLNA clients. The interaction is ad hoc and does not support multiple remote HANs simultaneously. Authentication is supported through SIP authentication but access control is not supported. The architecture is designed for DLNA devices and does not support extensions to additional service protocols. The system is deployed in the home gateway and is interacted with through a SIP client. No user

configuration is required, however the system requires interaction between human users, such that once a remote HAN requests a connection with the local HAN, the local HAN's user needs to accept or decline the request.

3.3.6.1. Analysis

VPN based approaches are drawn from experience from multi-site enterprise secure communication and its application to connecting home networks. Several approaches have been presented for setting up the communication and for using it for inter-HAN service interoperability as presented above. The main advantage of these approaches is that it provides higher security and can enable higher-level protocols to work as if they run within a single network. However this is not desired when access control is needed. None of the reviewed protocols support service composition or intra HAN service interoperability. NAT traversal support ranges from limited support in [10], local redirection in [121] and bidirectional NAT in [55]. Support for seamless integration is also limited, [10] does not support seamless integration, [121] partially supports it through aggregation to a virtual media server, and [55] supports seamless integration. Access control is only supported by [121] while all of the reviewed systems support authentication.

3.3.7. Proprietary protocols for sharing

P. Belimpasakis et al. propose an architecture based on a Home Media Atomizer (HMA) which is a component that can be either added to the home media server or run as a standalone component [12]. This element acts as a mediation layer between UPnP media servers content in the local HAN and HTTP based web feed protocols such as ATOM [90]. The HMA acts as a UPnP control point and loads data about local HAN's media content to the HTTP server that can handle requests from remote users about locally available media content. The HMA enforces access control so that remote users can only access services and content that is shared with them. This approach is very similar to that proposed in [10] (remote access to UPnP with ATOM based service discovery) however instead of using a VPN for the connection between remote HANs it is based on reconfiguration of the home gateway for allowing remote access to local HAN.

Siekkinen et al. present hBox in [119]. hBox is similar to [51] in its virtualisation approach

to remote resources. In this architecture a UPnP embedded device is introduced to all networks that wish to be interconnected. This device is capable of establishing and maintaining secure tunnels between local and remote HANs. This device is controlled via a mobile phone, which is used to establish trust with the remote HAN by sending SMS authentication messages. Selecting a remote user from a mobile phone address book makes the initial connection establishment. A request is sent as a text message to the target phone, which is required to accept the request for connection. The response includes the IP address to connect to and some shared security key. The mobile phone application is used to select services to be shared from those discovered in the local HAN. Once a device is shared with a remote HAN, hBox starts forwarding its presence announcement to the remote HAN's hBox, which announces it locally. Remote devices are represented in local HANs as embedded devices within the hBox UPnP device, thereby facilitating seamless integration with existing UPnP applications. Requests made by local control points to the hBox embedded devices are relayed to the remote hBox, which forwards them to the devices and returns back the result. Private networks are handled through STUN/TURN or ICE. hBox requires the introduction of an additional appliance to the home network. In addition, it uses non-standard communication, authentication, and user management protocols. Another disadvantage is the cost related to sending text messages for initiating the connection. The authors of [119] argue that the approach can be potentially extended to other types of service protocols however they have only implemented UPnP services. Finally, the architecture does not address intra-HAN service interoperability.

The SHARE architecture [76] extends the home gateway to manage the connections with peer networks and to exchange information about streaming services (UPnP AV servers) in local and remote HANs. In essence this home gateway extension relays SSDP messages to remote home networks by establishing a TCP connection, which eliminates the need to retransmit packets more than once. An additional component of the SHARE architecture is a virtual media server that contains an expanded control point and an expanded media server. The expanded control point discovers remote media servers and collects information about the media they contain. The expanded media server consolidates the media content received from multiple remote media servers. Media streaming is enabled using another component (Media Distributer) in this architecture that acts as a proxy for remote streams. The Media Distributer supports distributed streaming, and proxies streaming protocol data between multiple remote sources and the local UPnP media renderer. The system is focused on

enabling inter-home communication however it is not described in literature how it addresses NAT traversal. In addition the architecture does not address fine-grained access control over what content is shared. The system is designed specifically for UPnP A/V architecture and is not extensible to other service protocols and it does not support any intra-HAN service interoperability.

Intel's Device Relay software [65] enables connection between two UPnP networks by mirroring of UPnP devices across these networks. The connection between local and remote device relays requires the user to provide the local relay with the IP address of the remote relay. Then the device relay can mirror devices from local to remote HAN by relaying discovery messages to the remote HAN where they are virtualised by the local device relay instance, thereby enabling seamless integration with local control points. There is no indication in literature with regard to how Device Relay handles private HANs behind NAT or how the communication between Device Relay instances is performed. The Device Relay approach has a few limitations, such as only being able to connect to a single remote HAN simultaneously as opposed to several as supported by most other solutions presented here. Another drawback is that all devices are mirrored; there is no control over what is being shared. Finally the Device Relay requires open ports in both local and remote HANs and corresponding firewall configuration. The Device Relay is specifically designed for UPnP and does not support additional service protocols or intra-HAN service interoperability.

In [31] Chowdhury et al. describe a system for connecting multiple UPnP networks via an extension of the home gateway. A proprietary protocol is suggested for establishing a secure channel between local and remote HANs based on a pre-shared key that has to be available to all networks that wish to share devices and services with each other. This key is used for encryption of messages exchanged between the home networks. In addition, in order to establish connection with a remote HAN, its IP address needs to be available to the home user initiating the request. This requirement is problematic both because it is not realistic to require non technical users to be aware of other home network's IP addresses, but also because IP addresses may be changed dynamically (with DHCP). Once connected, the local and remote home gateways exchange information about devices in their network and access level for each device (permit/deny). Similar to the approach presented in [119], remote devices are added as embedded devices within the home gateway with necessary modification to URLs in the device location and control URLs so that they will be served

locally, facilitating seamless integration with local UPnP applications. The configuration of communication between the remote HANs is made through the modification of the home gateway and firewall when a connection request is accepted, and is updated whenever a device or service is shared with another network. The architecture relies on the home gateway to map from public IP address to services' private IP addresses when request from remote HANs are received to local HAN. The system is designed specifically for UPnP services and is not natively extensible for other service protocol and does not support intra-HAN service interoperability.

3.3.7.1. Analysis

Various specialised solutions have been suggested for enabling home-to-home sharing of services and content. The approaches presented in this section differ in the problem they focus on, as well as in the methodology towards the general sharing problem. The different approaches suggest mechanisms for establishing trust between remote HANs (e.g. hBox [119], Chowdhury [31]) or mechanisms for propagation of discovery messages (e.g. HMA with ATOM web syndication protocol [12]). None of the reviewed systems support service interoperability or service composition. Access control is not supported by [65, 76], and supported by [12, 31, 119]. Authentication is supported by all of the reviewed systems. All of them except for Home Media Atomizer [12] support seamless integration. With regard to NAT support there are different approaches: hBox [119] uses SIP TURN/STUN or ICE, Home Media Atomizer [12] relies on modifying the HG, and Chowdary et al. use local redirection. For [65, 76] there is no information in literature about how and if they support NAT traversal.

Architecture/ Criteria	Scope	Platform	Communication	Seamless integration	NAT traversal	Remote Service discovery	Remote service in local network	Security	System configuration	Intra-network support	Service composition
IP configuration	Services	Home gateway	N/A	X	HG manipulation	X	X	Requires manual config	Manual HG configuration	X	X
Web-based sharing	Content	Web	N/A	N/A	N/A	N/A	X	Authentication, authorization	N/A	N/A	X
Park et al. [97]	Services , content	JXTA	P2P	X	Relay peers	X	X	Authentication	N/A	X	X
Loeser et al. [77]	Services , content	OSGi + JXTA	P2P	X	Unknown	X	X	Authentication, encryption	N/A	√	X
Venkitaraman [137]	DLNA services	Proprietary P2P	P2P	√	Limited support	Relay	Direct access	Authentication, encryption	Creation of sharing groups	X	X
Wegner [139]	UPnP services	OSGi	R-OSGi	√	Local redirection	Relay	Virtual service	Authentication, authorization	R-OSGi relationships	X	X
W-DLNA [84]	DLNA services	Proprietary	SIP	X	Unknown	Relay	Content aggregator virtual service	Authentication, authorization	SIP users	X	X
Haber et al. [51]	Services	Proprietary	SIP	√	Home DNS	Relay	Virtual service	Authentication	SIP users, Home DNS	X	X
Belimpasakis et al. [10]	Services	Proprietary	VPN	X	Limited support	ATOM	Direct access	Authentication, encryption	N/A	X	X
DAS [121]	DLNA services	Proprietary	VPN + reverse proxy	√	Local redirection	SNS	Direct access	Authentication, authorization	N/A	X	X
Dial-to-Connect [55]	DLNA services	Home gateway	SIP+VPN	√	VPN, bidirectional NAT	Relay	Direct access	Authentication	N/A	X	X
Home Media Atomizer [12]	DLNA services	Proprietary	IP	X	HG manipulation	ATOM	Direct access	Authentication, authorization	N/A	X	X
hBox [119]	UPnP services	Proprietary	Proprietary	√	TURN/ STUN/ICE	Relay	Virtual service	Authentication, authorization, encryption	N/A	X	X
SHARE [76]	UPnP services	Home gateway	IP	√	Unknown	Relay	Direct access	Authentication	N/A	X	X
Intel Media Relay [65]	UPnP services	Proprietary	Unknown	√	Unknown	Relay	Virtual service	Authentication	IP addresses	X	X
Chowdhury et al. [31]	UPnP services	Home gateway	Proprietary	√	Local redirection	Relay	Virtual service	Authentication, authorization, encryption	IP addresses	X	X

Table 3 Inter-HAN Service Interoperability Architectures Comparison

3.3.8. Conclusions

The various approaches for inter-HAN service interoperability reviewed in the previous sections. Table 3 presents a summary of the comparison between them with regard to the comparison criteria defined in section 3.3

The first column in table 3 defines the *scope* of the reviewed architecture. Some architectures support sharing of services in general [10, 51, 77, 97], however most of the systems reviewed focus on a single service protocol – UPnP or even a subset of it support by DLNA, especially multimedia devices such as media player and media server. Some of those supporting UPnP services argue for broader generality (e.g. [119]) however they have only presented a design and implementation for UPnP services.

The second column in table 3 refers to the *computing platform*, which is used for the suggested architecture. The term computing platform refers to the hardware or software on which the system executes. A number of architectures suggest deploying the inter-HAN

service interoperability system as part of the home gateway. The advantage of this approach is its always-connected nature – being installed as part of a device that is always connected to the Internet, rather than on a desktop or an appliance that may be shutdown periodically. A potential drawback could be the lock-in to a single vendor, and the CPU and RAM limitations that may apply when provided as a home gateway extension. JXTA is a common platform used by P2P based approaches for establishing a P2P overlay. OSGi offers attractive platform due to its flexible module management system, however most OSGi experience is with centralised systems and it is not clear that R-OSGi is appropriate for use in the home-to-home setting due to the possibly complex configuration of relation between remote OSGi instances. Other solutions are based on proprietary architectures and platforms and are therefore poorly positioned to support service protocol interoperability and extensibility.

The *communication* column in table 3 defines the channel that is used between remote HANs to exchange information. Table 3 shows that there are several standard approaches for establishing communication channel between remote HANs. P2P approaches [77, 97, 137] rely on sending messages over the P2P network; such that each network must run at least one instance or more of a P2P middleware system that connects to the P2P overlay network. Distributed OSGi architectures using R-OSGi [139] rely on a communication channel that is established between remote instances of the platform. R-OSGi does not dictate a transport layer; therefore various secure communication channels could be used. SIP based approaches [51, 84] establish sessions between remote HANs and use these sessions to exchange information about services existing in these networks. VPN based architecture establish a VPN connection between the remote HANs enabling IP connectivity. Other communication channels rely on establishing secure IP channels between networks.

Seamless integration of remote devices with local HAN is an important feature required from an inter-HAN service interoperability system for allowing existing applications to interact with remote services similarly to the way they do with local services. Surprisingly not all of the architectures support seamless integration in this sense as can be seen in table 3. Another aspect of seamless integration as defined in section 3.3 is the ability to connect to multiple networks simultaneously. Most reviewed architectures support this behaviour with several exceptions: Dial-to-Connect [55] is on-demand and allows a network to be

connected to only a single remote HAN at a time. Intel Device Relay supports only one target network for mirroring at a time.

NAT Traversal is one of the most challenging problems for service inter-HAN service interoperability. The main challenge with private networks is that devices are assigned private IP addresses that are not reachable from outside the home network. In UPnP discovery, devices announce an address where they can be contacted. This address will typically be a private address. Therefore when shared with a remote HAN, by default this IP address is not reachable. Moreover, the remote HAN may have a device with identical private IP address locally. As can be seen in table 3, several techniques are used in the reviewed architectures. The local redirection approach indicates that private addresses are not reported to remote HANs, therefore there is no need to handle duplicate IP addresses, however there is no direct access from a remote HAN to the device, and traffic to, and from the device is proxied. Another technique is home gateway manipulation to allow remote access. This approach requires publication of the external IP address of the home gateway instead of the device private IP address, and configuration of the home gateway to redirect traffic on the device port on the public IP address to the device. One problem with this approach is that it does not allow fine-grained access control, when a device is shared with a remote HAN; all services are available for the remote HAN to use. STUN, TURN, and ICE are common techniques for NAT traversal. The problem with these approaches for NAT traversal is that they are not suitable to all scenarios and NAT configurations, and in addition there are specific issues related to each of these techniques as discussed in 3.3.5.1.

Remote service discovery (column 6) and *remote service in local HAN* (column 7) summarise how remote services are advertised in the local HAN and how they are accessed. The most common approach for announcing remote services in local HAN is through relay of discovery announcements received from the remote HAN. The announcement is not necessarily repeated in the local HAN, in some cases the announcement message is processed and is not repeated in the local HAN, e.g. W-DLNA [84]. Two other techniques for transmitting discovery information were suggested: ATOM web syndication protocol [10, 12] and social networking services in DAS [121]. Using ATOM for discovery relies on mapping between UPnP announcements and ATOM protocol and therefore does not integrate seamlessly with local client applications. Access to remote services can be divided into two groups: direct access, and virtual service. Direct access assumes IP connectivity

between the remote HANs, therefore relies on a VPN or on manipulation of the home gateway. The virtual service approach implies that the communication with remote services is proxied in the local HAN by a virtual service that interacts with the local HAN applications and encapsulates the interaction with the "live" device in the remote HAN.

As indicated in table 3, while *security* is an important aspect of inter-HAN service interoperability, not all of its aspects are fully supported. While authentication is supported by all reviewed architectures, access control is only supported in a few of them [12, 31, 84, 119, 121, 139]. P2P based architectures [77, 97, 137] do not support access control as it is assumed that once a service is shared, it is shared with the P2P network rather than with a specific user. In VPN based architectures, which use relay-and-repeat technique for discovery messages, there is no access control and all messages are delivered to all participants. While it is important to ensure the privacy and confidentiality of the traffic between the networks, only a number of architectures support encrypted communication channels [10, 31, 77, 119, 137].

The *system configuration* column in table 3 refers to the interaction of the system with the home user and how high level actions are translated into this interaction. For example adding a friend's network and initiating communication with them may require the IP address of the remote HAN [31, 65]. In SIP based architectures, system administration is performed by adding the SIP address of the remote friend's network. The distributed OSGi based architecture [139] requires configuration of the OSGi server instances such that they are able to communicate. In [137] the user is required to create and administer groups such that other users can join pending his approval. The IP configuration approach for sharing resources requires manipulation of the home gateway and potentially the firewall system. The above manual configurations required from the home user do not seem realistic for a non-technical home user.

The *intra-HAN interoperability* column in table 3 indicates that only [77] has some level of support for intra-HAN service interoperability and as mentioned earlier this is limited to being able to locate services from multiple service protocols locally. However none of the reviewed systems includes support for service composition.

The purpose of the review was to identify the gaps in existing approaches, however also to identify the requirements for an inter-HAN service interoperability system, that together with the requirements for intra-HAN interoperability system, define the requirements for an integrated service oriented architecture for HANs. As can be seen from table 3, no single system supports all of the reviewed aspects. The next section specifies the requirements for this architecture.

3.4. System Requirements

From the state of the art presented in previous sections we can draw the requirements for an integrated intra-HAN and inter-HAN service oriented system. It can be seen from table 2 and table 3 that no single system offers an integrated approach for both intra-HAN and inter-HAN service interoperability. This section presents a consolidated set of requirements for the proposed system. The overarching objective of the architecture is to enable services from multiple technologies to be shared across multiple HANs in a controlled manner and facilitate their interoperability and composition in local HANs with other services of multiple service protocols.

3.4.1. Intra-HAN service interoperability requirements

- (#1) Cross service protocol service composition – The reality of home networks indicates that no single service protocol dominates. In order to achieve richer user experience for home users, service composition must not be limited to composition of services from the same service protocol. The full potential of services in the home can be realised through connecting services from multiple application domains, and multiple manufacturers, which may support different service technologies. The system must support composition of services from both similar and different service protocols. Composite services must also be composable to allow further composition with other services.
- (#2) Shared composite services – There may be cases when users would like to share composite services rather than exposing the constituent low level resources. As composite services may offer some functions that may be usefully shared, the system is required to enable sharing of composite services in an analogous way to the sharing of atomic services.

- (#3) Cross HAN service composition – The system must support composition with both local as well as remote services (services from another home network) in a seamless fashion, hence composite services may include either local, remote or even external services (public domain services).

3.4.2. Inter-HAN service interoperability requirements

The requirements for inter-HAN service interoperability can be divided into a number of main categories: (i) seamless integration; (ii) private networks and firewalls; (iii) security; (iv) performance; (v) extensibility; and (vi) manageability.

3.4.2.1. Seamless integration

In order to integrate with existing technologies in the home network while providing the user with the new capability for sharing devices, services, and content with remote users, the following is required:

- (#4) The system must enable users to share devices from a local HAN with remote HANs.
- (#5) It must be possible for service protocol clients in the local HAN to automatically discover devices and services shared from remote HANs.
- (#6) There must be no restriction that prevents sharing the same devices and resources with multiple remote HANs.
- (#7) The interaction of applications in the local HAN with devices and services from remote HANs must be identical to the interaction with local devices. This is required so that existing service applications in the local HAN can interact with remote devices indistinguishably to the way they currently do with local devices.
- (#8) The system must be plug and play such that it is installed in the local HAN and can immediately interact with local devices and services without requiring any modifications to standardised device or service protocols.
- (#9) Networking – The system must be able to work regardless of the network technology supporting access connectivity (e.g. xDSL, cable).

3.4.2.2. Private networks and firewalls

Modern networks use NAT to assign private IP addresses to internal devices in the home network. Firewalls protect the home networks from unprivileged access and malicious attacks. The sharing system must be able to operate in such an environment.

- (#10) With the increasing popularity of private networks, the system must be able to discover and share devices with networks that are using NAT, supporting devices with identical IP addresses across multiple HANs.
- (#11) The system must able to communicate with remote HANs protected by firewalls without requiring any manual reconfiguration.

3.4.2.3. Security

Security is an important aspect of the system. This is to ensure that the home network does not become exposed to new threats, illegal access of resources or any other type of known threat. Security has a number of aspects:

- (#12) Authentication – All communication with remote HANs must be made by authenticated users only. Unauthenticated access must not be allowed.
- (#13) Access control – Sharing must not be automatic. Users must be able to control explicitly which resources are shared with which other remote HAN. The sharing control must not only allow a user to define per resource whether it is shared or not but also with which other remote HANs it is shared.
- (#14) Confidentiality – Access to private data must be restricted to privileged users, therefore all communication must be secure. In order to protect the privacy of transmitted data from malicious code sniffing the network traffic, all communication between the local and remote HANs must be encrypted.
- (#15) Vulnerability – Some service protocols for the HANs contain security vulnerabilities, however as they are not designed to work over unsecure networks such as the Internet, these vulnerabilities are relatively low risk when restricted to a single HAN. The multi HAN sharing system must not increase security vulnerability of the home network by introducing new threats or by extending or exposing existing vulnerabilities.

3.4.2.4. Performance

System scalability has several dimensions: the number of HANs with which sharing is enabled, the number of services in the local HAN, and the number of overall services shared with the local HAN. For the purpose of defining the performance requirements for the sharing system we anticipate that the number of HANs with which a HAN user will share devices, services and content will be in the range of 5-15 friends which represents a social circle of family and close friends. Parks Associates market research report from 2010 estimated the number of connected devices in the HAN globally to be over a billion. An IDC market research from 2007 predicted that the number of HANs would exceed 200 million households by 2011 [44]. A customer survey from 2008 by a HAN equipment retailer specialising in the technical hobbyist market recorded that an average of 7 connected devices in their customers' HANs [24]. Assuming not all devices can be shared (e.g. home gateway), we expect the number of shared devices will not exceed 5 devices per HAN. When considering UPnP devices, each device has typically no more than 3 services. Using the UPnP device to service ratio as a reference, the number of services shared from the HAN is estimated between 10-20 services which when shared from 5-15 remote HANs sums to up to 300 remote services with which the system is expected to operate with.

- (#16) Scale up (intra HAN) – the system must be able to represent up to 300 remote services with no significant latency. More services can be supported with reduced performance.
- (#17) Scale up (inter HAN) – the system must be able to scale to a small number of remote HANs corresponding to number of close family and friends. The number of remote HANs must not exceed 15 remote HANs.
- (#18) Scale down – the system should be deployable on various operating environment including Linux, Windows, OSX. The system's deployment requirements should be appropriate for home networks – it must be possible to deploy the system on low-end machines, where RAM does not exceed 1GB and CPU does not exceed 2GHz.
- (#19) Concurrent access – It must be possible for multiple remote HANs to interact with the same local HAN resource simultaneously.

3.4.2.5. Extensibility

- (#20) Given the heterogeneity of the HAN service protocols and its fast evolution it is desired that the system architecture will support extensions for additional service oriented protocols dynamically when such support for them is available.

3.4.2.6. Manageability

- (#21) While some relations between networks may be permanent, others may be ad hoc. Therefore the system must support dynamic addition and removal of remote HANs, i.e. change the set of remote HANs with which sharing is enabled during the lifetime of the system.

- (#22) The local HAN must be able to temporarily pause all sharing without removing the relationship with remote HANs. This could be required in cases of maintenance to local HAN or in case the user wants to use all of his or her bandwidth for another purpose.

- (#23) The system should not require manual configuration of home devices or of home gateway. Administration of the system should be appropriate for non-technical home users.

3.5. Summary

This chapter presented several aspects of the challenges in current home area networking. The intra-HAN and inter-HAN service interoperability challenges were explained and several approaches towards its solution were presented and assessed. Finally the requirements for an integrated service oriented architecture for home network addressing intra-HAN service interoperability with focus on service composition, and inter-HAN service interoperability were presented. Table 2 and table 3 indicate that no existing system in the literature meets all of these requirements. Based on the requirements presented at the end of this chapter, the next chapter present the *Krox* architecture for a service sharing and composition system and a design that realises the architecture.

Chapter 4
ARCHITECTURE AND DESIGN

The previous chapter presented the state of the art in intra-HAN service interoperability with emphasis on service composition and inter-HAN service interoperability. The relevant gaps were identified and presented, deriving the requirements for an integrated service-oriented architecture for HAN that supports both service composition and service sharing across multiple HANs. The key objective of this thesis is to design and implement an integrated service oriented architecture for HAN that supports secure and performant composition and sharing of HAN resources that can plug into existing HANs without requiring changes to network or device technologies. This chapter presents the *Krox* architecture and design based on the requirements presented in the end of the previous chapter.

This chapter has two parts: the first part introduces the *Krox* high level architecture for service sharing and composition; in the second part a design is presented for the realisation of the architecture with multiple service protocols: UPnP and Jini. While the *Krox* architecture is independent from any specific implementation, the realisation of *Krox* architecture with plug-ins for UPnP and Jini service protocol shows how a mapping between the architecture onto different service protocols is feasible. As discussed in chapter 2 UPnP and Jini are service-oriented protocols that support service discovery, service description, and service invocation. While UPnP is programming language agnostic, Jini relies on Java programming language. From market uptake point of view, UPnP devices are very common in HAN, while devices with Jini services are less common at the time of writing but have the potential to become more common with its support for Java. UPnP and Jini both support a communication mechanism for service invocation, however while UPnP supports SOAP

over HTTP, Jini supports Java RMI. Both UPnP and Jini support a parsable service interface that can be used for service composition and for sharing the service with multiple HANs.

The next section describes the high level concepts underlying the *Krox* architecture.

4.1. Architectural concepts

This section presents the main architectural concepts and patterns underlying the architecture and design proposed in this thesis in response to the requirements presented in section 3.4. The architecture is based on the following high-level concepts:

i) Plug-in architecture – In order to meet the extensibility requirements, an open architecture with plug-in modules for each supported service protocol is suggested for handling the diversity of service protocols which has been shown in chapter 2. The concept of plug-in architectures, more specifically OSGi based architecture was suggested in literature [77, 139] for enabling sharing of HAN resources with remote HANs. While OSGi provides adequate module management and support for dynamic loading, unloading, and management of plug-ins, it is mainly useful as a centralised service oriented architecture and it lacks the features required for use in a distributed environment as the HAN-to-HAN requires. Plug-in architectures require an infrastructure that can manage plug-ins lifecycle. A plug-in in the *Krox* architecture is a module that implements the interfaces required by the *Krox* architecture and supports the required extensible event model. A plug-in encapsulates the details of protocol specific discovery, automatic service virtualisation of services from remote HANs, and mapping from the service interface to web services. Each service protocol is supported through a separate service protocol plug-in.

ii) Resource virtualisation – Resource virtualisation has been shown in the literature to be useful for achieving seamless integration, as shown in the previous chapter. The *Krox* architecture is based on representing resources from remote HANs (devices and services) in the local HAN using "virtual resources". "Virtual resources" proxy the communication with the remote HAN and provide an interface to the local HAN that is identical to the interface provided by local devices or services of the same technology. For example a

remote UPnP media server would be represented in the local HAN using a virtual UPnP device. The virtual device would facilitate all the interaction with control point applications in the local HAN by tunnelling the communication over a secure IM&P connection to the remote HAN hosting the "live" device. There, the tunnelled messages are received by the *Krox* system and forwarded to the "live" device. The response messages are tunnelled back to the virtual device. The resource virtualisation is based on the service-oriented nature of HAN service protocols, where devices and services are represented by interfaces, which are abstracted from the implementation, such that the service interface lends itself to virtualisation.

iii) Automatic virtual resource generation – The *Krox* architecture automatically generates local virtual resources for devices from remote HANs that are shared with the local HAN. The automation is enabled due to the availability of service interfaces in a parsable format, e.g. Java (for Jini and HAVi services), WSDL (for DPWS) or XML (for UPnP services).

iv) Instant Messaging and Presence (IM&P) user metaphor – IM&P defines a user model which when adapted for the multi-HAN settings can greatly simplify required administration. When applying the IM&P user metaphor, remote HANs can be represented as IM&P users, such that if an IM&P user is in the "buddy list" it indicates that the local HAN is sharing devices with the remote HAN represented by this IM&P user. This approach abstracts lower level configuration from the home user such as remote HAN IP addresses or phone numbers, and configuration of sharing with a remote HAN, is reduced to adding the username of the remote HAN to a "buddy list".

v) Instant Messaging and Presence (IM&P) based communication – In order to establish secure communication and be able to exchange messages between remote HANs behind NAT and firewall systems, the *Krox* architecture builds on leveraging instant messaging and presence system for secure communication and messaging between remote HANs. Tables 3 in section 3.3.8 reviewed the communication mechanisms used by inter-HAN architectures. The main approaches reviewed were SIP-based communication, VPN based communication, and other proprietary mechanisms. The advantage of the IM&P based communication is its embedded security (authentication, encryption) with its simple setup and established scalability. In addition IM&P provides embedded presence features which can be used to trigger communication

initiation or termination between remote HANs. A disadvantage of IM&P is that it typically requires a server for communication between peers.

vi) Service orchestration – Through representation of HAN services as web services, enable the creation of reusable composite services from atomic ones.

The following sections describe the above architectural concepts and how they are used in *Krox* system architecture.

4.1.1. Plug-in architecture

The fast pace of service protocol and standards evolution implies that changes and introduction of new HAN service protocols are highly probable. The *Krox* architecture is limited in its scope to supporting HAN service protocols that define a parsable service interface (such that can lend itself to virtualisation), however it does not limit the underlying mechanisms defined by the service protocol for service discovery, parsable service description, control, eventing, and any additional protocol. The plug-in approach avoids trying to abstract all service protocols and instead suggests separation through bundling the support for a specific service protocol, such that the support for a service protocol is bundled in a plug-in that contains all the needed support for the specific protocol. Through the bundling of a HAN service protocol support as a plug-in, the system remains service protocol agnostic.

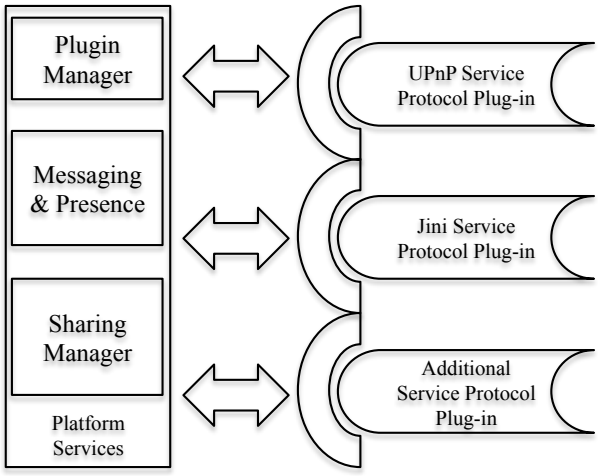

Figure 1 High-level Plug-in Architecture

A plug-in architecture, also referred to as a pluggable architecture [43, 141] is an architectural pattern that is desired when the full function set of the system is unknown during the system design. There are many recent examples to the usability of plug-in architecture such as OSGi plug-in framework, Eclipse plug-in framework[14], Firefox plug-ins[15] and others. The main advantage of a plug-in component architecture is that it enables dynamic assembly of the system from its components. To illustrate this approach, the support for a certain HAN service protocol such as UPnP can be packaged as a plug-in, and a given instance of *Krox* system can load a set of plug-ins. When a new service protocol is introduced and its corresponding plug-in has been developed (potentially by a 3rd party), it can be loaded into the system extending its support to the new service protocol without requiring modifications to the *Krox* system itself thereby providing the required extensibility.

Figure 1 illustrates the concept of an extensible plug-ins in *Krox* architecture. The support for a service protocol is packaged as a plug-in and is loaded into the system by a plug-in manager. The plug-in manager is responsible for management of the life cycle of plug-ins, i.e. loading, starting, stopping, and unloading of plug-ins. The plug-in architecture needs to be complemented by a set of framework services that a plug-in can interact with. It is worth defining more clearly what defines a plug-in and therefore what should be inside a plug-in and what is a framework service. A plug-in encapsulates the support for a HAN service protocol for enabling seamless sharing and interoperability with remote HANs. The plug-in must respond to a set of predefined events representing the core event model, which correspond to status changes of the local and remote HANs and discovery related events. A plug-in can support additional events as required for implementing the support for the service protocol. The plug-in framework defines how plug-ins interact with the rest of the *Krox* architecture the set of events that they are required to respond to. The plug-in framework is described in detail in section 4.2.2. As opposed to plug-ins, a framework service may not contain service protocol specific logic and must have a clear and single function. The messaging & presence service encapsulates the communication subsystem and is described in section 4.2.1. The plug-in manager is described as part of the plug-in framework in section 4.2.2. Finally the sharing manager is described in section 4.2.3.

[14] http://www.eclipse.org/articles/Article-Plug-in-architecture/plug-in_architecture.html

[15] https://developer.mozilla.org/en/Plug-ins

4.1.2. Instant Messaging & Presence

Instant Messaging and Presence (IM&P) is selected for the communication subsystem of *Krox* architecture and as a user metaphor for representing remote HANs. The following sections provide background for IM&P and motivation for this design decision.

4.1.2.1. History of Instant Messaging & Presence

Instant messaging is a well know communication pattern, where the basic idea is the ability to send and receive messages in real-time. Presence messages indicate the availability of the users, e.g. available, offline, busy. While Instant Messaging and Presence (IM&P) gained its immense popularity with the advent of instant messaging chat applications, first implementations of such systems were already introduced in the early days of UNIX operating system. For example UNIX enabled users to get limited presence information and to send and receive instant messages using UNIX commands, FINGER for presence, TALK for sending messages [54]. These protocols were suitable to the early days of the Internet, however they suffered from security and privacy issues. Next in the evolution of instant messaging was Internet Relay Chat (IRC). Introduced in the late 80s, IRC provided real-time conversation between users over a public network. Users could dynamically join and leave the chat room at anytime. In the 90s ICQ[16], America Online, and later Microsoft MSN Messenger[17] presented an Internet scale chat application, which eliminated the need for a chat room. Users could finally communicate directly with each other over a public network. These applications grew rapidly to millions of users, however they were not natively interoperable with each other and were built on proprietary protocols. Subsequently, this started the standardisation efforts. Jabber[18] is an open source project started in 1998 as a client and server that could communicate with several instant-messaging systems. As part of the standardisation efforts, two working groups were formed by the Internet Engineering Task Force (IETF) at different points in time, SIP for Instant Messaging and Presence Leveraging Extensions (SIMPLE)[19] – based on Session Initiation Protocol (SIP), and eXtensible Messaging and Presence Protocol (XMPP) [114] – based on XML streaming

[16] http://www.icq.com/

[17] http://explore.live.com/windows-live-messenger

[18] http://www.cisco.com/web/about/ac49/ac0/ac1/ac258/JabberInc.html

[19] http://datatracker.ietf.org/wg/simple/charter/

which was originally Jabber's underlying protocol. Instant messaging and presence communication pattern is not limited to plain text messaging and has been extended to audio, video, IP telephony and many other types of innovative services.

The concept of IM&P has a number of characteristics that make it suitable for this architecture:

1. Ease of use – IM&P based applications are easy to use and well adapted by non-technical home users.
2. Established massive scalability – IM&P systems have user bases of hundreds of million users and have been show to perform well.
3. Communication overlay – IM&P provides the overlay abstraction for the low-level communication mechanisms between remote HANs. Users of IM&P do not need to worry about low-level networking protocol for establishing connectivity between remote hosts.
4. Captures the concept of "buddies" – The IM&P terminology of users, and friends can be useful for defining relationships between the local HAN (i.e. the "user") and remote HANs ("buddies").
5. Support for point-to-point connections – Enabling messages to be exchanged between remote HANs, e.g. for transmitting service discovery messages.
6. Built-in presence alerting – can be used for being notified when a remote HAN comes online or goes offline.
7. Easy to secure – The experience gained in instant messaging and other applications described above provided well-established security models between clients and servers and between IM&P servers.

4.1.2.2. IM&P as a user metaphor

In order to adopt the IM&P user metaphor for defining the relationships between remote HANs we need to establish a vocabulary that will be used throughout this chapter defining consistently the terms *user*, *buddy*, and *buddy list*:

- **User** – In instant messaging system the user is identified using a username, and defines a human user signed into the overlay network formed by the IM&P system.
- **Buddy** – In instant messaging terminology, a "buddy" identifies a remote user with whom relationship has been agreed to enable users to exchange instant messages.

- **Buddy list or buddy roster** – In instant messaging, the buddy list identifies the set of buddies to which a user is connected.

In the *Krox* system architecture the IM&P user metaphor is adapted such that each HAN corresponds to a *user* in the IM&P system. To be accurate, if there is more than one network within the household, each could run a separate instance of the *Krox* system. In order for two *Krox* system instances to be able to share devices and services with each other, a buddy relation between the HANs must be established first in the IM&P network. Therefore a remote HAN is referred to as a *buddy* of the local HAN if the two HANs agreed to share devices. This does not indicate anything about the level of sharing that is derived from their sharing policies, but only that the messaging required for informing each other about shared resources is allowed. Similarly the *buddy roster* of a *Krox* system instance indicates all the remote HANs with which devices can be shared.

4.1.2.3. IM&P as a communication substrate

In the *Krox* system architecture, IM&P is used as the communication substrate for control messaging between the HANs. Each HAN has an IM&P identifier and is connected to a single IM&P server. If there are multiple network segments within the same household, each network segment will need to run its own instance of the *Krox* system with its own identifier for the IM&P server. As discussed in the previous section, HANs establish trusted communication channel by adding the remote HAN's *Krox* system identifier as a buddy. Once the buddy relationship has been agreed between the home users, a communication channel is established between the HANs, and they can start exchanging messages. An advantage of IM&P for the communication subsystem is the ability to use presence features indications for initiating or terminating communication with remote HANs. The presence feature of IM&P system provides a just-in-time notification of changes in the availability of remote users in the buddy list, without requiring any additional protocol; therefore it is appropriate for this purpose.

In the *Krox* system architecture, each HAN endpoints connects to a remote IM&P server that can be operated by a service provider, or deployed in one of the HANs, as long as it is accessible from all participating HANs. The IM&P server can be a standard public domain IM&P server and does not require any modifications. Having each HAN endpoint

implement an IM&P client fulfils three separate but complementary functions in this architecture:

1) Enable remote *Krox* system instances to exchange messages – The instant messaging capabilities are used for delivering messages between local and remote HANs that are configured as buddies.
2) Enable the local system to be notified on changes in the availability of remote HANs from its buddy list – Presence indications are used for detecting remote HANs' status and controlling local HAN's sharing status (e.g. by going offline, or busy).
3) Enable home users to administer their buddy lists in a familiar manner and control their sharing status – Buddy relations in IM&P are used to establish trusted communication channels between remote home networks.

4.1.3. Automatic resource virtualisation

The vision of pervasive home sharing of devices/service requires that services from remote HANs can be seamlessly made available in the local HAN without requiring any modification in network, device or service protocols. The ultimate goal is to enable seamless integration between existing applications and remote services that are made available in the local HAN. In order to support this behaviour it is suggested to use the concept of automatic resource virtualisation. As discussed in the previous chapter, several variations of this concept have been suggested in [51, 76, 119, 139]. Automatic device or service virtualisation means that remote devices or services can be automatically represented in the local HAN as virtual resources (devices or services) presenting a virtual interface in the remote HAN, i.e. be seamlessly discovered and interacted with by clients of the same service protocol running in the same HAN. The virtual resource can communicate with the "live" device through a communication infrastructure. The automation aspect indicates that once a resource is discovered, a virtual resource instance is automatically generated in remote HANs with which it is shared, that can represent the shared resource in these remote HANs without requiring additional configuration or coding. The virtual resource is made available only to networks with which the service or device is explicitly shared. The importance of virtualisation is in enabling virtual resources to be discovered and used in HANs in a manner identical to physical devices and services. With this approach existing clients can use local and remote services seamlessly without being aware of the location of

the service they are consuming, therefore they do not require modifications to be able to interact with remote service.

4.2. High-level architecture

The high-level system architecture depicted in figure 2 defines the proposed integrated architecture. The architecture contains several components:

- **Communication subsystem** - Provides the required secured communication channel and the presence and messaging infrastructure. The communication subsystem is described in section 4.2.1.
- **Plug-in framework** – Service protocol specific plug-ins encapsulate the extensions for a service protocol for multi-HAN and for service orchestration. The plug-in framework defines the requirements from a plug-in, an extensible event model, and

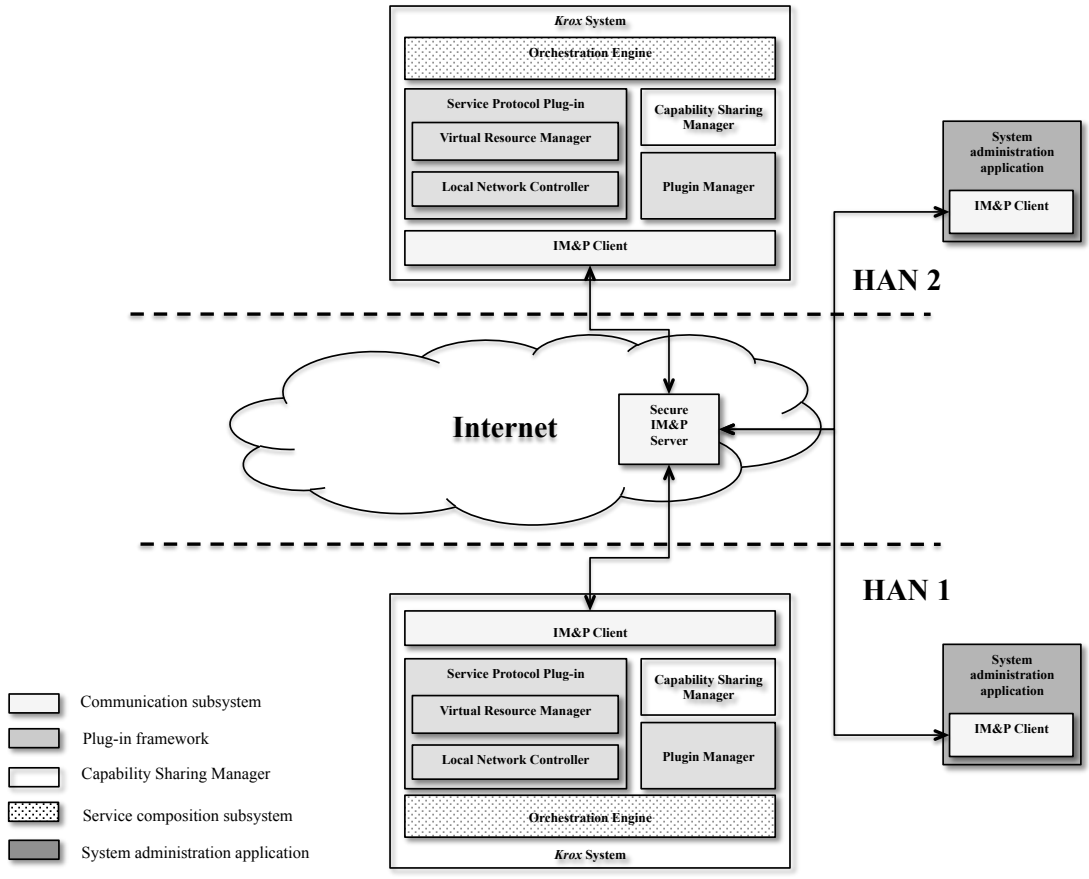

Figure 2 *Krox* System High Level Architecture

the infrastructure required to support and manage service protocol plug-ins. The plug-in framework is described in section 4.2.2.

- **Capability Sharing Manager** (CSM) – Responsible for controlling resource sharing. A capability defines a resource that can be shared. It can be a device, a service, content, or some grouping of such. The CSM is described in section 4.2.3.
- **Service composition subsystem** – Composing functionality that is made available by services in the local HAN – these could be either local services or remote services represented by local virtual resources. Service composition subsystem is described in section 4.2.4.
- **System administration application** – Used by the home user to administer the buddy list and sharing status and other aspects of system configuration. System administration is described in 4.2.5.

4.2.1. Communication subsystem

The role of the communication subsystem is twofold:

(i) To enable the *Krox* system instances in remote HANs to establish a trusted communication channel, such that they can securely exchange messages, and receive notifications on status changes of each other.

(ii) To enable components of a service protocol plug-in on a single HAN to exchange messages with components of a similar service protocol plug-in in remote HANs with which a trusted communication channel has been established, for example, the UPnP plug-in in HAN A, can send messages to the UPnP plug-in HAN B, if HAN A and HAN B agreed to share devices.

The communication subsystem architecture leverages concepts from IM&P for communication between multiple HANs and is based on an application layer protocol providing a scalable and secure overlay network connecting heterogeneous network infrastructure.

Using an IM&P based communication paradigm for interaction between instances of the system in multiple networks can benefit from various aspects of IM&P communication:

1. User abstraction – Simplicity is an important aspect for any system that is installed in the HAN. IM&P provides a useful abstraction of remote HANs as IM&P "users". This eliminates the need for home users to know about IP addresses or to exchange trusted keys. Additionally, many home users are experienced with using IM systems and are familiar with the concepts of buddy rosters, which can simplify the administration of the system.
2. Integration with presence – IM&P has a inbuilt support for presence notification about end points. The plug-in framework can use this presence mechanism for receiving notifications of remote HAN presence status changes that can trigger the initiation or termination of the communication with that HAN. In the context of *Krox* system in the HAN, presence indicates the willing of the home user to participate in sharing with remote HANs with which sharing has been agreed. Once signed in, the system is always on, unless intentionally changed. This means presences represents the sharing status of remote HAN rather than physical presence of the home user.
3. Standard communication interfaces – At minimum, IM&P systems enable end points to exchange messages. In *Krox* system architecture, this feature can be used for exchanging messages between instances of the *Krox* system deployed in remote HANs.
4. Embedded security – IM&P systems typically support at least authentication and encryption, which are important requirements for the *Krox* system architecture.
5. NAT traversal – IM&P systems typically provide standard mechanisms for NAT traversal, which is required for *Krox* system architecture.

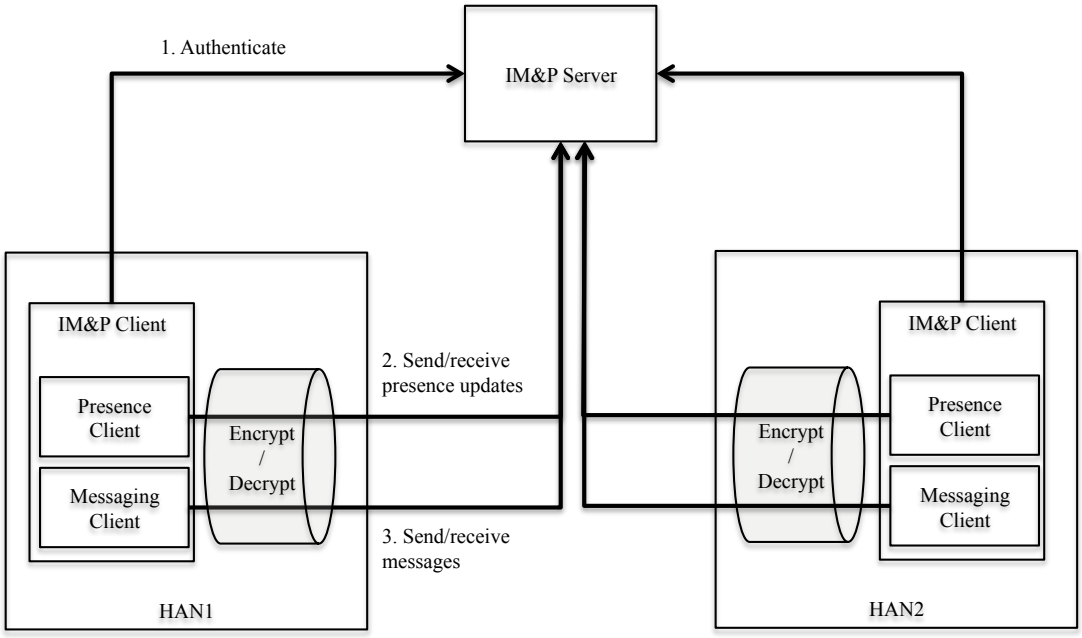

Figure 3 *Krox* Communication Subsystem

6. Support for unicast and multicast messaging models – IM&P servers can support both unicast and multicast messaging that can be useful for different communication paradigms.

IM&P systems also have a number of disadvantages that need to be considered:

1. Text based messaging – There are different types of communication interfaces supported by IM&P systems. The drawback of text based messaging for the communication in *Krox* system architecture is both its verbosity, and the limitation to message payload that can be expressed with textual representation. However, modern IM&P systems support other means of communication such as byte streams that can support any type of data being transferred.
2. Requires a server – IM&P architecture typically requires a server for communication between end points. While server-less architecture exist, they are less common than server based architectures.

As shown in figure 3, the communication subsystem contains an IM&P client that connects to a single IM&P server. There are no restrictions about where the IM&P server is deployed as long as connectivity exists between the server and all its clients in both directions. A service provider typically hosts the IM&P server, however it can be hosted inside the HAN

as long as it is accessible to other HANs using it. However, the IM&P network is not necessarily limited to a single server and can refer to a set of interconnected servers. The IM&P client is a software module that manages the connectivity with the server and enables its users to exchange messages with each other and be notified on presence status changes of other IM&P users from the buddy roster. Once the IM&P client connects to the IM&P server, it starts sending presence messages to the server such that the server can notify "buddies" of that HAN (i.e. *Krox* system instances in remote HANs with which sharing has been agreed) that the local HAN is ready to start sharing. When remote *Krox* system instances send messages to each other the messages go to the IM&P server and through the IM&P server network to the target client. The communication subsystem needs to support authentication to verify that only authenticated users can send or receive messages. Therefore the first step during the bootstrap of the *Krox* system is an authentication of the IM&P client with the IM&P server. Another aspect of security is encryption. The communication between the client and the server should support encryption of the sent messages such that anyone sniffing the traffic between the remote HANs, or between IM&P servers cannot intercept and use the information sent in a malicious manner. Typically IM&P systems support encryption at the network level, however even if not, then the communication subsystem needs to implement encryption and decryption before using the IM&P for sending messages, and after receiving a message.

4.2.2. Plug-in framework

Plug-ins encapsulate the support for a specific service protocol for service sharing and service composition. The plug-in framework defines the roles of plug-ins, their abstract interface, their interaction with the rest of *Krox* system architecture, and how they are managed. This section describes how the plug-in framework is used to achieve seamless integration between services from remote HANs and client applications in the local HAN. The service composition aspects of service protocol plug-ins are discussed in section 4.2.4.

The plug-in requires a parsable service interface that can lend itself to virtualisation, and can enable automatic generation of corresponding virtual resources. Therefore it is suitable only to those HAN service protocols with such a service interface – i.e. UPnP, DPWS, Jini, HAVi, and OSGi, and not suitable to ZeroConf, and SLP.

In order to achieve plug and play behaviour and seamless integration it is necessary that remote devices be represented in the local HAN in an identical way to local devices of the same service protocol. This can enable existing applications running in the local HAN to interact with remote devices and services in an identical fashion to the way they interact with local devices and services, thereby achieving the desired seamless integration. To support this with multiple service protocols, the open plug-in based architecture is used. Each service protocol plug-in provides service protocol specific resource virtualisation functions for achieving seamless integration and inter-HAN service interoperability. Each service protocol plug-in has two roles for resource virtualisation:

(i) Interact with local HAN resources (devices and services) that support the service protocol

(ii) Represent remote resource (devices and services) in the local HAN

The service protocol plug-in is responsible for discovering resources of the relevant service protocol in the local HAN and reporting them to all remote HANs with which this resource is shared. The communication subsystem provides the plug-in with a secure and efficient messaging infrastructure, and presence notifications indicating status changes from *Krox* systems in remote HANs. The Capability Sharing Manager provides an interface to the plug-in for checking sharing configuration of HAN resources with specific remote HANs, and enable plug-in components to listen to changes in the sharing policies. The resource virtualisation architecture does not define a concrete interface for service protocol plug-ins because their diversity implies that this can be over-restrictive. The plug-in framework defines an extensible event model to which the plug-in is required to respond and provides a mechanism for plug-ins to exchange additional messages – see 4.2.2.2 and 4.2.2.3 for a specification of the core event model.

Resource virtualisation results in a distributed system with "live" devices/services and virtual resources interacting through the framework via an IM&P communication system. A service protocol plug-in is comprised of two modules that represent the way a service protocol is supported through the virtualisation framework: a **Local Network Controller (LNC)** and a **Virtual Resource Manager (VRM)**. The plug-in framework is limited to defining the conceptual roles of the LNC and VRM and a minimal event model that must be supported by them, however the service protocol specific plug-in is required to extend this event model for its support for resource virtualisation.

The LNC is responsible for all interaction with the devices and services that support a specific service protocol in the local HAN. Typically this support includes discovery and invocation of actions, but can also include support for event subscription, retrieval of service description and others. The VRM is responsible for representing devices and services of a specific service protocol from remote HAN at the local HAN. When prompted by a remote HAN, the service protocol plug-in automatically generates a virtual instance of the remote resource that can be interacted with in the local HAN without requiring any additional steps.

The following sections describe the components of the plug-in framework, and their interaction.

4.2.2.1. Plug-in Manager

The Plug-in Manager is responsible for management of the life cycle of service protocol plug-ins, including their load, start, stop, and unload. When the *Krox* system is started, the Plug-in Manager loads and initialises the available plug-ins, and if any service protocol plug-ins are deployed during the runtime of the system, the Plug-in Manager is responsible for their loading and initialisation.

In order to support different plug-ins loaded in different HANs, when a remote HAN from the buddy list comes online, the local *Krox* system should send the list of plug-ins it supports to that *Krox* system instance in that remote HAN. The remote HAN should respond with the list of plug-ins it supports. This information should be used such that a remote HAN from the buddy list will be notified only on service protocols it supports. When a plug-in is added to the remote HAN it notifies all of its buddies. When a notification on added plug-in is received to the *Krox* system in the local HAN, if the same plug-in is loaded locally, the message is treated as an indication of the *Krox* system in the remote HAN coming online for this specific service protocol and the local plug-in is notified so that interaction can be initiated.

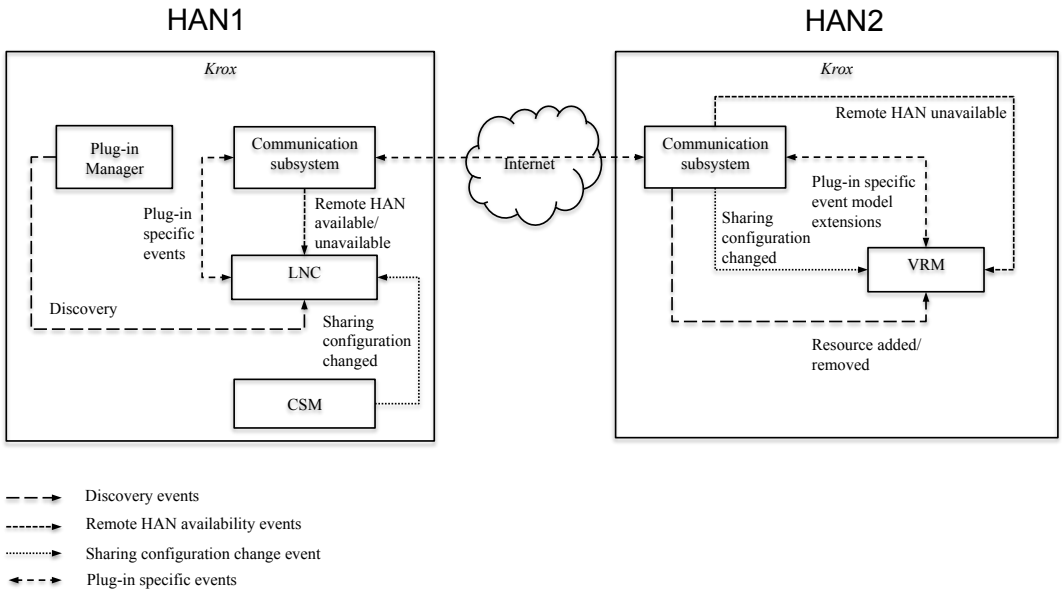

Figure 4 *Krox* Plug-in Event Model

4.2.2.2. Local Network Controller

The Local Network Controller (LNC) for each service protocol plug-in is responsible for all interaction with resources of the specific service protocol supported by the plug-in in the local HAN. The LNC can notify remote HANs, with which sharing has been agreed, about resources added or removed from the local HAN that are shared with these HANs via the communication subsystem. The LNC receives status change notifications from the communication subsystem when a *Krox* system in a remote HAN with which sharing has been agreed, comes online. This information is used by the LNC to initiate the communication with the remote HAN, e.g. by sending discovery information about existing resources in the local HAN that are shared with the remote HAN. The Capability Sharing Manager is consulted before the LNC notifies a remote HAN about an added or removed resource.

The plugin framework defines a core event model with events (described below) that must be supported by an LNC for a service protocol. In addition, a service protocol plug-in may extend the core model with additional events. Figure 4 illustrates the event interaction between the different components of the *Krox* system. For clarity, in each HAN only one plug-in component is shown to illustrate that the interaction is made between the local LNC and the remote VRM. In general the core model contains two types of events, discovery

events and remote HAN state change events. In addition, the protocol specific local LNC and remote VRM can extend the event model for their interaction.

LNC core event model

- **Discovery** - The heart of the LNC is the discovery of local HAN resources. When a plug-in is loaded, the Plug-in Manager triggers the LNC discovery event. The discovery event is only triggered once in the plug-in life cycle when the *Krox* system is started, however the plug-in may invoke it internally in order to keep in sync with the local HAN. To remain as generic as possible, the plug-in framework does not define the type or content of messages that need to be exchanged for reporting added or removed resources between networks. This message exchange is unique to the specifics of the service protocol supported by the plug-in.
- **Remote HAN available** - When a remote HAN with which sharing has been agreed (i.e. is a member of the buddy roster of the local HAN's *Krox* system) comes online, the LNC in the local HAN is called by the communication subsystem and is notified of the change. The availability of the remote HAN is detected by the communication subsystem presence mechanism of its underlying IM&P system, however this is transparent for the plug-in. The LNC in the local HAN is required to retrieve information about all its local HAN devices and services of the service protocol it supports that are shared with that HAN, and send this information using the communication subsystem to the remote HAN. The information describing each resource should be concise however sufficient to enable the remote HAN's VRM to advertise an instance of the virtual resource to its local HAN (as required by the specific service protocol). It is not mandated that each LNC maintains a cache of local devices and service, however it is highly recommended for achieving better performance.
- **Remote HAN unavailable** – When a remote HAN with which sharing has been agreed (i.e. is a member of the buddy roster of the local HAN's *Krox* system) goes offline, as indicated by the communication subsystem, the local LNC is notified by the communication subsystem. The event should be used for clean-up of local resources related to the remote HAN, such as remote subscriptions, and local mappings.
- **Sharing configuration changed (add/remove) –** The LNC must respond to dynamic changes in the configuration of sharing. The LNC must provide a listener

that is registered by the plugin manager with the Capability Sharing Manager during the plug-in bootstrap in order to be notified on changes in the sharing configuration of local resources. When sharing is permitted, additional resources may need to be sent to remote HANs. When permission to share is revoked, the LNC needs to notify remote HANs that the resource is no longer available and possibly clean local subscriptions made on behalf of remote HANs to those resources.

LNC event model extensions

The core event model can be extended by a service protocol plug-in for specifying the interaction between the plug-in components. The interaction between plug-in components of the service protocol plug-in in remote HANs (i.e. LNC and VRM) is made through messages. The LNC for a certain service protocol supports a set of message types that are agreed between the LNC and the VRM at design time of this plug-in. For example, the VRM can send a message for getting the service description. The LNC needs to understand the message body and be able to respond accordingly. The LNC needs to register the set of supported message types with the framework when it is loaded, so that the framework will know to forward these messages to the LNC. This is done by the LNC declaring the supported message types during the bootstrap process, whereby these message types are registered with the communication subsystem, such that received messages having this type are delegated to the LNC. To remain generic enough the plug-in framework does not define the other events that need to be supported by the LNC, therefore extensions of the event model are left for the service protocol plug-in design.

4.2.2.3. Virtual Resource Manager

The Virtual Resource Manager (VRM) represents remote resources (e.g. devices, services) in the local HAN. Given information from a remote HAN on a resource that is shared with the local HAN, the VRM creates a virtual instance of a device or service in the local HAN to represent the remote physical resource. From the point of view of applications in the local HAN, the virtual resource appears as an ordinary local device or service. The VRM uses the communication subsystem to relay messages and requests it receives from local applications to the remote HAN hosting the “live” device where the interaction with the device is facilitated by the corresponding remote LNC. When the VRM is capable of responding

immediately without communicating with the remote device it is encouraged to do so. This can be done for example by caching information – e.g. device descriptions.

The plugin framework defines a core event model with events (described below) that must be supported by a VRM for a service protocol. In addition, a service protocol plug-in may extend the core model with additional events.

VRM core event model

- **Remote resource added/removed** - The VRM must respond to resource discovery events when resources are added or removed in remote HAN that are shared with the local HAN. As part of the communication between the VRM and the LNC, the LNC of a remote HAN sends a message to the VRM of all of the remote HANs with which the resource is shared indicating the added or removed resource. When a notification is received from a remote HAN indicating an added remote device, the local HAN's VRM is responsible for using this information to announce a corresponding virtual resource in the local HAN to represent the remote resource. The virtualised resource instance must be able to respond to the relevant service protocol requests. The extent of the service protocol support is left to the design of the service specific plug-in. When a notification is received from a remote HAN's LNC indicating a removed resource, the VRM should respond by terminating the corresponding virtual resource's instance in the local HAN.
- **Remote HAN unavailable** - The VRM must respond to the event of a remote HAN changing its status to unavailable as indicated by the communication subsystem. When a remote HAN with which sharing has been agreed (i.e. is a member of the buddy roster of the local HAN's *Krox* system) changes its status to unavailable the virtual resource instances representing resources originating in that unavailable HAN must be terminated. All other local mappings in the VRM related to the unavailable remote HAN must be cleaned.
- **Sharing configuration changed (add/remove)** – The VRM must respond to the event of a change in the sharing configuration of a remote HAN that shares resources with its local HAN. This enables the VRM to clean relevant caches if needed.

VRM event model extensions

The VRM can use the communication subsystem to exchange additional types of messages with remote LNC, hosting "live" devices. The types of the messages that the VRM may accept must be registered with the communication subsystem, such that the communication subsystem can dispatch messages to the correct plug-in component. Similarly to the LNC, the VRM must declare the message types it supports. As part of the loading of the plug-in, the Plug-in Manager queries these message types and registers them with the communication subsystem, which then knows to forward received messages with this type to the VRM.

4.2.2.4. Summary

The plug-in framework supports remote representation of local devices through message exchange between remote HANs using the communication subsystem. The plug-in modules respond to changes in the status of remote HANs such that when a remote HAN comes online, the LNC component responds with plug-in specific messages to the corresponding remote HAN with discovery information for all the resources shared with this HAN. When a remote HAN changes its status to unavailable the VRM responds by terminating and removing all of the local virtual resources that correspond to resources from this remote HAN. Plug-in specific messages are exchanged between the LNC and VRM in different HANs for exchanging service protocol specific information related to discovery, invocation, eventing for supporting sharing of resources of the service protocol.

4.2.3. Capability sharing management subsystem

HAN service sharing requires fine-grained access control that allows the home user to grant/delegate access to certain devices/services/actions/content to a subset of its peer HANs. Access right management must be dynamic in order to accommodate the dynamic HAN environment, where remote HANs can be added and removed, and resources can join and leave the network frequently. Sharing management addresses identity management and access control. Supporting fine-grained access control requires modelling of the device/service capabilities. Such models could include descriptions of the capabilities and who may have access to them. In the *Krox* architecture, identity management is facilitated by the logical identities provided by the IM&P framework (i.e. unique IM&P usernames for each HAN).

In *Krox* system architecture the Capability Sharing Manager (CSM) is populated with information about the local HAN resources and the remote HANs with which sharing has been agreed. The CSM provides plug-ins with an interface through which they can query if a resource should be shared with a specific remote HAN or with which remote HANs a resource is shared. In addition, before a resource is used the CSM is queried to verify that the remote HAN trying to access the resource is authorised to do so. Finally the CSM provides notifications to registered listeners when the sharing policies are changed, so that the plug-in components can respond. As mentioned above (section 4.2.2.2), the LNC must provide a listener that is registered by the plug-in manager with the CSM for receiving update notifications when sharing policies change. The CSM design is described in section 4.3.2.

4.2.4. Service composition subsystem

The service composition subsystem is based on using *web services* as an interoperable and composable interface, and using an orchestration engine for their composition. Mapping between the service protocol and web services should be handled by the service protocol plug-in. Using web services as an interoperable service interface has a number of advantages and disadvantages. Web services are based on a service oriented architecture realisation based on open standards and their interoperability. The main advantage of using web service for representing HAN resources is their interoperability. The main drawback is that they may incur performance overhead in both memory as well as in CPU processing. An additional drawback to using web services is the potential loss of information during the mapping process due to the generality of the interface as opposed to the service protocol specific capability description model. Web services require a web server for hosting, which can also require significant amount of memory.

Aiello [2] argued that web services would have an important role in the future of the HAN as the key enabler for total interoperability between devices and services. While for the more powerful devices it suggests that the web services stack would be offered as part of the device, less powerful devices could connect to a controller implementing the web service stack for them. This approach is complementary to the *Krox* system architecture such that if devices offered web services, there would be no need for a transformation between the

service protocol and web services. Perumal et al. argue that SOAP and web services maximise the interoperability of heterogeneous resources and systems with satisfactory performance for smart home environment [102].

When designing the architecture for home service composition it is necessary to consider the relevant challenges of home networking: dynamicity (devices can dynamically join and leave the network), heterogeneity (variety of hardware, software and protocols) and distribution (devices can be located anywhere in the home). Heterogeneity is handled by using web services as a common interoperable interface. The discovery module of the LNC and VRM handles dynamicity and distribution. With the plug-in based architecture described in previous sections, the transformation from a service protocol to the representation as a web service is required as part of the service protocol plug-in implementation. The transformation takes place in both the LNC as well as the VRM. The LNC discovers devices and services in the local HAN, once a device/service is discovered it is required to generate a local web service for it. Similarly the VRM is responsible for generating web services for the corresponding resources in the remote HANs that it represents. The *Krox* system architecture does not define how the generation of web service for the specific service protocol should be made, however, given the service protocol has a self describing parsable service interface, as required by *Krox* system architecture, it should be possible to generate an automatic mapping from the protocol specific service interface to the web service interface. Once the web service corresponding a local live or virtual device/service has been generated it is deployed to a local web server. Similarly when it is no longer available in the local HAN (as a physical or virtual device/service) it is undeployed from the local web server.

4.2.4.1. HAN service orchestration

With HAN services mapped into web services, the *Krox* system architecture suggests service orchestration as an approach for constructing complex HAN services from existing simple services either live or virtual. A number of approaches for home service composition were described in section 3.2.3 and are summarised in table 2. The *Krox* system architecture extends the state of the art for service orchestration in supporting the orchestration of services from both the local HAN and remote HAN seamlessly. Through the automatic mapping of service protocols to web services by the corresponding plug-in components (LNC for local HAN services, VRM for remote HAN services), local web service proxies

(corresponding to either physical local or virtual remote services) are available in the local HAN for orchestration.

The *Krox* system architecture does not define the process of service orchestration for HAN services, but only provides a mechanism for making services available for service orchestration and a mechanism for executing service composition.

4.2.5. System administration application

As part of the *Krox* system architecture, an administration client application is required for controlling the various aspects of system management and monitoring:

- User management – Is required for adding and removing remote HANs for sharing, suspending and resuming relationship with remote buddies. This is done using an IM&P client, where nodes correspond to remote HANs. Changing the status of the *Krox* system enables pausing or resuming sharing with remote HANs.
- Sharing management – Is required to enable the configuration of sharing policy that defines which resources are shared with which remote HAN. For each remote HAN, identified by its *Krox* system identifier (i.e. its IM&P identifier) the home user can select which resources to share.
- Plug-in management – Is required to enable the user to control which plug-ins are currently installed and running, and to start and stop installed plug-ins.

4.2.6. Deployment considerations

The system could benefit from being deployed as part of the home gateway, however this is not mandated. The gateway has a number of advantages as a platform for running the *Krox* system: firstly it is available in most HANs and therefore it does not require an additional device to be introduced to the HAN. Additionally the home gateway is always connected, which is useful for the *Krox* communication subsystem. The drawback of deployment on the home gateway is the relatively resource constrained nature of home gateway platforms. Another potential drawback with deploying *Krox* system as part of the home gateway is the security risk – if *Krox* system is compromised, an attacker may gain access to configuration of the home gateway which is undesired. An alternative deployment option is as an application on the home user's desktop. The advantage of this approach is the much greater

computing power of the desktop computer compared to the home gateway. Modern multi-CPU and multi-core desktop PCs can provide much improved performance. The disadvantage of deployment on a desktop is that such an application is less likely to be always-connected. Another device that can host the *Krox* system is a Network-Attached Storage (NAS) device. Similarly to the home gateway, a NAS device would typically be always connected, however such devices are not as pervasive as home gateway devices. Finally the *Krox* system could be offered as an appliance for the HAN. The advantage of such an approach is that the appliance could be equipped with processing power that matches its requirements. The drawback is that it requires an additional appliance, which also implies additional cost for the home user. From an architecture point of view, none of these options should be excluded.

4.3. System design

The previous section presented the high level architecture for *Krox* system and its support for inter-HAN and intra-HAN service interoperability. In order to demonstrate the utility and applicability of this high-level architecture, in this section a corresponding system design is presented. The *Krox* system design (depicted in figure 5) makes two design decisions with regard to the high level architecture:

1) XMPP [114] as an IM&P system – XMPP provides a standard (RFC3920) secure, decentralised, and extensible implementation for IM&P. XMPP has many advantages, which make it an adequate choice for the communication subsystem of *Krox* system architecture as discussed in section 4.3.1.
2) Business Process Execution Language (BPEL) [5, 69] for HAN service orchestration – As discussed in section 2.2.2, BPEL is a standard language for expressing web service orchestrations. BPEL has several advantages for using in HAN and a few disadvantages. BPEL service orchestration can enable composition of HAN services, both local and remote (through their web service proxies), and additionally composition with external web services. In addition, BPEL renders the service orchestration as a web service itself, which enables its reusability for further composition. The main disadvantage of BPEL for the purpose of HAN service composition is that it may be too heavyweight in terms of its memory and CPU prerequisites. Another potential disadvantage of BPEL is that it supports only static service composition, where service binding is done at design time rather than in

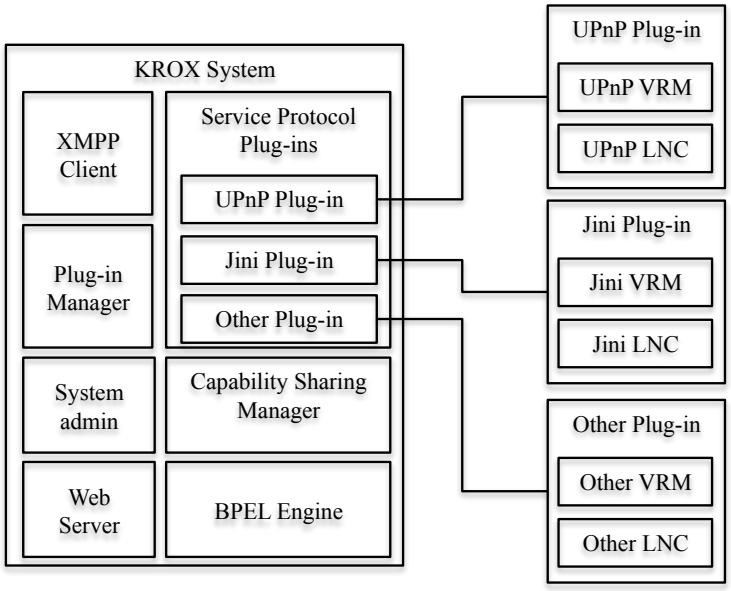

Figure 5 *Krox* System Design

runtime, which may be more appropriate for the HAN. These advantages and disadvantages are discussed in section 4.3.5.

To demonstrate the utility of the plug-in framework, which is part of the key contribution of this thesis, two service protocol plug-ins were designed for supporting sharing and composing of UPnP and Jini service protocols.

The following sections describe the details of the *Krox* system design including the design for the system components as they were described in the previous section and the UPnP and Jini service protocol plug-ins.

4.3.1. Communication subsystem

The high level architecture presented in the previous sections outlined the use of IM&P as a communication mechanism between remote HANs. In the system design, eXtensible Messaging and Presence Protocol (XMPP) [114] is suggested as the IM&P architecture. The following section describes XMPP and its applicability for the communication subsystem of *Krox* system architecture.

4.3.1.1. eXtensible Messaging and Presence Protocol

XMPP was initially designed as the underlying protocol for the popular instance messaging Jabber[20]. Jabber suggested an extensible protocol built on XML that allowed development of applications not only for its original purpose for IM but also for general purpose transport layer for distributed applications. The Internet Engineering Task Force (IETF) standardised the core protocol of XMPP to RFC3920 [114]. RFC3920 specifies how XML streaming protocol enables entities to exchange XML elements over the network. XMPP supports authentication and encryption in the streaming layer through Simple Authentication and Security Layer (SASL) [80] and Transport Layer Security (TLS) [109]. In the classical XMPP architecture clients communicate with each other through a server. Servers can also communicate with each other allowing connections of multiple cooperating domains. Each entity in XMPP has a unique address called JID in the following format: <*node*>@<*domain*>/<*resource*>. This approach enables the user to have multiple connections for the same user with the resource denoting a specific resource or location – e.g. laptop, desktop, home, work. Communication between XMPP clients is made through XML streams, which are envelopes containing XML stanzas (atomic unit of information). The core of XMPP is defined using 3 XML stanza types: message, presence, and info/query.

- Message XML stanzas enable an entity to send information asynchronously to another entity. Typically delivery is made in real time but there is also support for store-and-deliver later.
- Presence XML stanza is used for representing the availability status of the entity, e.g. online, busy, away. Presence stanzas are exchanged in a publish-subscribe manner such that entities can subscribe and be notified about changes in the presence status of another entity.
- Info/Query (IQ) stanza can be used by entities to make requests and receive responses from each other. IQ supports 4 types of request/responses: get, result, set, error. The different request and response are identified using an identifier per operation

The key element provided by the core stanza types is the near-real-time delivery semantics for communications, which is beneficial for the *Krox* communication subsystem to deliver messages between remote HANs with minimal latency. The content of a stanza is pure

[20] http://www.cisco.com/web/about/ac49/ac0/ac1/ac258/JabberInc.html

XML structured data rather than a specific type, thus XMPP enables the exchange of any type of data that can be represented in XML. As stated earlier, XMPP provides a server-based model but where the servers may be decentralised. This means that there is no central authoritative server. From the point of view of users, they still connect to a single server with which they authenticate and connect to the XMPP network. Numerous public domain XMPP servers exist, primarily in support of instant messaging applications but these servers can also be used for other purposes such as for *Krox* system communication. Public XMPP servers are available for everyone, but any user may run their own XMPP server on their own domain. Although peer-to-peer implementations of XMPP exist, the typical architecture of XMPP is a pure client-server model, whereby clients connect to a server and servers connect to other servers for inter-domain communications. Besides instant messaging applications XMPP has been used as communication substrate for various purposes including enabling complex communication between applications in the cloud (project Vertebra[21]), gaming (e.g. Chesspark[22]) and VOIP (e.g. GoogleTalk via the Jingle XMPP extension).

The following section describes how XMPP is used in the *Krox* system design and the advantages of selecting XMPP over other IM&P systems.

4.3.1.2. XMPP in *Krox* communication subsystem

In the *Krox* system design, following the *Krox* system architecture, each *Krox* HAN controller connects to an XMPP network. HANs are identified using an IM&P identifier (JID in XMPP architecture) and agreement to share with a remote HAN is indicated in adding the identifier of the remote HAN to the buddy roster of the local HAN. Buddy roster management and administration is a typical and familiar operation in instant messaging and therefore it can be considered simple enough for non-technical home users.

XMPP presence alerts indicate when XMPP users come online. In the *Krox* system design, a presence change corresponds to a change in the sharing status of the corresponding *Krox* system instance – a *Krox* system instance whose status is “available” indicates that the

[21] http://www.engineyard.com/

[22] http://www.chesspark.com

system is ready to accept notifications about devices and services shared with it. An "unavailable" presence status indicates that the corresponding *Krox* system instance is in non-sharing mode, and all devices and services that were shared from it are no longer available. The home user, as an administrative operation, can trigger the changes in the presence state of a *Krox* system instance, to start/stop/pause sharing of local devices with remote HANs. In addition, when the system is not connected, its presence status is "unavailable". Once the status of a remote HAN (with which sharing is authorised) changes to "available", the local HAN's *Krox* system initiates the interaction with it for sending it information about all the devices and services that are shared with that remote HAN. When the remote HAN changes its presence status to unavailable, all of the virtual resources in the local HAN that were shared from the remote HAN need to be destructed.

The communication subsystem uses XMPP message stanzas to wrap control messaging between *Krox* system instances in remote HANs, e.g. service discovery announcements, description requests, invocations, and results.

Several attributes of XMPP mentioned above make it an adequate for using as an IM&P system underlying the *Krox* communication subsystem design:

1. Scalability – XMPP has shown to exhibit scalability in instant messaging with world scale deployments and with large numbers of users and high volumes of traffic (REQ #16, REQ #17)
2. Security – With TLS and SASL, XMPP provides secure messaging infrastructure including authentication and encryption (REQ #12, REQ #14)
3. Extensibility – XMPP is open for extensions, which can be utilised for using a protocol extension instead of instant messaging for the message exchange between the LNC and the VRM
4. Presence – XMPP presence information can be used to trigger communication between remote HANs and enable prompt response to the event of HANs coming online or going offline (REQ #22)
5. Ease of administration for users – With XMPP all home users need is the identifier of the remote HAN they want to add as a sharing buddy. From user management point of view, XMPP provides a familiar and easy to use model that has been extensively adopted by users for instant messaging. (REQ #21, REQ #23)
6. NAT – Communication between an XMPP client and XMPP server is TCP-based

rather than UDP-based and is always initiated by the client, which indicates that the communication subsystem has no problem with NAT (REQ #10).

7. Multiple Point of Presence (MPOP) – XMPP allows a user to be connected with the same address in multiple locations, which are differentiated using a resource identifier. The benefit of this approach is that it can enable the *Krox* server as well as the administration application to connect to the XMPP server using different resources without any conflicts.

4.3.2. Capability sharing manager

The role of the Capability Sharing Manager (CSM) component is to enforce home users sharing policies for HAN services. During the system runtime the CSM is populated with resource sharing/capability models for remote HANs and with local resources. Remote HANs are identified using their IM&P identifier. Whenever sharing with a remote HAN is agreed, the CSM in both HANs need to be updated, such that sharing policies can be applied. Whenever resources are discovered in the local HAN the CSM resource model is updated with information about the resource, including the resource identifier, resource type, and any protocol specific information that can be used by home users in defining sharing policies, e.g. device vendor, device family (e.g. media device). When a service protocol plug-in discovers a resource it needs to query the CSM and check with which other HANs this resource needs to be shared based on the home user's sharing policy. The CSM responds with the set of identifiers of remote HANs. When a remote HAN with which sharing has been agreed changes its status to "available", service protocol plug-ins query the CSM for all local HAN resources that need to be shared with this remote HAN. Service protocol plug-ins are also required to check access permission before they perform an action on a local device/service on behalf of a remote HAN. The CSM only contains local information, i.e. information about devices and services in the local HAN (not including virtual services), and identification of remote HANs with which sharing has been agreed. Finally, the CSM needs to provide notifications to affected components when sharing configuration changes. However since access permission is checked before an action is executed on behalf of remote HAN on a local resource, if access permission was revoked, the action will be denied.

In order to give users enough flexibility, CSM should support fine-grained sharing specification that would allow users to share some services with some users and other

services with other users even for services from the same device. Similarly for content hosted in a media server, users can define sharing policies based on the name of a container or based on tagging. During runtime, the CSM is queried whether specific device/service/action/content should be shared with a given user and replies with positive or negative authorisation based on its internal sharing models. When the sharing policies are changed, the plug-in components are notified on the changes and can respond accordingly.

The actual specification of sharing policies is not part of *Krox* system architecture. In order to be able to specify sharing policies, integration with a policy based capability management system is planned as part of a research project (FAME[23]) but is explored in further work – see section 7.3.2.

4.3.3. UPnP

This section gives a description of the UPnP layered architecture (section 4.3.3.1) followed by a presentation of the corresponding UPnP service protocol plug-in for *Krox* system architecture (section 4.3.3.2).

4.3.3.1. UPnP Architecture

UPnP Device Architecture (UDA) [133] shown in figure 6 defines the layers of plug-and-play communication protocol: addressing, discovery, description, control, eventing, and presentation. These layers are described in the following sections.

4.3.3.1.1. UPnP addressing

Obtaining an IP address is the first step in UPnP communication, before a device can be discovered and interact with control points. UPnP addressing protocol enables devices to automatically obtain an IP address upon joining the local HAN without user administration. The IP address is obtained using Dynamic Host Configuration Protocol (DHCP) [36] if possible. If DHCP is not found then automatic IP addressing (Auto-IP) [30] is used. Auto IP allows the device to select an IP address from the 169.254/16 range and then it uses Address

[23] Federated Autonomic Management of End to End Communication Systems – SFI Strategic Research Cluster ("FAME"): 08/SRC/I1403 (www.fame.ie)

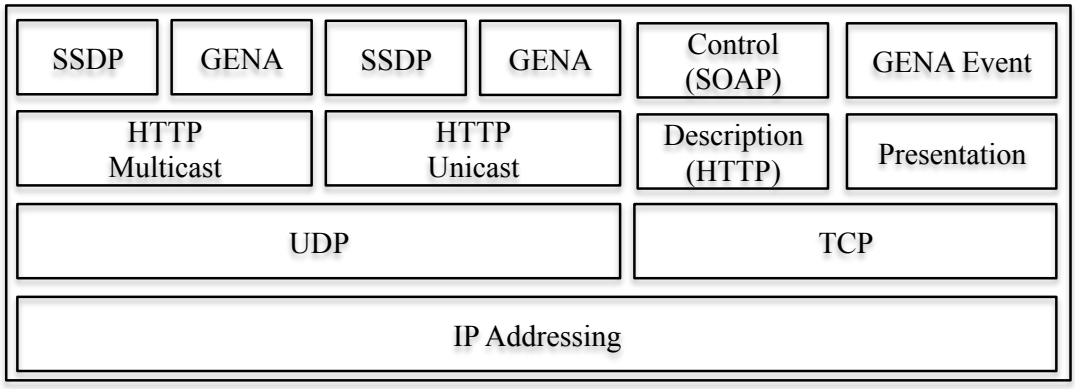

Figure 6 UPnP Architecture Stack

Resolution Protocol (ARP) [103] to check if the address is available. Once a device has an IP address it can take part in the UPnP discovery protocol.

4.3.3.1.2. UPnP discovery

A control point must know about the existence of a UPnP device in the local HAN before it can start interacting with it, therefore a discovery protocol is needed. UPnP discovery is based on Simple Service Discovery Protocol [133]. SSDP supports two complementary modes of operation: advertisement and search. When a device is added to the network it announces its presence, and similarly when a control point is added to the network it should be able to find devices. Search mode enables control points to initiate discovery in the network with a certain service type such that all the devices that support the requested service type must respond with a short message indicating its identification. This mode of operation is useful for applications that join the network and want to find certain type of services already existing in the network or for ad hoc searches, e.g. for showing the list of printers in a certain network. Search requests are HTTP packets sent over UDP to a local multicast address (239.255.255.250 port 1900) in the local HAN. The search request needs

```
NOTIFY * HTTP/1.1
Host: 239.255.255.250:1900
NT: blenderassociation:blender
NTS: ssdp:alive
USN: someunique:idscheme3
LOCATION: http://192.168.1.3/foo/bar
Cache-Control: max-age = 7393
```

Figure 7 UPnP Device Announcement Example

to describe the type of service of interest, such as root devices, or specific device type such as media server. All UPnP devices on that network must listen on the same multicast port and respond if they support the service described by the discovery request message. The response is made via a unicast HTTP POST over UDP packet directly to the requesting control point. When a new device joins the network it must announce its presence by sending an HTTP packet over UDP to the same multicast address, thereby enabling control points in the local HAN to learn about its existence. Control points interested in new devices can listen to the multicast address and thereby learn about new devices joining the local HAN. Devices must also announce when they leave the network. An example of a device announcement as defined in [133] is given in figure 7. The announcement contains basic details about the device such as its service type (blender in the example), its unique identifier (someunique:idscheme3), a URL to its description and expiration time which indicate for how long the information is valid for enabling applications to cache the device information. When a device leaves the network it only needs to send its service type and identifier. The purpose of presence announcements is to optimise the interaction with UPnP devices and to minimise the need for repeated searches that result in more control traffic and enable control points to learn about existing devices in the network as soon as possible.

4.3.3.1.3. UPnP description

UPnP devices and services are described via XML documents. A device description document contains information about the physical device such as its friendly name, manufacturer information, version, icons, and additional information about embedded devices and supported services. The device description (figure 8) document is organised according to a predefined schema that corresponds to the device profile. Once the device has been discovered by a control point, the control point can query the device description by sending an HTTP GET request to the URL specified in the discovery packet location header. The device responds by posting the description XML document in an HTTP response to the requesting control point. The control point needs to parse the XML document and extract more details about the supported services and how to interact with them. The device description may contain additional URLs for services, which can be further queried by control point applications to obtain more information about a specific service. The service description includes information about the actions supported by the service, their parameters and the state variables related to the service. State variables

```
<?xml version="1.0"?>
<root xmlns="urn:schemas-upnp-org:device-1-0">
 <specVersion>
  <major>1</major>
  <minor>0</minor>
 </specVersion>
 <URLBase>base URL for all relative URLs</URLBase>
 <device>
  <deviceType> urn:schemas-upnp-org:device:CDPlayer:1</deviceType>
  <friendlyName>short user-friendly title</friendlyName>
  <manufacturer>manufacturer name</manufacturer>
  <manufacturerURL>URL to manufacturer site</manufacturerURL>
  <modelDescription>long user-friendly title</modelDescription>
  <modelName>model name</modelName>
  <modelNumber>model number</modelNumber>
  <modelURL>URL to model site</modelURL>
  <serialNumber>manufacturer's serial number</serialNumber>
  <UDN>uuid:UUID</UDN>
  <UPC>Universal Product Code</UPC>
  <iconList>
   <icon>
    <mimetype>image/format</mimetype>
    <width>horizontal pixels</width>
    <height>vertical pixels</height>
    <depth>color depth</depth>
    <url>URL to icon</url>
   </icon>
   XML to declare other icons, if any, go here
  </iconList>
  <serviceList>
   <service>
    <serviceType>urn:schemas-upnp-org:service:SwitchPower:1</serviceType>
    <serviceId>urn:upnp-org:serviceId:SwitchPower</serviceId>
    <SCPDURL>URL to service description</SCPDURL>
    <controlURL>URL for control</controlURL>
    <eventSubURL>URL for eventing</eventSubURL>
   </service>
   <service>
    <serviceType>urn:schemas-upnp-org:service:ChangeDisc:1</serviceType>
    <serviceId>urn:upnp-org:serviceId:ChangeDisc</serviceId>
    <SCPDURL>URL to service description</SCPDURL>
    <controlURL>URL for control</controlURL>
    <eventSubURL>URL for eventing</eventSubURL>
   </service>
   <service>
    <serviceType>urn:schemas-upnp-org:service:PlayCD:1</serviceType>
    <serviceId>urn:upnp-org:serviceId:PlayCD</serviceId>
    <SCPDURL>URL to service description</SCPDURL>
    <controlURL>URL for control</controlURL>
    <eventSubURL>URL for eventing</eventSubURL>
   </service>
   <service>
    <serviceType>urn:schemas-upnp-org:service:Audio:1</serviceType>
    <serviceId>urn:upnp-org:serviceId:Audio</serviceId>
    <SCPDURL>URL to service description</SCPDURL>
    <controlURL>URL for control</controlURL>
    <eventSubURL>URL for eventing</eventSubURL>
   </service>
   Declarations for other services added by UPnP vendor (if any) go here
  </serviceList>
  <presentationURL>URL for presentation</presentationURL>
 </device>
</root>
```

Figure 8 UPnP Device Description Example

provide information about the state of the service, for example, a status state variable in a media player can indicate if the media player is playing/stopped/paused.

4.3.3.1.4. UPnP control

The UPnP control protocol specifies a methodology for remote procedure calls, enabling control point applications to invoke actions on UPnP devices in a platform and programming language neutral manner. Invocation is made by sending a SOAP request over HTTP to the relevant service's control URL as obtained from the service description. Once

the execution completes, a result or an error is sent back from the device to the requesting control point. Operations can either query some state or modify the state of the device.

4.3.3.1.5. UPnP eventing

The UPnP eventing protocol enables control points to subscribe for events indicating changes in state variables. UPnP supports two styles of event subscription: unicast subscription, which allows interested control points to subscribe for event updates, and multicast subscription, which enables all listening control points to listen to event updates which are sent to the multicast address in local HAN. Interaction with the device involves subscription for events, renewal of subscriptions, and un-subscription. The interaction is made through HTTP requests and events notifications are based on Generic Event Notification Architecture (GENA) [133] and delivered over HTTP.

4.3.3.1.6. UPnP presentation

The UPnP presentation layer enables control points to retrieve a web page from the presentation URL and present it in a web browser. Where the device supports presentation, a URL is provided in the description XML document to a web page hosted in the device's embedded web server. The web page can support dynamic updates to the device state and can potentially enable client to invoke actions on the device. UPnP presentation layer is optional and the extent of support may vary between device implementations.

4.3.3.1.7. UPnP security

While UPnP does not have an inherent security mechanism, it does offer some form of authentication and authorisation via the DeviceSecurity [131] and SecurityConsole [132] profiles. The UPnP specification suggests that the DeviceSecurity device enforces authentication and access control such that the access control policy is in fact stored in the SecurityConsole which is a single point in the home network that facilitates authorisation. The DeviceSecurity profile attempts to secure only the UPnP control protocol. If an action is defined to be access controlled, then a control point must be authorised to perform this action on the device before the device will accept the execution request. An authorised control point must digitally sign SOAP messages in order to provide integrity protection. The authorisation is made through a control point application that has a device interface as

well (SecurityConsole) that enables the user to configure security settings for devices on the network. DeviceSecurity supports confidentiality by enabling SOAP action messages to be encrypted. If authorised, the device decrypts and executes the request and then encrypts the result when it is sent back to the caller. DeviceSecurity also considers replay prevention by a sequence number for actions that are executed within the context of a secure session. For other actions in a non-session context a state variable value is used to generate a non-repeating value for replay prevention.

4.3.3.2. UPnP service protocol plug-in

This section gives an in-depth description of the design for multi-HAN extension for UPnP through device and service virtualisation, packaged as a service protocol plug-in, which contains two software modules: a Local Network Controller and a Virtual Resource Manager as required by the plug-in framework (see 4.2.2). UPnP discovery relies on UDP multicast – devices announcement are sent to a local multicast address and similarly control point search messages are sent to the same multicast address when such applications intend to find devices in the local HAN. UDP multicast is limited in its scope to a single HAN. In order to be able to deliver discovery notifications to remote HANs with which devices are shared, the communication subsystem is used. However a secure communication channel is not sufficient for extending UPnP across multiple HANs. The reason is UPnP discovery announcements include a device location, which represents a URL where more information about the devices could be retrieved. This location would represent a private IP address that is not reachable from remote HANs therefore repeating the original device announcement in the remote HAN is insufficient for achieving seamless integration with client control point applications.

The UPnP LNC collects information about the local HAN UPnP resources. The UPnP VRM aggregates information about devices and services from remote HANs and automatically generates corresponding virtual UPnP devices and service in its local HAN. It is a design choice to have a single instance of VRM managing all the local representations of UPnP devices from remote HANs, rather than having many VRM instances. The rationale behind this decision is to optimise communication and shared resources, such that there will be only a single listener to local search requests on behalf of remote devices. With a single VRM instance encapsulating all remote UPnP devices in the local HAN, when a search request is received, the VRM is required to respond on behalf of all of the devices and services that

correspond to service type indicated in the search request. When a request is directed to a single UPnP service or device (e.g. service description request), the VRM processes the request in the context of that device. From an external point of view (e.g. a control point), there is no central VRM entity but as many devices as and services as shared with the local HAN. The VRM entity is inaccessible, instead, the VRM masquerades as multiple UPnP devices.

Section 4.3.3.1 described the details of the various layers of UPnP protocol. The following sections describe how the UPnP layered protocol is mapped to the interactions between LNC and VRM for supporting multi-HAN UPnP networks.

4.3.3.2.1. UPnP Discovery

In order to extend the UPnP discovery protocol across multiple HANs, the multicast limitation to a single HAN needs to be circumvented. In this design the communication subsystem messages are used by the UPnP plug-in as envelopes to SSDP notifications between *Krox* system instances in multiple HANs. It must be noted that only HANs with which devices are shared are notified.

The UPnP Local Network Controller (LNC) listens to local SSDP announcements made by local UPnP devices. The LNC uses these announcements to keep an up-to-date repository of all announcements representing current devices/services in the local HAN. Periodic searches are used as a complementary mechanism in order to keep the repository in sync with the network and overcome cases of missed announcements due to network congestion or the unreliable nature of UDP communication (see steps 1-3 in figure 9), however the frequency of such searches is left for the implementation. While such complementary search is essential for guaranteeing that no announcement is missed for a too long time, it must be balanced against the overhead it places on the network devices in the local HAN.

The device and service announcements update the CSM and are cached for the duration indicated in their discovery announcement. When the duration expires they are removed from the local repository and update the CSM of their expiration. The purpose of caching local discovery announcements is to be able to report to a remote HAN about available

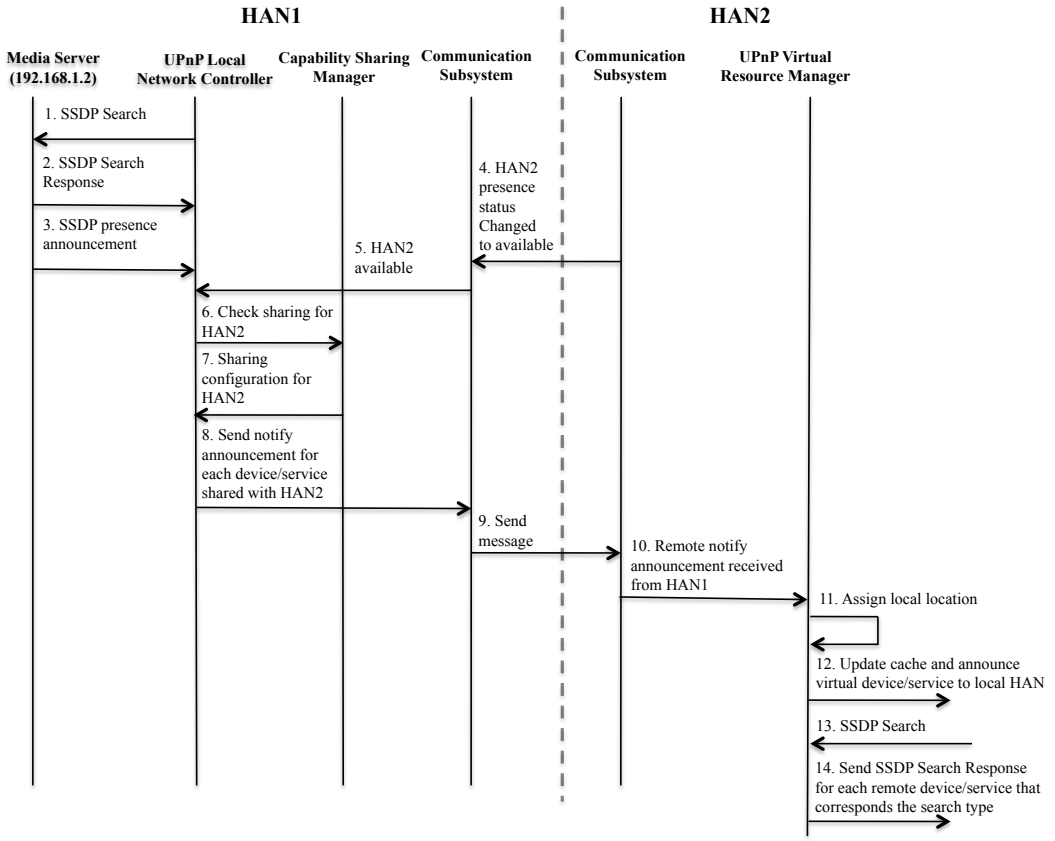

Figure 9 Multi-HAN UPnP Discovery Protocol Interaction

shared devices and services in the local HAN efficiently. When a remote HAN comes online as indicated by the communication subsystem (step 4-5 in figure 9), the UPnP LNC uses the communication subsystem to send an SSDP announcement to the "available" remote HAN for each UPnP device/service that is shared with that remote HAN and whose duration has not expired yet (steps 6-9 in figure 9). This behaviour emulates a search request being sent from the remote HAN to the local HAN. Similarly when a local device or service announcement is received at the LNC, it is sent to all remote HANs with which it is shared. Checking for sharing configuration is made by querying the capability-sharing manager, before any message is sent to the remote HAN.

When an SSDP announcement is received to *Krox* system from a remote HAN (from its UPnP LNC), it is delivered to the local UPnP VRM (step 10 in figure 9). The VRM needs to announce the SSDP device or service announcement in its local HAN. The SSDP announcement that was received from the HAN hosting the "live" device does not contain any location, which is not sent with the announcement for security and conciseness considerations. The reason is that the original location (as can be seen in figure 7) is not

accessible from the remote HAN. Therefore before the received SSDP announcement can be repeated in the receiving network, its location must be set to a locally meaningful one, i.e. a location that can be accessed later when a control point tries to retrieve the device description or subscribe for events (step 11 in figure 9). The UPnP VRM needs to assign the device/service a meaningful unique local address where control points can interact with it. The addressing scheme for device location is described in the next section. Finally, once the local location has been set, the VRM updates its local announcements cache and announces the device/service locally (step 12 in figure 9). The VRM maintains this cache so that it would be able to respond promptly to search requests made by applications in its local HAN on behalf of the remote devices and services it represents (steps 13-14 in figure 9). This caching eliminates the need to ever delegate search requests between multiple HANs and therefore reduces the control traffic on the expense of additional memory.

When a *byebye* announcement is received from a remote HAN for a device or service that is shared with the local HAN, the corresponding SSDP announcement is removed from the VRM local repository and the *byebye* announcement is repeated in the VRM's local HAN. Similarly when the duration of an *alive* device or service announcement received from a remote HAN to the VRM expires, the SSDP announcement is also removed from the VRM's repository.

When the UPnP LNC sharing configuration change listener receives a notification from the CSM on a change in the sharing configuration of a resource with relation to a set of remote HANs, the LNC must respond by sending the corresponding announcements to the relevant HANs. If sharing is added, then the device announcement should be sent, if sharing is removed, a *byebye* announcement should be sent to the relevant HANs. In addition a message should be sent to the remote HANs indicating the change in the sharing configuration, enabling them to clean related caches.

It is mentioned above that the LNC listens to discovery messages in the local HAN. A subtle point is that the LNC must ignore and discard announcements that represent virtual devices originated in the local VRM. Therefore when an SSDP announcement is received by the LNC it must verify first that it does not represent a remote device before it attempts to forward it to remote HANs. A remote announcement can be detected by examining the

```
NOTIFY * HTTP/1.1
Host: 239.255.255.250:1900
NT: blenderassociation:blender
NTS: ssdp:alive
USN: someunique:idscheme3
LOCATION: http://192.168.1.3/foo/bar
Cache-Control: max-age = 7393
```

```
NOTIFY * HTTP/1.1
Host: 239.255.255.250:1900
NT: blenderassociation:blender
NTS: ssdp:alive
USN: someunique:idscheme3
LOCATION:
http://192.168.1.2:4004/u1@server.com/uuid:ab
cdefgh-7dec-11d0-a765-00a0c91e6bf6
Cache-Control: max-age = 7393
```

Figure 10 Live Device Announcement (left) and the Corresponding Virtual Device Announcement (right)

location field. A remote announcement will conform to the remote addressing scheme therefore is easy to differentiate from a local announcement.

4.3.3.2.2. Device addressing scheme

A number of alternatives addressing scheme to be used by the VRM were considered: from assigning each UPnP device from a remote HAN a local port in the local HAN, through assigning a single port to each remote HAN and representing remote devices in local HAN using relative addresses. Another option, which was the design choice, was to listen on a single port for all remote devices from all HANs and use a consistent addressing scheme in the VRM. The first two options require listening on multiple ports and are more complex to manage and less bandwidth efficient. The third approach, which is the one selected for the UPnP plug-in design, allows listening on a single port however requires the location identifying a remote device to include more information identifying the HAN and the remote device.

In order to make the device location unique across the VRM's local HAN, and be able to extract the id of the devices along with the remote HAN hosting it from prospective calls, the device is announced the VRM's local HAN with a location that is comprised of the communication subsystem identifier for the HAN hosting the "live" device and the device unique identifier. The resulting URL is an address in the local HAN that represents the remote device.

Figure 10 illustrates how the original announcement (as shown in figure 7) sent from a blender in Bob's HAN will look like when shared with Alice and announced by the VRM in

Alice's HAN. The announcement is directed to the multicast address (239:255:255:250:1900) and contains the location of the blender device. In Bob's HAN, the blender has the location http://192.168.1.3/foo/bar. When announced in Alice's network it will have the location as can be seen on the right hand side of figure 10: http://192.168.1.2:4004/u1@server.com/uuid:abcdefgh-7dec-11d0-a765-00a0c91e6bf6. The location in the VRM's local HAN (Alice's HAN) is made of three parts: host:port – the host corresponds to the private IP address of the machine running the VRM and the port identifies the HTTP port on which the VRM listens for communication with remote devices represented in the local HAN. The second part of the address identifies the remote HAN hosting the "live" device by using its communication subsystem identifier. This identifier uniquely identifies the remote HAN and can be used with the communication subsystem for sending messages to the *Krox* system in that HAN. The third part of the location identifies the device in the remote HAN using its Universally Unique Identifier (UUID). It is important to note that from the discovery point of view there is no significance or complexity with the fact that both HANs in the example (Bob and Alice) are behind NAT and the original location of the blender in Bob's HAN may be used in Alice's HAN for a completely different device.

4.3.3.2.3. UPnP description

Once a control point has learned about the existence of a remote device (represented locally as a virtual device) it can start interacting with it. It is important to note that the control point is not aware if the device with which it interacts is remote. The control point follows its regular interaction protocol with the UPnP device it sees, so no change is required to the control point. This device is the virtual device represented by the VRM, which relays messages to the remote HAN hosting the "live" device. The interaction depicted in figure 11 begins after the remote UPnP device has already been announced by the VRM in the local HAN with a location that corresponds to a URL served locally by the VRM. A control point that has already received the device presence announcement, or search response, is interested in getting more information about the device through its description document. This is done by issuing an HTTP GET request by the control point and sending it to the location URL as it was defined in the device announcement (step 1 in figure 11). The HTTP GET is received by the VRM, which is expected to respond by posting the XML device description document. In order to accomplish that, the VRM in the local HAN (HAN1) needs to relay the request to the remote HAN hosting the "live" device corresponding to the

request (HAN2). The request that is sent to the remote HAN using the communication subsystem containing an identifier for the request, the UUID of the device whose description is requested, and the relative path to the description. The relative path is extracted from the URI given in the HTTP request. As explained in the previous section, the location that the request was directed to in the local HAN contains the communication subsystem identifier of the HAN hosting the "live" device, therefore this identifier can be extracted from the location and the request for description can be redirected to the correct remote HAN over the communication subsystem (step 2-3 in figure 11). In the remote HAN (HAN2) the description request is delegated to the UPnP LNC (steps 4-6 in figure 11). The LNC in HAN2 parses the XML description of the device or service and consults with the capability-sharing manager for checking which services or actions are shared with the requesting HAN (HAN1). Devices, services, or actions that are not shared with the requesting HAN are filtered from the returned device description document. For a device description, services may be filtered, and from a service description, actions may be filtered. Filtering is made possible by the conformance of device and service description to the device and service schema defined as part of the UPnP Device Architecture document [133]. The filtered XML document is then sent using the communication subsystem to the requesting HAN (steps 7-10 in figure 11). The VRM in HAN1 receives the returned description and before it can be posted to the requesting control point, URLs contained in it must be "localised". Modifications are required for all URLs that are to be accessed locally, such as the SCPDURL (service location), control (address for SOAP requests), eventSubURL (address for event subscription). These URLs must be prefixed similarly to the way the device location is constructed with the communication subsystem identifier of the originating HAN hosting the device, and the device UUID. Once URLs have been modified, the VRM in HAN1 posts the description to the requesting control point (steps 11-12 in figure 11).

In order to reduce the communication between remote HANs, several optimisations can be applied. The typical behaviour of a control point is to respond with a description request for every device/service it discovers. Therefore when the VRM announces a device or a service, it is highly likely that many control points in the local HAN that are interested in the service type will request the same description. To reduce the inter-HAN communication overhead and the load on the "live" device, device and service description can be cached in the VRM. The cache remains valid for the duration of the device announcement and enables the virtual

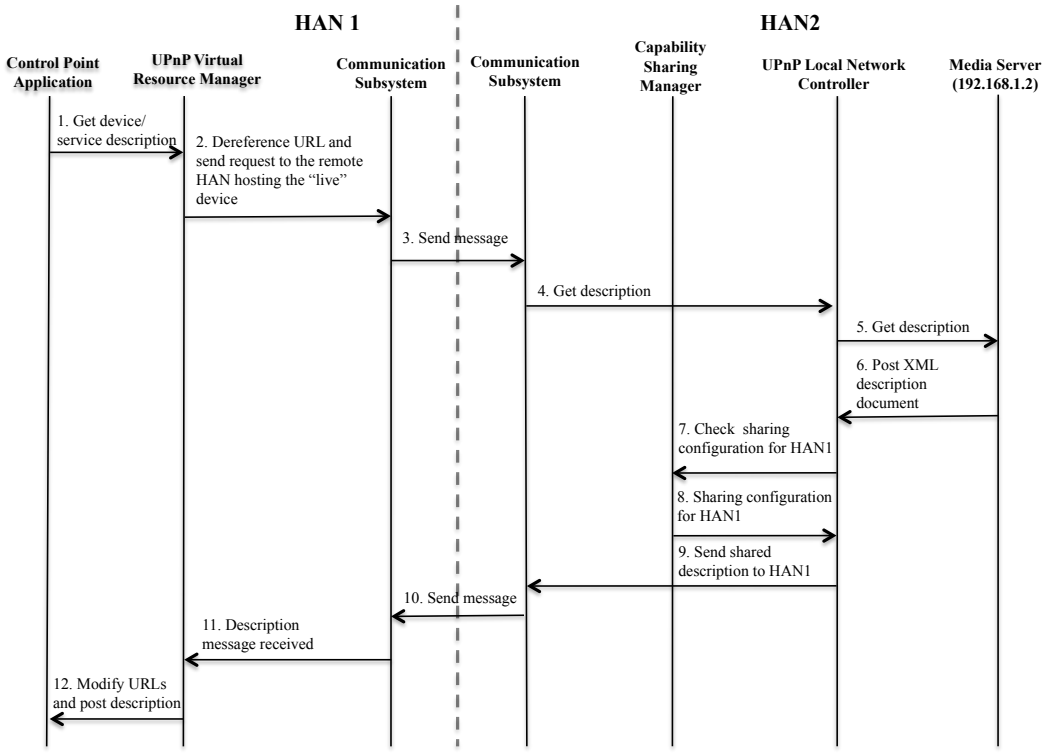

Figure 11 Multi-HAN UPnP Description Protocol Interaction

device to serve control points requesting the device description of remote devices very quickly. It also reduces potential contention on the actual device. While caching boosts performance it also incurs memory overhead, therefore it has to be balanced. It would be part of the evaluation to determine the benefit of caching in performance vs. its associated memory overhead. In case there is a change in the sharing permissions in remote HAN, the remote HAN can request peer HANs to invalidate their VRM cache due to a change that may impact the description of devices and services shared with them. However even if the remote HAN would see a description that is broader than what it might be able to access, the access control will be enforced when unauthorised access is attempted.

4.3.3.2.4. UPnP control and post processing

If the UPnP VRM receives an HTTP SOAP request for execution, it extracts the communication subsystem identifier of the HAN hosting the live device (HAN2) and the UUID of the remote live device from the URI header (steps 1-2 in figure 12). The SOAP execution request, along with an invocation identifier is forwarded to the corresponding hosting HAN using the communication subsystem where it is routed to the UPnP LNC (step 3 in figure 12). After verifying that the HAN (HAN1 in figure 12) attempting to execute the action has sufficient permissions to do so, the UPnP LNC in HAN2 posts a SOAP request to

the corresponding “live” device (steps 4-7 in figure 12). In case of an error, either an error received from the device, a network error, or a timeout, it is immediately forwarded back to the HAN requesting the invocation (HAN1 in figure 12). If the execution is successful, further filtering may be required in case the sharing configuration policy was defined for content sharing (step 8-9 in figure 12).

The UPnP VRM automatically generates a local proxy for a remote resource regardless of the device or service type. The UPnP VRM runtime entity is minimal and for all interaction it only serves as a generic proxy for the “live” device. However there may be a need for device or service specific extensions to the execution of the virtual resource. For example the result from a media server browse or search actions may contain content directories or URLs which the home user may or may not wish to share with a specific remote HAN. Therefore special handling needs to parse the result of such a query in its context and filter it based on sharing configuration as defined in the capability sharing manager. Another scenario is when the result of a query or action contains URLs, such as the printer queue URL in a printing service, or a URL for media in a media server. These URLs will not be accessible in remote HANs therefore additional processing needs to be performed either in

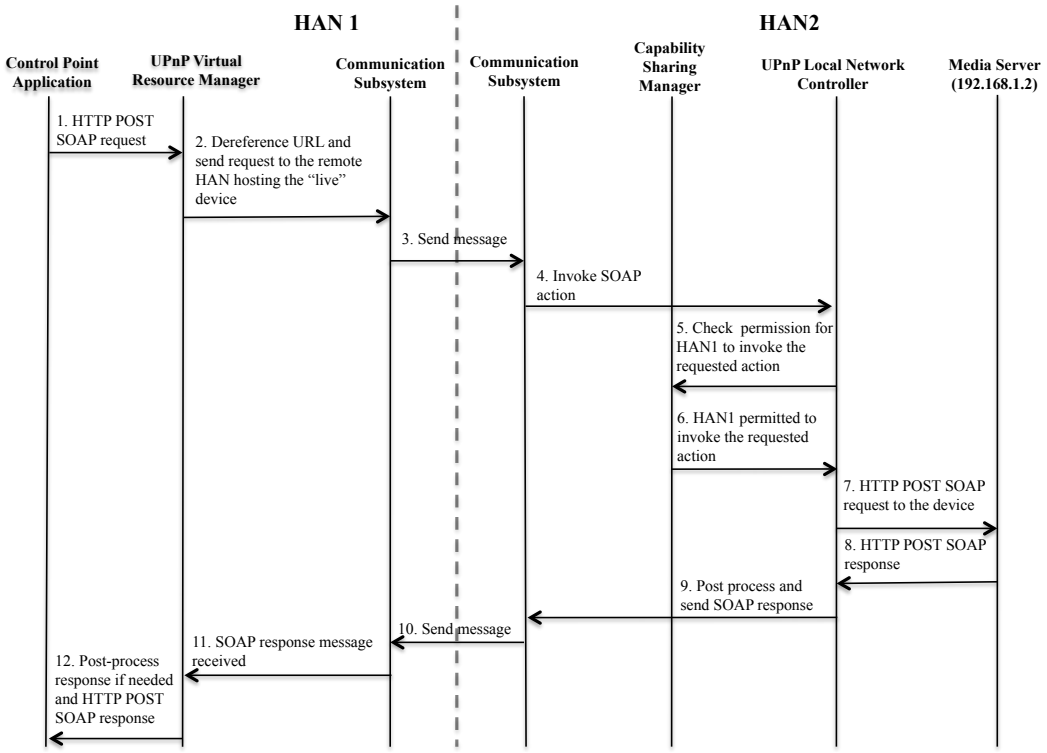

Figure 12 Multi-HAN UPnP Control Protocol Interaction

originating network or in remote HAN in order to enable control point applications in the remote HAN to use these URLs to print or access media streams. In order to allow greater flexibility, the *Krox* system architecture enables a service protocol plug-in to include extensions that can be invoked at predefined join points. Such a join point defines an event, which allows pre or post processing code to be inserted [33].

In order to support pre/post processing, a service protocol plug-in can provide pre or post processors that can be attached to a set of join points specified by the plug-in. Join points are mapped to the event model of the plug-in, either the core event model or plug-in specific extension of the model. The device/service specific processors can be attached at various levels such as for all devices of the service protocol, for a specific device type, for a specific service type and for a specific action in a service. This granularity, in addition to the available extension points, provides flexible and extensible behaviour complementing the generic functionality provided by the UPnP virtual device. The difference between a joint point and logic that is part of the regular execution flow (e.g. handling control request), is that an extension point may depend on the device type, and may have various strategies that can be separately composed in each deployment, while the regular execution flow is generic and independent of the device type. The configuration for pre/post processors is defined in the plug-in configuration and is registered when the plug-in manager loads the plug-in.

4.3.3.2.5. Out of band access to UPnP resources

As discussed above, the result received from a UPnP service can contain URLs which are meaningful and accessible only in the local HAN. These URLs must be replaced with URLs that can be accessed from the remote HAN that expects the result of the UPnP action. While the data access (e.g. media streaming) is out of band from UPnP point of view and is not standardised by the protocol, for the completeness of the solution the following approach is presented: the LNC replaces the URLs before sending them with a URL that has a unique identifier that the LNC maps to the original URL. http://192.168.1.3:53262/exportItem?id=5 it will be replaced with the URL: http://ExportItem?id=5 where the private IP address of the "live" device is removed. When the VRM receives the result containing URLs it adds as prefix to the URLs the identifier of the *Krox* system in the remote HAN hosting the device and the device unique identifier in addition to the host name and port for the UPnP VRM. The VRM would return the above URL to a client in the local HAN as:

http://192.168.1.1:4004/uuid:111233445555/resource/ExportItem?id=5. When the VRM receives a request that corresponds to a resource (as indicated by the structure of the URL), it extracts the identifier of the *Krox* system in the remote HAN, and forwards the request to that remote HAN with the device identifier. In the remote HAN, the LNC resolves the device from the device unique identifier and checks with the CSM that the remote HAN is allowed to access the resource and if so, it forwards the request to the device. The response from the device is tunnelled using the communication subsystem. In order to separate control traffic (for UPnP traffic) and data traffic (out of band communication between devices and control points) a separate channel in the communication system is used for transferring data. While the bandwidth is still limited, the separation of channels can enable assigning different priorities to the different communication channels. The advantage of this approach is that it avoids opening additional ports in the firewall of the HAN and does not require any manual configuration.

4.3.3.2.6. UPnP eventing

Control points that are interested in receiving event notifications from a UPnP service can subscribe by sending an HTTP subscription request to the virtual device's event subscription address as published by the VRM in the device description XML document. (step 1 in figure 13). The VRM extracts from the subscription URL the communication subsystem identifier of the HAN hosting the "live" device, and the device UUID similarly to the way this is handled for SOAP requests (Step 2 in figure 13). A subscription request is then sent to the remote HAN hosting the "live" device (HAN2 in figure 13) where it is received by the LNC (steps 3-4 in figure 13). The LNC checks with the CSM if the remote HAN (HAN1) is allowed to subscribe for events for the given device, and if so, it sends a subscription request to the device giving a local callback interface for subscription, which means the LNC would be the notification target for this subscription (steps 5-7 in figure 13). The device responds with a subscription identifier, which is cached by the LNC and mapped against the communication subsystem identifier of the HAN requesting the subscription, and then sends the subscription identifier to the requesting HAN using the communication subsystem (steps 8-10 in figure 13). The subscription identifier is received at the VRM in HAN1, which updates its local mappings with a subscription identifier and the callback interface that was given by the control point application in the original subscription request. The VRM then posts the subscription response with the subscription identifier to the requesting control point (steps 11-12 in figure 13). When the device sends a notification related to this

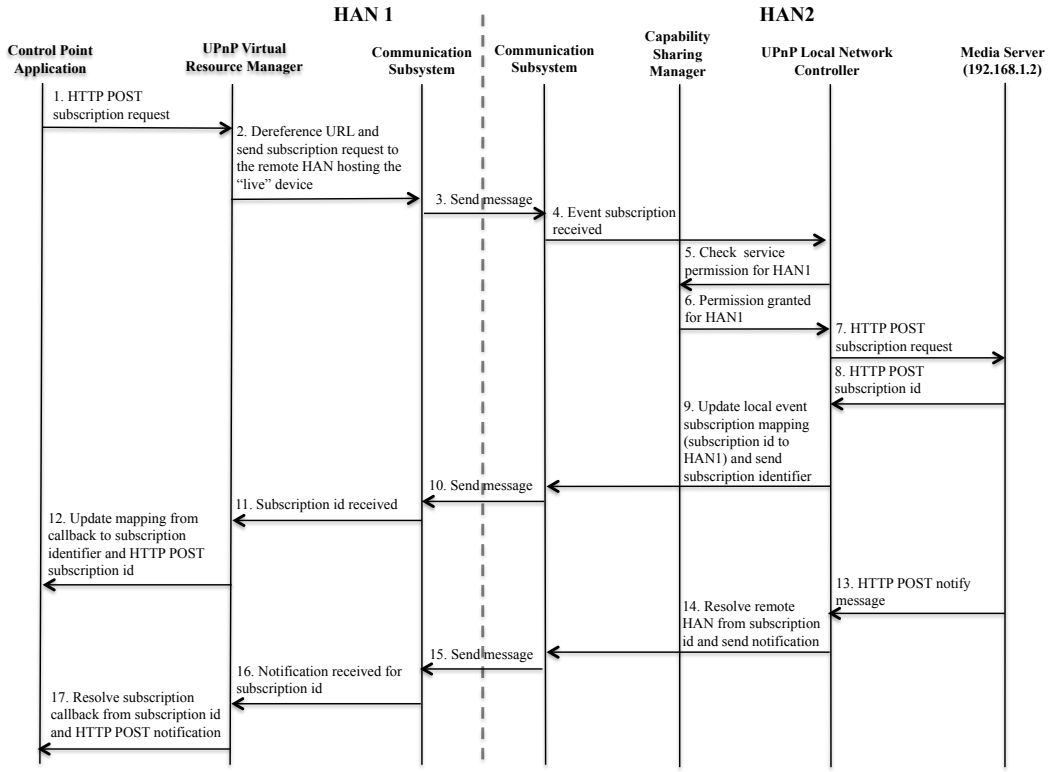

Figure 13 Multi-HAN UPnP Eventing Protocol Interaction

subscription, it is received by the LNC (in HAN2). The LNC extracts the subscription identifier from the notification and resolves the communication subsystem identifier of the subscribing HAN from its local mapping. The LNC then forwards the notification using the communication subsystem to that HAN (steps 13-15 in figure 13). When the appropriate VRM receives the notification, it resolves the callback interface given by the control point corresponding to the subscription identifier and posts the notification to this callback URL (steps 16-17 in figure 13).

When a control point unsubscribes, the VRM in HAN1 removes the mapping between the callback interface and the given subscription identifier, and forwards the unsubscribe request using the communication subsystem to the remote HAN (HAN2). Similarly, the LNC in HAN2 updates its local mapping and sends an unsubscribe request to the device.

If the sharing configuration was changed such that a subscribed remote HAN is no longer allowed to subscribe for events for the relevant service, the LNC should respond by removing the event subscription corresponding the remote HAN.

4.3.3.2.7. UPnP service protocol plug-in security guidelines

UPnP can be subject to many different types of attacks (see section 6.4.2). The UPnP service protocol plug-in design should be able to defend against common types of attacks from spreading beyond the scope of a single HAN. This section define a set of defence techniques:

1) Discovery – The VRM handles search requests in the local HAN for the remote devices it represents. In order to defend against misbehaving control points, the VRM should restrict the frequency of search requests to which it responds. Once the threshold is crossed, additional requests should be ignored. The LNC must verify that the location for the device specified in the device announcement is in the local HAN. If the URL is not local then the device announcement should be ignored.
2) Description – The LNC parses the service description before it sends it back to the requesting remote HAN. The device description should only be sent if parsing succeeds. In addition, if the size of the description document is suspiciously long – over a predefined threshold, it should be discarded and an error should be sent instead to the requester.
3) Control – The VRM should parse the SOAP request and verify that it is valid. It should identify suspiciously long SOAP requests based on a configuration of the maximum acceptable SOAP request. The LNC should parse the SOAP response and identify suspiciously long SOAP responses that are longer than a predefined threshold. If such response is identified, it should be discarded and an error should be returned.
4) Eventing – The VRM should verify that the callback URL is in the local HAN. The LNC should not subscribe for events more than once per each service irrespective of the number of remote HANs requesting subscription. Additional subscription request should be notified based on the existing subscription with the device. When the last remote HAN subscription is removed, then the LNC should unsubscribe from the device. If an event notification attack is identified (e.g. when the number of event notifications per time window crosses a threshold) in the local HAN event notifications from this device should be discarded. Similarly, the VRM can have a maximum of one subscription per remote service sent to the remote LNC, and map all additional subscriptions to this single subscription.

5) Due to the potential high risk of sharing the Internet Gateway Device, it is recommended that the IGD is not shared with any remote HANs. The UPnP plug-in implementation can either prohibit sharing of IGD or give a warning to the home user indicating the risk.

4.3.3.3. Summary

The above sections described the details of a design for a service protocol plug-in that can be used with the *Krox* system architecture for an extension of the UPnP protocol to multi-HAN through service virtualisation using the *Krox* communication subsystem. This plug-in design supports seamless integration through resource virtualisation, such that network protocols do not need to be modified, and the UPnP protocol extension is designed in a way that is transparent to existing applications. The design for the UPnP plug-in gracefully handles private networks by using the *Krox* communication subsystem and an addressing scheme that translates between private network addresses and published addresses. Since all UPnP protocol traffic is carried over the *Krox* communication subsystem, the plug-in does not require additional ports to be open and therefore firewalls do not present an additional challenge for this design. The secure communication subsystem guarantees that only authenticated *Krox* system instances can interact with each other and all UPnP traffic between HANs is encrypted. Authorisation is achieved by using the capability-sharing manager. While performance needs to be evaluated, it can be observed that the control traffic carried on the wire does not add significant overhead to the UPnP payload to/from the control point or device, since only an additional device UUID and a request identifier is added to the constant overhead of each message. The design for out-of-band communication enables control points in the local HAN to send and receive data from devices in remote HANs securely using a separate secure communication subsystem channel. The UPnP plug-in does not include support for UPnP presentation layer because it is not very common in HAN devices, however the presentation layer is optional in the UPnP protocol stack.

Section 5.5 in the next chapter presents the details of a prototype implementation of the UPnP service protocol plug-in for *Krox* system architecture.

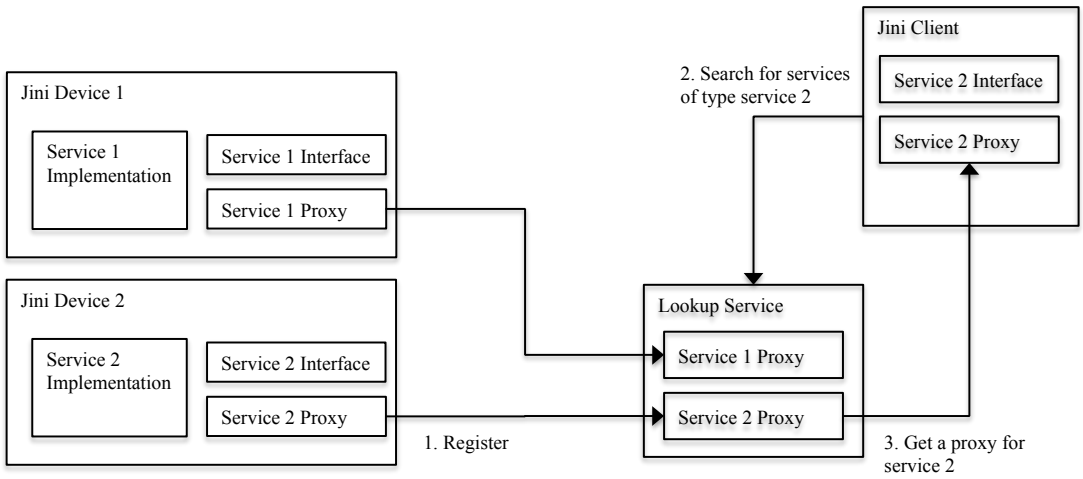

Figure 14 Jini Architecture

4.3.4. Jini

In order to demonstrate the generality of the *Krox* system architecture and its applicability for HAN service protocol plug-ins, this section presents the design for Jini service protocol plug-in. As described in chapter 2 (section 2.3.3) Jini is a Java based service oriented distributed architecture. Through the definition of protocols for look up and discovery, Jini enables clients to find and invoke services in the local HAN. Jini takes a service-centric approach whereby services can be software or hardware, and can communicate with each other and facilitate the composition of complex applications from several atomic services. This section gives a description of the Jini architecture (section 4.3.4.1) followed by a presentation of the corresponding Jini service protocol plug-in for *Krox* system architecture (section 4.3.4.2).

4.3.4.1. Jini architecture

Jini architecture, presented in figure 14 includes Jini services and Jini clients. The architecture is based on service interfaces, service implementations and service proxies. A service interface defines the contract that the service supports. A service implementation implements this interface. A service proxy is a stub that can be downloaded to the client from the lookup service for interaction with the service. The advantage of this approach is the loose coupling between the client and the server, and that the client does not depend on the implementation of the service. Jini is Java based and requires devices to include a Java Virtual Machine (JVM).

4.3.4.1.1. Service discovery

The first step in Jini service discovery is finding a lookup service. A lookup service enables devices to register their services, and clients to locate services. As shown in figure 14 (step 1), once a lookup service has been found, the service provider can use it to register the service, and a client can use it to search for services of interest by the service interface and additional characteristics, such as manufacturer, and a friendly name (step 2). At the time a service is registered, it can attach a set of attributes that can be used by the lookup service for matching against client queries. These attributes can be modified at a later time. The service interface is defined in terms of Java interfaces and there is no standardisation to date for classes of devices or services, therefore a device, e.g. printers from different vendor may have very similar yet different Jini service interfaces. When the service registers itself with the lookup service it can do so with a "lease", indicating the time it is available for. The service can then renew or cancel the lease. The lookup service "leases" a proxy object to the client for a period of time, such that when it expires the client needs to renew the lease. Similarly when the device leaves the network the lease is automatically expired. For high availability, multiple lookup services may exists in the HAN, and the devices can register their services in more than one lookup service, to ensure their availability even if one lookup service becomes unavailable.

4.3.4.1.2. Service invocation

Once the client obtained a service proxy for a Jini device/service, it can invoke methods on this proxy. The proxy implementation communicates with the remote service implementation via a network protocol, which is abstracted from the service client. Jini does not dictate the specific protocol for interaction between the service proxy on the client side and service object on the server side. However, typically this involves Java Remote Method Invocation (RMI). The service developer can choose between a "thin" proxy that only communicates with the remote object or a "fat" proxy that implements some or even the whole business logic on behalf of the remote object.

4.3.4.1.3. Jini security

Security is an important aspect of Jini due to its extensive use of mobile code, which is downloaded from lookup services and potentially HTTP servers. Early versions of Jini supported various security mechanisms through Java security for controlling access to

resources. Using Jini (both in client and service) requires using the Java Security manager. The Java security manager uses a policy file that controls permission that enable the client or server to protect them in a hostile environment. The permission file is additive, therefore a client or server cannot exclude permission, but rather it needs to specify permission explicitly. Java security policy can restrict access to activities, such as connection to sockets. In addition, activities can be allowed based on the host from where the code is downloaded, and finally, access to activities can be associated with digital signatures.

Jini 2.1 introduced a number of mechanisms for integrity, confidentiality, and authentication, in addition to the standard mechanisms described above. Integrity ensures that classes and instances were not modified on their way between the server and the client. Confidentiality ensures that unauthorised access to the data was not allowed. Authentication ensures that the data comes from whom you expect. These mechanisms are supported through the definition of constraints. Jini security defines a set of constrains that can be applied on the server or the client, and can be defined on a method level. This could be used such that when the client receives a notification from the lookup service on a new service discovered, it can use a custom proxy preparer such that enables using the service only if the client constraints are met. A client can require the proxy to support integrity by defining the integrity constraint. Another risk when using Jini is that code may be downloaded from an HTTP server. When a client discovered a service and downloaded the proxy from the lookup service, it in fact received a URL from where class files can be downloaded. In order to verify that the downloaded class actually corresponds to the version on the server, MD5 hashing technique is used such that the client can verify that the jar file that was downloaded has the same hash signature as was given in the URL. In addition the client can require the proxy to be verified by a local trust verifier. For confidentiality the client can constrain the invocation to require encryption.

4.3.4.2. Jini service protocol plug-in

Jini fits very well into the design approach for extending services beyond the scope of a single HAN described in this thesis. In the Jini programming model the actual way a proxy service object interacts with the service instance implementation is abstracted from the client of the service (either human or machine). Therefore the client of the service is not aware of which underlying network protocol is used to interact with the actual service.

Following the *Krox* system architecture described earlier, the approach proposed by this thesis is to represent shared resources from remote HANs as virtual resources in the local HAN. A virtual resource that represents a remote Jini service interface must support all actions on this interface. For each Jini service that is shared with a remote HAN, a virtual service is automatically generated in the corresponding remote HAN. This virtual service can be discovered in its hosting HAN in an identical way to other Jini services with no further configuration or administration. Once discovered, actions can be invoked on it, such that each call is delegated to the "live" service in its hosting HAN. The Jini LNC is responsible for all interaction with local HAN's Jini services including the discovery and invocation. The Jini VRM is responsible for representing services shared from remote HANs in the local HAN. The *Krox* communication subsystem is used for securely exchanging messages between the Jini LNC and Jini VRM.

The Jini VRM is designed as an aggregator for all remote HANs' Jini services. The Jini VRM caches information about remote services shared from other HANs and dynamically generates an implementation and a proxy (conforming to the Jini architecture) that correspond to the interfaces of the shared services. The generated proxy communicates locally with the instance implementation and relays the invocations request and response. The Jini VRM caches the service implementation and registers the proxy with the local HAN's lookup services.

The full details of the design for Jini service protocol plug-in for *Krox* system architecture are given in following sections.

4.3.4.2.1. Service discovery

The Jini LNC is responsible for discovering Jini services in the local HAN. This is done by first locating lookup services via a multicast message to a well-known multicast address. Once the Jini LNC finds lookup services, it subscribes for service updates (step 1 in figure 15). In order to enumerate all types of services in the local HAN, the LNC does not specify a certain service type but instead listens to all types of services in the local HAN. When a local Jini service is added, the LNC is notified by the lookup service, which in turn updates its local repository with the added service proxy (steps 2-3 in figure 15). When a remote HAN changes its status to "available", the LNC checks with the Capability Sharing

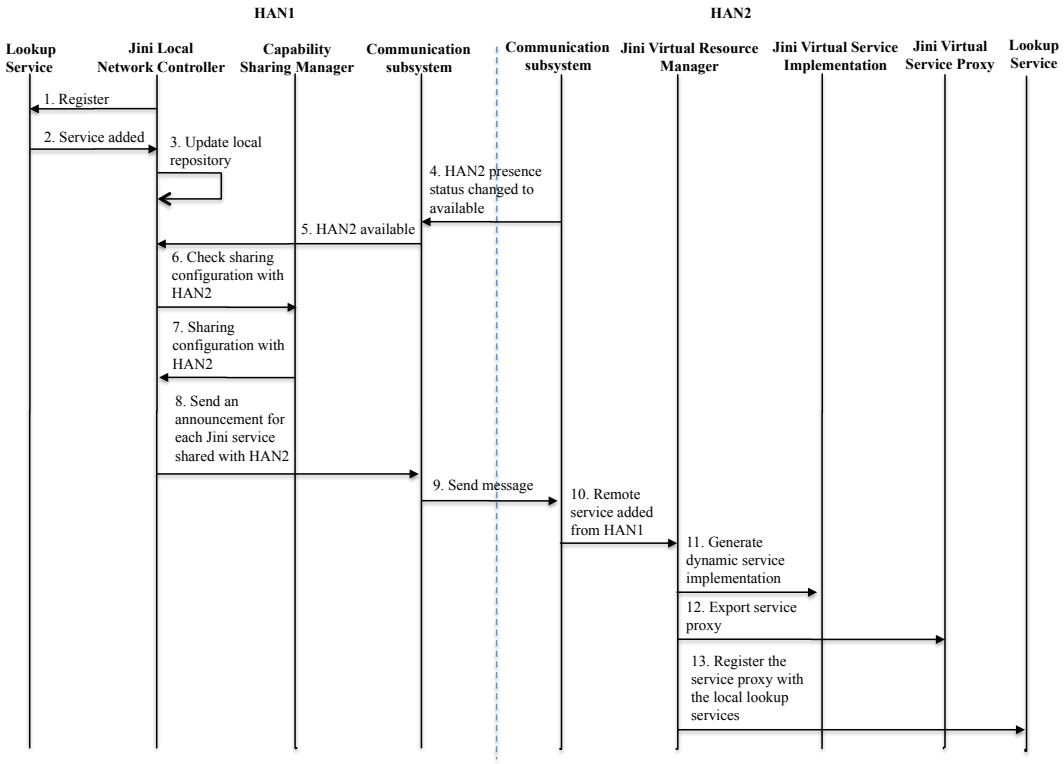

Figure 15 Multi-HAN Jini Service Discovery

Manager which services should be shared with it, and in turn and sends a notification about these services to that HAN using the communication subsystem (steps 4-9 in figure 15). Similarly when a Jini service is added to the local HAN, the LNC checks with the Capability Sharing Manager with which remote HANs it should be shared, and consequently sends an announcement about the added service to each of these HANs. The service announcement contains only the service interface name and a service identifier. Unlike UPnP where the service description could be modified to reflect partial sharing of actions from a service, in Jini this is not possible. The Jini service "client" and the Jini service "server" must pre-share the Jini service Java interface - therefore it cannot be modified to reflect sharing of parts of the interface, at least not in the interface level. This does not mean however that home users must share all the methods of a Jini interface. It only means that all the methods of a Jini service interface are visible to a remote HAN with which is it shared. If an action is not shared with a remote HAN, when the remote HAN will attempt to execute the method, the execution will fail in the LNC due to insufficient privileges.

Once the Jini VRM is notified of an added Jini service, it is responsible for virtualizing the remote service by creating a local instance of the Jini service that forwards requests to the network hosting the "live" Jini service. In order to create a local service that will act as described above, the Jini VRM needs to register a proxy for the service implementation with the local lookup service. As explained above, a Jini service is a trio of service interface, service implementation, and a service proxy (steps 10-13 in figure 15).

When a local Jini service becomes unavailable, the Jini LNC notifies the remote HANs with which the service is shared, and the corresponding VRM in these HANs cancels the lease for the virtual service that was registered with the lookup service in those HANs.

When the sharing configuration is changed the Jini LNC is notified by the CSM and sends a corresponding update to the affected remote HANs where the Jini VRM needs to create or destruct relevant Jini service implementations and service proxies.

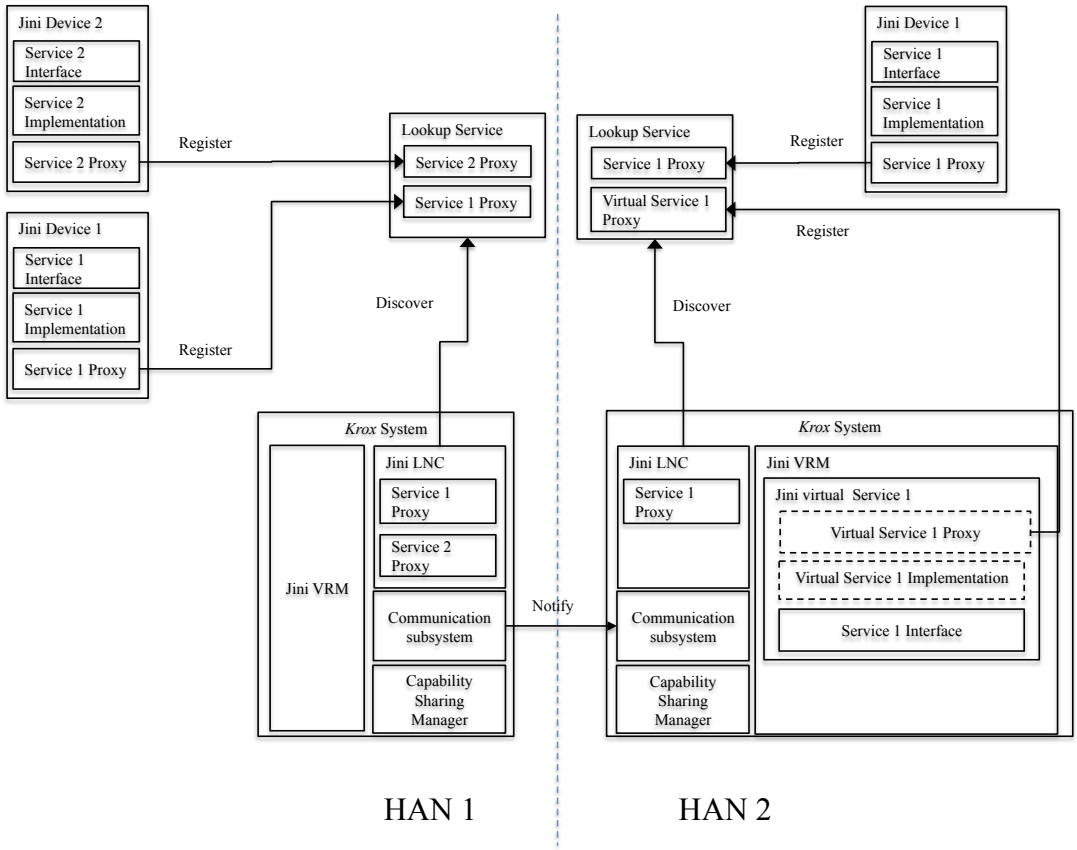

Figure 16 Jini Service Plug-in Example

Figure 16 illustrates the discovery process of Jini across multiple HANs. HAN1 has two "live" Jini enabled devices, each with one service. Each service is a trio of a service interface, a service implementation (hosted by the device) and a service proxy. The service registers itself with a lookup service, which results in a service proxy being stored in the lookup service. HAN1's Jini LNC locates the lookup service and then retrieves information about Jini services in the HAN and updates its repository with the service proxies for the 2 services. HAN1 shares one service (service1) with HAN2. HAN2 runs one "live" Jini enabled device with one service. Similarly to HAN1 the service has a service interface, a service implementation and a service proxy. The service registers with the lookup service in HAN2 and as a result the lookup service repository is updated with the service proxy. When HAN2 comes online, the Jini LNC in HAN1 sends information about the HAN1's shared service (service1). The information is received in the HAN2 VRM and results in the automatic dynamic generation of a virtual service implementation and a virtual service proxy in HAN2. Once the virtual service has been generated, the service implementation is kept in the VRM repository in HAN2, and the virtual proxy is registered with the local (HAN2) lookup service. At this point, HAN1 has 2 Jini services, all of them are local, and HAN2 has 2 services, one of them is local, and the other one is virtual, shared from HAN1.

4.3.4.2.2. Service invocation

Once the Jini VRM has registered a local virtual service proxy for the remote Jini service, client applications searching for services having the particular service interface will be able to find the virtual service in the lookup service in the same way as local services. Figure 17 illustrates the interaction that takes place between the client application and the other parts of the system. Once an application in the local HAN discovers the virtual service, it downloads the service proxy and invokes methods from the service interface (steps 1-3 in figure 17). When a method is invoked, the virtual service proxy relays the request to the virtual service implementation that is hosted in the Jini VRM. The virtual service delegates the call using the communication subsystem to the network hosting the "live" Jini service (steps 4-6 in figure 17). When the Jini LNC in the remote HAN receives an invocation request, it resolves the local service proxy. Before the service is invoked the Jini LNC must verify that the remote HAN has sufficient permissions to execute this method by checking with the capability-sharing manager. If the requesting HAN (HAN1 in figure 17) is not allowed to call this method then an error is sent back. Otherwise the LNC invokes the

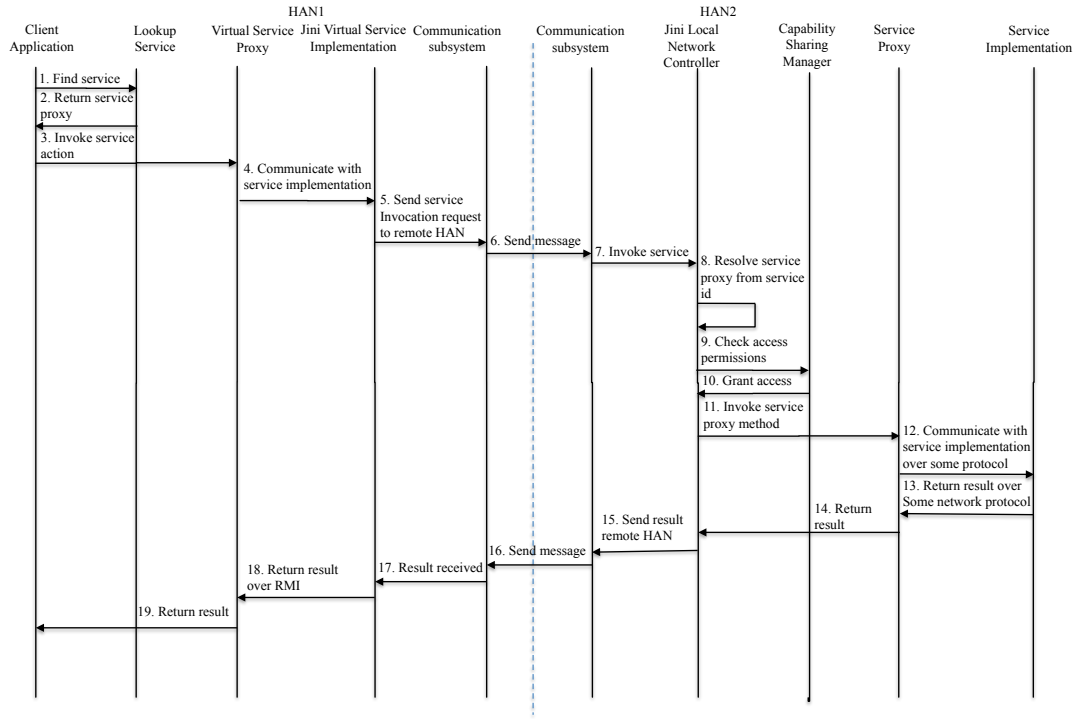

Figure 17 Multi-HAN Jini Service Invocation

required method on the local "live" service proxy (steps 7-11 in figure 17). The local service proxy interacts with the actual "live" service implementation and a result is returned to the Jini LNC. The result is then delegated using the communication subsystem to the remote HAN with the result or error (steps 12-16 in figure 17). The Jini VRM (in HAN1) receives the result and passes it to the virtual service implementation, which returns the result or error if any to the virtual service proxy. Finally the virtual service proxy returns the result or error back to the client application (steps 17-19 in figure 17).

4.3.4.2.3. Jini service protocol plug-in security guidelines

While Jini service protocol can be considered secure with the mechanisms described in section 4.3.4.1.3, these mechanisms must be used in order to prevent a hostile or misbehaving entity in a Jini environment to create an attack that would spread beyond the scope of a single HAN.

The Jini LNC interacts with the lookup service and service proxies in the local HAN. Since the LNC runs a proxy to the lookup service as part of its code, it should accept proxies only if they (the corresponding registrar) are signed by a trusted authority. The same rule applies

to the Jini VRM, which interacts with the lookup service in its HAN in order to register virtual services. The trust is configured in the security configuration files of the system.

When the Jini LNC downloads a proxy to a service it must do so only if it is signed by a trusted authority. In addition, it can use a local verifier to verify its trust in the service. Finally the LNC should use integrity and confidentiality constraints when interacting with local services to ensure that data is not tampered with on the way to and from the service, and that communication is encrypted.

4.3.4.3. Summary

The above sections describe the detailed design for a service protocol plug-in that can be used with the system architecture for an extension of Jini to multi-HAN through service virtualisation, using the secure communication subsystem. This plug-in design supports seamless integration through resource virtualisation, such that no HAN service or network protocols needed to be modified. The extension for supporting sharing of Jini services across multiple HAN networks does not require modifications to Jini architecture or to applications consuming Jini services. The design for multi-HAN service discovery and invocation enables existing and future Jini applications to seamlessly discover and invoke remote services in a similar fashion to local services. Authorisation is achieved by using the capability-sharing manager.

While the presented approach for extending Jini service protocol can be useful in many cases, it may not support all types of Jini services. Jini services are not limited in their parameter types and return values and can use any Java type. While for parameter types there is no restriction, as long as the object type is serializable, Jini service return type can be a networked object that communicates with the physical device. For example, a printer service PrintService[24] returns a DocPrintRequest[25] where the doc for printing needs be set. Therefore the interaction of a client with the printer would require two steps:

[24] http://www.jini.org/files/specs/print-api/net/jini/print/service/PrintService.html

[25] http://www.jini.org/files/specs/print-api/net/jini/print/job/DocPrintRequest.html

```
// prepare the document for printing
Doc docToPrint=new InputStreamDoc(…);
// locate the Jini printer service
PrinterService printer=…;
// create a print request job (step 1)
DocPrintRequest request=printer.createDocPrintRequest();
// print the document (step 2)
request.setDoc(docToPrint);
```

The object that is returned to the caller of printer service in step 1, communicates with the physical printer for setting the document, when the appropriate method is invoked in step 2.

The Jini service protocol plug-in can be extended to support such scenarios with two mechanisms. If the return type of a Jini service is an immutable object, therefore cannot be modified after its construction, there is no need to modify the plug-in, and as long as the object is serializable, there is no limitation. To support scenarios such as the printing service described above, the Java dynamic proxy technique can be used. Instead of returning the actual object from the service, the virtual service implementation should return an interface implementation dynamically generated using dynamic proxy, such that each call to the interface is delegated over the communication subsystem to the remote HAN where it is invoked on the actual object that was returned from the method invocation on the physical device. For example, if a Jini PrintService returned a DocPrintRequest object when creating a print job, instead of sending the actual DocPrintRequest Object between the remote HANs, the object should remain in the HAN hosting the physical printer (in the LNC), and a virtual DocPrintRequest object would be automatically created in the remote HAN's VRM. When methods are invoked on the DocPrintRequest object, they are tunnelled using the communication subsystem to the remote HAN hosting the physical printer. With this approach the Jini plug-in can support a return type for a service as long as it is serializable, and either immutable or implements an interface that can be replaced with a dynamic proxy implementation.

Section 5.6 in the next chapter presents the details of a prototype implementation of the Jini service protocol plug-in for *Krox* system architecture.

4.3.5. Service composition subsystem

Service composition for HAN services is needed to allow construction of complex rich featured functionality from multiple atomic home services. While services can be composed in a service protocol specific manner, e.g. using a control point application for constructing UPnP applications, or a Java program for constructing a Jini application, these approaches are not reusable themselves as services, i.e. a control point application is not a UPnP device or service, therefore it cannot be discovered or invoked. Similarly a Java application using Jini services for an application is not rendered as a Jini service. In addition, the heterogeneity of devices and services in the HAN and the lack of interoperability between different service protocols prevent services of different protocols from being composed. In order to facilitate the composition of services in the HAN, an abstraction layer is required between the low-level device/service technology, and the service composition layer. This abstraction layer should hide the differences in format, protocol, and network related details from the consumer and expose the device services in a common standard way.

The design for the service composition subsystem is based on SOAP web services as a service representation for HAN services and a BPEL service orchestration engine for composing and executing composite services. A major advantage of the web-based approach, in addition to the standard and uniform access to various types of services, is that it enables the composability of such services using existing service composition standards, which is an important asset for HAN applications. Using BPEL has the advantage that the composite service is by itself a reusable web service, which can enable its further composition. Another advantage is that it can enable the composition of external web services with HAN services. Finally based on the resource virtualisation in *Krox* system architecture, composite services can enable orchestration of local and remote HAN services.

The proposed design for a HAN service composition system contains 3 main components:

1. **Orchestration engine** – E.g. a standard BPEL engine hosting the executable composite services

2. **Web services** – Automatically generated services that correspond to the home services discovered in local HAN. With the virtualisation framework, such services can be in fact either local or virtual services available in the local HAN
3. **Service Protocol plug-in** – A plug-in component is needed for mediating between the service technology, e.g. UPnP, and the service representation. Following the plug-in architecture described earlier in this chapter, the plug-in for a service protocol includes support for mapping between the service protocol and web services

In *Krox* system design BPEL is used for composing home services from various sources, local HAN services, remote HAN services, and external web services. Composite services are defined in BPEL using dynamic and automatically generated web service proxies that represent HAN services (either physical or virtual) in the local HAN. As discussed in section 3.2.3, using BPEL for HAN service composition was suggested in literature in [17, 52, 106]. Redondo et al. [106] present a BPEL based dynamic service composition with OSGi. It assumes a corresponding bundle exists for mapping the service protocol to OSGi and is focused on service composition itself, using BPEL to express the orchestration of OSGi services. The architecture described in [106] does not support service orchestration of remote services with local ones. Bohn et al. [17] and Hackmann et al. [52] suggest service protocol specific extension of BPEL, but unlike in *Krox* system design, this approach is limited to local services of a single service protocol.

Web services form a generic service representation and need to be mapped from the original service protocol. For example capabilities offered by a UPnP-enabled device discovered in the HAN, would be exposed as a set of web services corresponding to the services supported by this device. These services can then be composed with other services including UPnP services, Jini services, external web services, and other composite services.

In order to support the mapping of various service protocols to web services, the service protocol plug-ins discussed earlier in this chapter are extended with additional support for web service generation and mapping. The advantage of this approach is that it encapsulates all the details of a certain service protocol under individual pluggable modules. Another advantage is that the transformation functionality can leverage other capabilities already existing in the service protocol plug-in as part of the service virtualisation, e.g. service

discovery. By creating an interoperable proxy web service interface, different HAN service protocols can be used and composed in a uniform manner ignoring the underlying differences in protocol, data structure, platform and network.

Once a service is discovered by the LNC/VRM, this extension is invoked and generates dynamically and automatically a corresponding technology neutral SOAP web service. The web service is deployed into a local web server, which is bundled with the *Krox* system.

4.3.5.1. UPnP to web service mapping

In order to support the mapping of UPnP services to web services, the LNC and the VRM are extended. Once a device/service is discovered (a local device/service in the LNC, a remote device/service in the VRM), its XML description is fetched. By parsing and inspecting the XML description of the service, a corresponding web service is generated such that for each UPnP service, a single web service is created, and each UPnP action corresponds to a web service operation. The parameters of the operation match the parameters of the UPnP action. Because the invocation of the web service is performed using SOAP, which is text-based, the web service parameters can be constrained to Strings. In cases where there is an error in the service invocation, the web service throws an exception. In case of success, the web service returns the SOAP response as received from the service. The similarity between the way web service interfaces and UPnP services are described enables the simple mapping, and the parsable format of a UPnP service description enabled the automation of the web service generation.

Once a proxy web service has been generated it is built and packaged as a web application and deployed to a local web server. As soon as the device or service disappears from the network, its corresponding web service proxies are immediately undeployed from the web server and removed. When an action is invoked through the web service, the corresponding device action is called and the result or error code is returned to the caller.

4.3.5.2. Jini to web service mapping

Similarly to the extension of UPnP service protocol plug-in, the mapping from a Jini service to a corresponding web service is introduced as an extension to the Jini service protocol

plug-in. Whenever a Jini service is discovered, either locally by the Jini LNC or for a remote service - in the Jini VRM, the transformation from the Jini service interface to a web service is performed. Once a service interface has been discovered, it is inspected and a web service is automatically generated for it. Each Jini service interface corresponds to a single web service such that each method in the Jini service corresponds to a single operation on the web service. The signature of each web service method corresponds to a single method in the Jini service interface. The automatic generation of a web service from a Jini service interface is made possible by the parsable Java interface of a Jini service using Java reflection API.

The generated web service implementation interacts with the Jini service by locating the service by its service identifier in the lookup service. Once the web service has been built and packaged, it is also deployed to the local web server. When the web service is invoked, it communicates with the Jini service and returns the result/exception as returned from the Jini service execution. With the integration of service protocol to web service mapping to the plug-in's discovery mechanism, as soon as a new services are added, a corresponding web service is generated and deployed. As soon as the service is no longer available in the HAN, its corresponding web service is undeployed.

4.3.5.3. Composing home services

After the web service proxy for UPnP and Jini services has been deployed to the web server, a composite service can then be defined in BPEL involving the web service proxies, and may include other arbitrary web services, either internal or external. Such a composite service can be deployed in a BPEL runtime engine and be executed in the home environment, either as part of a client application, or as part of another composite service. The advantage of a web service proxy based approach is that it enables seamless composition of web-based services with UPnP and Jini services. Another important advantage of this design is that remote and local services (both represented in the local HAN as web services) can be seamlessly composed.

Composition can be made either by the home user or by a service provider and only instrumented. For example, a service provider may run a process in the home network that discovers home services and based on a knowledge base of template composition can

identify potential composite services that can be offered to the home user. When the user has all the prerequisites for a given composite service, the composite service can be deployed. It is not the purpose of this thesis to define a methodology for user driven service composition but only to establish a framework that can enable this process.

4.3.5.4. Composite services as UPnP services

BPEL composite services are accessible to local HAN clients in the form of SOAP web services. A client can invoke them directly with SOAP requests. While the previous sections discussed a mechanism to map a service protocol (e.g. UPnP) to a generic web service, in this section the reverse process is presented for composite services, i.e. by adding a UPnP interface to a composite service, this enables seamless sharing of BPEL composite services with remote HANs. There may be cases where composition could be a mechanism to share a function, rather than share the devices and services implementing this function. A useful side effect of this approach is that these UPnP services can be discovered by UPnP clients in the local HAN and interact with other UPnP services. The idea of representing workflows as UPnP devices was suggested by Bobek et al. in [16]. However the design suggested here is different in that composite services are available as independent UPnP devices rather than as embedded devices as part of the workflow engine device as suggested in [16]. In *Krox* system design the purpose is not to enable management through UPnP of the workflow engine, but rather to enable sharing of composite services. Since a sharing framework for UPnP has already been suggested as part of the design, the mapping from BPEL to UPnP piggybacks existing functionality and seamlessly enables sharing of composite services with remote HANs. This means once the BPEL composite service has been mapped to a UPnP device/service and announced in the local HAN, the UPnP LNC will discover it and share it as a UPnP device if required, with remote HANs.

In order to support sharing of composite services, a transformation between a BPEL service and a corresponding UPnP device/service is required. When a BPEL service is deployed to the BPEL engine, a corresponding UPnP device is automatically generated and can be discovered and accessed through UPnP. By the parsing of the WSDL of the composite service, the corresponding service name and actions are taken and used for generating the corresponding UPnP device, having services corresponding to the BPEL service operations. The process for generating a UPnP device from a BPEL service is depicted in figure 18. When a control point invokes the UPnP service, its implementation invokes the composite service by sending a SOAP request to the process URL.

4.3.5.5. Summary

The above sections described the design for the *Krox* service composition subsystem. It is suggested that services can be composed via combining BPEL for expressing and executing composite services with a plug-in approach to mapping between service protocol and web services. The design demonstrates how the plug-in approach integrates the discovery modules in both the VRM and the LNC and the functionality for generating automatically a corresponding web service for a UPnP or Jini service based on inspection of the parsable service interface. Once web services are available and deployed in the local HAN, service composition with BPEL can take place, such that it can compose same technology services, as well as cross technology services, and external web services. Using BPEL for service composition supports the requirement for further composability of the composite service. The final requirement from the service composition subsystem is to enable sharing of composite services. In order to support this requirement, a mapping between BPEL and UPnP has been defined such that when a composite service is deployed to the BPEL engine, a corresponding UPnP service is automatically generated and advertised in the local HAN.

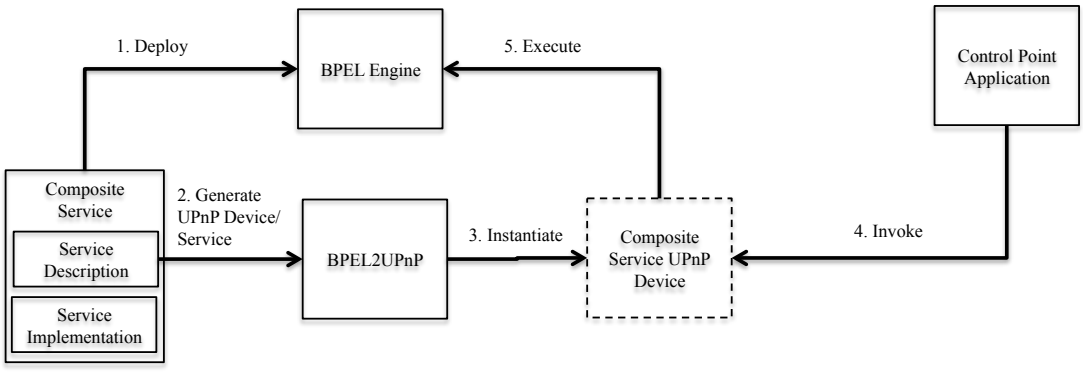

Figure 18 Generating a UPnP Device Proxy for a BPEL Service

If the sharing policy allows it, the service will be shared.

4.3.6. Security considerations

The above sections describe the *Krox* system design. The following sections provide more details about the security mechanisms used as part of the *Krox* architecture and design.

4.3.6.1. Authentication

Authentication is the first step in the system bootstrap. The system must authenticate itself against the IM&P server so it can communicate with other instances of *Krox* system in remote HANs with which sharing has been agreed (i.e. they are in the local HAN's *Krox* system buddy roster). With XMPP, authentication is made using Simple Authentication and Security Layer (SASL) [80] as defined in RFC 3920 [114]. SASL does not define a specific mechanism and XMPP supports all of the mechanisms defined in RFC 4422 [80], however XMPP recommendation is to use the EXTERNAL mechanism with end user certificates for client to server authentication.

4.3.6.2. Confidentiality

The inter-HAN traffic must be encrypted to avoid information disclosure and prevent eavesdropping. With XMPP, as soon as the client (*Krox* communication subsystem), authenticated and is connected to the XMPP server, all communication from the client to the server is encrypted using Transport Layer Security (TLS) [109].

4.3.6.3. Authorisation

The Capability Sharing Manager described in this chapter is responsible for management of access control. Before a device/service is shared with remote HANs, the CSM checks the sharing configuration and permits or denies sharing with the remote HAN. Similarly when a remote HAN requests an action to be executed, permission is checked with the CSM before action is executed. The CSM can further filter content from the response sent back to a requester based on the fine-grained sharing configuration.

4.3.6.4. Rate limiting

In order to avoid situations of one instance of *Krox* system flooding other instances in remote HANs with messages due to a misbehaving service protocol plug-in, or a software bug, or an attack, the communication subsystem is required to support rate limiting. Rate limiting should constrain the number of messages a *Krox* system can send/receive in a given amount of time. If more messages are sent or received they will be dropped. This could be also implemented as a plug-in for the IM&P server, such that the limit will be per a client connection, or between two endpoints. The advantage of this approach is that it does not affect the performance of the *Krox* system in the local HAN.

4.3.6.5. Miscellaneous

XMPP defines a set of best practices and recommendations that should be applied by XMPP implementations to defend against multiple types of denial of service attacks however these guidelines are not mandatory [142].

4.4. Conclusions

This chapter presented the *Krox* integrated system architecture and a design for addressing the requirements for intra-HAN and inter-HAN service interoperability as they were presented in the previous chapter (section 3.4). *Krox* is a service-oriented architecture that enables HAN services from different multiple HAN service protocols to be composed and shared with remote HANs. *Krox* system architecture defines a plug-in framework that enables plug-ins to support various service oriented HAN service protocols via an extensible event model.

There are a number of restrictions that need to be considered in regard to the *Krox* system architecture presented in this chapter:

1) While the plug-in framework is flexible, it is suitable for a specific subset of HAN service protocols. The *Krox* system architecture can support HAN service protocols that define a service interface in a parsable format. The *Krox* system architecture does not require a specific service interface format, however the format must conform to some standard, e.g. an XML schema, Java interface. A parsable service

interface enables the *Krox* system architecture, or more specifically, a service protocol plug-in, to automatically virtualise remote resources in the local HAN by using the service interface for virtualisation. In addition, the automatic mapping from a service protocol specific interface to web service relies on being able to automatically parse and inspect the service interface. When considering this characteristic against the HAN service oriented architectures we can identify *Krox* system architecture as being suitable for UPnP, DPWS, Jini, HAVi, and OSGi, and not suitable for ZeroConf, and SLP due to their lack of parsable service interface.

2) The design presented for the Jini service plug-in does not cover the full range of potential Jini services because it only supports simple return type, however section 4.3.4.3 described how this could be extended to support any Java type that is serializable and either immutable or implements an interface.
3) While *Krox* system architecture defines a mechanism for intra-HAN service interoperability, and service composition, it does not address the data type inconsistencies and incompatibility between service protocols (the semantic interoperability problem).
4) The support for web service mapping in Jini service protocol plug-in design is limited to simple serializable immutable data types. If the return type of a service interface method is mutable (as discussed in section 4.3.4.3) the automatic mapping to web service is insufficient.
5) The *Krox* system architecture does not support more than a single instance of *Krox* system in the same HAN, e.g. in the case of multiple Internet connection for the same household. A *Krox* system identity represents a HAN rather than a human user, therefore it is not useful for managing different parts of the same HAN by multiple home users.

The next section reiterates the requirements and how they are addressed by the *Krox* system architecture and the design for the service plug-ins for UPnP and Jini.

4.4.1. Requirements

4.4.1.1. Intra-HAN service interoperability

- REQ #1 – *Cross service protocol service composition* – *Krox* system architecture supports service composition of local HAN services through service orchestration of

web services, and an automatic mapping between a service protocol and web service. As discussed above, such a mapping can exist when the service protocol supports a service interface with a parsable format. By mapping the service interface to web services, service composition is enabled for services from different service protocols.

- REQ #2 – *Share composite services* – Through the automatic mapping between BPEL composite service description (WSDL) and UPnP service description, the *Krox* system can dynamically generate UPnP devices that correspond to composite services deployed in the local HAN. With a UPnP service protocol plug-in these dynamic UPnP devices can be seamlessly shared similarly to "live" UPnP devices.
- REQ #3 – *Cross HAN service composition* – The service protocol plug-in maps both local HAN services (in the LNC) and remote HAN services (in the VRM) to web services. Therefore, while service composition only refers to local HAN web services, some of them can in fact be virtual services representing remote services in the local HAN.

4.4.1.2. Inter-HAN service interoperability

4.4.1.2.1. Seamless integration

- REQ #4 – *Enable sharing of HAN services from the local HAN with remote HANs* – Sharing of HAN services in *Krox* system architecture is supported through the implementation of service protocol specific plug-in that is required to implement service virtualisation for the HAN service protocol. In order to allow remote HANs to communicate and identify each other, the *Krox* system architecture has a communication subsystem that enables secure message exchange between remote HANs. Sharing relationships between remote HANs are established by adding a remote HAN to the buddy roster of the local HAN's *Krox* system instance.
- REQ #5 – *Automatic discovery of resources from remote HANs shared with the local HAN* – This is the heart of seamless integration, which is supported in *Krox* system architecture through the use of the automatic service virtualisation techniques. When local HAN services are discovered in the local HAN, remote HANs with which the service is shared are informed using the communication subsystem about the added services and the service is automatically introduced in the remote HAN as a virtual service. The service virtualisation (as demonstrated with UPnP and Jini service protocol

plug-ins) announces a device/service that supports the corresponding service protocol, and therefore can be automatically discovered by service protocol clients in that HAN.

- REQ #6 – *There must be no restriction that prevents sharing the same devices and resources with multiple remote HANs* – The *Krox* architecture does not place limitation on sharing HAN devices and services with multiple HAN. It should be noted that at the same time *Krox* system architecture does not restrict or coordinate simultaneous access to the device by design.
- REQ #7 – *Interaction of applications with remote devices must be identical to the interaction with local of the same service protocol* – This is supported in the *Krox* system architecture by the automatic resource virtualisation of service protocol plug-ins, as demonstrated with UPnP and Jini. The interaction with remote devices and services, facilitated by the service protocol plug-in's VRM is identical to the interaction with a local device/service of the same service protocol.
- REQ #8 – *The system must not require modification to service protocols and must support plug-and-play* – The *Krox* system architecture and the service plug-ins that were designed for UPnP and Jini do not require modifications to be made to the service protocols. When the *Krox* system is deployed, based on its installed service protocol plug-ins it immediately starts to discover devices and services and share them based on the defined sharing configuration.
- REQ #9 – *Independence of access network technology* – the *Krox* system architecture and design does not require specific access connectivity. Only a single Internet connection per HAN is supported.

4.4.1.2.2. Private networks and firewalls

- REQ #10 - *The system must be able to discover and share devices with networks that are using NAT even in the existence of devices with identical IP addresses in multiple HANs* – In *Krox* system architecture and design, private addresses are not used beyond the scope of the local HAN. When informing remote HANs on local devices their IP address is not sent. This is demonstrated with the design for the UPnP service protocol plug-in such that the location of the device in the local HAN is not sent to the remote HAN, and rather the VRM assigns the device a local URL in its network. This approach can be adapted for additional service protocols. Since IP addresses are not sent between HANs, the existence of the same private IP addresses in multiple HANs is not problematic. Another mechanism is required for the UPnP service protocol plug-in to

support out-of-band traffic. Section 4.3.3.2.5 presented a possible solution to this problem.

- REQ #11 – *The system must be able to communicate with remote HANs behind firewalls* – All service protocol traffic in *Krox* system architecture is sent using the communication subsystem. With XMPP based communication subsystem, *Krox* system requires port 5222 to be open (XMPP client to server communication port), however this is a standard instant messaging port (defined in RFC 3920). No additional ports are required for *Krox* system communication between remote HANs.

4.4.1.2.3. Security

- REQ #12 – *All communication with remote HANs must be authenticated* – All messaging between *Krox* system instances is only made through the communication subsystem, which means it is made after authentication has succeeded.
- REQ #13 – *Access control – Sharing must not be automatic and must enable home users to control which resources are shared with which remote HANs* – In the *Krox* system architecture access control is enforced by the Capability Sharing Manager through the interaction with the service protocol plug-ins. Given the home user's configuration of which resources should be shared with which remote HAN, as demonstrated in the design of the service protocol plug-ins for UPnP and Jini, the CSM is consulted before resources are shared, as well as before actions are invoked on local resources on behalf of remote HANs to prevent unauthorised access. While the *Krox* system does not define sharing policies, it is designed to enable enforcing fine-grained access control to device/service/action/content level as illustrated in the design for service protocol plug-ins.
- REQ #14 – *Confidentiality - all traffic between remote HANs must be encrypted* – In *Krox* system architecture the communication subsystem guarantees that all data exchange between remote HANs is encrypted. With XMPP as the underlying IM&P system, this is support at the transport layer with TLS.
- REQ #15 – *Security vulnerability* – some service protocols have inherent security vulnerabilities, in a multi-HAN setting, such vulnerabilities must be confined to a single HAN. While it follows from supporting a multi-HAN system that vulnerabilities for the HAN increase, it must be shown how *Krox* system defends against such threats and vulnerabilities. In order to analyse the security threats and define methods for defending

against them both at the system level and in the service protocol plug-in level, a security analysis is given in section 6.4.

4.4.1.2.4. Performance

A separate performance evaluation will be performed to demonstrate that *Krox* system satisfies the performance requirements (REQ #16-REQ #19) using a prototype implementation (see sections 6.2-6.3).

4.4.1.2.5. Extensibility

- REQ #20 – *Extensibility to additional HAN service protocols* – One of the key characteristics of the *Krox* system architecture is its extensibility through the plug-in based architecture and the plug-in framework. The design for *Krox* identified the characteristics of HAN service protocols that can be supported in the *Krox* system architecture, and the plug-in framework provides the structure for such support complemented by the communication subsystem and the capability management.

4.4.1.2.6. Manageability

- REQ #21 – *Dynamic relation management with remote HANs* – through the use of IM&P and more specifically XMPP, agreement to share local HAN resources with remote HANs is reduced to adding the *Krox* system identifier of the remote HAN (i.e. its IM&P identifier) to the buddy roster of the *Krox* system in the local HAN. Remote HANs can be added, deleted, and blocked, following IM&P user management patterns.
- REQ #22 – *Pause/resume sharing with remote HANs* – through the use of IM&P and more specifically XMPP in *Krox* system design, the home user can change the status of the otherwise "always on" *Krox* system to "unavailable" which results in termination of all virtual resources in remote HANs that correspond to the local HAN's resources.
- REQ #23 – *The system must not require manual configuration of the home gateway and its administration must be appropriate for non-technical users* – *Krox* system architecture does not require any manual configuration to the home gateway. Configuration of the *Krox* system architecture is restricted to adding and removing remote HANs, which is equivalent to managing a buddy roster for IM applications and

controlling the sharing status of the system, which is equivalent to the connection status of an IM system.

4.4.2. Summary

The previous section described how *Krox* system architecture and design address the system requirements as they were defined in section 3.4. Some of the requirements (REQ #15-19) still need to be evaluated with a prototype implementation.

This chapter presented the *Krox* integrated plug-in based architecture with a plug-in framework and an extensible event model that supports the design of multiple HAN service protocol plug-ins for intra-HAN and inter-HAN service interoperability. The design approach was demonstrated using multiple service protocols, UPnP and Jini. With resource virtualisation techniques, leveraging an extensible event model, resources from remote HANs are made available to the local HAN as virtual resources enabling seamless integration with client applications in the local HAN. By mapping from a service protocol service interface to web services, and service orchestration, *Krox* system architecture supports intra-HAN service interoperability and service composition. *Krox* system architecture builds on the IM&P user metaphor for defining the relationships between remote HANs that agree to share resources. IM&P also provides a secure and scalable communication subsystem for exchanging messages between *Krox* system instances in remote HANs. Finally the capability-sharing manager provides an infrastructure for enforcing fine-grained sharing policies through interaction with the service protocol plug-ins.

An important aspect of the *Krox* system architecture is its modularity through the loose coupling between its components. The clients of the communication subsystem are abstracted from the fact that it is based on IM&P system, such that it can be replaced with an alternative system that provides the necessary features of messaging and status change notifications. The modularity of the service composition subsystem enables changing the format of the generated services without changing the system, while also supporting evolution and addition of service protocol plug-in releases. Similarly the BPEL service orchestration engine can be replaced with an alternative service orchestration approach.

The next chapter presents a prototype implementation of *Krox* system architecture with service protocol plug-ins for UPnP and Jini, and service composition supported by web services and BPEL.

Chapter 5
IMPLEMENTATION

The previous chapter presented the *Krox* high-level architecture and a system design for integrated intra-HAN and inter-HAN service interoperability supporting the requirements presented earlier in this thesis. This chapter presents a prototype implementation of the *Krox* system architecture and design with service protocol plug-in implementations for UPnP and Jini. The system implementation has the following objectives:

- Demonstrates and validate the utility of the system design for solving the problem addressed by this thesis.
- Provide a performance measurement instrument.
- Provide a grounding for a security analysis.

The following sections describe the details of the system prototype implementation as well as the service protocol plug-ins and client application prototype.

5.1. Prototype system

5.1.1. Technology selection

- Programming language – the system prototype is implemented with core Java using JDK6[26]. Java provides the desired platform independence and enables deployment on multiple operating systems.
- IM&P server – the system prototype uses the open source OpenFire XMPP server version 3.6.4[27]. OpenFire was chosen because of its simplicity for install and administration and its well-established developers community.
- IM&P client – the system prototype uses the open source Smack SDK version 3.1.0[28] for the implementation of client-side XMPP application. Smack provides an easy to use Java API for interaction with the XMPP server.
- Orchestration engine – the system prototype uses Apache ODE 1.3.5[29] for executing service orchestrations. The advantage of Apache ODE is that is an open source BPEL implementation, is lightweight and extensible.
- Application server – The system prototype uses Apache Tomcat version 6.0.18[30], which is a lightweight servlet container, both for hosting the ODE orchestration engine, as well as the automatically generated web services.
- UPnP SDK – the system prototype uses CyberLink Java version 2.0[31] as an SDK for the interaction with UPnP devices and services. CyberLink is a lightweight library that provides tools infrastructure for interaction with UPnP networks.
- Jini – Jini service protocol plug-in was implemented using Jini 2.1[32].

[26] http://www.oracle.com/technetwork/java/javase/downloads/index.html

[27] http://www.igniterealtime.org/downloads/index.jsp

[28] http://www.igniterealtime.org/downloads/index.jsp

[29] http://ode.apache.org/getting-ode.html

[30] http://tomcat.apache.org/

[31] http://sourceforge.net/projects/cgupnpjava/files/clinkjava/

[32] http://www.jini.org/wiki/Jini_Starter_Kit_2.1_-_Java

5.1.2. Implementation components

The main components of the system implementation (depicted in figure 19) are described in the following sections:

1) MessagingPresenceManager – implements the communication subsystem, responsible for the messaging and presence infrastructure (section 5.2)
2) CapabilitySharingManager – implements the capability sharing management for checking access control for local resources (section 5.3)
3) KroxGateway – the main class of the implementation, responsible for instantiating the MessagingPresenceManager and the PluginManager (section 5.4)
4) PluginManager – responsible for management of service protocol plug-ins – loading, unloading, and register the plug-ins with the MessagingPresenceManager (section 5.4)
5) ILocalNetworkController – an interface definition for an LNC, must be implemented by service protocol plug-in (section 5.4.1)
6) IVirtualResourceManager – an interface definition for a VRM, must be implemented by service protocol plug-in (section 5.4.2)
7) UPnPLocalNetworkController – UPnP service protocol plug-in implementation for LNC (implements the ILocalNetworkController interface) (section 5.5)
8) UpnPVirtualResourceManager – UPnP service protocol plug-in implementation for VRM (implements the IVirtualResourceManager interface) (section 5.5)
9) JiniLocalNetworkController – Jini service protocol plug-in implementation for LNC (implements the ILocalNetworkController interface) (section 5.6)

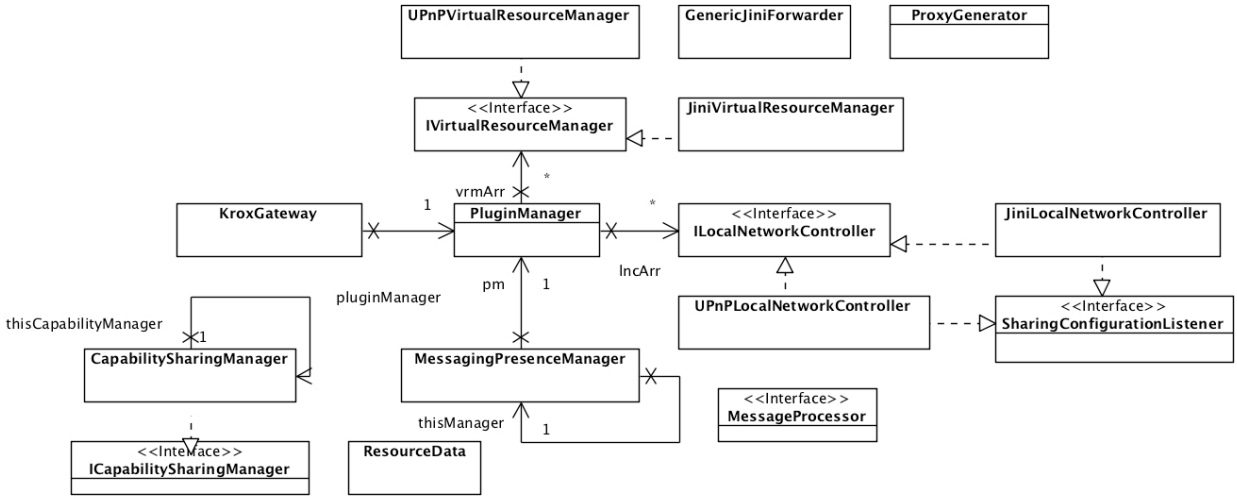

Figure 19 *Krox* System Prototype UML Class Diagram

10) JiniVirtualResourceManager – Jini service protocol plug-in implementation for VRM (implements the IVirtualResourceManager interface) (section 5.6)

5.2. Communication subsystem

The communication subsystem is encapsulated in the MessagingPresenceManager object, providing messaging and presence services to other components of the system, specifically to plug-ins. It is important to note that while the MessagingPresenceManager relies on IM&P for exchanging messages with *Krox* system instances in remote HANs, it abstracts the use of IM&P technology from the users of its API, therefore the interface it provides to its users is implementation agnostic. The implementation could be changed without affecting the users of the communication subsystem. The users of the communication subsystem API, e.g. plug-in components can send and receive messages using the MessagingPresenceManager, and in addition they are notified on changes in the list of HANs with which they need to communicate and the status (presence) of these HANs.

The MessagingPresenceManager (figure 20) implements several interfaces from the Smack IM&P client library in order to provide messaging and presence capabilities:

1) RosterListener – Once authenticated, the MessagingPresenceManager subscribes

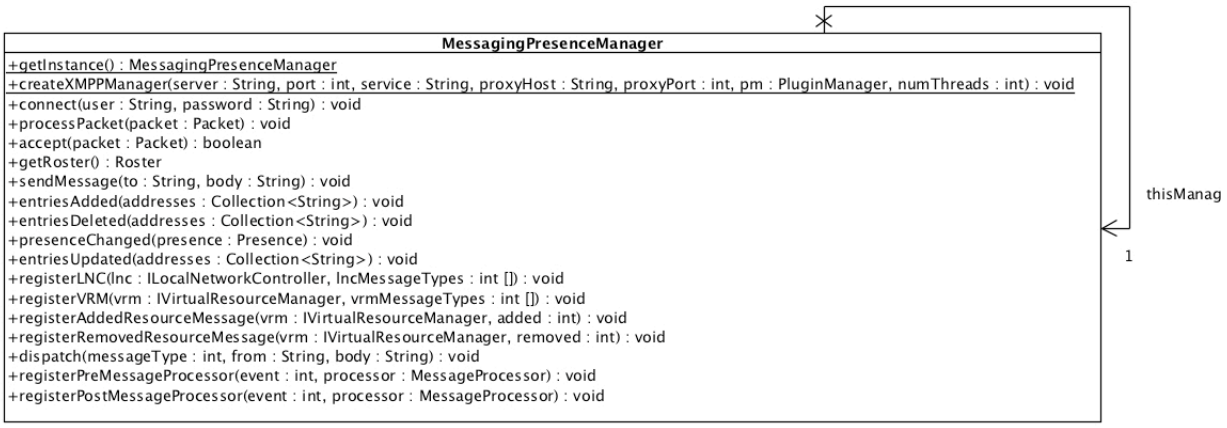

Figure 20 Communication Subsystem Implementation

itself as a listener for changes in the buddy roster. This is important for being notified when buddies are added or removed from the buddy list, which correspond to remote HANs with which sharing should be initiated or terminated. When buddies are added or removed from the list, the Capability Sharing Manager must be updated such that its model of remote HANs with which sharing is allowed or disallowed is updated. The RosterListener also allows the MessagingPresenceManager to respond to presence changes for members of the buddy list. The MessagingPresenceManager handles the cases where buddies change the state to “available” and when a buddies change their state from “available” to anything else.

2) PacketListener – The MessagingPresenceManager registers itself with the XMPP server as a packet listener, which means it intercepts received messages. The handling of messages is explained below.

Krox system instances have a unique identifier that is used for message exchange between them. For simplicity, this identifier reuses the communication subsystem identifier, which corresponds to a *Krox* system instance, however this is opaque to users of the API (i.e. service protocol plug-ins) that use this identifier for sending messages to system instances in remote HANs. The communication system uses this identifier to route messages to the correct system instance in a specific HAN.

5.2.1. Messaging and presence

The core functionality of the MessagingPresenceManager is to enable sending messages to the *Krox* system in remote HANs and dispatching and processing messages received from *Krox* system instances in remote HANs. Another functionality provided by the MessagingPresenceManager is access to the buddy list (the list of remote HANs with which

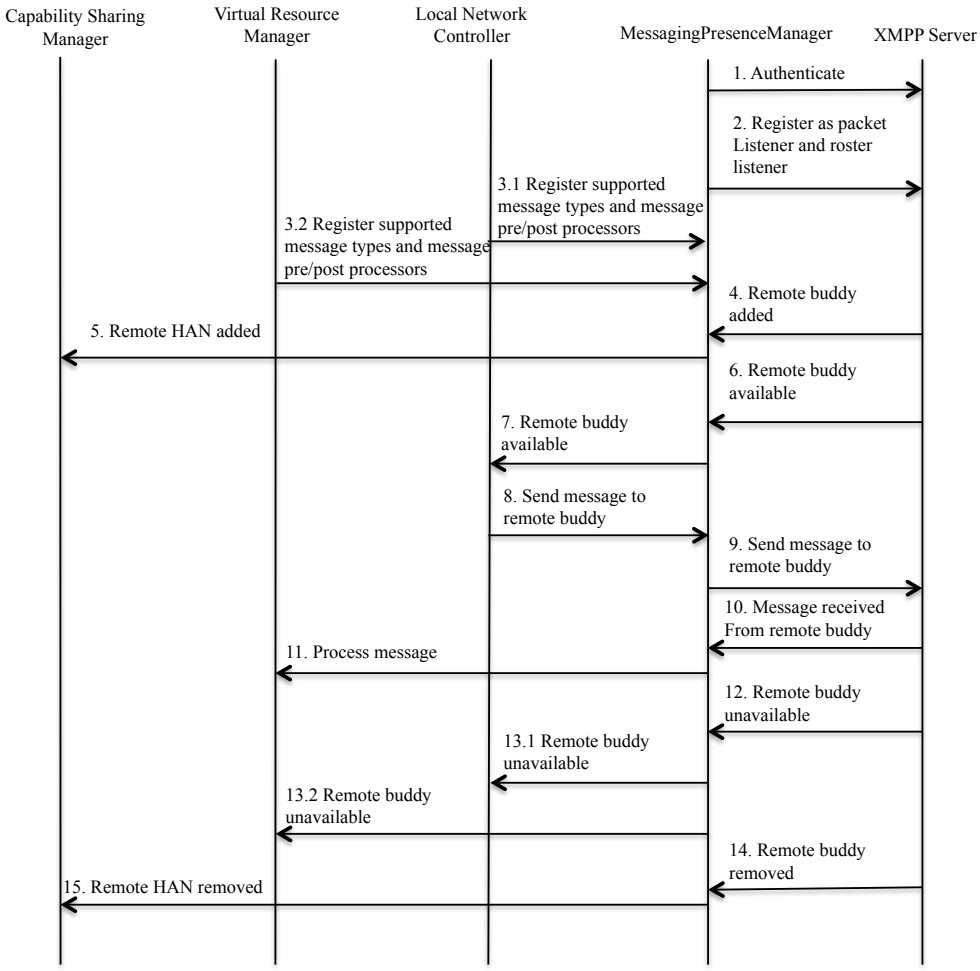

Figure 21 *Krox* Communication Subsystem Interaction

services/devices can be shared) which includes up to date presence status of its members. This is needed for propagating discovery announcement to remote HANs with which devices/services are shared. The sendMessage() method takes several parameters: the address of the destination, the type of the message, and the body of the message. The MessagingPresenceManager concatenates the type as a prefix to the body and sends the messages to the destination. Message dispatching in the destination is based on maintaining a mapping of supported message types against which component needs to be called for processing the message. When the system loads a plug-in, the plug-in's supported message types are registered with the MessagingPresenceManager along with the interface that needs to process them. In addition the plug-in components can register pre/post processors that will be called by the MessagingPresenceManager before an incoming message is processed by the relevant component, or before a message is send to a remote HAN. A plug-in's interaction with the MessagingPresenceManager is depicted in figure 21.

For performance purposes the actual processing of the message (other than message dispatching) is performed in another thread, rather than the message dispatching thread. For this purpose the MessagingPresenceManager has a thread pool and the processing of a messages is made by wrapping the call to the processing method in a runnable task wrapper and handing it over to the thread pool for execution. This enables the MessagingPresenceManager to offer full concurrency in handling incoming messages. Incoming and outgoing messages are also handled in separate threads. At all times there is only a single instance of the MessagingPresenceManager in the system. All messages exchanged between the authenticated *Krox* system instances in remote HANs are encrypted, which is handled by the underlying XMPP framework.

5.3. Capability sharing manager

As mentioned in the previous chapter, fine-grained modelling of resources and the actual definition of sharing policies is out of the scope of this thesis. However in order to demonstrate the interaction between the service protocol plug-in and the CapabilitySharingManager, a default stub was implemented defining the contract between the plug-in implementation and the CapabilitySharingManager. The interface and implementation described in figure 22 as a UML class diagram, enables updates and query

the managed resource model. The CapabilitySharingManager is expected to have a persistent storage such that when it is started it loads the information about sharing policies and remote HANs with which sharing has been configured. The CapabilitySharingManager interface enables the communication subsystem to update the model when sharing with remote HANs is agreed as implied by adding a remote HAN to the local HAN's roster by calling *remoteHANAdded* with the corresponding remote HAN identifier. Similarly when the relation is terminated, the communication subsystem updates the model with the removed HAN identifier by calling *remoteHANRemoved*.

Service protocol plug-ins update the CapabilitySharingManager when resources are added and removed giving the resource identifier, and possibly additional data that can be used when applying sharing policies, such as device model, device vendor, or any other protocol specific information that can be useful as represented using the *ResourceData* object. When the plug-in's LNC discovered a new resource, it queries the CapabilitySharingManager for the identifiers of the remote HANs with which the resource is shared. When a remote HAN changes its status to available, the LNC queries the CapabilitySharingManager for the resources that need to be shared with that remote HAN.

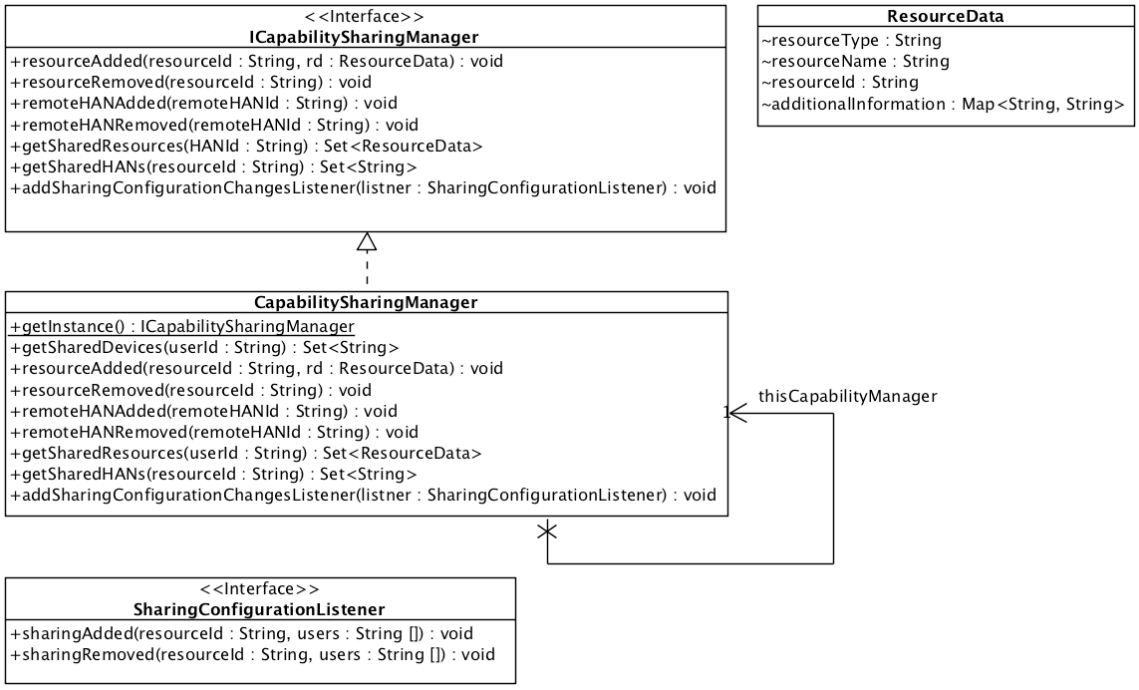

Figure 22 Capability Sharing Manager UML Class Diagram

The CapabilitySharingManager enables components to register a listener (*SharingConfigurationListener*), which is called when the sharing configuration for a resource is changed. This enables the LNC components of service protocol plug-ins to be notified when access-sharing configuration is changed and respond appropriately.

For the purpose of performance evaluation with maximal load, the only sharing policy supported by the *Krox* prototype implementation is share-all which means all resources are shared with all other remote HANs.

5.4. Service protocol plug-in framework

The outer most object of the implementation is the KroxGateway (see figure 23). The KroxGateway is responsible for instantiating the necessary components of the *Krox* system and initiate their bootstrap process. In its bootstrap, the KroxGateway starts the MessagePresenceManager and PluginManager and initiates their bootstrap. The PluginManager is responsible for loading, unloading and managing service protocol plug-ins. The mechanism used by the PluginManager to locate plug-ins is abstracted from the rest of the system. When the system is started, the PluginManager starts looking for available plug-ins. In the prototype implementation the UPnP and Jini service protocol plug-ins are pre-loaded. The PluginManager registers plug-ins with the MessagingPresenceManager to support message exchange with system instances in remote HANs. In addition the PluginManager registers the LNC's sharing configuration change listener with the CapabilitySharingManager.

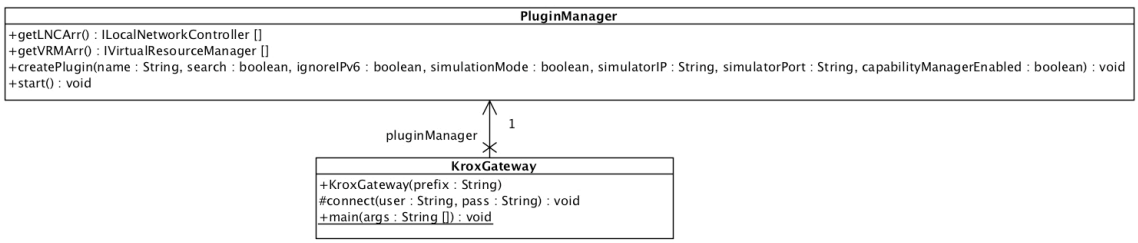

Figure 23 Plug-in Framework Implementation

A service protocol plug-in must contain two modules, the LNC and VRM. The Local Network Controller must implement the Java interface: ILocalNetworkController and the Virtual Resource Manager must implement the java interface: IVirtualResourceManager. For the purpose of the prototype it is assumed that all HANs have the same configuration of plug-ins, therefore unsupported messages are never received to the MessagingPresenceManager. If they were received, the MessagingPresenceManager will ignore all messages which have a type for which no LNC or VRM have registered.

The design of the LNC and VRM interfaces (figure 24) follows the core event model described in section 4.2.2.2-4.2.2.3. The LNC and VRM implementation extend the core event model with additional events required to facilitate the required interaction between local and remote components. Additional events are expressed using message types that are dynamically registered with the PresenceMessagingManager, which are forwarded to the registered LNC. The VRM interface provides a tighter event model, where the events of resource added/removed are part of the generic event model. The VRM implementation is required to declare the message type that corresponds to add/remove of remote resource. This enables the plug-in framework to link this message type to the required VRM implementation handling addition or removal of remote shared resources, rather than leaving this as an internal contract between plug-in components. The reason for requiring the VRM to explicitly support resource added/removed events is that this is an essential part

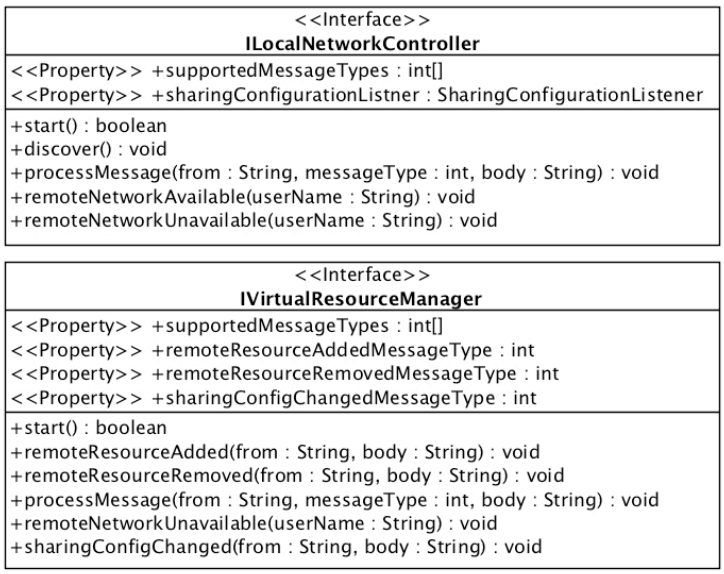

Figure 24 (a) LNC Interface (b) VRM Interfaces

of the plug-in behaviour. The VRM is also required to specify the message type for sharing configuration change, such that it can be linked to the handling of changes in the sharing configuration in remote HAN.

The extensible event model enables the definition of specific join points (as discussed in section 4.3.3).

5.5. UPnP service protocol plug-in

The UPnP service protocol plug-in includes an implementation for the ILocalNetworkController (UPnPLocalNetworkController) and the IVirtualResourceManager (UPnPVirtualResourceManager). Figure 25 is a UML class diagram corresponding to the plug-in implementation. The UPnP LNC and VRM use the CyberLink Java UPnP SDK for interaction with UPnP devices. The UPnP LNC implements the INotifyListener and ISearchResponseListener interfaces which enable it to listen to search responses (triggered by its search requests) and device announcements (by listening to the multicast address in the local HAN). The UPnP LNC also implements an HTTPRequestListener for listening to event notifications corresponding to event subscriptions made on behalf of remote HANs. In addition the LNC implements the SharingConfigurationListener to be notified by the CSM when the sharing configuration of a resource in the local HAN changed. The registration with the CSM is made during the bootstrap of the LNC. The UPnP VRM implements the SearchListener interface for being able to receive and respond to search requests made in the local HAN. The UPnP VRM implements the HTTPRequestListener for responding to HTTP requests including description requests, SOAP requests, and event subscription requests.

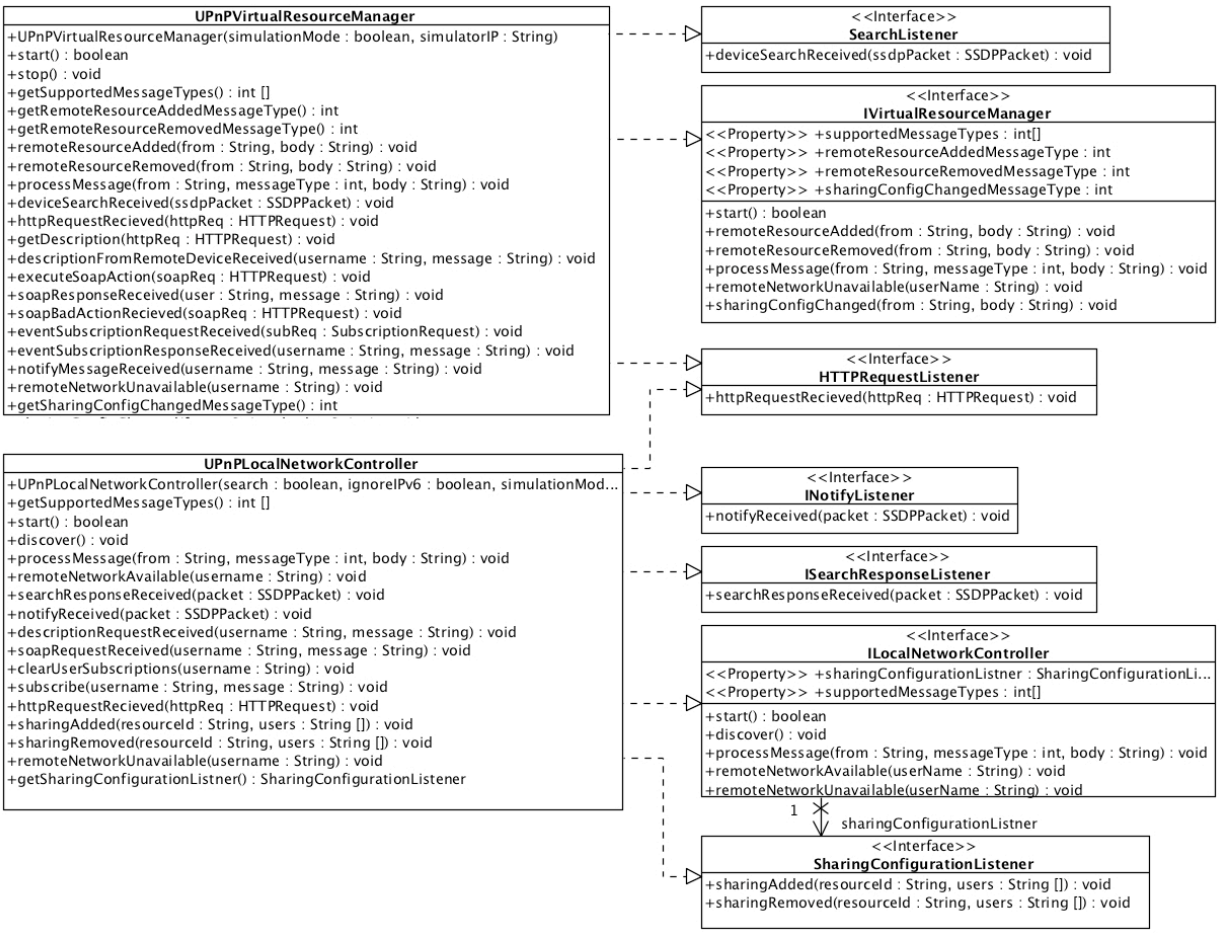

Figure 25 UPnP Plug-in Implementation

The following sections describe how the design of UPnP service plug-in as presented in the previous chapter was implemented.

5.5.1. Discovery

The UPnPLocalNetworkController implements discovery through two mechanisms:

1) Sending SSDP search request and listening to responses
2) Listening to device presence announcements sent to the SSDP multicast address

The discovery interaction between the plug-in components in remote HANs is illustrated as a sequence diagram in figure 26. Steps 1-11 in figure 26 describe how device and service announcements in the local HAN (HAN1) are reported to all other HANs (HAN2 in figure 26) with which the device or service is shared. Steps 12-19 in figure 26 represent how SSDP *byebye* messages are forwarded to remote HANs and handled in their VRM after the LNC updated its local repository. The VRM updates its local announcement repository and repeats the *byebye* for the corresponding virtual device/service in its local HAN (HAN2 in figure 26). Steps 20-21 describe how the VRM responds to search requests made by control points in its local HAN (HAN2). The VRM uses the search request processing for cleaning stale announcements that have expired and announcing corresponding *byebye* messages.

When a remote HAN changes its presence status to available the MessagingPresenceManager calls the UPnP LNC *remoteNetworkAvailable*() giving the remote HAN identifier. The LNC responds by sending the remote HAN all of the

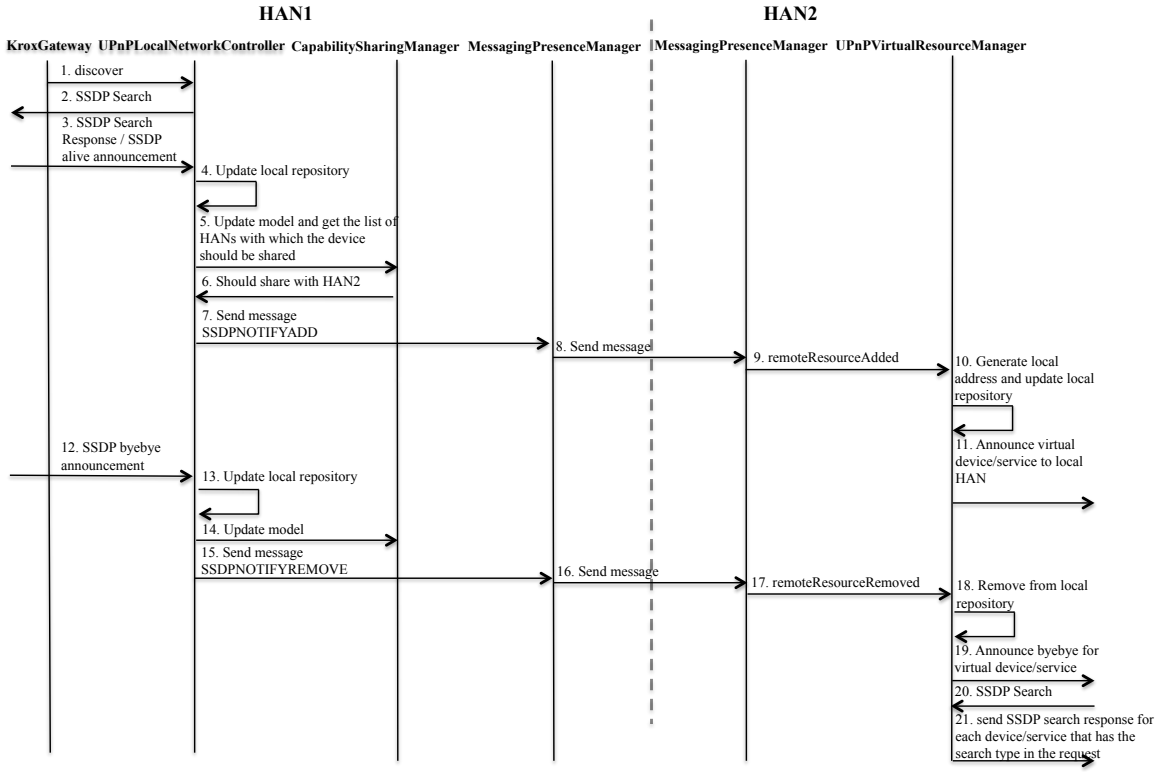

Figure 26 UPnP Plug-in Discovery Protocol Implementation

announcements in the repository that are shared with that remote HAN. Since the LNC's repository should only cache SSDP announcements for their duration in the network as was reported in the SSDP message (*cached-control* parameter in the SSDP announcement), before the message is sent to the remote HAN, its expiration time is checked and if it expired, it is removed from the repository and will not be sent to the remote HAN.

If the LNC is notified on a change in the sharing configuration it needs to update the affected remote HANs on the change. If sharing was added for a resource, the LNC sends an announcement about this resource to the remote HAN. If sharing was removed, the LNC sends a *byebye* announcement for this resource to the corresponding HAN.

As discussed in the previous chapter, the LNC ignores announcements made by the VRM in its local HAN, avoiding unintentional re-sharing of remote devices, which are not owned by the local HAN.

5.5.2. Description

In order to provide description documents for devices and underlying services on behalf of virtual remote devices in the local HAN, the local VRM listens on a local HTTP port. Figure 27 describes the interaction between the control point in HAN1 and the VRM representing locally (in HAN1) a device from a remote HAN (HAN2). The VRM in HAN1 extracts the identifier of the *Krox* system hosting the "live" device and the device UUID from the HTTP GET request. Before the request is sent to the remote HAN, the VRM checks in its description cache if it already has a description for this device or service as received from the remote HAN, if so, the result is immediately posted to the HTTP requester. If not it follows steps 3-18 in figure 27. Caching can significantly shorten the time it takes to retrieve device/service description from remote HANs and consequently reduce the inter-HAN traffic and the load on devices.

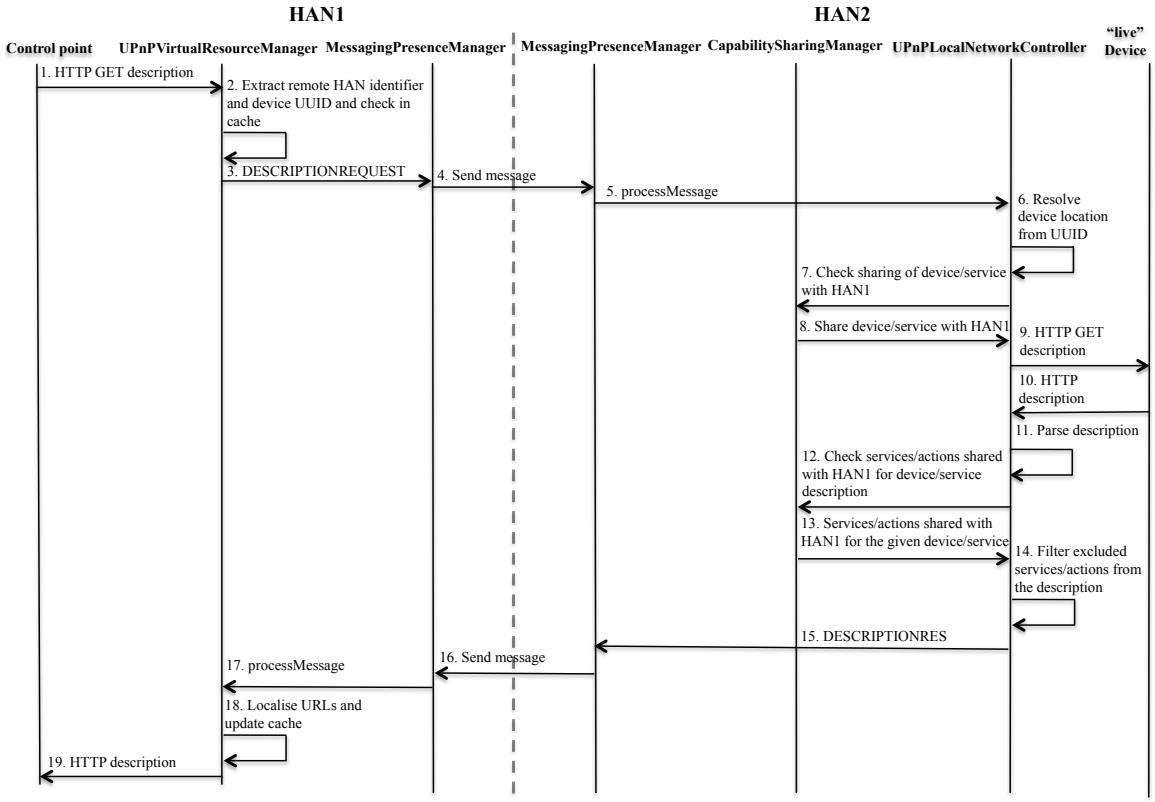

Figure 27 UPnP Plug-in Description Protocol Implementation

In order to optimise the performance of requests for description processing, the VRM also piggybacks multiple requests for description on pending requests blocking for results. When a request is currently waiting for a result from a remote HAN, and additional description request is received for the same resource, the request will block, but will not send another request to the remote HAN. Instead when the result returns for the first request the same result will immediately be published to all other waiting requesters. This optimisation as well as the caching of descriptions saves both time and inter-HAN traffic, and reduces the load on the remote "live" device.

When a description request is sent to a *Krox* system instance in a remote HAN, the thread handling the description request in the VRM blocks for as long as the result has not returned from the remote HAN or a timeout expires. When the message is received in the remote HAN, the VRM notifies the waiting thread and the result is posted back the waiting control point and cached for future reference. If the timeout expires or an error message was returned from the remote LNC, then the error is posted back to the local control point with

an HTTP status reflecting the error. While the thread is blocked, concurrent description requests can be processed.

If a message from a remote HAN indicating a sharing configuration change that affects the local HAN is received to the VRM, it is required to remove the relevant resource's cached description, as it may have been invalidated.

5.5.3. Control and post processing

The control interaction with remote devices is depicted in figure 28. When an HTTP POST containing a SOAP request is received by the VRM from a control point (HAN1 in figure 28), the VRM extracts the identifier of the *Krox* system hosting the "live" device and the device UUID from the HTTP SOAP request and sends this information to HAN2 where it is processed by the local UPnP LNC (steps 1-5 in figure 28). The LNC in the remote HAN (HAN2) resolves the device location from the given device UUID and verifies with the

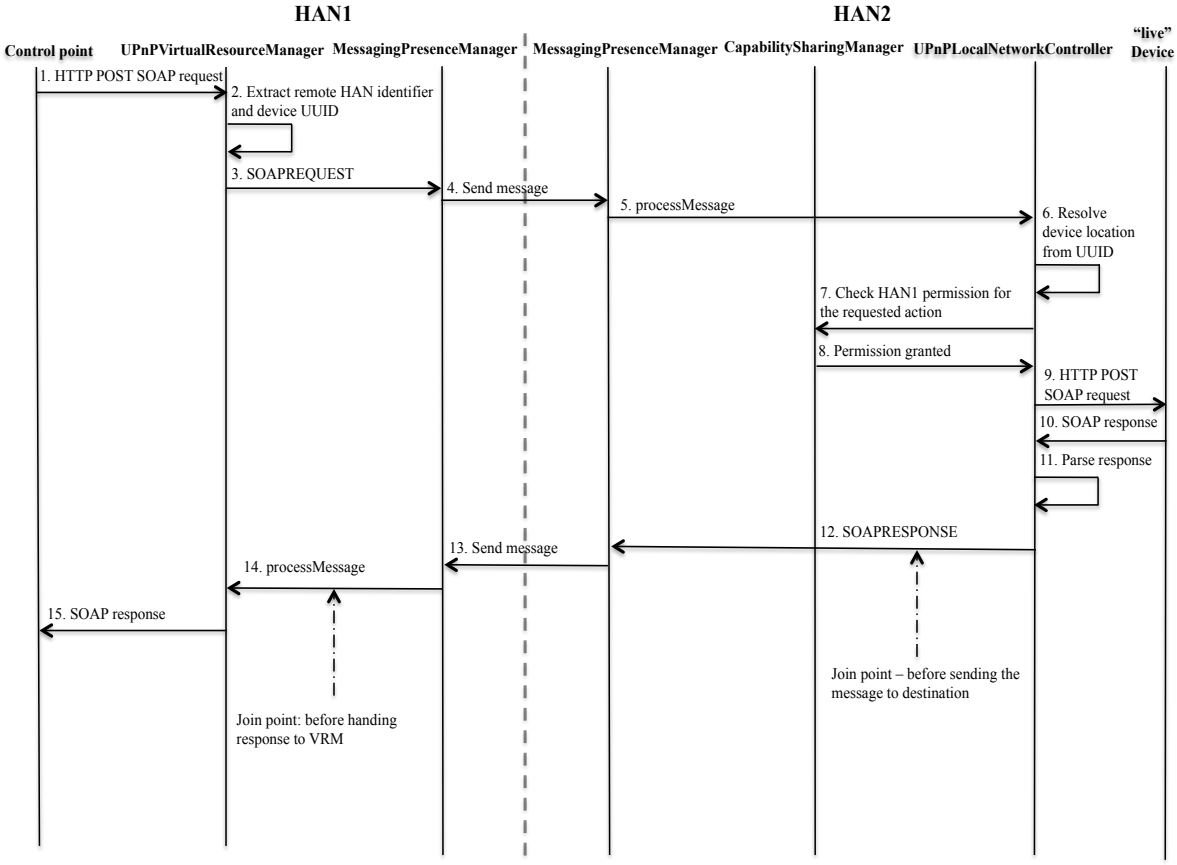

Figure 28 UPnP Control and Post Processing Protocol Implementation

CSM that the requesting HAN is permitted to execute the requested action. If it is, the SOAP request is sent to the "live" device. When the SOAP response is received from the device, the LNC parses the result and then sends the result back to the requesting HAN (steps 6-12 in figure 28).

Before the SOAP response is sent by the communication subsystem to the remote HAN (HAN1), the UPnP plug-in implemented an extension in the scope of all UPnP devices (as opposed to a specific device type) to replace all private IP addresses in a SOAP response (if any) with the external IP address of the HAN. This approach is different than the design as presented in section 4.3.3.4. As discussed in the previous chapter, the actual streaming of data is not an integral part of the UPnP protocol, and hence is not an integral part of the UPnP plug-in, however for being able to test the plug-in and to demonstrate that it behaves as expected, some approach for enabling client application to stream media from remote HAN is required. For simplicity of testing the implementation assumes that the firewall and home gateway have been configured to allow access from the remote HAN with which the device is shared to the device. While this implementation is not appropriate for the HAN because of the required manual configuration, it was sufficient for testing and demonstration purposes of the *Krox* system architecture.

Similarly to the description protocol implementation, while waiting for the response from the remote HAN, the local VRM's thread processing the SOAP request is blocking. When the SOAP response is received, the waiting thread is notified. Again, multiple concurrent SOAP requests can be invoked. In cases where a timeout expires before the response is received an error will be posted. In the prototype system, 5 seconds is the default timeout for a remote device SOAP request to return. This timeout was defined as the maximum time a control point application is willing to wait for an invocation to return.

5.5.4. Eventing

The final protocol in the UPnP service plug-in implementation is eventing (figure 29). An event related request (delivered over HTTP) could be either an event subscription request or request for removal of an event subscription. An event subscription request can either be a new subscription request or a subscription renewal. When the VRM receives an event subscription request from a local control point for a remote service it represents, the VRM

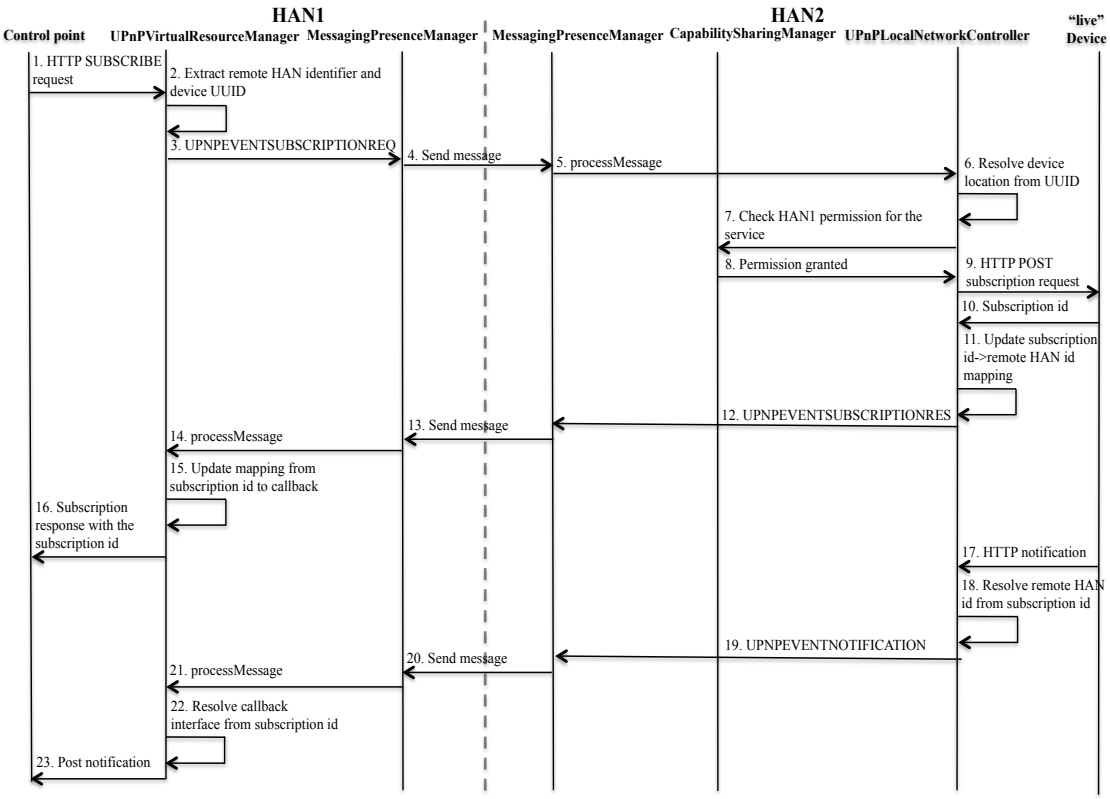

Figure 29 UPnP Plug-in Eventing Protocol Implementation

extracts the identifier of the *Krox* system hosting the "live" device and the device UUID from the subscription URI. The VRM then sends a subscription request to the remote HAN hosting the "live" device (steps 1-4 in figure 29). In the remote HAN (HAN2 in figure 29), the LNC handles the subscription request by resolving the device from the given UUID. If the requesting HAN (HAN1) is allowed to subscribe for events on the requested service, an HTTP subscription request is sent to the device, which responds with a subscription id (steps 5-10 in figure 29). The LNC maps the subscription id to the identifier of the *Krox* system requesting the event subscription. The subscription id is then sent using the MessagingPresenceManager to the requesting HAN (HAN1) where it is passed to the VRM (steps 11-14 in figure 29). The VRM maps the subscription identifier to the callback interface, provided in the original subscription request, such that the subscription identifier can be used to resolve event notifications and direct them to the subscribing control point. The subscription identifier is then sent back to the subscribing control point (steps 15-16 in figure 29). The VRM's thread processing the event subscription blocks until the response is received from the remote HAN (HAN2) with a subscription identifier (SID). This blocking does not prevent additional HTTP requests from being processed concurrently as the processing is multithreaded.

When a notification corresponding to a subscription is received from the "live" device to the LNC, the LNC extracts its subscription id and resolves the identifier of the remote *Krox* system instance corresponding to this subscription id. The notification is then forwarded to that HAN where the VRM processes it. The VRM resolves the callback interface from the subscription id, and notifies the corresponding control point (steps 17-23 in figure 29).

If the LNC is notified on a removal of sharing of a resource on device/service on which subscription has been made from remote HAN, the subscription is terminated and an error is sent to the subscribed remote HAN.

5.5.5. Testing

The *Krox* system was tested with UPnP plug-in in a setup that included multiple HANs distributed in multiple geographical locations. The purpose of testing was to validate that the implementation conforms to the plug-in design as described in the design chapter.

The testing technique used logging such that each event from the core event model and plug-in extension is printed to a log file with a timestamp and additional information. Each message was assigned an identifier, such that the messages across multiple HANs' logs can be correlated. Through this correlation, it verified that the implementation of the event model for the UPnP plug-in follows the design by validating that the set of related printed log messages follows order indicated by the design. This technique was used to validate the implementation of the discovery, description, control, and eventing of the UPnP plug-in in a controlled environment with a small number of UPnP devices and control points. Testing has shown that the implementation worked as expected and that the event model defined by the design of the plug-in was sufficient to express the required interaction between the plug-in components for supporting sharing of UPnP devices across multiple HANs. Additional testing was performed using the client application described in section 5.8.

5.5.6. Summary

The UPnP service plug-in implementation, described in the above sections, demonstrates the feasibility of the *Krox* system design for a UPnP service plug-in. UPnP service protocol plug-in was implemented supporting service discovery, description, invocation, and eventing layers of the UPnP service protocol. Table 4 summarises the event model (core and extensions) that is included in the UPnP plug-in implementation.

A number of performance optimisations have been applied in order to support high throughput with low latency:

1) *Description caching* – the VRM caches description documents retrieved from remote HANs such that they only need to be fetched once.
2) *Description fetch piggyback* – in order to optimise the first retrieval of a remote device/service description, even in the case of multiple concurrent requests for the same device/service description from local control points, the description is fetched only once and is published to all requesting control points in the same HAN.
3) *Local device/service announcement caching* – in order to respond promptly when a remote HAN changes its status to "available", all local device/service announcements are cached, such that the communication with the remote HAN does not require the LNC to initiate a UPnP search in the local HAN.
4) *Search response bundling* – When communication is started between two HANs, the information about shared devices/services is sent in a single message containing information about all shared devices/service, rather than sending a single message corresponding to each shared device/service. This optimization enables to reduce the overhead of many small messages (each message corresponding to a single device/service).

Event	Initiated By	Processed By	Description
DISCOVER	Plug-in Manager	UPnP LNC	Initiate discovery in the local HAN through search and listen to device announcements
REMOTEHANAVAILABLE	Communication subsystem	UPnP LNC	A remote HAN from the buddy roster became “available” - need to send it information about all local shared resources
REMOTEHANUNAVAILABLE	Communication subsystem	UPnP LNC, UPnP VRM	A remote HAN from the buddy roster became “unavailable” – need to clean related resources in LNC and VRM
SHARINGCONFIGURATIONADDED	Capability Sharing Manager, UPnP LNC	UPnP LNC, UPnP VRM	Sharing configuration changed, additional resources should be shared now with remote HANs. VRM needs to clean description cache
SHARINGCONFIGURATIONREMOVED	Capability Sharing Manager, UPnP LNC	UPnP LNC, UPnP VRM	A resource that was previously shared is not longer allowed for sharing. The LNC receives a notification from the CSM and sends an update to the relevant remote VRMs. Both LNC and VRM need to clean relevant resources
UPNPSSDPNOTIFY	UPnP LNC	UPnP VRM	Device/service announcement. Mapped to the core event of resource added/removed in the VRM
UPNPDESCRIPTIONREQUEST	UPnP VRM	UPnP LNC	Request for remote device/service description
UPNPDESCRIPTIONRES	UPnP LNC	UPnP VRM	Device/service description response
UPNPSOAPREQUEST	UPnP VRM	UPnP LNC	Remote service action invocation request
UPNPSOAPRESPONSE	UPnP LNC	UPnP VRM	Remote service invocation response
UPNPEVENTSUBSCRIPTIONREQ	UPnP VRM	UPnP LNC	Remote event subscription request
UPNPEVENTSUBSCRIPTIONRES	UPnP LNC	UPnP VRM	Remove Event subscription response
UPNPEVENTNOTIFICATION	UPnP LNC	UPnP VRM	Remote event notification

Table 4 UPnP Plug-in Event Model

All of the UPnP control protocol traffic is sent over the secure communication subsystem and is therefore encrypted and can be sent and received only by authenticated participants (i.e. *Krox* system instances).

The testing performed with a number of remote HANs has shown that the UPnP plug-in works as expected, and that the event model described above is sufficient to represent the required interaction between the LNC and VRM to support seamless service integration of remote services with local client applications.

5.6. Jini service protocol plug-in

The Jini service protocol plug-in (figure 30) includes an implementation for the ILocalNetworkController (JiniLocalNetworkController) and the IVirtualResourceManager (JiniVirtualResourceManager). The GenericJiniForwarder is used by the Jini VRM for generating dynamic implementation for remote shared Jini services. The ProxyGenerator is used by the Jini VRM to export a Jini proxy for the generated service. In addition the Jini

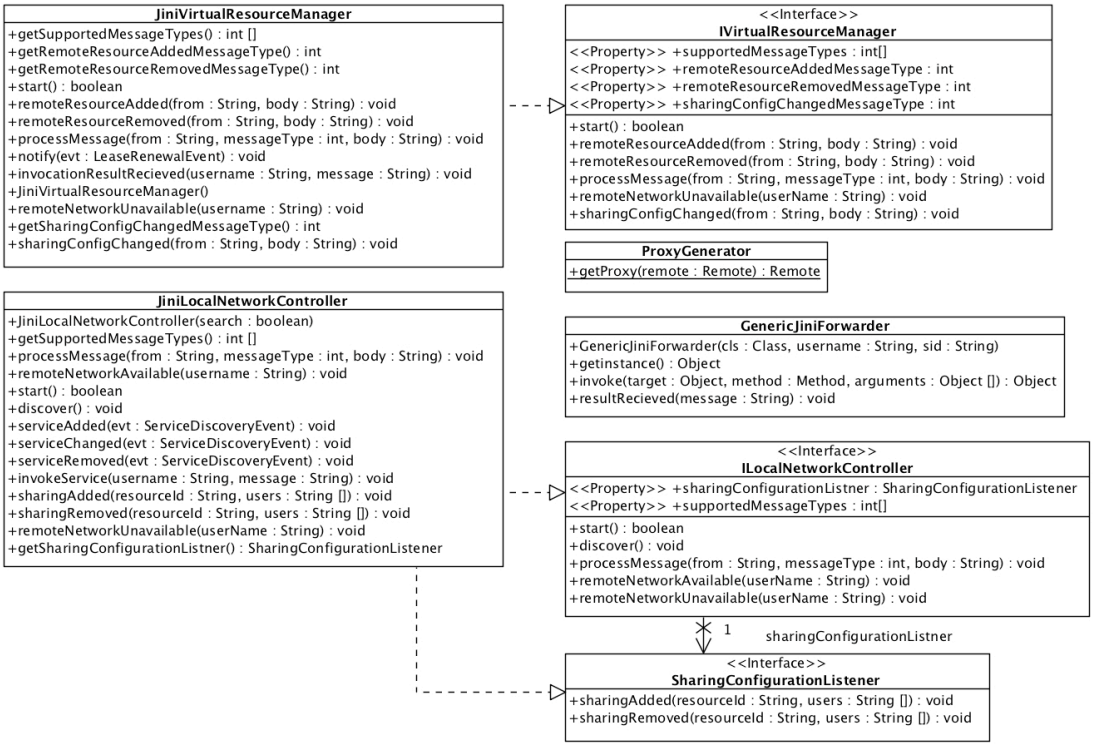

Figure 30 Jini Plug-in Prototype Implementation UML Class Diagram

LNC implements the SharingConfigurationListener to be notified by the CSM when the sharing configuration of a resource in the local HAN changes. The registration with the CSM is made during the bootstrap of the LNC.

The Jini plug-in event model is completely different from the one described in the previous section for UPnP. In addition the mechanism used by the Jini VRM to represent remote services in the local HAN is significantly different than the one used by the UPnP plug-in due to the differences between the service protocols.

The following sections describe how the design of Jini service plug-in as presented in the previous chapter was implemented. Section 5.6.1 describes the implementation of Jini service discovery and automatic service virtualisation. Section 5.6.2 describes the implementation of Jini service invocation.

5.6.1. Service discovery

Jini service discovery implementation, illustrated in figure 31 begins when the plug-in is started and the KroxGateway (through the PluginManager) calls the *discover*() of the plug-in. The Jini LNC registers itself with the local lookup services in the local HAN and is notified therefore on all existing services and added and removed services. Steps 1-14 in figure 31 describe the discovery process and registration of a virtual service proxy in the remote HAN (HAN2).

The Jini VRM is required to dynamically generate a synthetic local implementation for the service interface. The message received from HAN1 contained the name of the interface corresponding to the shared service and the Java implementation dynamically created by the Jini VRM implements (in the Java programming language sense) this service interface. Before the service can be used in the local HAN (HAN2), a proxy must be exported and registered with a local lookup service. The full details of how a service implementation is

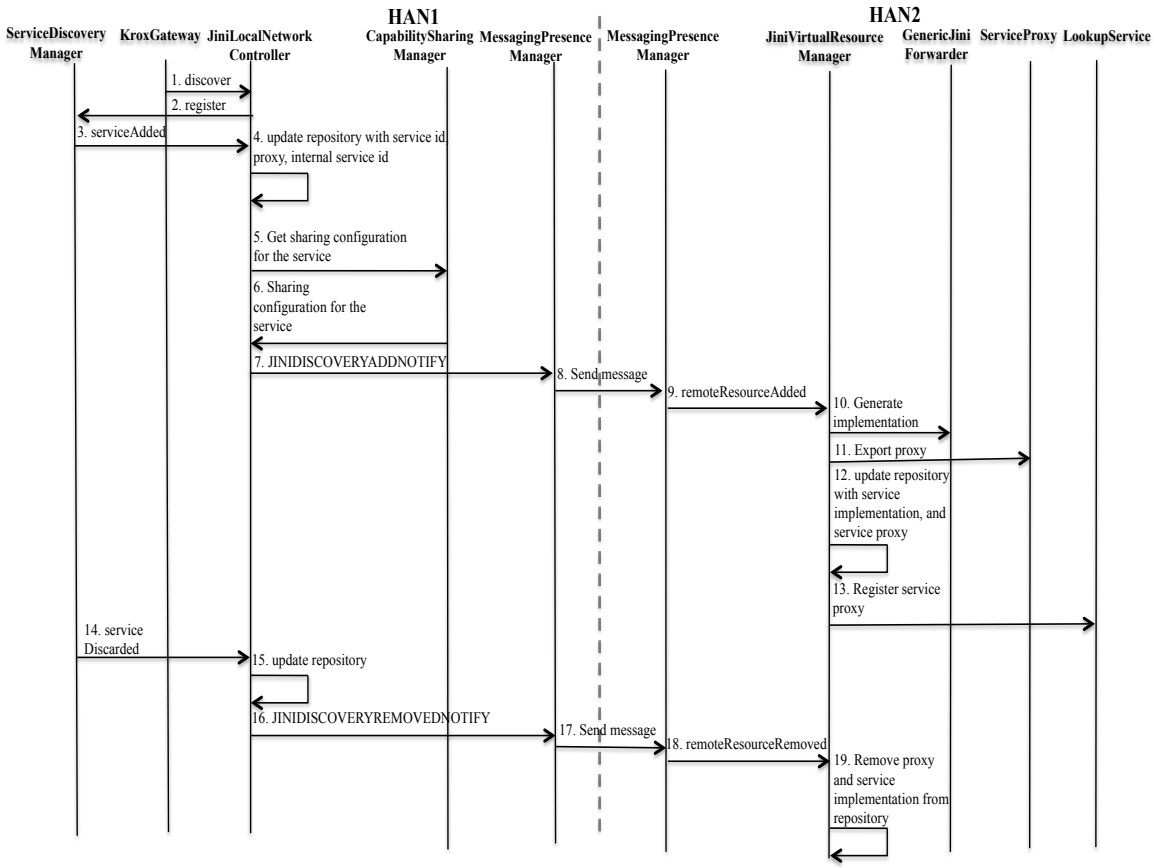

Figure 31 Jini Plug-in Service Discovery Prototype Implementation

generated in the remote HAN (HAN2) and how it is exported are given in the next section.

When a "live" service is removed, the LNC (in HAN1) receives a notification from the lookup service. The LNC removes the local service proxy from its repository and notifies remote HANs about the removed service. In the remote HAN (HAN2) the proxy and the service implementation are removed from the repository and the VRM (in HAN2) cancels the lease of the virtual service with the local lookup service (steps 14-19 in figure 31).

When a remote HAN comes online, the LNC sends *JINIDISCOVERYADDEDNOTIFY* for all services currently available in the HAN that are shared with that remote HAN. When a remote HAN goes offline, the VRM removes all of the service it received from this HAN from its local mapping.

As discussed in the previous chapter, the LNC ignores announcements made by the VRM in its local HAN, avoiding unintentional re-sharing of remote devices, which are not owned by the local HAN. In order to be able to distinguish between remote and local services, a new service characteristic was added. When the VRM in the local HAN registers the remote service, it registers it with a "remote" attribute. The Jini LNC can then ignore and suppress notifications about added services with the "remote" attribute.

If the LNC is notified on a change in the sharing configuration, it needs to inform the relevant remote VRMs on the update. If sharing was added for additional service, the LNC needs to notify the remote HANs on the added service. If sharing for a service was removed, then the remote HAN's VRM should destruct the service implementation and proxy and deregister it from the lookup service.

5.6.1.1. Jini service implementation dynamic generation

When a *resourceAdded* notification for a shared remote service arrives at a VRM from remote HAN, it contains information about the remote HAN, which owns the "live" service and the service interface. Given the Jini service interface name, the VRM uses the Java dynamic proxy technique [60] to dynamically create an implementation for this interface. The Java dynamic proxy technique enables implementing several design patterns including

Figure 32 Dynamic Proxy Implementation for a Jini Service Interface

the façade, proxy, decorator [42], and others in a dynamic manner. Dynamic proxy objects enable the dynamic implementation of one or more interfaces and dispatch the calls of those interfaces using the Java reflection API. This provides a mechanism for intercepting method calls and redirecting them or extending their functionality dynamically. The dynamic proxy approach does not involve code generation; instead it uses Java reflection to offer a runtime dynamic implementation for a given interface.

Using this technique, a generic implementation acts as an implementation of the given interface. This is done by instantiating the generic GenericJiniForwarder class (see figure 32), which implements Java InvocationHandler[33] and therefore supports the dynamic proxy pattern. The GenericJiniForwarder is a generic forwarder that takes an interface class object, an identifier for remote *Krox* system instance, and a service identifier in its constructor and uses the dynamic proxy technique to instantiate a proxy that dynamically implements the given interface. The *getInstance*() method of the GenericJiniForwarder returns an implementation of the Jini service interface with which the GenericJiniForwarder was constructed, such that the implementation of each method of the interface simply forwards the call over the communication subsystem to the remote HAN where the "live" service is hosted.

[33] http://download.oracle.com/javase/6/docs/api/java/lang/reflect/InvocationHandler.html

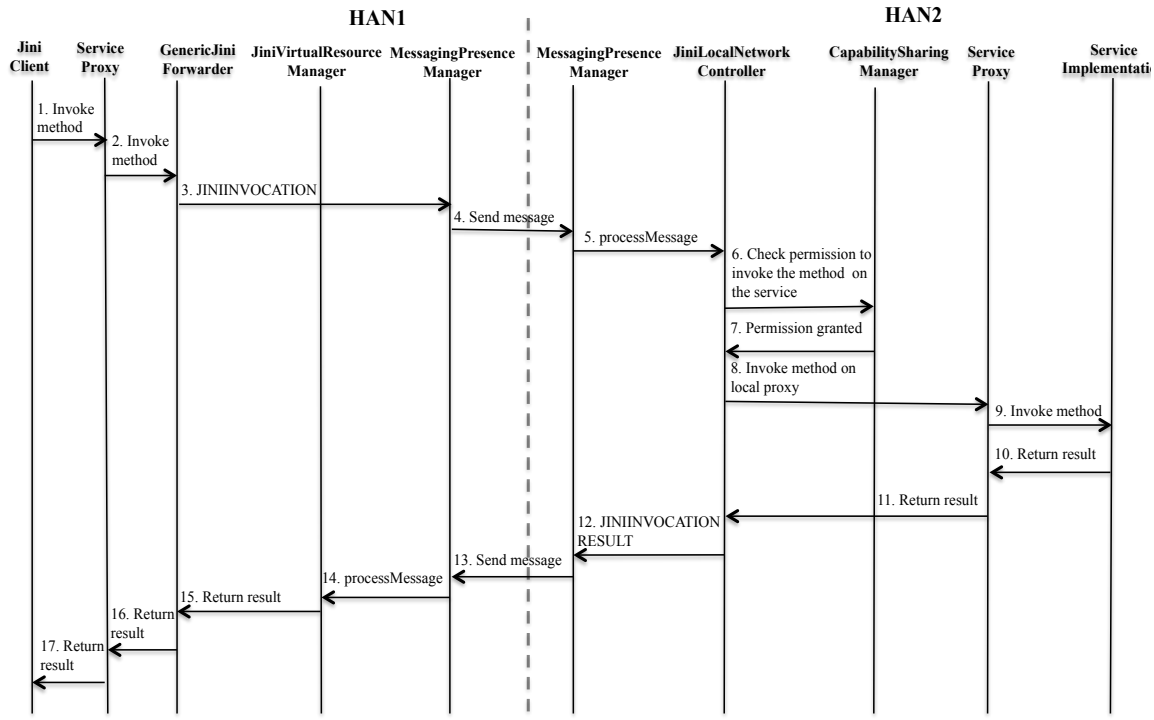

Figure 33 Jini Plug-in Service Invocation Prototype Implementation

5.6.1.2. Jini service proxy dynamic generation

Once the service implementation has been generated, its Jini service proxy needs to be generated automatically and registered with the lookup service. The Jini service proxy is exported from the service implementation by the ProxyGenerator using Jini Extensible Remote Invocation (JERI[34]), and is registered with the local lookup service. The service proxy is a generated automatically by the Jini exporter and contains only RMI[35] communication support to the service implementation. The service implementation in turn serves as a delegator of the method invocation using the MessagingPresenceManager to the remote HAN's Jini LNC. Once the service implementation and service proxy have been generated and registered, the VRM updates its local repository with a mapping between the service identifier and the proxy, and service identifier and the implementation.

5.6.2. Service invocation

[34] http://www.jini.org/files/specs/porter/api/net/jini/jeri/package-summary.html

[35] http://www.oracle.com/technetwork/java/javase/tech/index-jsp-136424.html

Jini service invocation (figure 33) is started once an application in the local HAN discovers a Jini service, and downloads a service proxy. Then it is ready to invoke methods on this service proxy that implements the Java service interface. When the service proxy interface method is invoked, the proxy communicates the method invocation request to the service implementation. As was mentioned in the previous section, the service implementation is maintained in the JiniVirtualResourceManager mappings. The implemented Jini virtual service dynamic proxy class is generic and forwards the call over the communication subsystem to the remote HAN hosting the "live" service (steps 1-4 in figure 33). In the remote HAN the message is processed by the LNC. The LNC finds the Jini service proxy in its local repository and checks with the CSM if the requesting HAN has appropriate permission to execute the requested method. If so, the method is invoked on the real service proxy and the result is sent back over the communication subsystem to the remote HAN (HAN1) where it is passed to the VRM in the requesting HAN (steps 5-14 in figure 33). The local VRM hands the result to the service implementation instance (the GenericJiniForwarder), which sends it back to the service proxy, which in turn returns the result to the Jini client (steps 15-17 in figure 33).

From a performance point of view the service implementation (GenericJiniForwarder) blocks until the response from the remote HAN is received, however it does not prevent other service clients invoking methods on the same service object. When the result is received at the VRM, the blocked service implementation is notified and it can return the result to the client via the service proxy.

5.6.3. Testing

In order to test the Jini plug-in, the *Krox* system was tested with a number of participating remote HANs. The purpose of testing was to validate that the implementation conforms to the plug-in design as described in the design chapter. For testing purposes a number of simple Jini services were implemented and registered in each local HAN with the lookup service.

Similarly to UPnP plug-in testing, the Jini plug-in code was instrumented to print log messages, which correspond to the event model, such that local events can be correlated with remote events. For example when the LNC receives a notification on added service, it

prints a message to the log with the identifier that is then used to notify remote HANs on this service. In the remote HAN, the VRM prints the message with the identifier before and after it registers the virtual service with the lookup service, such that it can be verified that flow of events corresponds to the one defined in the design. In order to test Jini invocation, a Jini client was developed such that it registers with the lookup service and is notified on added services, and for each added service it invokes a method from its interface.

Testing has shown that the implementation worked as expected and that the event model defined by the design of the plug-in was sufficient to express the required interaction between the plug-in components for supporting sharing of Jini services across multiple HANs. Additional testing was performed using the client application described in section 5.8.

5.6.4. Summary

The Jini service plug-in implementation, described in the above sections, demonstrates the feasibility of the *Krox* system design for Jini service plug-in. Jini service protocol plug-in was fully implemented supporting service discovery, and service invocation as required by the Jini service protocol. The implementation is lightweight in that it sends over the wire only the minimal information required to represent the Jini service in the remote HAN (the service identifier and the name of the service interface). In addition, the generated service implementation only relays the request to the remote HAN where the "live" service proxy is invoked; therefore it is both lightweight in memory as well as in processing.

Table 5 summarises the event model (core and extensions) that is included in the Jini plug-in implementation.

The testing performed with a number of remote HANs and emulated Jini services has shown that the Jini plug-in works as expected, and that the event model described above is sufficient to represent the required interaction between the LNC and VRM to support seamless service integration of remote services with local client applications.

Event	Initiated By	Processed By	Description
DISCOVER	Plug-in Manager	Jini LNC	Initiate discovery in the local HAN through registration with lookup services
REMOTEHANAVAILABLE	Communication subsystem	Jini LNC	A remote HAN from the buddy roster became "available" - need to send it information about all local shared resources
REMOTEHANUNAVAILABLE	Communication subsystem	Jini LNC, Jini VRM	A remote HAN from the buddy roster became "unavailable" – need to clean related resources in LNC and VRM
SHARINGCONFIGURATIONADDED	Capability Sharing Manager	Jini LNC	Sharing configuration changed, additional resources should be shared now with remote HANs. The LNC receives a notification from the CSM and sends an update to the relevant remote VRMs.
SHARINGCONFIGURATIONREMOVED	Capability Sharing Manager	Jini LNC	A resource that was previously shared is not longer allowed for sharing. The LNC receives a notification from the CSM and sends an update to the relevant remote VRMs.
JINIDISCOVERYADDEDNOTIFY	Jini LNC	Jini VRM	New Jini service was discovered, notify remote HANs on the added service
JINIDISCOVERYREMOVEDNOTIFY	Jini LNC	Jini VRM	Jini service is no longer available in the HAN, notify remote HANs on the removed service
JINIINVOCATION	Virtual service implementation (GenericJiniForwarder)	Jini LNC	A request for an invocation of a method on a remote Jini service
JINIINVOCATIONRESULT	Jini LNC	Virtual service implementation (GenericJiniForwarder)	The result of an invocation of a method on a remote Jini service

Table 5 Jini Plug-in Event Model

5.7. System administration

The first aspect of system administration for the *Krox* system prototype is management of sharing relationships with remote HANs. Based on the IM&P user metaphor, relationships with remote HANs are reduced to adding or removing "buddies" from the buddy roster. When the home user wishes to add a friend's HAN to his sharing list, he needs to know the identifier of the remote HAN. Adding a buddy to the buddy list enables the communication subsystem to send and receive messages between the local HAN and the remote "buddy" HAN. For the purpose of the prototype all configuration of users and relations between users was made via direct user administration interface in the IM&P server rather than through a client application. IM&P user configuration is used daily by a large number of untrained users, therefore, this provides a demonstration of how relatively easy user administration can be based on the IM&P user metaphor applied to sharing of HAN resources.

When the *Krox* system is started the KroxGateway reads its configuration from a configuration file (figure 34). The system configuration for the prototype contains

```
#user configuration
username=sim-user-1
password=11111

#server configuration
xmpp_server=134.226.62.17
xmpp_port=5222
xmpp_service=KROX


#proxy configuration
proxy_mode=true
http_proxy_host=www-proxy.cs.tcd.ie
http_proxy_port=8080

#plug-in configuration
plug_in_dir=

#performance configuration
number_of_threads=20
event_grace=2000

#logging configuration
logger=true
```

Figure 34 *Krox* System Configuration File

information about the username and password, the IM&P server to connect to, http proxy if needed, and information about where to search for plug-ins to load. In a production system the configuration file should not be manually modifiable or human readable, however for prototyping purposes the configuration file was implemented using a text file to make it easier to modify for the frequent configuration changes during testing and evaluation phases.

The event_grace parameter defines the amount of time to wait for the completion of event subscription when an event notification is received. This is related to the race condition in UPnP event handing that was described in section 5.5.4.

5.8. Client application prototype

The term client application refers here to service protocol specific clients that interact with services that support the service protocol, e.g. a UPnP control point is a UPnP client application, a Java program that discovers and interact with Jini services is referred to as a Jini client. As seamless integration of remote services with local client applications in the local HAN is an important aspect of the *Krox* system architecture, an application was developed (figure 35) to test and demonstrate this seamless integration. The client application enables the discovery and invocation of specific service protocol local and virtual services. The client application organises virtual devices and services by the user

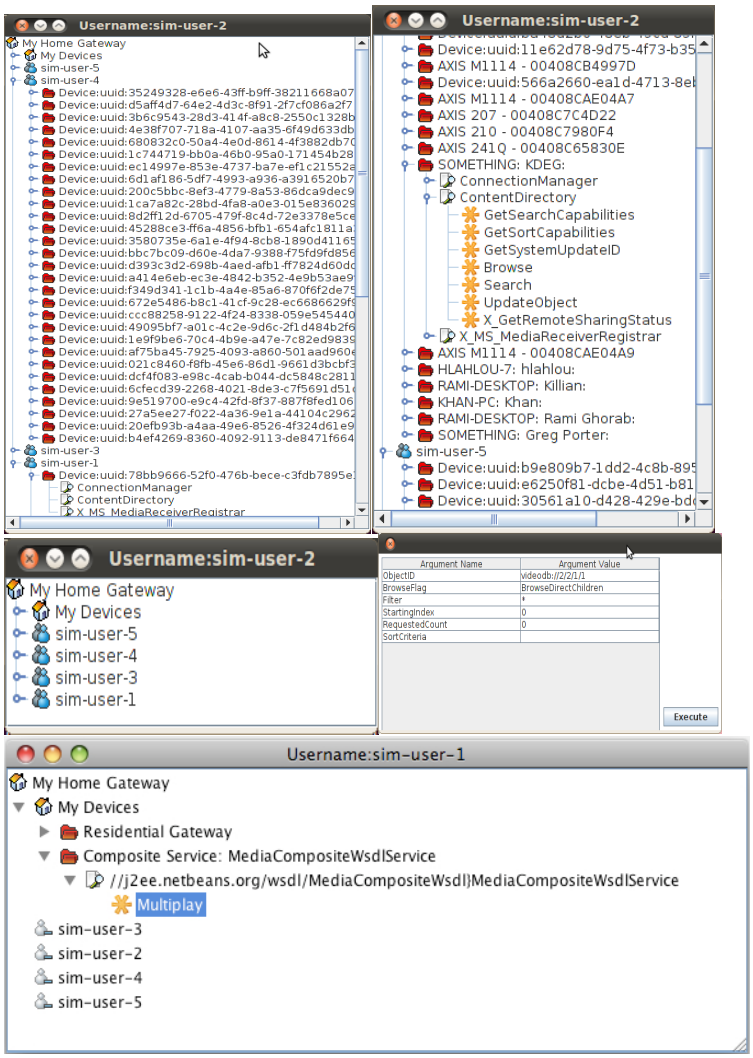

Figure 35 *Krox* Client Application Prototype

from which they were shared, enabling to distinguish between local and remote devices and services for testing purposes. The discovery capabilities of the client application enable to validate that all devices from remote HANs are virtualised and shown under the user that shared them. It enables to verify that when a device is no longer available it is not shown anymore in the local HAN. For discovered devices and services, the client application enables inspection of the device/service and invocation of actions. For example the screenshots in figure 35 were taken from a HAN sharing devices and services with four other HANs. The local HAN devices are shown under “My Devices”, and devices and services from remote HANs are shown under the identifier of the remote HAN that shared them.

The client application embeds an IM&P endpoint for receiving updates on added users (i.e. remote HANs with which sharing has been agreed) and the status of remote HANs.

To support UPnP services, the *Krox* client embeds a UPnP control point which locates UPnP devices and services in the local HAN and shows them either under the HAN that shared them, or under the local HAN. When a UPnP device is discovered, its description is fetched and parsed and the supported services are shown in the client application under the device. For each service, its corresponding description is fetched and parsed and its corresponding actions are shown under the service. This behaviour enables testing and validating the discovery and description implementation of the UPnP plug-in. The client application enables manual invocation of UPnP actions for both local and remote devices in a fully transparent fashion. The invocation is facilitated by sending a SOAP request to the local device, which can be either a "live" device or a virtual remote device. This behaviour enables testing and validation of the invocation implementation of the UPnP plug-in. The client application supports subscription for event notifications for UPnP services. This capability enables testing and validation of the eventing implementation of the UPnP plug-in. Finally, the client application was used in validation of the implementation of representation of composite services as UPnP devices and services. Since composite services are represented as UPnP devices, they are discovered, inspected, and can be invoked using the client application as can be seen in figure 35.

For supporting Jini services, the *Krox* client application finds a lookup service in the local HAN and listens to changes. Jini services that are discovered are shown under the network that shared them. When a service is discovered, the *Krox* client retrieves the service proxy from the lookup service and parses the service interface, extracts the methods and parameters and presents it as the child of the service in the client application. Similarly to UPnP, Jini service invocation is supported. The client application was also used in functional testing for Jini discovery and invocation for HANs from remote HANs shared with the local HAN.

5.9. Home service composition with BPEL

In order to demonstrate the service composition subsystem design, a prototype implementation was developed for mapping UPnP and Jini services to web services, and

then composing them to BPEL processes and deploying and executing such services. The following sections provide details about this prototype implementation.

5.9.1. WS-BPEL

The prototype builds on service composition with BPEL. BPEL 2.0 is supported through Apache ODE version 1.3.5. ODE was deployed on a Apache Tomcat version 6.0.18 servlet container in the local HAN. The following sections describe how UPnP and Jini service protocols were mapped to web services for automatic generation and deployment of web service proxies corresponding to HAN services.

5.9.2. UPnP to web service mapping extension

The implementation of the UPnP to web services mapping is based on three steps:

1) Device inspection – fetching the device description, and for each service fetch the service description. Each service will correspond to a web service; each action in the service description will correspond to a method in the generated web service.
2) Code generation – generating the code for a web service, given the device name, service type, and names of parameters (input and output) for the UPnP actions.
3) Web service package and deployment – the generated Java web service is compiled, built and deployed to the local web server.

When a UPnP device was discovered by the UPnP LNC or UPnP VRM, its description is fetched. For each supported service, the corresponding Java code for a JAX-WS web service is generated (see figures 36-37). Once the Java file for the web service has been saved to disk, a script is executed to compile, package and deploy the web service to the servlet container .

Figure 36 shows a generated web service WSDL corresponding to an AVTransport UPnP service (from a media renderer device). The screenshot on the left shows the AVTransport service description XML as retrieved from the UPnP media renderer device. On the right, is the WSDL (shown from the web server) of the deployed web service that corresponds to the UPnP AVTransport service. It can be seen that each action in the UPnP service is mapped to

Figure 36 UPnP AVTransport Service Description (right) and Corresponding Web Service (left)

an operation in the web service, and the entire UPnP service is mapped to a single web service.

Figure 37 shows the generated code for the web service, which corresponds to the UPnP service shown on figure 36. The generated code has a method for each action in the UPnP service and an additional generic invocation method, which is called from all of the action implementations. The class is generated with a number of identifiers that are hardwired: the control URL for the service, which should be used when invoking actions (ctrlURL in figure 37), the service type, which is used in creating the SOAP request, and the host/port of the device, which are used for sending the SOAP request. All action implementations are similar: they create two arrays of Strings: one for the variable names and one for the variable values – the variable names for the methods correspond to the names of the variable

in the UPnP action and the name of the method corresponds to the name of the UPnP action. The last parameter for the action invocation is the action name, which is hardwired to the method during the code generation. Each method calls the generic *invokeMethod*() method which handles the interaction with the device and either returns the SOAP response or throws a UPnPException. The *invokeMethod*() prepares the SOAP request and sends it to the device. If the invocation was successful, the String SOAP response is sent back to the calling method and back to the web service client. If there was an error, a UPnPException is constructed with the corresponding UPnP error code and error description and is thrown back from the web service. For each UPnP service, once the code is automatically generated, the code is saved to disk and compiled. The artefacts are packaged in a Web Archive (WAR) file that is copied to the auto-deploy directory of the servlet container (Tomcat), which initiates the deployment of the web service.

```
package Services;
import javax.jws.WebService;
import javax.jws.WebParam;
import upnp.UPnPException;
import org.cybergarage.upnp.control.ActionRequest;
import org.cybergarage.upnp.control.ActionResponse;
import org.cybergarage.xml.Node;
import org.cybergarage.upnp.control.Control;
import org.cybergarage.soap.SOAP;

@WebService(targetNamespace = "http://j2ee.netbeans.org/wsdl/XBMC__M-1638718874_AVTransport")
public class MediaRenderer_AVTransport {
    private String ctrlURL="/AVTransport/e20cd02e-e67c-f07a-4e3a-8a824d41bc29/control.xml";
    private String serviceType="urn:schemas-upnp-org:service:AVTransport:1";
    private String host="192.168.1.3";
    private int port=58563;
    private String invokeMethod(String actionName, String[] args, String[] values) throws UPnPException {
        ActionRequest ctrlreq = new ActionRequest();
        ctrlreq.setURI(ctrlURL, true);
        ctrlreq.setHost(host, port);
        ctrlreq.setRequestHost(host);
        ctrlreq.setRequestPort(port);
        ctrlreq.setEnvelopeNode(SOAP.createEnvelopeBodyNode());
        Node envNode = ctrlreq.getEnvelopeNode();;
        Node bodyNode = ctrlreq.getBodyNode();
        Node actionNode = new Node();
        actionNode.setName(Control.NS, actionName);
        actionNode.setNameSpace(Control.NS, serviceType);
        for (int n=0; n<args.length; n++) {
            Node argNode = new Node();
            argNode.setName(args[n]);
            argNode.setValue(values[n]);
            actionNode.addNode(argNode);
        }
        bodyNode.addNode(actionNode);
        ctrlreq.setContent(envNode);
        String soapAction = "\"" +serviceType+"#"+actionName+"\"";
        ctrlreq.setSOAPAction(soapAction);
        ActionResponse ctrlRes = ctrlreq.post();
        if (!ctrlRes.isSuccessful()) {
            throw new UPnPException(ctrlRes.getUPnPErrorCode(),ctrlRes.getUPnPErrorDescription());
        }
        return ctrlRes.toString();
    }
    public String GetCurrentTransportActions (@WebParam(name = "InstanceID") String InstanceID) throws UPnPException {...}
    public String GetDeviceCapabilities (@WebParam(name = "InstanceID") String InstanceID) throws UPnPException {...}
    public String GetMediaInfo (@WebParam(name = "InstanceID") String InstanceID) throws UPnPException {...}
    public String GetPositionInfo (@WebParam(name = "InstanceID") String InstanceID) throws UPnPException {...}
    public String GetTransportInfo (@WebParam(name = "InstanceID") String InstanceID) throws UPnPException {...}
    public String GetTransportSettings (@WebParam(name = "InstanceID") String InstanceID) throws UPnPException {...}
    public String Next (@WebParam(name = "InstanceID") String InstanceID) throws UPnPException {...}
    public String Pause (@WebParam(name = "InstanceID") String InstanceID) throws UPnPException {...}
    public String Play (@WebParam(name = "InstanceID") String InstanceID,
                        @WebParam(name = "Speed") String Speed) throws UPnPException {...}
    public String Previous (@WebParam(name = "InstanceID") String InstanceID) throws UPnPException {...}
    public String Seek (@WebParam(name = "InstanceID") String InstanceID,
                        @WebParam(name = "Unit") String Unit,
                        @WebParam(name = "Target") String Target) throws UPnPException {...}
    public String SetAVTransportURI (@WebParam(name = "InstanceID") String InstanceID,
                                     @WebParam(name = "CurrentURI") String CurrentURI,
                                     @WebParam(name = "CurrentURIMetaData") String CurrentURIMetaData) throws UPnPException {
        String[]args={"InstanceID","CurrentURI","CurrentURIMetaData"};
        String[]vals={InstanceID,CurrentURI,CurrentURIMetaData};
        String actionName="SetAVTransportURI";
        return invokeMethod(actionName, args,vals);
    }

    public String SetPlayMode (@WebParam(name = "InstanceID") String InstanceID,
                               @WebParam(name = "NewPlayMode") String NewPlayMode) throws UPnPException {...}
    public String Stop (@WebParam(name = "InstanceID") String InstanceID) throws UPnPException {...}
}
```

Figure 37 AVTransport Web Service Generated Code

Whenever the *Krox* system is restarted all existing services are undeployed. If they are rediscovered they are redeployed. When a device is removed, its corresponding services are undeployed from the local servlet container.

5.9.3. Jini to web service mapping extension

The generation of web services for Jini services is very similar to the technique used for UPnP services with a minor difference. While UPnP requires inspection of the service description document in order to determine the names of methods and types of parameters, for Jini this is much simplified using the Java reflection API. When the Jini LNC or Jini VRM discover a service, the generation of a web service is triggered. Given the interface, the names of the methods and types of parameters are retrieved from the interface using the Java reflection API. Unfortunately using Java reflection API it is not possible to retrieve the

```java
package jiniserviceinterfaces;

import java.rmi.Remote;
import java.rmi.RemoteException;

public interface NameGuessingJiniService extends Remote{
    public String guessLastName(String firstName) throws RemoteException;
    public String guessFirstName(String lastName) throws RemoteException;
}
```

```java
package Services;

import javax.jws.WebService;
import javax.jws.WebParam;
import java.rmi.*;
import net.jini.core.lookup.*;
import net.jini.core.discovery.LookupLocator;
import jiniserviceinterfaces.NameGuessingJiniService;

@WebService(targetNamespace = "http://j2ee.netbeans.org/wsdl/jiniserviceinterfaces.NameGuessingJiniService")
public class NameGuessingJiniServiceWS {

    private long serviceIdLsb = -9202661330224450901l;
    private long serviceIdMsb = 4256320035323923891;

    public java.lang.String guessLastName(@WebParam(name = "param1") java.lang.String param1) throws Exception {
        ServiceID sid = new ServiceID(serviceIdMsb, serviceIdLsb);
        System.setSecurityManager(new RMISecurityManager());
        LookupLocator lookup = new LookupLocator("jini://localhost");
        ServiceRegistrar regi = lookup.getRegistrar();
        ServiceTemplate template = new ServiceTemplate(sid, null, null);
        NameGuessingJiniService proxy = (NameGuessingJiniService) regi.lookup(template);
        if (proxy != null) {
            java.lang.String result = (java.lang.String) proxy.guessLastName(param1);
            return result;
        }
        return null;
    }

    public java.lang.String guessFirstName(@WebParam(name = "param1") java.lang.String param1) throws Exception {
        ServiceID sid = new ServiceID(serviceIdMsb, serviceIdLsb);
        System.setSecurityManager(new RMISecurityManager());
        LookupLocator lookup = new LookupLocator("jini://localhost");
        ServiceRegistrar regi = lookup.getRegistrar();
        ServiceTemplate template = new ServiceTemplate(sid, null, null);
        NameGuessingJiniService proxy = (NameGuessingJiniService) regi.lookup(template);
        if (proxy != null) {
            java.lang.String result = (java.lang.String) proxy.guessFirstName(param1);
            return result;
        }
        return null;
    }
}
```

Figure 38 Jini Service Interface (top) and Corresponding Web Service (bottom)

names of the parameters, therefore they are named param1, param2, etc. For each method in Jini service interface, a method is created in the web service. Figure 38 illustrates a Jini service interface and its corresponding generated web service code. In the Jini generated web service code there is no generic invocation method. Instead in the implementation of each method, the Jini service proxy is retrieved from the lookup service using the hardwired service identifier. Given this proxy and the hardwired service interface, the corresponding method is invoked on the Jini service and the result is returned. Once the Java web service code generation is completed, the service is saved to disk and the same script mentioned above is used to build, package, and deploy the service. Undeployment of services is triggered both on system bootstrap and when services are removed from the local HAN.

5.9.4. Composing home services

When the HAN services are deployed as web services they can be composed using BPEL regardless if they represent UPnP, Jini, or any other HAN services protocol. In addition, since both the LNC and the VRM in the local HAN trigger automatic web service generation, local and remote services (through their local representations) can be composed seamlessly. A further advantage of using ordinary web services is that it is possible to also compose external 3rd party web services and other composite services.

In order to demonstrate the feasibility of the design approach, a composite service involving a local HAN UPnP service, a remote UPnP service, a local HAN Jini service, and an external web service were composed. The composite service used the following services:

- Local UPnP device – an XBox Media Centre (XBMC[36]) media renderer, specifically AVTransport service.
- Remote UPnP device – a MediaGate[37] media server, specifically ContentDirectory service.
- Jini Service – an emulated Jini service
- External web service – Microsoft translation service[38]

[36] xbmc.org

[37] http://www.cybergarage.org/twiki/bin/view/Main/MediaGateForJava

[38] api.microsofttranslator.com/V2/Soap.svc

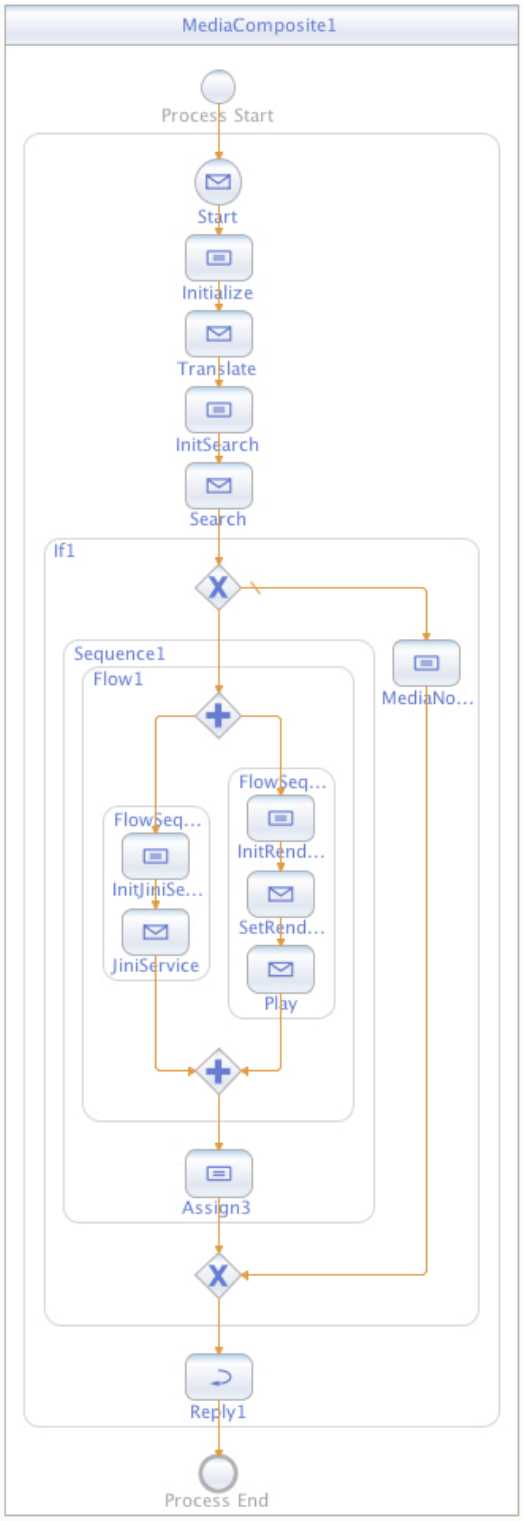

Figure 39 HAN Service Orchestration with BPEL

The composite service (figure 39) takes a String parameter corresponding to a title of media. The name is translated to French and then searched in the remote media server. If the media is found on the remote server, the local media renderer is setup to play the media, and in

parallel the Jini service is invoked. If the media is not found, an error message is returned indicating that the media was not found. The purpose of the example composite service is not to show the utility of the design for creating a specific composite service, but instead to demonstrate the variety of options for composition of services from different service protocols, and from multiple HANs, with external web services. The composite service was deployed to the ODE BPEL engine and testing has shown it worked as expected.

5.9.5. Representing BPEL services as UPnP devices

The last part of the implementation is a standalone application that enables the representation of BPEL composite services as UPnP devices in the local HAN, and by that enable their sharing with the UPnP service protocol plug-in. The only purpose of this application is to enable sharing of composite services, which was defined as a requirement (#22). The representation as UPnP devices was chosen because UPnP does not require the client to know in advance on the existence of the service. Unlike Jini, which requires the service client to have the service interface in advance, when representing composite services as UPnP devices, they can be safely advertised and consumed by applications without requiring previous information about the service. BPEL2UPnP is a singleton process running in the local HAN that is responsible for generating and maintaining virtual UPnP devices that correspond to composite services. When a composite service is deployed, the BPEL2UPnP application is called with a URL that corresponds to the WSDL document of a BPEL composite service. Using a WSDL parser, the document is parsed and information about the service name, input and output parameters, port and operation names and more is extracted. The service name is used to create a unique device identifier. Each port type in the composite service WSDL corresponds to a new UPnP service for the virtual UPnP device. BPEL2UPnP uses the service information extracted from the WSDL document to advertise the new virtual device and services to the local HAN. The device is advertised as having type: CompositeService:1 (version 1), however in the future this could be customised to a more meaningful service type with metadata given for the composite service. The device location is set to <host>:<port>/<device unique identifier>. When a SOAP request is received requesting the invocation of an action, the request is parsed and a corresponding SOAP request is composed and sent to the BPEL process URL. The SOAP response from the service is used to populate the output variables of the UPnP service and the corresponding SOAP response is returned to the requesting control point application. Figure 40 shows the device description corresponding to the BPEL composite service

```
<?xml version="1.0"?>
<root xmlns="urn:schemas-upnp-org:device-1-0">
    <specVersion>
        <major>1</major>
        <minor>0</minor>
    </specVersion>
    <device>
        <UDN>uuid:MediaCompositeWsdlService</UDN>
        <friendlyName>Composite Service: MediaCompositeWsdlService</friendlyName>
        <deviceType>urn:schemas-upnp-org:device:Composite-Service:1</deviceType>
        <manufacturer>TCD</manufacturer>
        <manufacturerURL>http://www.tcd.ie/</manufacturerURL>
        <modelName>Composite Service UPnP Device</modelName>
        <modelNumber>4.0</modelNumber>
        <modelURL>http://www.tcd.ie/</modelURL>
        <serialNumber>{27FDCA50-022A-440E-829D-0739C48F03C9}</serialNumber>
        <serviceList>
            <service>
                <serviceType>urn:schemas-upnp-org:service:{http://j2ee.netbeans.org/wsdl/MediaCompositeWsdl}MediaCompositeWsdlService:1</
                <serviceId>urn:upnp-org:serviceId:MediaCompositeWsdlService</serviceId>
                <controlURL>/uuid:MediaCompositeWsdlService/control</controlURL>
                <eventSubURL>/uuid:MediaCompositeWsdlService/eventing</eventSubURL>
                <SCPDURL>/uuid:MediaCompositeWsdlService/service</SCPDURL>
            </service>
        </serviceList>
    </device>
</root>
```

Figure 40 UPnP Device Description Corresponding to a BPEL Composite Service

presented in figure 39 as automatically generated and hosted by the BPEL2UPnP application.

When the composite service is undeployed, it is removed from the cache of BPEL2UPnP, which in turn announces a *byebye* on behalf of the virtual device and its constituent services.

For the purpose of the prototype, the integration between the BPEL engine and BPEL2UPnP is manual. This means that when a new process is deployed, BPEL2UPnP needs to be invoked separately with the URL representing the WSDL document in the application server. The purpose of BPEL2UPnP is to demonstrate how BPEL services can be shared with remote HANs as UPnP services, therefore it is sufficient to demonstrate that a given BPEL composite service described as WSDL can be mapped to a UPnP device. The BPEL2UPnP was tested and worked as expected.

5.10. Summary

This chapter presented the prototype implementation for *Krox* system architecture and system design with plug-in implementations for UPnP and Jini. The *Krox* system prototype implementation demonstrated the utility, and feasibility of *Krox* system design, specifically the plug-in framework and the design for the specific HAN service protocols. The level of testing, which was conducted for the prototype system, as described in section 5.5.5, 5.6.3, 5.8, and 5.9.4, indicates that the system worked as expected and conformed to the design specification, except for specific areas where this is explicitly discussed.

Req #	Description	Addressed by	Tested with
#1	Cross service protocol service composition	Using a BPEL engine and the plug-in's generated web services that were deployed in the local servlet container	Tested with a composite service that included Jini, UPnP, and web service
#2	Share composite services	BPEL2UPnP standalone application addressed the mapping from the WSDL of the composite service and its stateful representation as a UPnP device	Tested and demonstrated via the client application
#3	Cross HAN service composition	Using the service virtualisation and the web service generation in the VRM of the UPnP and Jini plug-ins, enabled the composability of remote services	Tested with the composite service that included a local media renderer and a remote media server
#4	Enable sharing of HAN services from the local HAN with remote HANs	The service virtualisation implementation of the plug-ins through the core event model and its extensions	Tested for each plug-in separately using the log validation technique and demonstrated via the client application
#5	Automatic discovery of resources from remote HANs shared with the local HAN	The implementation of the different implementations of the VRM in the UPnP and Jini plug-ins.	Tested for each plug-in separately using the log validation technique and demonstrated via the client application
#6	There must be no restriction that prevents sharing the same devices and resources with multiple remote HANs	The implementation does not restrict sharing of the same devices to multiple remote HANs	Testing was made with all HANs sharing all devices with all other HANs, thereby showing that the same devices can be shared with multiple HANs
#7	Interaction of applications with remote devices must be identical to the interaction with local of the same service protocol	The implementation of the different implementations of the VRM in the UPnP and Jini plug-ins.	Tested with the client application
#8	The system must not require modification to service protocols and must support plug-and-play	Supported through the plug-in framework event model, and the service virtualisation implementation of the different plug-ins	Tested with the client application that was implemented as a UPnP and Jini regular client, thereby showing that client applications do not require modifications to discover and interact with remote devices in the local HAN. In addition, this was tested and demonstrate using a standard control point application included with XBOX Media Centre (XBMC)
#9	Independence of access network technology	The implementation is independent from any access technology and does not make assumptions on a specific access technology	The system was tested with both DSL and cable access technologies
#10	The system must be able to discover and share devices with networks that are using NAT even in the existence of devices with identical IP addresses in multiple HANs	This is addressed by multiple components: • The communication subsystem uses TCP connection • The UPnP plug-in does not use private IP addresses beyond the scope of the local HAN, thereby supporting duplicate addresses in connected HANs. • Out of band communication was addressed by replacing the internal IP address with external IP address assuming that the appropriate configuration of firewall has been pre-made	Tested and demonstrated via the client application with multiple HANs having identical private IP addresses. Devices were appropriately discovered, and the client enabled to invoke actions on them. Streaming was tested and demonstrated with remote media server connected to a local media renderer
#11	The system must be able to communicate with remote HANs behind firewalls	The communication subsystem only requires the instant messaging port to be open. For the prototype purpose only, for out of band communication the firewall needs to be preconfigured	Tested with the client application with remote HANs behind firewall.
#12	All communication with remote HANs must be authenticated	The first step in the system's bootstrap is authentication with IM&P server	Unit testing, if authentication fails the system exists.
#13	Access control – Sharing must not be automatic and must enable home users to control which resources are shared with which remote HANs	This was addressed partially by this prototype – the implementation mandates that only resources that are shared are notified to remote HANs with which sharing has been agreed, however for this version, the only policy that was supported was share all devices with all remote HANs.	The system was tested with a stub implementation of access control, such that access control is checked but always returns true.
#14	Confidentiality - all traffic between remote HANs must be encrypted	The XMPP server was configured such that all communication between the client and the server is encrypted	The system was tested with the client-to-server encryption handled by the XMPP client and server.
#15	Security vulnerability	Was not addressed specifically by the prototype implementation. This is the subject of the security analysis presented in the next chapter	N/A
#16-#19	Performance requirements	The prototype implementation was implemented with performance considerations however, the actual performance evaluation is presented in the next chapter	N/A
#20	Extensibility to additional HAN service protocols	The extensible event model was extended for the implementation of UPnP and Jini service protocols. The differences between the service protocols, and their similarities with other protocols give good indication about the generality of the system architecture	Tested with UPnP and Jini plug-ins and demonstrated with the client application
#21	Dynamic relation management with remote HANs	The IM&P user model implemented with the Smack XMPP client library provides notifications when a remote HAN is added to the buddy list of the local HAN. In the prototype adding and removing remote HANs to/ from the buddy list was made using the XMPP server administration user interface, however the Smack API support these operations.	Tested using the XMPP server administration user interface, for adding/removing remote HANs from the buddy list of a local HAN and validating that the notification is received and processed correctly. Demonstrated via the client application user interface
#22	Pause/resume sharing with remote HANs	The IM&P user model implemented with the Smack XMPP library enables changing the presence status of the local HAN and by that pause all sharing with remote HANs. "Blocking" the sharing with a specific remote HAN enables to suspend the sharing with this remote HAN temporarily	Was not tested
#23	The system must not require manual configuration of the home gateway and its administration must be appropriate for non-technical users	Configuration of the home gateway is not required for the system, however for demonstration of streaming, the home gateway was preconfigured as discussed above. An administration application was not implemented however the IM&P user metaphor demonstrated with the client application gives good indication on the appropriateness of the level of administration required for operating the *Krox* system	N/A

Table 6 *KROX* System Prototype Requirements Addressing

Table 6 summarises the requirements as they were defined in section 3.4 and their addressing by the prototype implementation and their level of testing.

In accordance with the design science research methodology the key contribution of *Krox* is the architecture and design for integrated intra-HAN and inter-HAN service interoperability. *Krox* system architecture extends the state of the art presented in chapter 3 in the following aspects:

1) An integrated approach to service interoperability – Enables services from multiple service protocols to be shared with remote HANs and composed in both local and remote HANs with other services. This has not been addressed by any of the systems reviewed in chapter 3.
2) An open pluggable architecture and an extensible plug-in framework that enables multiple different HAN service oriented protocols to be supported for intra-HAN and inter-HAN service interoperability without requiring modifications to the service protocols, while supporting seamless integration with client applications in the HAN. The utility of the plug-in framework was demonstrated with multiple service protocols.
3) *Krox* service composition subsystem enables seamless composition of local and remote HAN services as well as further sharing of composite service. The *Krox* system architecture enables services of multiple HAN service protocols to be composed through the mapping between the HAN service interface and web service, which enables service orchestration. Through service virtualisation both local and remote HAN services can be composed. While the ability to compose services across multiple service protocols was suggested in literature [106, 2], the orchestration presented in this thesis for services from local and remote HANs is novel. Finally it has been shown how composite services can be shared with remote HANs through mapping to virtual UPnP services and then seamlessly shared similarly to actual UPnP devices and services.

The following chapter presents a performance evaluation of the system prototype and a security analysis of the potential system vulnerabilities and mechanisms for defending against them.

Chapter 6
EVALUATION

Chapter 4 presented the *Krox* system architecture and system design supporting integrated intra-HAN and inter-HAN service interoperability. In chapter 5, a prototype implementation of *Krox* system was presented, demonstrating the feasibility of the design approach. The design science research methodology requires the quality, and efficacy of a design artefact to be rigorously demonstrated and then assessed though a well executed evaluation method. Section 6.1 defines the evaluation goals based on the relevant performance and security requirements. In this research two evaluation methods, experimental evaluation, and analytical evaluation were used to assess the system design and implementation. The experimental evaluation intends to study the behaviour of the *Krox* system in a controlled environment with regard to performance with respect to the system requirements using simulation. Section 6.2 presents a performance evaluation of inter-HAN service interoperability (HAN service sharing), and section 6.3 describes the performance evaluation of intra-HAN service interoperability (HAN service composability). For evaluating the security of the system architecture and design, section 6.4 presents an analytical evaluation that examines the potential security vulnerabilities and threats and how the system can prevent an attack on an individual HAN from using the *Krox* system to spread to additional HANs. Finally section 6.5 presents concluding remarks.

6.1. Evaluation goals

The requirements for the *Krox* system architecture as specified in section 3.4 identified several performance and security requirements:

6.1.1. Performance

1. REQ #16 – Scale up (intra HAN) – The system must be able to represent up to 300 remote services with no significant latency. More services can be supported with reduced performance.

1) REQ #17 – Scale up (inter HAN) – The system must be able to scale to a small number of remote HANs corresponding to close family and friends. The number of remote HANs must not exceed 15 remote HANs.
2) REQ #18 – Scale down – The system should be deployable on popular operating environments including Linux, Windows, Mac OS X. The system's deployment requirements should be appropriate for home area networks – It must be possible to deploy the system on low-end machines, specifically home gateways where RAM does not exceed 1GB and CPU does not exceed 2GHz.
3) REQ #19 – Concurrent access – It must be possible for multiple remote HANs to interact with the same local HAN resource simultaneously.

Based on the above requirements, the goal of the performance evaluation is to assess the different performance aspects of the *Krox* system with between 5 and 15 remote HANs and up to 300 remote services represented in the local HAN. There are two main areas for evaluating the performance of *Krox* system: inter-HAN service interoperability i.e. the ability to represent remote services and interact efficiently with remote HANs and intra-HAN service interoperability and composability, i.e. the ability to efficiently represent remote and local HAN services as composable and interoperable services with no significant latency. More specific criteria for inter-HAN and intra-HAN performance evaluation are given in section 6.2 and section 6.3 respectively.

6.1.2. Security

The security requirements for authentication (REQ #12), access control (REQ #13), and confidentiality (REQ #14) were already addressed in the previous chapters. The focus of this chapter is the analysis of security vulnerabilities and the corresponding required mitigation mechanisms:

REQ #15 – Vulnerability – Some service protocols for the HANs contain security vulnerabilities, however as they are not designed to work over unsecure networks such as the Internet, these vulnerabilities are relatively of low risk when restricted to a single network. It is an essential requirement that the multi HAN sharing system will not increase security vulnerability of the home network by introducing new threats or by extending existing vulnerabilities.

6.2. Inter-HAN service interoperability performance evaluation

The inter-HAN service interoperability performance evaluation assesses the ability of the *Krox* system to efficiently discover resources in the local HAN, share them with remote HANs and represent them in these HANs as virtual resources. In order to assess the effect of each service protocol on performance, two separate experiments were designed for UPnP and Jini. Due to its verbose nature, and its support for event notifications, it is expected that UPnP plug-in would provide a worst-case scenario for performance evaluation. Therefore the main performance evaluation of the *Krox* system was made with UPnP service protocol plugin and UPnP devices. A complementary evaluation for the *Krox* system with the Jini service plug-in was performed to assess specifically the effect of using the dynamic proxy technique on the performance of invocation of remote Jini services. The performance evaluation of the *Krox* system with UPnP service protocol plug-in is described in section 6.2.1 and the performance evaluation of the *Krox* system with Jini is described in section 6.2.2.

6.2.1. UPnP

Following the experimental evaluation methodology [57], the performance of *Krox* system implementation was evaluated using a controlled experiment using an emulation of UPnP devices and control points. The emulation of UPnP networks enables the experiment to evaluate the *Krox* system architecture as the design artefact in a controlled environment where device and control point behaviour can be controlled and tuned to the experiment requirements. The performance requirements as described in section 3.4 specify two dimensions that need to be assessed: the number of HANs with which services are shared,

and the number of remote services shared with the local HAN. The first experiment evaluates the system performance with an increasing number of services while the number of participating HANs remains fixed, however the results of this experiment are also analysed to predict the impact of adding more HANs.

The following sections describe the performance evaluation parameters and the experimental design.

6.2.1.1. Evaluation parameters

When considering the inter-HAN performance aspects of *Krox* system architecture, several parameters need to be evaluated:

- **CPU utilisation** – What is the processing overhead added by the system? Regardless of whether the *Krox* system is deployed on a powerful server, embedded within the home gateway, or running on a standalone appliance, it must be verified that CPU usage of the system is relatively low.
- **Memory utilisation** – How much memory is consumed by the *Krox* system? It is important to analyse the system's memory utilisation and observe the impact of caching (e.g. of description documents in the VRM) on memory consumption. Both the LNC and the VRM affect the memory utilisation of the system. The LNC maintains a repository of all local announcements, and the VRM maintains a repository of all remote announcements and caches description documents fetched from remote HANs. It is therefore expected that the memory utilisation will grow linearly with the number of services available in the local HAN, both local devices and remote devices and services shared with the local HAN.
- **Responding to local HAN's search requests** – How long does it take for the UPnP VRM to respond to search requests in the local HAN with responses corresponding to remote devices and services? The VRM in the local HAN listens to search requests made by control points in the local HAN. When search request is received, the VRM needs to respond with every device/service announcement that corresponds to the service type in the search request. More services shared with the local HAN lead to potentially more processing in the VRM.

- **Discovery delay** – How long does it take to process a remote device announcement from when it is received to the local HAN, until it is announced? Once the LNC discovers a local device it sends notifications to all remote HANs with which it is shared, where the corresponding VRMs announce it. The purpose of the evaluation is not to measure the delay introduced by the external network time, which depends on the downstream and upstream bandwidth of the HAN and on the public network. Instead the purpose is to evaluate the delay introduced by the processing of the VRM, and observe the effect of having more devices and services shared with the local HAN with respect to the time it takes for the VRM to announce remote services in the local HAN especially with machines without high concurrency – i.e. machines with no multiple processors or multiple cores, where the processing of received messages is streamlined.
- **Remote description delay** – How much overhead is added by the VRM to description request processing (with and without caching)? Fetching device/service description requires a number of steps. While the system does not control the time it takes to fetch the description from the device, or to send it over the network, it is important to evaluate the delay added by the VRM itself. This delay equals the time spent in the VRM – i.e. until a message is sent to the remote HAN requesting the description, and from the time a response message is received with the description is received from the remote HAN, until the description is sent to the requesting control point.
- **Remote invocation delay** – How much overhead is added by the VRM to SOAP request processing? i.e. of the overall time that a SOAP request processing took, how much was spent in the VRM, before a message was sent to the remote HAN requesting the invocation, and after the response was received. While the number of SOAP requests may be relatively small comparing to the number of description requests, the handling of control requests must be very efficient to minimise the overhead to allow true seamless integration, abstracting the remoteness of a shared device.
- **Event notification delay** – How long it takes from the time a subscribed event notification from a remote device is received in a VRM, until the notification is sent to the subscribed control point? Remote event notification processing must be very efficient to abstract the remoteness and enable applications to get real-time status updates about service changes.

- **Bandwidth utilisation** – How much bandwidth is consumed by the *Krox* system for its message exchange between remote HANs? The bandwidth use of the Krox system is assessed by measurement of the number and size of messages sent between *Krox* system instances in remote HANs. The bandwidth used by *Krox* system depends on the behaviour of control points and their interaction with the system. More interaction leads to more SOAP requests, which lead to more inter-HAN communication, such that the size of the response can affect the consumption of bandwidth. The purpose of this parameter is to assess the levels of the bandwidth used by the *Krox* system for inter-HAN communication.

6.2.1.2. UPnP emulated network design

When considering a network of UPnP devices there are a number of parameters that may impact performance:

1) **Number of devices** – More devices lead to more SSDP announcements, which leads to more processing overhead on the LNC. More shared devices from remote HANs imply more inter-HAN traffic and more overhead on the VRM in the HANs with which devices are shared.
2) **Number of services per device** – More services lead to more SSDP announcements, which have the same impact as more devices.
3) **Duration of device availability in the HAN** – It is expected that some of the devices in the HAN are stationary and not moving, while others are mobile and can join and leave the HAN more frequently. The duration of the device availability, as indicated in its SSDP announcement affects discovery and inter-HAN communication. More mobile devices mean more devices with shorter duration, and possibly devices that leave and re-join the network frequently. The impact on performance is not limited to discovery but also affects description requests that can be initiated by control points every time a device joins the HAN.
4) **Frequency of search requests** – Search requests in the local HAN require the VRM to respond with search response for each remote service that corresponds to the service type given in the request. In case the control point is interested in all service types, this means that the VRM must respond with a search response for all remote services. Therefore more search requests lead to more overhead on the VRM.

5) **Frequency of description requests** – The typical behaviour of a control point is to request the description of all services that match a certain type – e.g. if the control point sent a search request for media servers to the local HAN, it will request the description for each root device that replied to the search request. More description requests imply more processing overhead on the VRM, more inter-HAN traffic (until description cache is populated), and more processing overhead in the remote LNC.
6) **Frequency of SOAP requests** – There is no typical behaviour with regard to SOAP requests as they correspond to actions orchestrated by control points on behalf of users, however it is clear that more SOAP requests lead to more overhead on the VRM, more inter-HAN traffic for requests and responses, and more overhead on the remote LNC. In addition, SOAP responses can be of variable size, such that longer SOAP responses can have bigger impact on performance.
7) **Number of event subscriptions** – More event subscriptions may lead to more event notifications being sent, and for shared devices it increases both the inter-HAN communication as well as the processing in both the LNC and the VRM. In addition, since event subscription is proxied by the LNC on behalf of the remote *Krox* system instance, it also affects memory consumption.
8) **Number of event notifications** – More event notifications lead to more processing in the LNC that receives the notification, more inter-HAN traffic, and finally more processing in the VRM that is subscribed on behalf of a control point.
9) **Number of shared devices vs. unshared devices** – More local devices shared with remote HAN lead to more inter-HAN traffic and more processing in the VRM of the remote HAN. More local device shared with remote HANs, also imply more overhead on the LNC for processing remote requests related to those shared devices.
10) **Number of remote HANs with which sharing is allowed** – More HANs with which sharing is allowed leads to more inter-HAN traffic and more processing on the local system's LNC for remote description, invocation, and event subscription requests.

In order to be able to control all of the above parameters in an experimental environment, an emulation of UPnP devices and control points was designed and implemented. The following sections present the design for the emulated UPnP device and control point.

6.2.1.2.1. Emulated UPnP device

The emulated UPnP device (figure 41), designed and implemented for the performance evaluation of the *Krox* system, implements the UPnP specification for discovery, description, control, and eventing, and can be configured to load a number of services and embedded devices.

The emulated device is constructed with a duration parameter that defines for how long it is available in the HAN. When the emulated device receives a search request, it responds with a search response packets for each supported service that corresponds to the service type in the request. Following the UPnP specification and the MAX_REPLY_DELAY_TIMEOUT (MX) parameter of the search request, the emulated device waits a random number of seconds (between 0 and the value of MX) before it sends its search response. In addition to responding to search requests in the local HAN, the emulated device announces its presence regularly in an interval that corresponds to its preconfigured duration by sending presence announcements to the SSDP port in the local HAN (UDP port 1900) in regular intervals. When the emulated device leaves the HAN it sends *byebye* SSDP announcements. This provides a mechanism to emulate both stationary devices, i.e. available for longer duration in the HAN, and mobile devices, i.e. available for shorter duration in the HAN, joining and

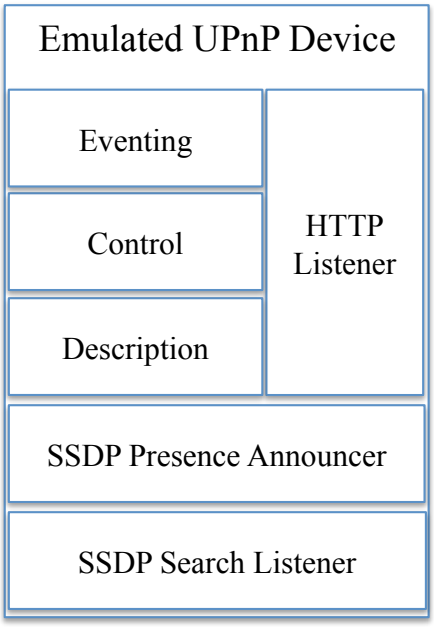

Figure 41 Emulated Device Architecture

leaving the HAN frequently.

The emulated device listens to HTTP description requests and responds with a well-formed XML description document. For the purpose of performance evaluation, the coarse-grained description request is sufficient to assess the performance impact of description requests on the *Krox* system, therefore it was not necessary to implement different types of description request/response as they are processed similarly across the *Krox* system. The returned XML description document is based on a template of a media server instrumented with the specific device details during the processing of a description request.

The emulated device supports the UPnP control protocol only in a limited way. Support is restricted to accepting UPnP control SOAP requests and returning one of two successful SOAP responses. The responses, selected in turn, differ in their size: the long response is 150 kilobytes corresponding to a "*browse*" response with 150 items and their metadata, and the short response is 4 kilobytes, corresponding to a "*browse*" response of one directory entry.

The emulated device also supports the UPnP eventing protocol, enabling control points to subscribe for changes in the service state variables. Every execution of a SOAP request on the emulated device triggers event notification to its subscribers. In "live" UPnP devices some SOAP actions lead to changes in the state of the devices and consequently in its state variables, which in turn leads to notifications being sent to event subscribers. In this sense the event model of the emulated device is a superset of these cases.

6.2.1.2.2. Emulated UPnP control point

The emulated UPnP control point, designed and implemented for the performance evaluation of the *Krox* system, is a UPnP client application that sends periodic search requests and listens to device announcements in the local HAN. In order to maximise the load on *Krox* system, all search requests sent by the emulated control point are sent with a search type: *ssdp:all* that indicates that all devices in the HAN must respond with all of their services. In fact, based on the UPnP specification, a root device (a device that is not embedded inside another device) with 3 services must respond with 6 search response

announcements, one per service, and 3 for the device - one corresponding a root device, one corresponding to the device unique identifier (UUID), and one corresponding to the device type. The control point sends search requests to the local HAN at regular predefined intervals. Following the typical control point behaviour, for each search response that corresponds to a root device, the control point sends a description request. For a preconfigured percentage of the root device announcements, the control point sends 2 consecutive SOAP requests. This corresponds for example to the interaction required for setting up streaming between a media server and a media renderer. Finally, if the device duration (as given in the device announcement) is above a preconfigured threshold, the control point subscribes for event notifications.

6.2.1.3. Experimental setup

The UPnP emulation described in the previous sections specifies a number of parameters that need to be configured for the *Krox* performance evaluation experiments. This section discusses the value selection and the reasons for selecting these values.

1) **Device duration** – As discussed earlier, UPnP devices can be divided into two groups: stationary devices and mobile devices. Stationary devices do not leave and re-join the network with any frequency e.g. a UPnP enabled refrigerator, a UPnP TV, a residential gateway. The UPnP device architecture document [133] recommends that 1800 seconds is the minimum interval between UPnP device announcements and recommends that stationary devices should have a much longer interval, typically one day. For the purpose of the experiment, stationary emulated devices are constructed with announcements interval of 900 seconds. The purpose is to increase the number of announcements and therefore the overall traffic to enable stress testing of the *Krox* system implementation. Mobile emulated device are constructed with a short duration randomly selected from the set (in seconds) {60, 120, 300, 450, 600}.
2) **Search requests** – The emulated UPnP control point is configured to send a search request every 2 minutes. The purpose of the search request is to force the VRM to process these requests and consequently respond on behalf of remote services, which leads to the control point requesting description for root devices, which finally leads to more inter-HAN communication. The emulated control point is interested only in remote devices, therefore it ignores all responses for local

emulated devices. Typically a control point does not send a broad search request for all devices in the HAN. Control points are typically in a specific type of device, e.g. media server, or Internet Gateway Device. However since only a single control point is running in the HAN, and in order to evaluate the system under stress this value was selected.

3) **Description requests** – As discussed above, whenever a search request is received for a remote root device, the emulated control point sends a description request. In addition, whenever an announcement is received from an emulated device, another description request is sent.
4) **SOAP requests** – The emulated UPnP control point is configured to send SOAP request for 10% of the device announcements it receives from remote emulated root devices. SOAP requests correspond to an action requested by a human user via a control point, or through a composite service. Therefore by artificially increasing the number of SOAP request, the load on the system via SOAP requests can emulate more the load from more HANs that actually participating in the experiment. The SOAP request is triggered whenever a device announcement or search response is received.
5) **Event subscription requests** – The emulated UPnP control point is configured to subscribe to event notifications for all stationary devices (i.e. all devices whose duration is greater than or equal to 900 seconds). The subscription request is triggered when a device announcement or search response was received. Therefore the control point may be subscribed more than once to the same service.
6) **Event notifications** – The emulated UPnP device is configured to send event notifications to subscribed clients. In a normal UPnP setup, a change to the state of the service leads to an event related to the state variable that changed. In order to increase the inter-HAN traffic, each SOAP request that is served by the emulated device results in a number of events being sent to all of the subscribers of the service.
7) **Shared vs**. **unshared devices** – for the evaluation all devices are shared with all other remote HANs. This is useful for increasing the inter-HAN traffic, and the load on each of the participating HANs.

The following section describes the hardware that was used for performing the UPnP experiment.

6.2.1.3.1. Hardware

The hardware for the evaluation included 5 desktop components such that each one corresponds to a HAN controller and runs the *Krox* system as well as the UPnP network emulation during the experiment. The desktops differ in their processors, cache size, memory, and network card. It is expected that the evaluation results will be proportional to the computing resources available in the machine running the *Krox* system:

- 5 Desktop machines with Linux Ubuntu[39] 10.04 operating system
    - Desktop 1: Intel Pentium 4, 2GHz, 512KB cache, 1GB Memory, 100Mb/s network card
    - Desktop 2: Intel Pentium 4, 3GHz, 2MB cache, 2GB Memory, 1000Mb/s network card
    - Desktop 3: Intel Pentium 4, 3.2GHZ, 2MB cache, 2GB Memory, 1000Mb/s network card
    - Desktop 4: Intel Pentium 4, 2GHZ, 512KB cache, 750MB Memory, 100Mb/s
    - Desktop 5: Intel Pentium 4, 2.4GHZ, 512KB cache, 1GB Memory, 100Mb/s
- Macbook Pro with OS X 10.6.7 operating system, Intel Core 2 Duo, 2.6GHz, 6MB cache, 4GB Memory running XMPP OpenFire server 3.6.4.

In the Desktop machines running Linux, no other software is installed besides the *Krox* system.

6.2.1.3.2. Experiment design

The evaluation setup (shown in figure 42) consists of the hardware described above, such that each desktop computer corresponds to a single HAN and runs the *Krox* system, a single emulated control point, and a set of emulated devices. A number of experiments were run,

[39] http://www.ubuntu.com/

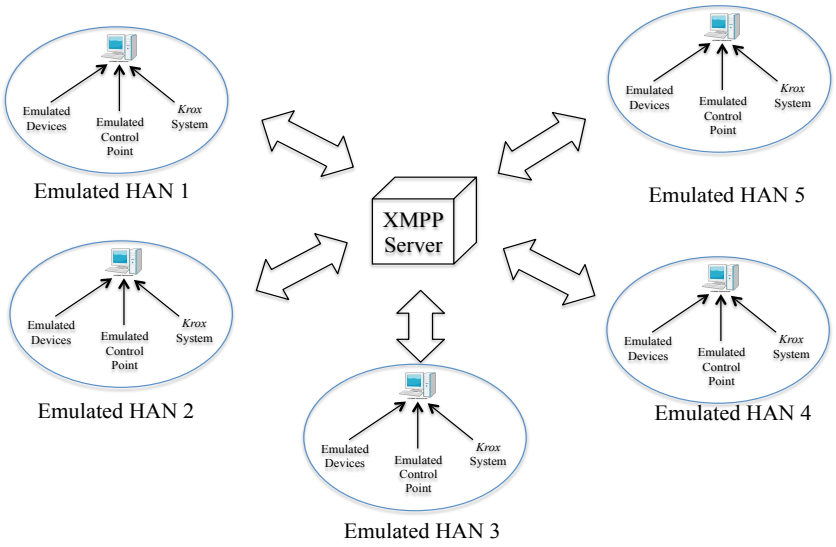

Figure 42 Experimental Setup

and the number of devices and the ratio between stationary and mobile devices depends on the specific experiment. All the participating *Krox* systems are configured to share all of their devices with all of the other participating HANs. This is in order to increase the inter-HAN traffic during the experiments. The XMPP server, installed on the 6th machine (macbook laptop), is accessible to all of the *Krox* system instances.

The following sections describe the experiments that were performed with UPnP network emulation and their results.

For these UPnP experiments, the number of HANs is fixed to 5, such that each HAN runs the emulated UPnP network and an instance of *Krox* system. In order to evaluate the performance of *Krox* system implementation with a growing number of services shared from remote HANs, the experiment included 10 steps such that each step is run for an hour. In every step of the experiment, the number of emulated devices in each HAN is increased by 5. Since there are 5 HANs and since all devices are shared with all other HANs (i.e. 4 other HANs), this results in 20 remote devices shared with each of the HANs added during each step of the experiment. Each emulated UPnP device has 3 services; therefore 20 remote devices correspond to 60 remote services shared with each of the HANs, added during each step of the experiment. For consistency, each step of the experiment is separate and involves restarting the *Krox* system and the emulated UPnP devices and control point followed by an hour of execution.

At the first step, each HAN has 5 local emulated UPnP devices, and in the final step, each HAN has 50 local UPnP emulated devices, which implies 200 remote shared devices, and 600 remote services shared in each HAN. While the requirement is for maximum number of 300 remote services shared with the HAN, testing with this larger load gives a higher degree of comfort in evaluating the system's performance under the target load.

The experiment was repeated twice with different mixes between stationary and mobile devices. The first iteration of the experiment was performed with 80% stationary devices and 20% mobile devices. The second iteration was performed with 60% stationary devices and 40% mobile devices. Each iteration repeated the 10 steps described above such that the number of devices starts from 5 per HAN in the first step, and reaches 50 devices per HAN in the last step. The purpose of running two iterations with different mobile to stationary devices ratio, was to validate the assumption that the impact of the proportion of mobile to stationary devices on the overall performance is negligible. The results show that the differences are indeed insignificant, therefore for presentation purposes the averaging of the two iterations is used, in order to focus on the significant performance issues.

Stationary devices are constructed at the beginning of each step of the experiment and announce their presence every 900 seconds. Mobile devices are constructed in the beginning of each step but are short lived and once their duration expires they are removed from the network. In order to maintain a fixed number of devices in each HAN for consistency, whenever a mobile device is removed from the HAN, another mobile device is created immediately.

6.2.1.4. Experimental results

The following sections describe the results of the experiments.

6.2.1.4.1. CPU utilisation

In order to measure the CPU utilisation of the *Krox* system, the CPU utilisation is sampled every 30 seconds. Figure 43 shows the increase in CPU utilisation of the *Krox* system with

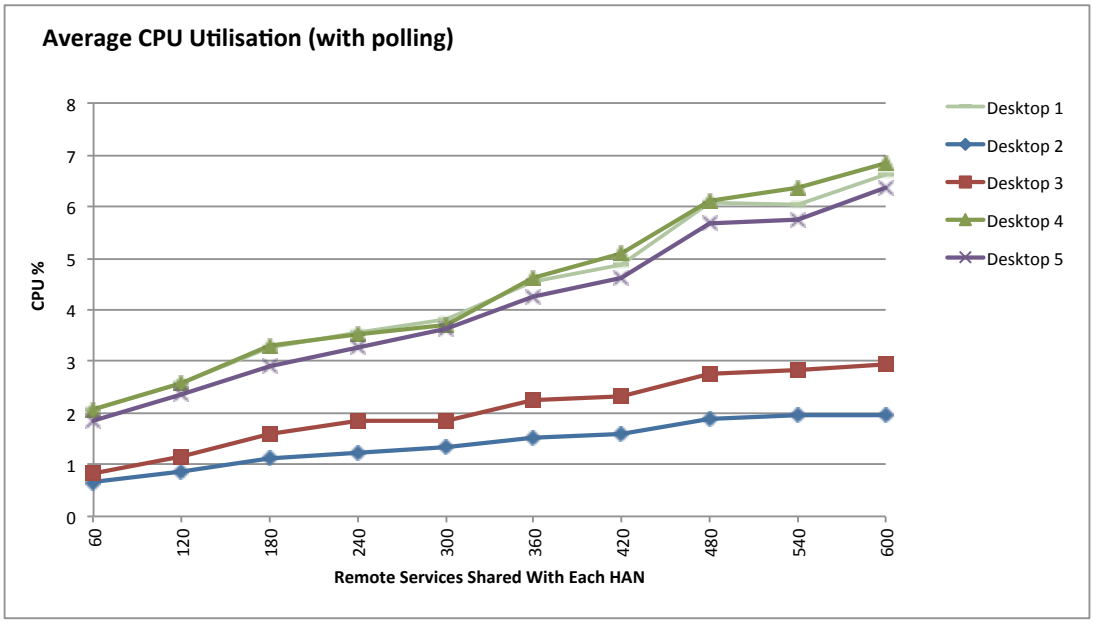

Figure 43 *Krox* System CPU Utilisation with polling

the increased number of services shared with the local HAN. It can be seen that the CPU with 300 services shared with the HAN (which is the maximum required) in all of the desktops is below 4%. Even with 600 remote shared services the CPU grows linearly to less than 7% for all of the desktops in the setup.

6.2.1.4.2. Memory utilisation

The heap memory utilisation of *Krox* system was sampled every 30 seconds. Before the heap size was sampled, full garbage collection was performed, to ensure de-allocated memory is freed. Memory is used by the LNC to store local device announcements and remote HAN's event subscriptions. The VRM stores in memory remote device announcements, cached descriptions, and event subscription information. Therefore it is expected that the heap memory will grow linearly with the number of available services (both local and remote). Figure 44 shows the growth in heap memory with the increase in remote services shared with the local HAN. It should be noted that this heap memory also includes instrumentation required for collection of results during the experiment; therefore the actual heap memory usage of the *Krox* system is even lower than what is shown in figure 45. The heap memory of the system does not exceed 6 megabytes with 300 remote services shared with the local HAN in all machines and with 600 remote shared services, the heap memory is still below 10 megabytes.

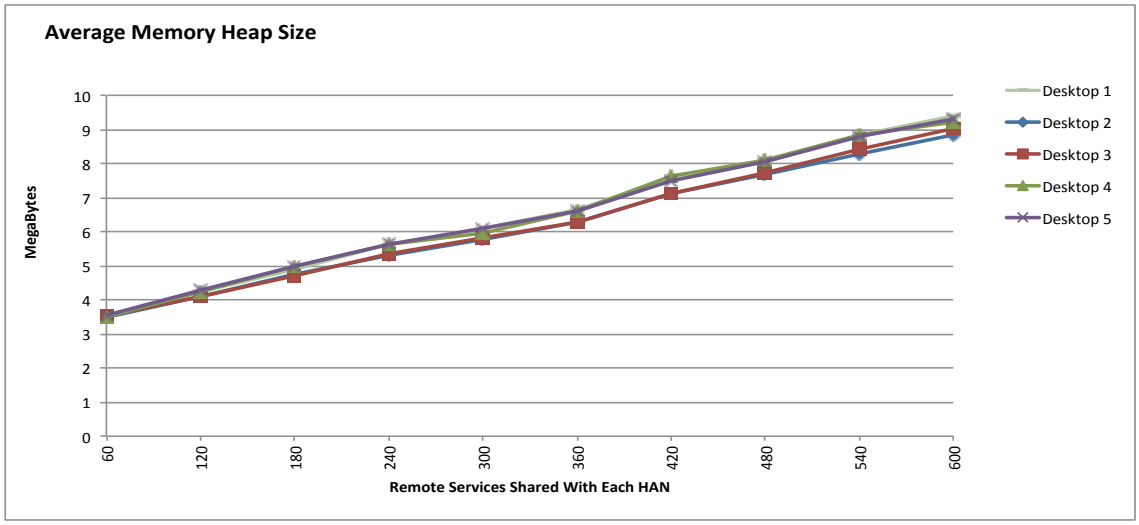

Figure 44 *Krox* System Heap Memory Utilisation

6.2.1.4.3. Local search processing

As described earlier (6.2.1.2.2), the emulated UPnP control point sends an *ssdp:all* search request every 120 seconds to the local HAN, which requires all devices in the local HAN to respond. Specifically this requires the VRM in the HAN to respond on behalf of all remote devices and services it currently represents (i.e. that have not expired). Since the VRM caches remote device/service announcements, handling search requests does not require inter-HAN communication. Instead it requires the VRM to check all cached remote device/service announcements and announce a search response for each one that has not expired. If the announcement has expired, it is removed from the cache and its removal is announced (i.e. *byebye* announcement). It is therefore expected that the processing time for search requests will be proportional to the number of remote services shared with the local HAN. According to the UPnP specification, this means for example, if the number of remote devices the VRM represents is 100, each with 3 services, the VRM needs to send 600 search responses to the requesting control point, i.e. 3 per device + 1 per each service (i.e. 3 more). The time here is measured from when a search request was received, until the VRM finished its processing, therefore sent the last search response. Figure 45 shows the average and maximum results of the measurement of the VRM's search processing time. As can be seen the processing time grows linearly with the number of services. It can also be observed that the more powerful desktops (desktop 2, desktop 3 with more powerful processors) outperform the others (desktop 1, desktop 4, desktop 5). With 300 remote

services shared with the local HAN, a search request is processed in less than 1.5 seconds. In fact, in the more powerful desktops (desktop 2, desktop 3), this is completed in less than 750 milliseconds. The maximum search request processing time for 300 remote services is less than 4 seconds. With 600 services shared with the local HAN, the average processing time reaches 2.5 seconds, which is still reasonable time for a control point to wait for device results.

The VRM implicitly implements the MX behaviour, which requires a device to wait between 0 and MX seconds randomly before sending the response to a search request (the MX is given in the search request). Therefore the sequential processing of the VRM can be considered to implement the desired MX behaviour without explicitly waiting before sending the result. To make this more accurate, when the number of remote services shared with the local HAN is small, the VRM could introduce some random wait in order not to overwhelm the local HAN with search responses. Indexing the remote device announcements by common service types could further optimise the results. While this would not change the results for the case of *ssdp:all* service type, it would improve the performance for specific service types that are commonly searched by control points.

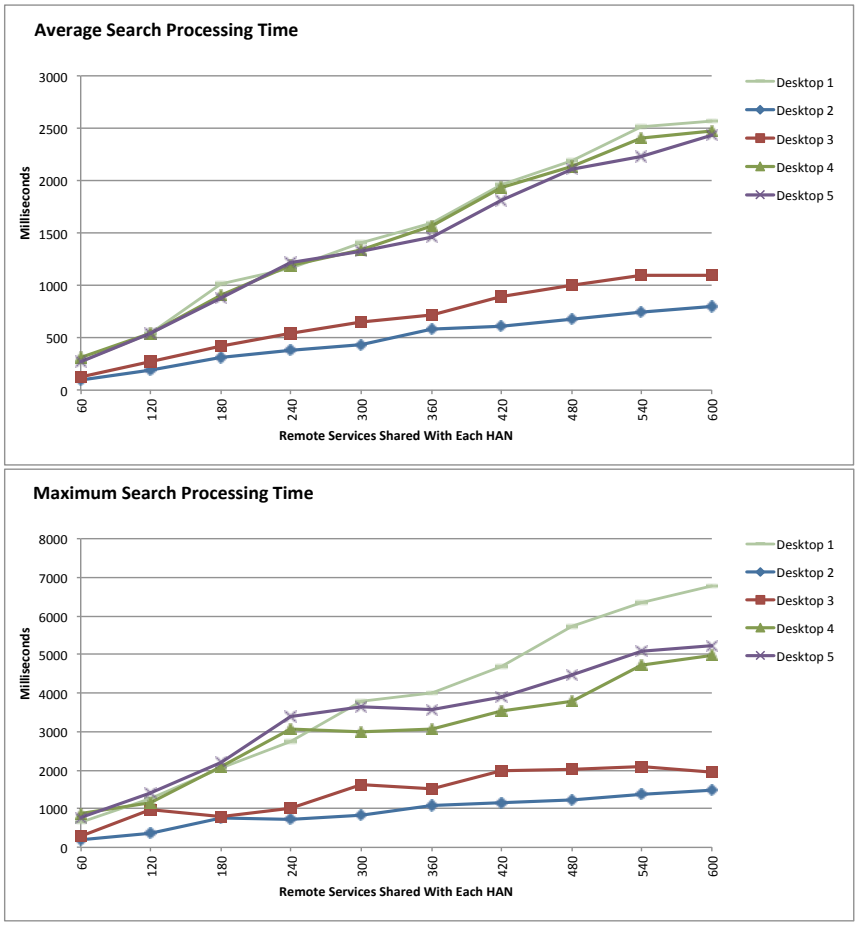

Figure 45 *Krox* System Search Processing Time

6.2.1.4.4. Discovery delay

The discovery processing of the VRM starts when a message is received in the communication subsystem indicating a remote resource has been added or removed. In the case where the resource was added, the VRM needs to construct a local announcement, with a location that corresponds to the remote device/service in the local HAN and announce it in its local HAN. Figure 46 shows the increase in discovery processing time that corresponds to the increase in remote devices/services shared with the local HAN. The processing includes construction of a device/service announcement from the original remote announcement that was sent, assignment of a local location to it and sending it to the local HAN's multicast address. It can be seen that the average processing time remains constant

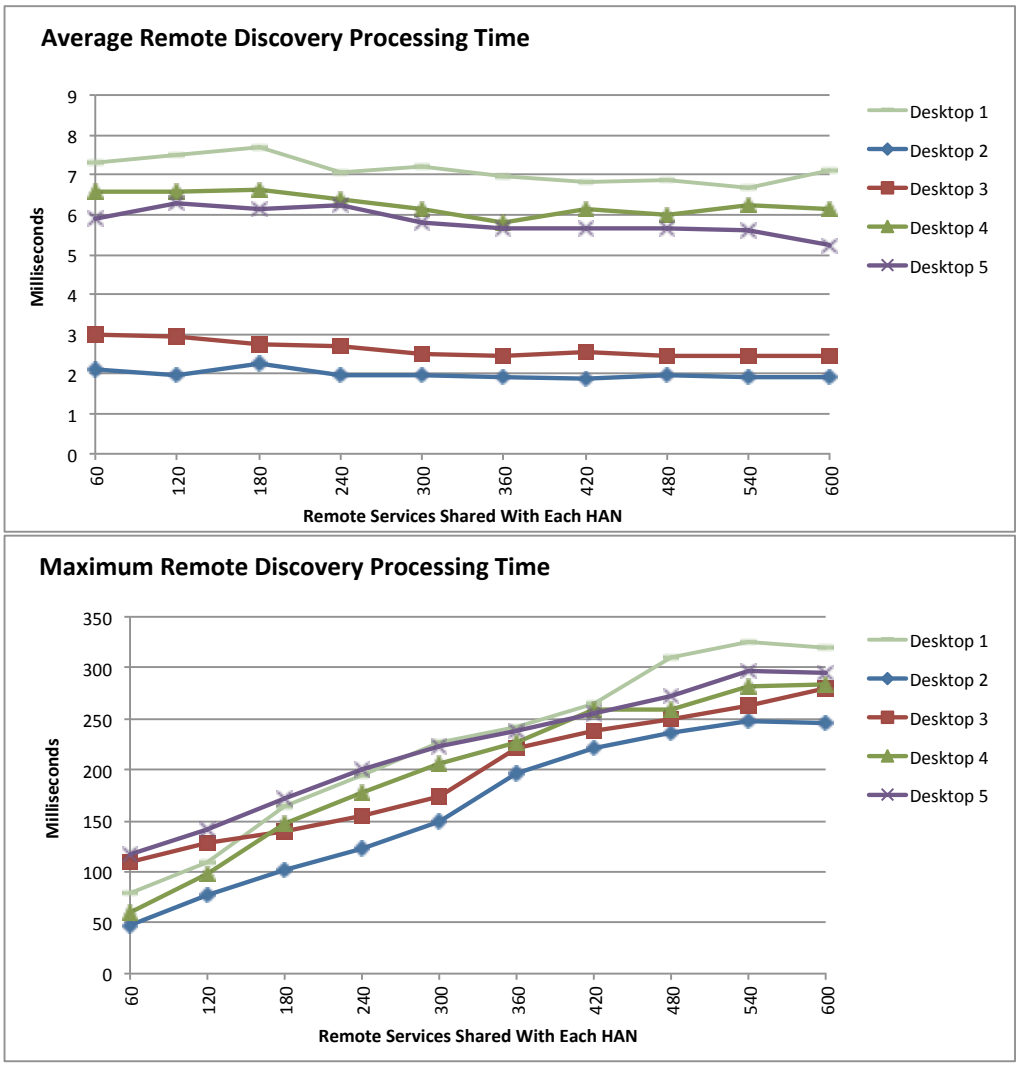

Figure 46 *Krox* System Remote Discovery Processing Time

at less than 8 milliseconds however the maximum processing time grows to less than 250 milliseconds with 300 remote services and less than 320 milliseconds with 600 remote services shared with the local HAN. The increase in the maximum while the average remains constant is explained by the fact that with more load, there are only occasional peaks that increase the maximum, however there are more samples due to the increased number of shared devices, and as most of them are handled efficiently and the average remains constant even under the high load. Moreover, the median is also constant and is very close to the average, and similarly the 90th percentile remains constant with the increase in the number of devices shared with the local HAN.

6.2.1.4.5. Remote description delay

As discussed earlier, the remote description processing time corresponds to the processing of a description request for a remote device/service in the VRM before sending a request to

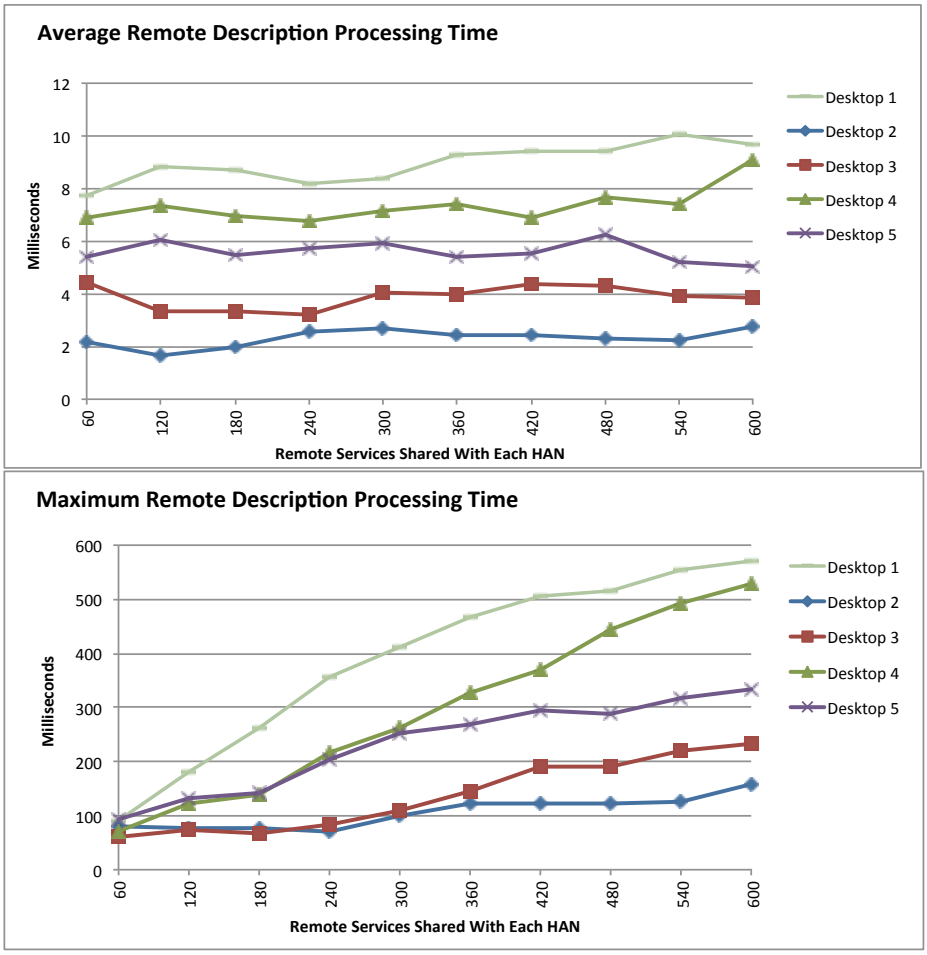

Figure 47 *Krox* System Remote Description Processing Time

the LNC, and from the time the VRM receives the corresponding description document from the LNC until it sends it to the control point in the local HAN. Once the VRM receives the description from the LNC it needs to parse it and add prefixes to the service URLs. While description latency is important, with the caching mechanism in the VRM, the description for each stationary device is only fetched once from the remote HAN. For mobile devices the description is fetched once during the lifetime of the device, however mobile devices are short lived. Therefore, the processing time of requests from the cache is even more significant. Figure 47 shows the average processing time for description requests remains relatively constant with the increased number of services shared with the local HAN. The average processing time is less than 10 milliseconds for 300 services shared, for all desktops. The increase in the load on the system with more shared services does increase the maximum processing time for remote description request. While for 300 remote services the maximum processing time is less than 450 milliseconds for all desktops, it goes above 500 milliseconds when 600 services are shared with the local HAN. For cached description request processing, the processing time (average and max) is always below 5 milliseconds for all desktops, which is useful for reducing the inter-HAN traffic and processing overhead in the VRM.

6.2.1.4.6. Remote invocation delay

Remote invocation processing time measures the time it takes the VRM in the local HAN to process a SOAP request – from the time a request is received in the VRM, until it sends a message to the remote HAN requesting the invocation, and from the time it receives the SOAP response, until the message is sent back to the requesting control point. Figure 48 shows the average and maximum processing time for SOAP requests in the VRM. It can be seen that the average SOAP processing overhead increases slowly with the growing number of devices. In addition it can be seen that for all desktops in the experiment the maximum processing time of invocation requests in the VRM is less than 250 milliseconds for 600 shared services and under 175 milliseconds for 300 shared services.

The number of SOAP requests made by local control point for remote devices grows from 78 (with 60 remote services shared with the local HAN) to 780 (with 600 remote services shared with the local HAN), which produces the resulting additional load on the local *Krox*

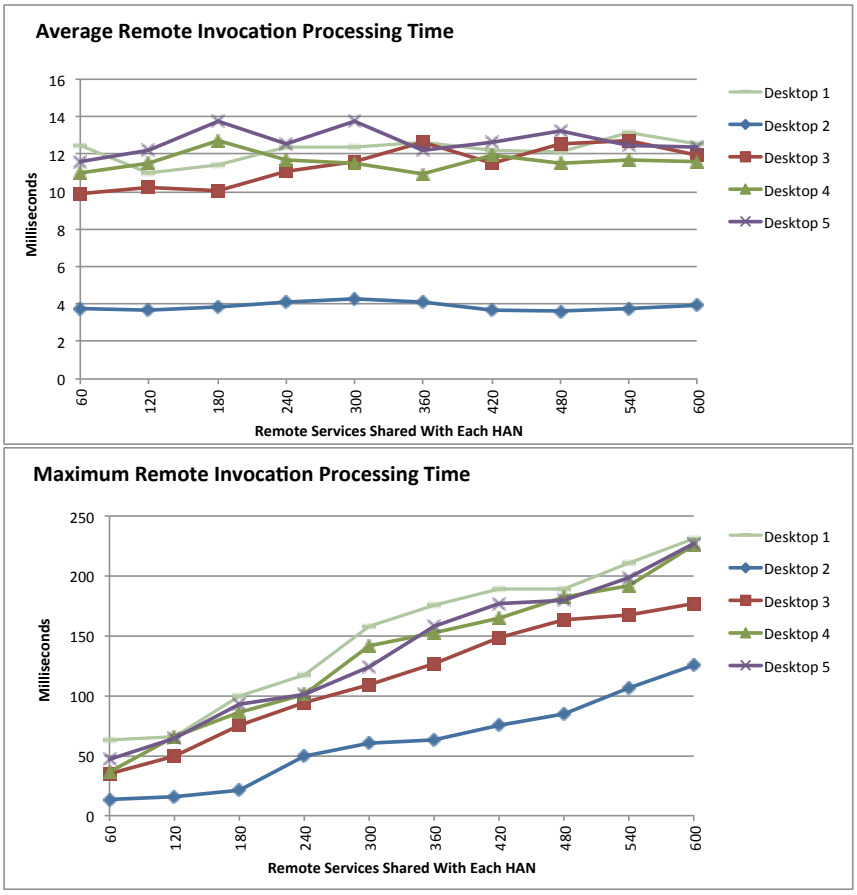

Figure 48 *Krox* System Remote Invocation Processing Time

system which affects all of the measurements, not only the processing time of SOAP requests.

6.2.1.4.7. Event notification processing time

When the LNC receives an event notification that corresponds to a subscription made by a remote instance of *Krox* system, the LNC forwards the event using the communication subsystem to the subscribed *Krox* system. In the remote *Krox* system, the VRM resolves the actual subscriber and sends the event notification. The event processing time is the time it takes the VRM from when it received the event notification message from the remote LNC, until when the VRM sends the notification to the subscribed control point. Figure 49 shows that the event notification processing time in the VRM increases slowly with the growing number of devices with less than 15 milliseconds per event notification with up to 600 remote services shared with the local HAN. It should be noted that the number of event notifications grows with the number of services (or more precisely with the number of

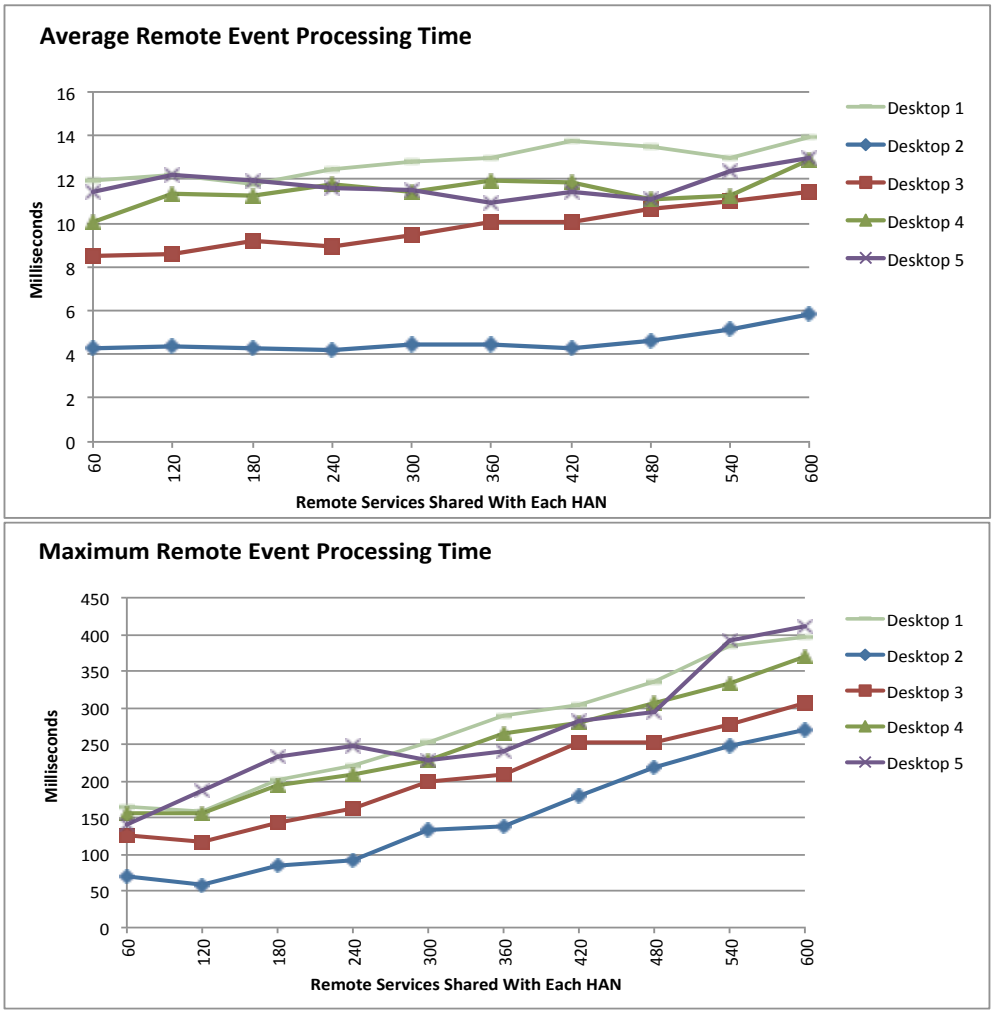

Figure 49 *Krox* System Event Notification Processing Time

SOAP requests and event subscriptions made by the control point). The number of event notifications grows from 300 per hour (for 60 services shared with the local HAN) to 3000 per hour (for 600 services shared with the local HAN). With more services in the HAN the maximum event notification processing time grows to 250 milliseconds with 300 remote services shared with the local HAN and 412 milliseconds with 600 remote services shared with the local HAN.

6.2.1.4.8. Bandwidth utilisation

Figure 50 shows the increase in inter-HAN bandwidth, utilising the upstream and downstream Internet connection of the household, with more devices shared devices. The messages correspond to inter-HAN discovery announcements, description request and responses, control requests and responses, and event subscription and notifications. The

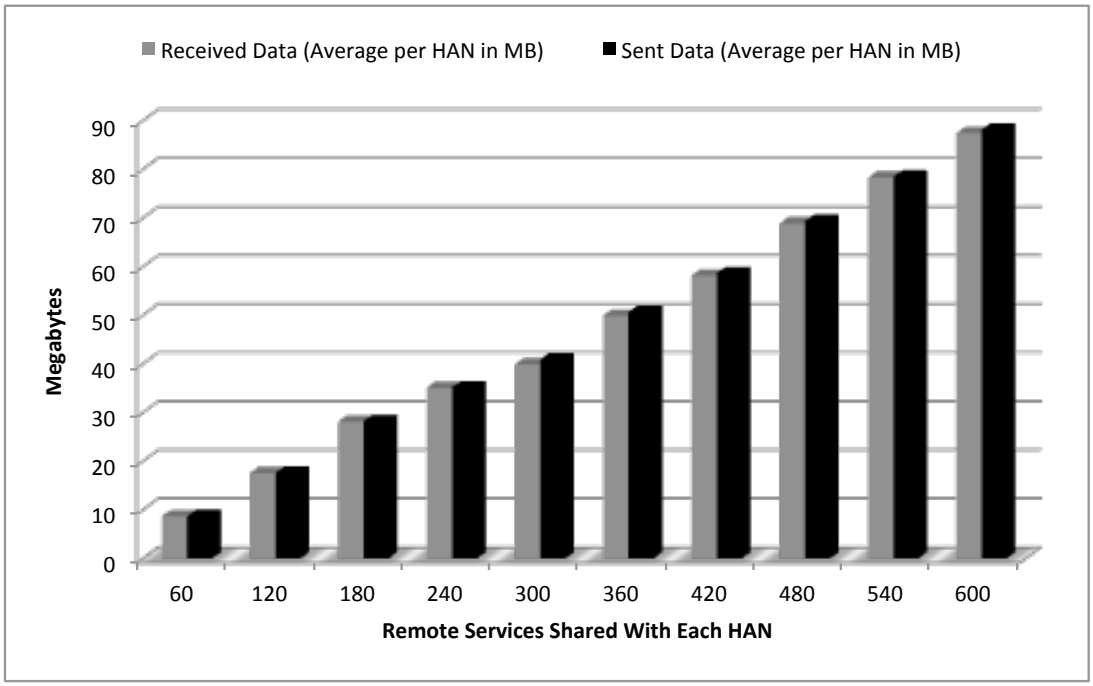

Figure 50 *Krox* System Bandwidth Utilisation

messages are counted in the communication subsystem, before they are sent and after they are received, and their size in bytes is measured. It should be noted that the amount of bandwidth consumed in this experiment is dominated by the large SOAP responses. With 5.8-58 megabytes used for 39-390 large SOAP responses (in one hour), the bandwidth used for discovery, description, eventing, and shorter SOAP responses, which is the remaining bandwidth used is relatively small.

6.2.1.4.9. Summary

The sections above presented the results of the controlled experiment for *Krox* system implementation with UPnP plug-in and a UPnP emulated network with a setup of 5 HANs, such that each HAN is sharing all devices with all other 4 HANs. The results show linear scalability of the system both on average as well as on the maximum. The increase of the CPU utilisation shows that with up to 300 remote services in the local HAN the CPU is still relatively low at less than 4% on average on all desktop machines. The heap memory required to run the *Krox* with up to 300 remote services is less than 6 Megabytes. While this is relatively low, in addition to the memory required for the intra-HAN service interoperability, this may exclude potential deployment platforms, which are more resource constrained. The VRM response time to search requests made every 2 minutes with a search

type of *ssdp:all* grows to less than 1.5 seconds on average with up to 300 remote services. With a maximum of up to 4 seconds these result indicate that control points in the local HAN can seamlessly discover remote services with no significant difference from local devices and services. From discovery point of view, an important factor is how long it takes before a device is discovered in its original HAN until it is announced in remote HANs with which it is shared. From the results shown in this experiment, it is apparent that this delay will be dominated by the network latency between the HANs because the local processing time of the VRM given a remote announcement is relatively small. Remote description delay is an important factor because it affects the seamless interaction with control points. The results show that on average this delay will be dominated by the network latency between the HANs and the time it takes the actual device description to be fetched. However when adding these delays, the overall delay exhibited by the VRM may be excessive in some cases. Therefore the use of caching is important and significantly improves the latency seen by the control point. Remote invocation delay affects the interaction with a human user or a composite service that is waiting for its completion. The results of the experiment show that on average the delay introduced by the VRM is negligible, however the maximum delay add to the network latency and the actual time taken for the device to return the response may impact the user experience. This is especially important in nested composite services where each layer can introduce a latency that when considered separately is negligible but the accumulative delay may be excessive. Event notification is an important mechanism for enabling control points to keep up to date with the state of devices they interact with. For example, a control point may want to know the status of a media player (playing, stopped, paused) so it can present it to its users, in addition a composite service may be using events to trigger some functionality. Therefore event notification delay should be relatively small. The experiment shows that on average the event delay is dominated by the network latency, and can enable seamless integration with control point applications with no significant delay. Similarly to remote invocation delay, in a nested composite service, when the maximum delay is exhibited in addition to the network latency, this delay may be excessive and may need to be reduced. Finally the experiment shows that the levels of bandwidth used for the communication required for enabling remote services to be discovered in local HANs is relatively small, therefore the actual amount of bandwidth used for communication between the HANs will be dominated by the amount of data used for interaction with devices, e.g. retrieval of play lists, or actual streaming of data.

As discussed earlier, since the setup is limited to 5 HANs sharing devices and services, in order to be able to reason about the scalability to more HANs, the emulated control point increases the load on the local HAN artificially. If the number of services shared with remote HANs remains fixed, more HANs with which devices are shared lead to more overhead on the LNC, with more description requests, SOAP requests, and event subscriptions from these remote HANs. Therefore the emulated HAN artificially increases the load on the LNC by sending more requests than is expected for sharing devices and services with 4 remote HANs

Finally, the experiment was performed in a controlled environment, with no additional network activity such as downloading or uploading of content done in parallel to the experiment using the HAN's bandwidth. Therefore it is apparent that the results only provide lower bounds on the potential performance of the system under realistic conditions.

6.2.2. Jini

In the implementation described in the previous chapter, remote Jini services are represented in the local HAN with lightweight objects (GenericJiniForwarder). Additionally, Jini requires both HANs to have the Java interfaces of the services they wish to interact with in their path, therefore during discovery, only the name of the interface, the service instance identifier and the service attributes are sent using the communication subsystem. This guarantees that discovery related inter-HAN traffic is efficient. The main challenge with regard to remote Jini services is the invocation latency, as the number of hops from a client calling a method on a proxy to the service implementation being invoked in a remote HAN is relatively high as discussed in section 5.6.2.

In order to evaluate the overhead of invoking actions of remote services, a sample Jini service was implemented (this is the name guessing service shown in figure 38). In this experiment two HANs were used such that one HAN (HAN1) was hosting the implemented Jini service and sharing it with the other HAN (HAN2), therefore the Jini service from HAN1 was shown in HAN2 as a remote Jini service. The service takes a String as a parameter and returns a String. To check the remote overhead of sending the request and response over the communication subsystem, the local execution time was compared with the remote execution time. In 100 executions, the method took 0.92 milliseconds on average

with the local Jini service. This time does not include the initial lookup but only the invocation of the method on the proxy object that was returned from the lookup service. In 100 executions of the method on a proxy obtained for a virtual Jini service took 35.17 milliseconds on average. This includes the full implementation of the remote invocation. The remote invocation added an overhead of ~34 milliseconds per each call. Considering that the method tested simply returns a string concatenation and is very light weight, it is a reasonable assumption that with a method that implements some more complicated function and with running on a public network infrastructure and the latency it may add, this overhead would be negligible.

6.2.3. Summary

In the previous sections, the results from the inter-HAN service interoperability performance evaluation performed using the *Krox* system implementation were described. In this section the performance requirements as defined in section 3.4 are assessed based on the results presented in the previous sections.

6.2.3.1. Scale-up (intra-HAN) – REQ #16

The requirement (REQ #16) is to be able to represent 300 remote services in the local HAN. The results of the experiment described in 6.2.1.4 show that the *Krox* system with UPnP service protocol plug-in performs well with up to 300 remote services and scales linearly beyond that. The average CPU is below 5% for all systems in the evaluation, heap memory is below 8MB. Local search requests are completed on average in less than 1.5 seconds and the maximum processing time is less than 4 seconds. The overhead of the VRM for discovery, description, control and event notification is negligible – on average it is below 15 milliseconds, and the maximum processing time is below 400 milliseconds. The results of the experiment for Jini services described in section 6.2.2 shows that the overhead for Jini service invocation is negligible and would be dominated by the network latency rather than by the processing overhead of the VRM or LNC. It is important to restate that the experiment does not evaluate the actual latency of the network between the HANs. The purpose is to evaluate the performance of the system in a controlled environment. A production system would be affected by the network latency and the upstream and downstream of HANs, however the relevant metrics measured in this evaluation should still hold and can therefore be compared against other systems implementations.

6.2.3.2. Scale-up (inter-HAN) – REQ #17

The requirement (REQ #17) is to be able to share devices and services from the local HAN with up to 15 remote HANs. As discussed in the experimental design (section 6.2.1.3.2), the experiment was designed such that load on the LNC will represent more than the 4 HANs with which devices are shared. The results of the evaluation indicate that the number of remote HANs can be extended without incurring significant performance overhead. The number of HANs (as opposed to the number of services shared from remote HANs) affects only the LNC and only subset of the LNC behaviour. The VRM is only affected by the number of services shared from remote HANs rather than by the number of remote HANs. The following aspects of the LNC are affected by sharing with more remote HANs:

1) The number of messages sent by the LNC about added/removed resources – The overhead of sending additional messages is negligible. The communication subsystem inherits messaging scalability from its underlying IM&P system, and given the inherently small size of discovery messages, the overhead of an additional message is negligible.
2) The number of description requests received in the local HAN – More remote HANs with which local services are shared will lead to more description requests from client applications in these HANs. The maximum number of these description requests therefore corresponds to the number of shared services, such that each HAN would request the service description no more than once.
3) The number of SOAP requests received in the local HAN – More remote HANs with which local services are shared could also potentially lead to more SOAP request made by client applications from these HANs to interact with the local HANs shared devices and services.
4) The number of event subscriptions received in the local HAN – More remote HANs with which local services are shared could potentially lead to more event subscriptions in the local HAN made by client applications in remote HANs, which could lead to more event notifications being sent. However the overhead of more event notifications on the LNC is restricted to resolving the identifier of the *Krox* system in the remote HAN and sending the event notification using the communication subsystem. Since event notification has a very low overhead, and

only indicates a change in a state variable, it is limited in size, therefore it does not require excessive bandwidth, even with additional HANs.

While it cannot be directly argued that it follows from the scalability of the results that the system can scale to a larger number of HANs, it can be reasoned from the design of the experiment and from the linear scalability observed that the system is capable of scaling to more than 4 HANs with reasonable performance.

6.2.3.3. Scale-down – REQ #18

The Java implementation of the *Krox* system enabled the system to be deployed on multiple platforms. The performance evaluation was performed on Linux, however the system was also tested successfully on Windows and OSX and worked as expected. The requirement (REQ #18) is to be able to run the system on machines with processor with clock speed of no more than 2GHz and no more than 1 gigabyte of RAM. Out of 5 desktops that were used for the performance evaluation 3 of them have processors with 2GHz (desktop 1, desktop 4, desktop 5). The same desktops have 1 gigabytes of RAM or less (desktop 4 has 750 megabytes). Desktop 2 and Desktop 3 have more processing power and RAM. The reason for using more powerful machines in the experiment was to assess the effect of more processing power on the results.

6.2.3.4. Concurrent access – REQ #19

As indicated in the design and implementation chapters, there is no inherent mechanism preventing concurrent access to local HAN's resources (REQ #19). During the UPnP experiment, and Jini experiments concurrent requests were made and they are not synchronised in any way by the *Krox* system.

6.2.3.5. Conclusions

As shown by this inter-HAN service interoperability experiment and summarised in this section all of the performance requirements (REQ #16-REQ #19) are addressed by the design and implementation.

6.3. Intra-HAN service interoperability performance evaluation

The service composition subsystem affects performance both in terms of processing and memory. The service composition subsystem requires a servlet container for hosting the generated web services, and an orchestration engine for hosting and executing composite services. From performance point of view there are number of parameters that need to be evaluated:

1) The overhead of generating of web services – How long it takes to generate web service proxies and deploy them
2) What is the memory overhead of a deployed web service on the servlet container
3) What is the latency introduced by the additional layer of indirection for interaction with a service – i.e. interacting with a service through a web service adds another chain of processing that handles the request

The following sections present an evaluation of the *Krox* service composition subsystem with these parameters.

6.3.1. Experiment Environment

Web services are typically hosted in a servlet container, although with Java 6 this is not mandated. However, the service orchestration engine, Apache ODE cannot run as a stand-alone Java program and is deployed as a web application in a servlet container. The prototype used Tomcat servlet container version 6.0.18, which requires only 10-15MB of RAM including the orchestration engine installed as a web application.

6.3.2. Web services

The process of automatic web service generation and deployment includes several steps as described in the previous chapter. When a device is discovered an automatic JAX-WS web service is generated and built for each service that the device contains. The web service is then packaged and deployed in the local application server. This whole process requires

	UPnP	Web Service Wrapped UPnP	Round-trip (Client, Web Service, UPnP)	Web Service Overhead
Play	112.36ms	117.86ms	202.44ms	90.08ms
Pause	365.46ms	370.98ms	614.85ms	249.39ms
Stop	342.35ms	347.44ms	585.0ms	242.64ms

Table 7 Performance Overhead of UPnP Web Service Proxy

between 2-5 seconds per service as measured on the generation of a web service for a UPnP AVTranport service (measured 1000 times).

Table 7 shows indicative performance overheads introduced by the web service proxy for UPnP services prototype implementation for 3 media renderer AVTransport service operations ("Play", "Pause" and "Stop"). The first column (UPnP) shows the time taken to directly invoke the operation using SOAP messaging. The second column shows the server-side time required to perform the same operation when it is wrapped in a web service interacting with the UPnP control point application, here the additional redirection overhead introduced by the server-side processing is approx. 5 milliseconds. Column three (round-trip) shows the time taken to invoke the operation from a Java web service client on the same host as the web service and demonstrates the additional overhead introduced by the client side processing to form the web service request, passing it to the web server, and then waiting for the web service response. The overall overhead introduced by the web service based approach is equal to round trip time (column 3) minus the UPnP time (column 1) as shown in column 4. On average this time equals to ~200 milliseconds, and it is important to note that it includes the marshalling and un-marshalling of Java objects from XML which is required only for a Java client and is not relevant for a composite service using the web service.

6.3.3. Summary

The results shown in the section above comparing the direct invocation UPnP services against the web service proxy invocation of the same services indicate that the overhead added by the web service is relatively small – 200 milliseconds on average for UPnP services. Table 7 shows the overall time spent in the service implementation in the second

column and the overall time for a call to the service to return in the third column. It can be seen from table 7 that the overhead added by the web service implementation is negligible and constant while the overhead related to the calling the web service as shown in the third column is 200 milliseconds on average.

The web service generation is time consuming (2-5 seconds) as indicated in 6.3.2. However, the web service generation does not need to slow down discovery and can be streamlined as a parallel process. The time is dominated by the time it takes to compile, package, and deploy the service, which are done in another process. In addition, the implementation did not use caching of web services to reuse the same web service when a service that was already available in the HAN re-joins. With such caching, when a service that was already available in the HAN once is available again, its corresponding web service (packaged as a web archive) is copied from the caching directory to the auto deploy directory of the servlet container enabling fast deployment, saving the time required for compiling and archiving the web service.

This shows that the web service approach taken by the *Krox* system design works as expected and does not incur significant overheads. It is extensible through the service protocol plug-ins and enables intra-HAN service interoperability and service composability. The web service approach is limited to addressing the syntactic service interoperability and does not address the semantic interoperability.

6.4. Security analysis

HANs are exposed to various types of attacks for example malicious code (such as Trojan horses, viruses and worms), Denial of Service (DoS) attacks and eavesdropping. These attacks may differ in their approach, their goal and the way they interact with the HAN. However, the HAN is typically considered by its users to be a closed and trusted domain. With the *Krox* system for inter-HAN service interoperability, it needs to be guaranteed that only trusted HANs can see shared devices and services and that only a trusted HAN can share devices and services. In addition, extending the boundary of the HAN can introduce new threats that need to be analysed in order to defend against attack on a single HAN from spreading to HANs with which devices and services are shared.

The first step in protecting a system is to have a deep understanding of the potential threats. In the context of the *Krox* system, this means identify both potential attacks on the *Krox* system itself as well as assess the impact of existing threats and vulnerabilities relevant to UPnP and Jini service protocols on the *Krox* system, specifically in preventing attacks on these service protocols from using the system to spread to multiple HANs.

In order to consider the potential attacks of the system, the following sections provide a systematic analysis of the known threats that are relevant in the context of the *Krox* system architecture and mitigation techniques. The term weakness refers to a security vulnerability in the specification or behaviour of an entity. The term vulnerability refers to a flaw or a security weakness in an asset that can be exploited by a threat [40]. A negative impact can be the result of a malicious or an accidental action [98]. Attack trees provide an intuitive and systematic way to represent the weaknesses and risks of a given system [115]. In the following sections attack trees are used to identify the potential threats to the *Krox* system, based on decomposition of the system to its components that are susceptible to attacks: the communication subsystem, and the service protocol plug-ins. The communication subsystem uses IM&P as its underlying communication protocol for providing messaging and presence capabilities to the *Krox* system instances, therefore the security analysis for the communication subsystem can extend existing work in the area of IM&P security in the context of the *Krox* system, as discussed in section 6.4.1. An assessment of the potential attacks on UPnP devices and their impact on the *Krox* system, specifically the potential for attacks to spread to multiple HANs is given in section 6.4.2. Finally an assessment of the potential attacks on Jini services, and the potential of these attacks to exploit the *Krox* system to spread to multiple HANs is presented in section 6.4.3.

6.4.1. Communication subsystem

The communication subsystem in the *Krox* architecture enables instances of the system in multiple HANs to exchange messages and be notified on status changes of the system in remote HANs for HANs with which sharing has been agreed. For the purpose of this thesis, the focus is on how to confine an attack made to a single HAN and prevent it from spreading to multiple HANs with which it is sharing or using devices and services. The communication subsystem embeds a number of security mechanisms to facilitate secure

communication as described in the design security considerations (section 4.3.6) including authentication, encryption, and auditing. The analysis in this section is adding to the security mechanisms that are recommended for IM&P and specifically XMPP based communication. The main focus of this section therefore is on DoS attacks and their potential to spread using the communication subsystem to additional HANs.

A Denial of Service (DoS) attack is defined by RFC 4732 as an attack in which the attacker attempts to cause the victim's machine or resource to become unavailable to its legitimate users [53]. DoS can be caused by high demand, faulty implementation or a malicious attack. For *Krox* communication subsystem an DoS attack can be made from within the local HAN, from another HAN with which devices and services are shared, or from a foreign HAN. A DoS can be caused by a software bug, either in the IM&P endpoint, or in other components, service protocol plug-ins, or by vulnerabilities in service protocols exploited by an attacker. While the above could lead to the denial of service of the local *Krox* system, unless defence mechanisms are used, these attacks can spread beyond the scope of *Krox* system to other HANs. Figure 51 illustrates the potential DoS attacks on *Krox* communication subsystem using an attack tree. The following sections describe the nodes on this attack tree from left to right.

6.4.1.1. Messages flood

A flood of messages sent to the *Krox* system in the local HAN can cause a denial of service in the system. If the messages correspond to device announcements in the local HAN, they may attempt to spread the attack to all remote HANs with which the service is shared and cause denial of service in them, not only to the *Krox* system, but also to other systems in those HANs. The following sections assess the effect of various scenarios of messages flood

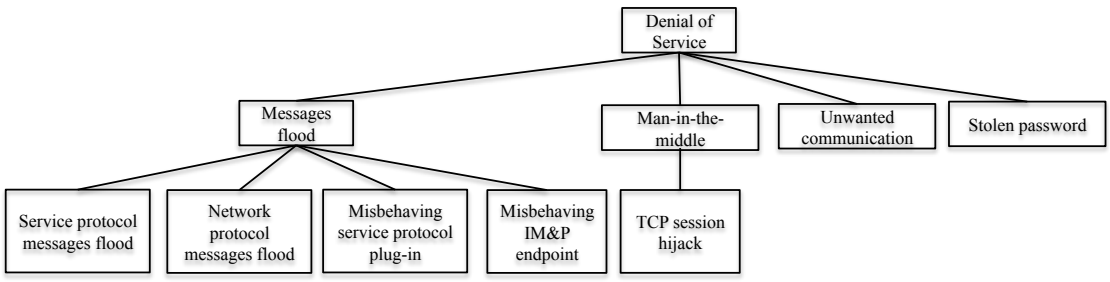

Figure 51 Communication Subsystem Attack Tree

on *Krox* system and suggest corresponding defence mechanisms.

6.4.1.1.1. Service protocol messages flood

Description: If the service protocol is vulnerable in such way that can lead to a service protocol messages flood, such a scenario, caused by either an attacker exploiting the vulnerability, or by a software bug in a device or a service could result in the corresponding *Krox* service protocol plug-in flooding the communication subsystem with messages to *Krox* systems in remote HANs.

Effect on *Krox* system: A service protocol causing a messages flood can result in denial of service in the local *Krox* system, but can also cause denial of service in remote *Krox* systems, and client applications and lookup services in those remote HANs.

Recommended defence mechanisms: In order to prevent a messages flood from causing denial of service in the system or hosting HAN, the communication subsystem should support rate limiting as described in more details in section 4.3.6.4. The rate limiting mechanism was not implemented as part of the prototype implementation. Rate limiting is useful only from preventing the messages flood from spreading to remote HANs. The *Krox* system in the local HAN will eventually run out of memory.

6.4.1.1.2. Network protocol messages flood

Description: Network protocol messages flood refers to an attacker trying to shut down the port which is used by the communication subsystem for exchanging messages with remote HANs.

Effect on *Krox* system: In case the attack is successful the communication system will be disconnected from the IM&P server and will not be able to participate in sharing with remote HANs.

Recommended defence mechanisms: While an arbitrary network protocol attack on the communication subsystem can definitely shut it down and prevent it from being able to communicate with remote HANs, it may not spread to remote HANs, therefore no specific defence mechanisms are defined.

6.4.1.1.3. Misbehaving service protocol plug-in

Description: A misbehaving service protocol plug-in can initiate messages flood to remote HANs either due to a bug or a malicious plug-in. For example, the service protocol plug-in can send more messages to remote HANs than required by the service protocol, e.g. by duplicating discovery announcements.

Effect on *Krox* system: The effect on the system is similar to that of a messages flood caused by a service protocol.

Recommended defence mechanisms: While it is assumed that service protocol plug-ins running in the HAN are well tested, bugs can still exist, and in order to defend against this vulnerability, the rate limiting mechanism discussed above should be used.

6.4.1.1.4. Misbehaving IM&P endpoint

Description: A misbehaving or malicious IM&P endpoint is an authenticated client of the IM&P server that initiates a messages flood to the *Krox* system in the local HAN. The attack can come from within the HAN, from another HAN with which the local HAN shares or uses devices and services, or from a foreign HAN. The messages flood could contain presence messages, service protocol specific messages, or other types of messages.

Effect on *Krox* system: The effect on the system is similar to that of a messages flood caused by a service protocol.

Recommended defence mechanisms: This type of attack should be handled at the level of the IM&P server by restricting the number of connections and the bandwidth that a client (an IP address) can use in a given time. In addition, the IM&P endpoint in the local HAN should ignore all messages from other endpoints that are not in its buddy roster. While the local HAN can still be affected, this would defend against an attack from a foreign HAN from spreading to remote HANs from the buddy roster of the HAN.

6.4.1.2. Man in the middle

Man in the middle (MITM) attack is an attack where the attacker intercepts and tampers with the communication between the victim and the host. By injecting themselves between the victim and the server, the attacker can invoke an attack on the server while impersonating to the victim, or an attack on the victim while impersonating the server.

6.4.1.2.1. Client TCP session hijack

Description: An attacker can intercept a conversation between the client and the server, and potentially manipulate it and create a man-in-the-middle attack. This could be done using TCP session hijack techniques as demonstrated in [145]. By tampering with the messages sent between a *Krox* instance and the IM&P server and modifying them with malicious content an attack against a remote instance of *Krox* system, or an IM&P server can be established.

Effect on *Krox* system: Such an attack could have different effects on the local and remote *Krox* systems depending on the goal of the attacker. This could lead to disconnection of the local HAN from the IM&P server or to corrupt or modify data to appear as if sent from the *Krox* system in the local HAN to remote HANs.

Recommended defence mechanisms: In [105] several mechanisms were identified to improve the robustness of TCP against session hijacking. Since these techniques are applied in the implementation of the TCP library, and since all communication in the *Krox* system architecture is made via IM&P communication, these techniques should be considered for the IM&P client and server implementations.

6.4.1.3. Unwanted communication

Description: Unwanted communication refers to an authenticated IM&P endpoint that sends messages to the *Krox* system, although it is not in its buddy roster. Such an IM&P endpoint can send arbitrary messages it to the *Krox* system, long messages consuming its resources.

Effect on *Krox* system: Unwanted communication from an IM&P client can result in denial of service in the local *Krox* system, however it cannot spread to multiple HANs. It cannot result in information disclosure because actions will only be invoked on behalf of remote HANs with which sharing has been agreed, based on the sharing configuration of the home user.

Recommended defence mechanisms: The communication subsystem of *Krox* needs to guarantee that the local system only receives messages from remote *Krox* systems with which sharing has been agreed. The filtering can be done in the IM&P client side of *Krox* such that messages are checked before they are processed, however this could be

implemented more efficiently as an extension to the IM&P server before messages are sent to the IM&P client.

6.4.1.4. Stolen password

Description: If an attacker obtained the user/password information for *Krox* system in HAN1, he can use this information to access remote devices shared with the HAN1. In addition the attacker can use the stolen identity to facilitate denial of service attack while assuming the identity of the victim.

Effect on *Krox* system: An attacker with a *Krox* credential could connect from any HAN and either attack remote HANs with which the victim's *Krox* system was sharing resources, or consume resources from these HANs without permission. The attack as well as consuming resources shared with the victim's HAN would seem as if they are made from the victim's HAN.

Recommended defence mechanisms: In the *Krox* system, authentication takes place after TLS has been negotiated therefore the password cannot be obtained during the authentication process by sniffing the communication. However, there is no inherent mechanism in *Krox* architecture to defend against stolen password. The password should be stored in *Krox* system in such way that it is hard to compromise. The system could provide indication of when the system was last authenticated and the IP address, which was used for the authentication. In addition the system can provide an indication when another login was made with the same identity, which can give the home user an indication that his password was compromised. These mechanisms however were not implemented in the *Krox* prototype system.

6.4.2. UPnP service protocol plug-in

UPnP architecture has several potential security vulnerabilities and weaknesses. Some of the vulnerabilities are inherently related to the way the UPnP protocol works, while others are related to specific device implementations. Several vulnerabilities are related to specific classes of devices such as Internet gateway while others are more generic. Following the attack tree methodology, UPnP attacks can be classified by the goal of the attacker into two groups: denial of service, and malicious management actions. The purpose of this section is not to define mechanisms to make UPnP more secure, but rather to defend attacks that

exploit UPnP vulnerabilities from spreading to remote HANs using the *Krox* system. The following sections present these potential attacks, the goal an attacker wishes to achieve, the technique for achieving this goal, the effect on HANs running the *Krox* system, and mechanisms to defend against the attack from spreading across the boundaries of a single HAN. Section 6.4.2.1 discusses the potential DoS attacks on UPnP and their effect on the *Krox* system. Section 6.4.2.2 discusses the effect of malicious management actions. Finally section 6.4.2.3 discusses the effect of eavesdropping to UPnP protocol in the local HAN.

6.4.2.1. Denial of Service

In the context of UPnP, DoS can cause UPnP devices to shut down, and can cause control points to run out of memory, or crash. Figure 52 shows an attack tree that was created for this analysis summarising the potential techniques for DoS attacks on UPnP networks. The following sections are organised according to the nodes in the tree from left to right, such that for each leaf node, the type of attack is described, with the potential effect on the *Krox* system, and the recommended defence mechanism.

6.4.2.1.1. Misbehaving device

A misbehaving UPnP device can either be a faulty device, a device with a bug, or a malicious device installed by an attacker or by the home user by mistake, such as by installing a Trojan horse that can act as a malicious device. In addition, the misbehaving device can be a friend's mobile device joining the HAN. There are several ways that a misbehaving device could lead to denial of service in the local HAN as described in the following sections.

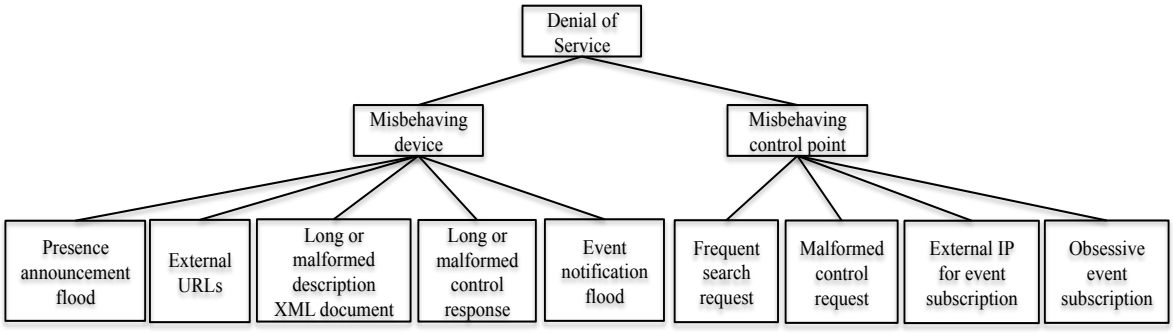

Figure 52 Denial of Service Attack Tree for UPnP

6.4.2.1.1.1. Presence announcement flood

Description: A misbehaving device can announce its presence (through SSDP *alive* and *byebye* messages) with a high frequency and thereby cause denial of service in control points in the local HAN. The UPnP discovery protocol is based on local multicast of presence announcements and search requests. A misbehaving device flooding the multicast address can cause denial of service to all control point applications simultaneously. The specific frequency of announcements that leads to denial of service is undefined and depends on the implementation of the control point. There is no inherent workaround in the UPnP protocol that protects against this type of misbehaving device. The UPnP discovery specification discusses the need to be able to automatically shut off the discovery algorithm however this feature has not been defined in the discovery specification [133].

Effect on *Krox* system: The UPnP LNC listens to device announcements on the local HAN. Therefore, if a misbehaving device exists in the local HAN, the LNC is affected. In the case where the device is explicitly shared with remote HANs, this means that every such announcement would be forwarded to those remote HANs. If the device is not shared, the LNC only updates its local repository with the device update, however, if the device is shared with remote HANs, such a flood of device announcements can lead to denial of service not only of the local *Krox* instance, but also of remote instances of *Krox* system and of control points in remote HANs.

Recommended defence mechanisms: In case the misbehaving device is shared with remote HANs, *Krox* system must be able to limit the attack to the HAN hosting the misbehaving device. This could be achieved using the rate limiting capability of the communication subsystem as described in section 4.3.6.4. The local *Krox* system may still suffer denial of service however it will not be propagated to *Krox* system instances in remote HANs. If the *Krox* system can identify the attack on time, it can disconnect from the IM&P server and by that stop sharing its devices with remote HANs. Similarly, if a *Krox* system identifies that a remote HAN is under attack, it can block it by means of the underlying IM&P system by pause sharing with this “buddy” (i.e. remote HAN), and by that ignore all messages received from that HAN.

6.4.2.1.1.2. External URLs

Description: UPnP presence announcements contain a URL for the location of the device description. The device description contains URLs for interacting with services and potentially for icons. These URLs must be local to the network containing the control point and the device, however many control point application do not validate the location before they try to interact with it. In the case of a malicious device, this could lead to denial of service to multiple control points in the local HAN when trying to communicate the device in the given location. An example of this attack is shown in [37] for several versions of Microsoft Windows. Microsoft released a patch to defend against this attack.

Effect on *Krox* system: Such an attack can only affect the local instance of *Krox*. The reason is that the actual encoded URL of the location is never sent to remote HANs, instead, remote devices are announced in the local HAN with location URLs that are local to the network hosting the *Krox* system. However external location URL can cause the local *Krox* system to suffer denial of service, and therefore needs to be protected against.

Recommended defence mechanism: The LNC can validate the presence announcements and search responses for non-local addresses in the HAN. If they are not local, they can be discarded. All other URLs are expected to be relative to the device IP address and device port, therefore if they are not, the device can be ignored, and a warning can be propagated to the HAN user. This mechanism is included in the design security guidelines for the UPnP service protocol however it was not implemented prototype implementation (section 4.3.3.2.7).

6.4.2.1.1.3. Long or malformed description document

Description: When a control point sends an HTTP GET request for the device/service description, the device responds with a device description document. A misbehaving device can send a sufficiently long description that can cause the control point to run out of memory. Some embedded control points may not be equipped with powerful parsers for XML and can fail due to a misbehaving device sending a malformed device/service description document that contains illegal characters or white spaces.

Effect on *Krox* system: A sufficiently long description can cause the *Krox* system to run out of memory. Even if the *Krox* system does not run out of memory immediately, if the description is sufficiently long, and if it is sent to a requesting remote HAN, it may cause it

to run out of memory. A malformed description is not problematic because *Krox* parses the description document before it is sent to the requesting remote HAN. A malformed description document will fail to parse and therefore will not be sent to the requesting *Krox* system instance in the remote HAN.

Recommended defence mechanism: While the system cannot defend locally against running out of memory when receiving long descriptions, it can avoid sending suspiciously long descriptions to remote HANs and instead report an error. This mechanism is included in the design security guidelines for the UPnP service protocol however it was not implemented prototype implementation (section 4.3.3.2.7).

6.4.2.1.1.4. Long or malformed SOAP response

Description: A misbehaving device can return a sufficiently long SOAP response that can cause a control point to run out of memory. Similarly to the problem with malformed description, a malformed SOAP response can cause control points to crash due to weak XML/SOAP parser implementation.

Effect on *Krox* system: Similarly to the long description problem, even if the *Krox* system does not run out of memory, by sending the long SOAP response to the remote HAN, it can cause the remote *Krox* system instance to run out of memory, or cause the requesting control point in the remote HAN to run out of memory. A malformed SOAP responses are not a problem because the SOAP response is parsed in the local HAN and if it is malformed, an error will be sent to the requesting remote HAN.

Recommended defence mechanism: Similarly to the handling of description requests the LNC can check the size of the SOAP response and if it is suspiciously long, it may discard it and return an error to the requesting remote HAN. This mechanism is included in the design security guidelines for the UPnP service protocol however it was not implemented prototype implementation (section 4.3.3.2.7).

6.4.2.1.1.5. Event notification flood

Description: A misbehaving device can flood subscribed control points with event notifications and cause denial of service. Event notifications are supposed to be sent only when state variables change their value. A misbehaving device can send many event

notifications in a short interval to all subscribed control points and cause them to run out of memory or exhaust their resources.

Effect on *Krox* system: If a control point from a remote HAN is subscribed to event notifications on a misbehaving device in the local HAN, the notifications are tunnelled to the remote control point through the *Krox* system in the local HAN. An event notification flood could lead to *Krox* system to run out of memory, but if it can still send the messages to the remote HAN, it could achieve the effect of denial of service on the subscribed control point.

Recommended defence mechanism: Rate limiting on the communication subsystem would prevent the attack from spreading to the remote HAN subscribed for event notifications. In addition, when such a misbehaving device sending notifications flood is identified, the LNC block event subscription for the misbehaving device. This mechanism was not implemented, however it is included in the design security guidelines (sections 4.3.6.4 and 4.3.3.2.7)

6.4.2.1.2. Misbehaving control point

A misbehaving control point can either be a faulty or malicious application installed by an attacker or by the home user by mistake, such as by installing a Trojan horse. In addition, the misbehaving control point application can join the HAN via a mobile device. There are several ways that a misbehaving control point could lead to denial of service in the local HAN.

6.4.2.1.2.1. Search request flood

Description: A control point can learn about devices in the local HAN by sending periodic search request. A misbehaving control point can flood the HAN with frequent search requests that force all devices that support the service type in the request to respond, which in turn can cause the devices to suffer denial of service.

Effect on *Krox* system: The VRM in the local HAN acts a proxy for search requests made by control points. Therefore when a search request is sent by a control point in the local HAN, the VRM needs to respond on behalf of all devices and services that correspond to the search type in the request. This could lead to the denial of service in the *Krox* system in the local HAN, however it does not affect remote HANs with which the local HAN shares devices and services.

Recommended defence mechanism: The VRM can define the maximum number of search request for a period of time, i.e. if a series of search requests are too frequent, the VRM would ignore the requests for a certain time. The rate limiting definitions can be configured to accommodate legitimate loads. This mechanism is included in the design security guidelines for the UPnP service protocol however it was not implemented prototype implementation (section 4.3.3.2.7).

6.4.2.1.2.2. Malformed or sufficiently long SOAP request

Description: Some UPnP devices are not equipped with powerful SOAP/XML parsing capabilities, which can result in their crash when they receive a malformed SOAP request. When the SOAP request is sufficiently long, it can cause the device to run out of memory.

Effect on *Krox* system: When the VRM receives SOAP request for an action invocation on a remote device, the request is parsed locally before being sent, therefore, if it is not well formed, an error will be sent to the local misbehaving control point, and the request will not be sent to the remote HAN. If the request is very long, it may cause the *Krox* system to run out of memory. If it is very long but did not cause the local *Krox* system to run out of memory, when sent to the remote HAN it can still cause the *Krox* system in the remote HAN or the remote device to run out of memory.

Recommended defence mechanism: In order to defend against excessively long SOAP requests, the same techniques discussed for protecting against long description response should be applied (see section 4.3.3.2.7).

6.4.2.1.2.3. External IP for event subscription

Description: A misbehaving control point can try to exhaust the resources of a device by sending event subscription requests with external IP addresses. In case the device does not check that the IP address is not in the local HAN, it will accept the subscription and will never remove it until it expires. Event subscriptions consume device resources and devices typically limit the number of event subscriptions allowed, therefore a misbehaving control can lead to genuine subscriptions being refused by the device as was demonstrated in [134].

Effect on *Krox* system: Event subscriptions to remote devices are made by the LNC in that HAN, such that the callback interface that is given by the control point to the VRM is not passed to the remote HAN and therefore is ignored in the HAN hosting the live device.

However, in the HAN hosting the misbehaving control point, the VRM forwards notifications received from the live device in the remote HAN, to the callback interface given in the subscription. Therefore if the given callback URL is external, this could lead to denial of service in the local HAN's *Krox* system.

Recommended defence mechanism: When the VRM receives a subscription request with a callback interface that has an external URL it can refuse the subscription and return an error message. This mechanism is included in the design security guidelines for the UPnP service protocol however it was not implemented prototype implementation (section 4.3.3.2.7).

6.4.2.1.2.4. Excessive event subscription

Description: The UPnP protocol does not limit a control point to subscribing for event notifications only once. Therefore a control point can send many subscriptions to the same device. When the maximum number of allowed subscription (device specific) is reached, the device will not allow any more subscriptions to be made, and the existing subscriptions will remain. With enough event subscriptions the device reaches denial of service, such that the device remains alive however UPnP is shut down.

Effect on *Krox* system: More event subscription lead to more inter-HAN traffic, and more memory overhead, both in the LNC and in the VRM. Given enough subscription, both the device for which subscription is made, as well as the corresponding *Krox* systems will reach denial of service.

Recommended defence mechanism: For each device the LNC could keep a maximum of one subscription per UPnP service on behalf of all remote HANs with an infinite expiration time. When additional requests for subscription are received, they will be mapped in memory against the physical subscription that was already made with the device. When a remote HAN cancels a subscription, unless it is the last one, it only updates the memory mapping. In addition, the VRM could subscribe for each service maximum once in the remote LNC and map all additional subscriptions internally to this remote subscription. When the last subscription is removed from the VRM, the remote LNC can be notified. This mechanism is included in the design security guidelines for the UPnP service protocol however it was not implemented prototype implementation (section 4.3.3.2.7).

6.4.2.2. Malicious management actions

There are various goals behind malicious management actions attacks and the damage of such attacks can vary significantly. Malicious management actions are actions made using UPnP to an Internet Gateway Device (IGD). The reason for the emphasis on IGD is because of its sensitive role in the HAN in controlling traffic and in that it has certain vulnerabilities that when leveraged by attackers can have severe impact on the HAN.

The UPnP IGD profile enables control points to change several configurations, some of which could be harmful if made by an attacker. As opposed to DoS attacks that require the attacker to be present in the HAN (e.g. as malware), for attacks against the IGD this is not necessarily required. Such an attack can be performed either by a malicious control point already located in the local HAN, or by a script invoked accidentally by the user through a web browser as was shown in [45]. The latter is enabled because IGD is typically assigned a standard IP address or host name (e.g. 192.168.1.1) that can be guessed by an attacker. Such a script contains a SOAP request that the user unintentionally causes to be sent to a preconfigured IP address that the attacker guesses to belong to the IGD. If successful, the attacker can invoke any action supported by the UPnP IGD specification. A UPnP IGD offers convenient control over the home gateway allowing control points to configure port forwarding, and configure the DNS. The purpose of this interface is to enable applications to configure the router with the configuration they require instead of requiring manual intervention from the user who is typically not qualified in such tasks. For example IGD interface enables applications such as MSN Messenger, XBOX, BitTorrent clients, and others to configure port forwarding instead of asking the user to interact directly with the router. While enabling convenient access to the residential gateway, the IGD suffers from

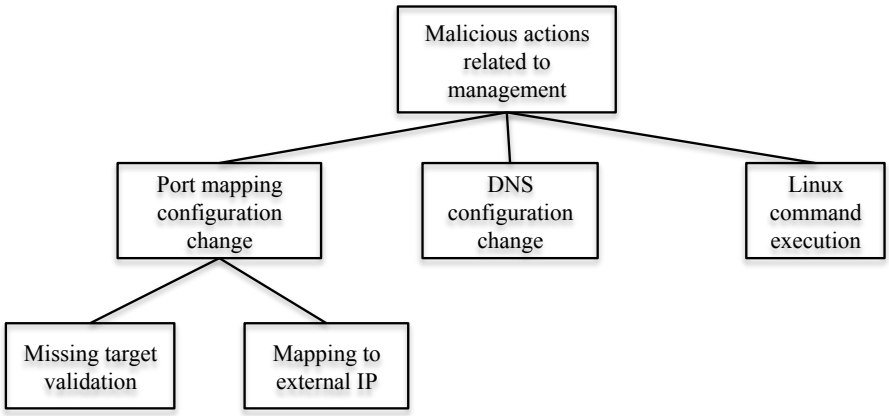

Figure 53 Malicious Management Actions Attack Tree

several flaws that can be leveraged by attackers and present severe security threats. Since the IGD controls some of the most vulnerable features of the home router, an attack on it may have significant consequences. While the *Krox* system implementation does not enforce it, the security guidelines of the UPnP plug-in design (section 4.3.3.2.7) recommend to not share IGD devices because of the potential damage that can be created if it is attacked. Figure 53 presents an attack tree that corresponds to malicious management action attacks. The following sections are organised according to the tree nodes from left to right.

6.4.2.2.1. Port mapping configuration change

Port mapping in the IGD enables redirection of incoming traffic on a specific port of the IGD to another IPAddress/port. Attacks that attempt to modify the port mapping of the IGD send a UPnP SOAP request to the device. The techniques for configuring the port mapping of the IGD rely on weaknesses in the implementation of these devices that have been shown to exist in commercial IGD implementations [134].

6.4.2.2.1.1. Missing validation

Description: There is no validation in the IGD that the device requesting the mapping is the target of the mapping – i.e. that a device is asking to configure forwarding for itself. This means any device/control point can ask to forward any public IP/port to any other private IP/port. A control point installed as part of a virus or Trojan horse could use this weakness to direct traffic to applications that are not expecting it and by that disrupt applications normal behaviour, or to enable external access by attackers to devices or services in the HAN.

Effect on *Krox* system: The effect of changes in the configuration of an IGD device affects the HAN in which the IGD resides. There is no global effect of this on the *Krox* system.

Recommended defence mechanism: Sharing of IGD should not be allowed.

6.4.2.2.1.2. Mapping to an external IP Address

Description: The interface of the port mapping receives the public IP address and port, and the local IP address and port. While the UPnP specification allows only the mapping of the external IP address to an internal one, in some IGD implementations, specifically ones

based on the Broadcom platform as shown in [134], there it is no validation that the internal IP is indeed internal, thereby enabling mapping traffic to an external IP address. This is a major risk as it enables an attacker to forward all or part of incoming traffic to a remote host. This kind of attack can enable the attacker to hijack traffic, such as email or web, and enables phishing and other types of fraud. It also allows the attackers to route their traffic through the victim's network, for example using the victim's network as spam zombie. As mentioned above, this threat is not inherent in UPnP specification of IGD profile but is a bug in some IGD implementations.

Effect on *Krox* system: There is no specific effect of this that is related to the *Krox* system.

Recommended defence mechanism: Sharing of IGD should not be allowed.

6.4.2.2.2. DNS configuration changes

Description: Malicious changes can be made to the Domain Name Server (DNS) through a UPnP service provided by IGD. An attacker can reconfigure the DNS to direct traffic from the home network to a destination controlled by the attacker potentially enabling phishing. The victim then uses the browser to visit web sites, sends or receives emails, or uses instant messaging communication, which are in fact is controlled by the attacker who can redirect the traffic to any site he chooses. This can also enable the attacker to setup man-in-the-middle attack.

Effect on *Krox* system: There is no specific effect of this that is related to the *Krox* system.

Recommended defence mechanism: Sharing of IGD should not be allowed.

6.4.2.2.3. Linux Command Execution

Description: In some Linux based IGD it is possible to pass a Linux command instead of a required parameter. This is possible because of old implementations where the parameters are given to a shell script to execute. Given this implementation, the parameter can be any Linux command, e.g. '/bin/shutdown –r 0' which will cause the IGD to reboot.

Effect on *Krox* system: There is no specific effect of this that is related to the *Krox* system.

Recommended defence mechanism: Sharing of IGD should not be allowed.

6.4.2.3. Eavesdropping

Description: Eavesdropping in the context of UPnP protocol, refers to an application that uses sniffing to listen to communication with devices in the HAN. UPnP protocol does not use encryption for communication with local devices, therefore an eavesdropper in the HAN can "listen" to all communication with devices.

Effect on *Krox* system: All communication between remote HANs is encrypted, however since the UPnP protocol does not support encryption, therefore all communication between the LNC and UPnP devices in the local HAN can be accessible to eavesdroppers.

Recommended defence mechanism: There is no specific defence mechanism against eavesdropping in the local HAN, as it is not supported by the UPnP protocol.

6.4.3. Jini service protocol plug-in

Unlike UPnP, Jini has several embedded security mechanisms for authentication, authorisation, code level protection, privacy and integrity as described in section 4.3.4.1.3. However these mechanisms are not enforced by the Jini service protocol. The reliance of Jini on mobile code makes it especially susceptible to malicious services and malicious service proxies. The following sections describe the potential attacks on Jini services, the effect on *Krox* system and recommended defence mechanisms. Denial of service is discussed in section 6.4.3.1, and eavesdropping is discussed in section 6.4.3.2.

6.4.3.1. Denial of Service

Figure 54 illustrates the relevant DoS attacks as described in the following sections.

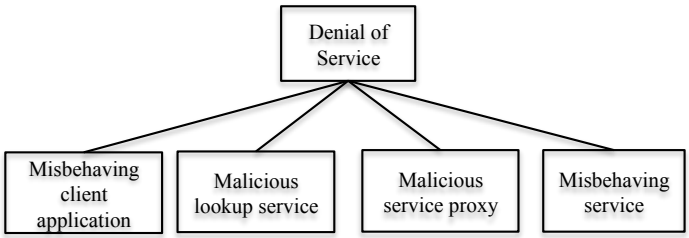

Figure 54 Denial of Service Attack Tree for Jini

6.4.3.1.1. Misbehaving client application

Description: A misbehaving or malicious client can shutdown the lookup service or any other service it can gain access to in the local HAN as a result of a software fault or a malicious attack. Another type of attack could be performed if the service interface method takes parameters that are not final (in the Java modifier sense, i.e. they can be extended), and the malicious client includes a parameter in a service call that extends the parameter type in such way that a call to a method on the parameter would lead to an attack on the service.

Effect on *Krox* system: If the lookup service is shutdown, no Jini services can be announced in the local HAN, and therefore be shared with remote HANs. An attacker can attack services including the lookup service in the local HAN, however it does not have direct access, to remote "live" services, therefore a direct attack against remote Jini service is not possible. For defending against a malicious attack that is using inherited parameters, the service provider must ensure that the service parameter classes are final and cannot be extended.

Recommended defence mechanism: In order to defend against clients that attack services using inherited objects, the *Krox* system can ignore services, which contain in their interfaces parameters that are not defined as final. The LNC will inspect the service interface of a discovered service and will not share it if it has parameters whose classes are not final. This mechanism was not implemented in the prototype system.

6.4.3.1.2. Malicious lookup service

Description: A malicious lookup service can act as a lookup service in the HAN and return malicious service proxies implementing the required interface. In addition, since when services register with the lookup service, they use a proxy to the lookup service, which runs in their Java Virtual Machine, which means if the lookup service is hostile, this could result in a denial of service or other severe consequences for the service/device itself. Similarly if a client registers with the lookup service for notifications on new services, this could result in the same effect on the client machine.

Effect on *Krox* system: The risk of malicious lookup service is twofold: it could host malicious service proxies that implement common Jini services, such that when downloaded from the lookup server, they could harm the Jini LNC. The other risk is that when the VRM

registers a service proxy, the service registrar that it gets (a proxy of the lookup service) is malicious and could harm the VRM, or as it is executing within the *Krox* system it can try to initiate an attack on remote HANs.

Recommended defence mechanism: In order to defend against malicious lookup service, the LNC and VRM should only accept service registrars signed by trusted authorities. This is controlled by the security policy configuration of the *Krox* system.

6.4.3.1.3. Misbehaving service proxy

Description: A misbehaving or malicious service proxy is dangerous because it is Java code that runs on the client process. This means it can damage the process, e.g. by calling *System.exit()*, or harm resources on that machine.

Effect on *Krox* system: Since the LNC invokes methods on the service proxy in the local HAN in response to a request message sent from a remote HAN, a malicious service proxy can harm the *Krox* system in the local HAN, however it cannot harm the remote HAN.

Recommended defence mechanism: In order to defend against malicious service proxy, the LNC should only accept service proxies if they are signed by trusted authorities as configured in the security policy file of the *Krox* system. In addition, the LNC can require the proxy to be verified by a local trust verifier.

6.4.3.1.4. Misbehaving service

Description: A misbehaving service or malicious service can cause denial of service in multiple ways. The first way is by attacking the lookup service, by leasing and cancelling its lease frequently. This could potentially result in a denial of service of the lookup service. In case an attacker did not register a proxy, but instead registered the service itself, the same risk as explained in the section above for misbehaving proxies exists. The service in this case would run as part of the client JVM, and could easily cause denial of service, or even worse – for example, it could erase the hard disk on the machine running the client.

Effect on *Krox* system: A misbehaving service that registers and cancels its registration can result in inter-HAN traffic and therefore affect both the local *Krox* (LNC) and the remote *Krox* system in HANs with which the service is shared, until the lookup service fails. If the service is used as proxy it can harm the local *Krox* system.

Recommended defence mechanism: In order to defend against a message flood caused by discovery announcement from the lookup service, the rate of messages can be limited in the communication subsystem, therefore preventing from the denial of service attack from spreading. The second type of attack by a misbehaving service described above is limited to the local HAN's LNC. In order to defend against such a service, the LNC can require the service to be signed by a trusted authority and request a verifier to approve the service.

6.4.3.2. Eavesdropping

Description: Eavesdropping refers to an unauthorised application listening to the communication between a Jini client and a Jini service. The severity of such an attack depends on what type of information is sent between the client and the server. For example if the Jini service represents a printer and the document is confidential this might be problematic. Jini 2.1 enables client to define constraints on the communication between the client and a service such that it may require communication to be encrypted.

Effect on *Krox* system: The communication between remote HANs is encrypted using the communication subsystem, however if the communication with the proxy locally is not encrypted, an eavesdropper in the HAN hosting the "live" service can get access to confidential content from the remote HAN.

Recommended defence mechanism: The LNC should require the communication with a service proxy to be encrypted. This can be accomplished by defining a confidentiality constraint on method invocations using the Jini security mechanism.

6.5. Conclusions

This chapter presented an evaluation of the *Krox* system architecture and design, which included a performance evaluation and a security analysis. The utility of the design artefact was demonstrated with the implementation of the prototype system; therefore the main purpose of this chapter was to demonstrate the quality and efficacy of the *Krox* system architecture and design through a performance assessment and an analysis of the security threats and mitigation techniques.

The high level goals for the evaluation (defined in section 6.1) were derived from the system requirements. For performance this requires the system to enable home user to share resources with up to 15 remote HANs, such that no more than 300 remote services are shared with the local HAN. For security the main focus of the evaluation was to analyse the potential threats and ensure they are not spread using the system to remote HANs when the local HAN is attacked. The following sections discuss how the evaluation goals were addressed.

6.5.1. Performance

A number of specific evaluation parameters (see 6.2.1.1) were derived from the high-level evaluation goals defined. These parameters define the key performance indicators for the ordinary performance of the *Krox* system that correspond to the requirements for 300 remote services shared with the local HAN, such that sharing can be made with up to 15 remote HANs. Table 8 summarises the results of the experiment described in section 6.2.1.4 with 300 remote services shared with the local HAN and with 4 remote HANs. Column 1 in table 8 presents the evaluation parameter and column 2 and 3 present the experiment average and maximum result respectively. The results demonstrate the linear scalability of the system to adding more devices and services. As argued in section 6.2.3.2, the evaluation was made with increased load on each local system to assess the scalability of the system to more than 4 remote HANs as participated in the evaluation. The Jini plug-in evaluation demonstrated the relatively small overhead of the plug-in implementation for remote service invocation.

Additional performance evaluation assessed the overhead of web service orchestration – specifically the generation and deployment of web services and the overhead introduced by the additional layer of indirection. The results were presented in 6.3.2. The web service generation and deployment takes between 2-5 seconds. This includes the code generation, compilation, packaging and deployment in the servlet container. While this is relatively long process, it should be noted that it is not blocking the system's processing and can be done in the background. The overhead introduced by the web service indirection is shown to be relatively small with less than 200 milliseconds on average per invocation as measured with a Java client. When invoked from an orchestration engine performance could be even faster because of using pure SOAP without requiring to marshal and un-marshal to Java objects.

Parameter	Average	Maximum
CPU utilisation	<4%	N/A
Heap utilisation	<6 megabytes	N/A
Search request processing	<1.5 seconds	<4 seconds
Discovery processing delay	<10 milliseconds	<250 milliseconds
Remote description delay	<10 milliseconds	<450 milliseconds
Remote invocation delay	<15 milliseconds	<175 milliseconds
Event notification delay	<15 milliseconds	<250 milliseconds

Table 8 Performance Evaluation Results Summary

However in a nested composite service when these small delays are aggregated this could lead to a noticeable delay that can affect performance and usability of the services.

It should be noted that the evaluation only provides a lower bound and an assessment of the scalability of the *Krox* system rather than providing absolute performance indicators. A full implementation of the design needs to include additional access control support and an implementation of the recommended security mechanisms. The system was evaluated with access control that returns in constant time, however a full access control implementation may scale linearly with the number of resources and remote HANs. On the other hand in a realistic scenario the HAN user will not share all resources with all of his friends therefore the inter-HAN traffic will be reduced in respect to the performance evaluation. The additional security mechanisms required for defending against potential attacks can also affect performance with additional checks and validations required; therefore these must be implemented with special care with regard to performance.

6.5.2. Security

The security analysis (see section 6.4) used a decomposition of the *Krox* system architecture, and corresponding attack trees to the relevant components in order to identify potential weaknesses of the system and recommend on defence mechanisms. Since the communication subsystem of *Krox* facilitates the transmission of messages between HANs over the Internet, it is more susceptible to attacks. On the other hand, since it is based on an IM&P system, it can reuse existing standard security mechanisms to protect the *Krox* system in the HAN. The *Krox* system does not intend to make the HAN more secure, but only to

defend from attacks on a single HAN from spreading using the *Krox* system to multiple HANs. Therefore, while it may be possible to cause denial of service to the local *Krox* system, the security analysis shows that by following the security design guidelines, such an attack will not spread to other remote HANs.

The security analysis included assessment of UPnP and Jini service protocols for known potential attacks. The potential attacks on the service protocols were identified with their potential effect on *Krox* system and defence mechanisms were recommended. By using the defence mechanisms against these common attacks the *Krox* system can confine the scope of these attacks to the local HANs and prevent it from spreading to remote HANs using the system, as required.

Finally, the security requirements as defined in section 3.4 are addressed with the security considerations described in the design chapter and the defence mechanisms described in section 6.4.

6.5.3. Achieving of evaluation and security requirements

Chapter 1 defined the following challenges for this research:

- Extending HAN service protocols to *multiple HANs* – since HAN service protocols do not extend natively beyond the scope of a single HAN, the challenge was to extend the HAN service protocols beyond the scope of a single household, such that they can seamlessly interact with client applications in those HANs. The *Krox* system architecture, described in chapter 4, and the corresponding prototype implementation that was evaluated in this chapter, provide a plug-in based architecture that enables service protocols to be extended to multiple HANs using the service virtualisation technique. This approach is appropriate when the service protocol has a self-describing service interface that can be used for automatic virtualisation. The feasibility and utility of using the *Krox* system architecture to extend HAN service protocols for multiple HANs was demonstrated using a design and implementation of plug-ins for UPnP and Jini.
- Enabling services from *multiple service protocols* to be *composed together* – service composition can enable reusable units of functionality to be available in the HAN

through the composition of multiple atomic services. For service interoperability, the *Krox* system relies on web services as an interoperable service format. For service composition, the *Krox* system architecture uses service orchestration engine. In order to automatically map HAN services to web services and enable their participation in composite services, the plug-in approach is used, such that a service protocol plug-in is required to support the mapping from a service protocol (e.g. UPnP) to a web service, such that the output of the mapping is a Java code for web service that can be compiled and deployed to the local servlet container. The automatic web service generation approach is powerful in providing interoperability and enabling service composition. Through its integration with the service protocol plug-in, it enables seamless generation of web service proxies for local and remote services that are available in the local HAN. The web service generation has limitations however. It is suitable only for HAN service protocols that support a parsable self-describing service interface.

- *Performance* – as a system that runs in the HAN, the system must not overuse computing resources such as bandwidth, CPU, and memory. In addition, in order to be useful it is required to enable sharing of HAN resources with a representative number of family members and friends with no significant latency. Moreover, the performance show linear scalability well beyond the intended target, which indicates that it will scale gracefully in response to more demanding future requirements. The performance evaluation presented in this chapter demonstrated the linear scalability of the *Krox* system in regard to the number of remote devices that are shared with it with no significant effect on the CPU utilisation (<5% for 300 remote services). The heap memory used by the system for representing 300 remote services is less than 6 Megabytes, however this is added to the 10-15 Megabytes required for the servlet container and orchestration engine. The evaluation was performed on desktop machines that are representative of home PCs which are a potential platform for future deployment of the *Krox* architecture. However the least powerful of these still represented computing power slightly beyond that available at the top end of contemporary home gateway devices, which are another important class of platforms for potential *Krox* deployment. Therefore, it cannot be claimed that the system can be deployed as part of a home gateway at the time of writing, however modern home gateways are increasingly seen as a platform for hosting additional applications, e.g. for value added services by ISPs, therefore device manufacturers are pressured to develop models with more processing power and

RAM. The performance evaluation indicates that the delays added by the system are relatively small, and that the actual delays will be dominated by the network latency. This enables remote services represented by the system in the local HAN to seamlessly interact with control points without incurring significant delays. However, while each delay separately may be negligible, when aggregated in a composite service, especially a nested one, these delays may become more detectable by the user. In such case the benefits of composite service should be balanced against the performance cost.

- *Security* – when the HAN service protocols are extended beyond the scope of a single HAN, it must be made with minimal additional potential vulnerability for the HAN. The security assessment presented in this chapter identified the known threats, their potential effect on the *Krox* system and devised recommended defence mechanisms against them. Such attacks can still harm the local HAN if an attacker succeeds in introducing them to the HAN, but *Krox* system would prevent them from using the connection between the HAN and remote HANs from spreading. These mechanisms must be implemented carefully to minimise the impact on the performance of inter-HAN service interoperability.
- *Simple configuration and administration* – HAN users expect configuration of systems to be minimal and simple. *Krox* system architecture and design supports intuitive configuration through the use of IM&P user metaphor with which HAN users are known to be familiar as demonstrated in chapter 4 and 5 of this thesis. The development and usability assessment of an optimised user interface is beyond the scope of this thesis, and is not claimed as a contribution, beyond the illustrative client presented in section 5.7.

In summary, the *Krox* system architecture and design was shown to address the requirements specified in section 3.4 and the research challenges as shown above beyond the known systems in literature as reviewed in chapter 3. The next chapter will discuss further work and concluding remarks.

Chapter 7
CONCLUSIONS AND FUTURE WORK

This chapter concludes this thesis with a summary of the thesis, an overview of the contributions presented, and a brief description of a number of open research topics not tackled in this thesis.

7.1. Overview of this thesis

A design science research must produce an artefact created to address a problem. This thesis addressed the unsolved problem of integrated inter-HAN and intra-HAN service interoperability. In accordance with the design science research methodology the design artefact resulting from this research is the *Krox* system architecture and its design for supporting sharing services of multiple service protocols with remote HANs, and enabling their interaction and composition with other services in the HAN.

Following the design science methodology, the research objectives design in Chapter 1 were addressed as follows:

1) Review the state of the art in intra-HAN and inter-HAN service interoperability to establish the requirements for an integrated approach – A HAN service protocol review (section 2.3) asserted that the inter-HAN service interoperability and the intra-HAN interoperability were not addressed to a full extent by the service protocols for the HAN. The state of the art (presented in chapter 3) resulted in the identification of a gap in support for integrated intra-HAN and inter-HAN service interoperability. In addition the

state of the art review identified the advantages and disadvantages of existing approaches and derived a set of requirements for an integrated solution (section 3.4).

2) Design an architecture that extends current intra-HAN systems with support for service composition with multiple service protocols, and inter-HAN sharing of service scalable to a number of HANs appropriate for sharing with a household's personal circle of family and friends - This thesis presented the *Krox* system architecture, which supports integrated approach for intra-HAN and inter-HAN service interoperability and addresses the requirements presented in section 3.4. The architecture supports multiple service protocols through its plug-in framework, such that the support for each service protocol is encapsulated in a *Krox* plug-in. The plug-in framework defines a concise and extensible event model for plug-ins, which dictates an interaction model between remote HANs for achieving the desired seamless integration with applications in remote HANs. The *Krox* system architecture supports seamless integration through its use of automatic service virtualisation. A service shared from a remote HAN is represented in the local HAN using a virtual service that implements the service interface on behalf of the remote "live" service and tunnels the communication from the local HAN to the remote HAN hosting the "live" service. The plug-in also supports a mapping from the service protocol to a web service, which makes it available to the service composition subsystem. Both remote and local services are mapped to web services, thereby enabling seamless interoperability between local and remote services through their web service representation. Once mapped to web services, composite services can be created and deployed to a service orchestration engine in the local HAN. The *Krox* system architecture uses IM&P as a user metaphor, such that an IM&P user correspond to the local HAN, and HANs that agree to share devices between them are represented using buddy relation in the IM&P network. In addition IM&P is used for the secure communication subsystem connecting remote HANs. In order to demonstrate the utility of the *Krox* system design, two service protocol plug-ins were implemented for the UPnP and Jini service protocols. These plug-ins demonstrate the feasibility and applicability of the plug-in framework event model. The implementation of two plug-ins for representative service protocols demonstrate the completeness and the extensibility of the plug-in framework event model, and the mapping of these protocols to web services demonstrated their interoperability and composability. While a number of system in the literature address the problems of intra-HAN service interoperability and inter-HAN service interoperability separately, their integration is an unsolved problem. The work in this thesis is therefore novel in presenting and evaluating a comprehensive

integrated architecture that addresses this problem. In the area of inter-HAN service interoperability, the *Krox* architecture and design adopts the service virtualisation approach that is used in a number of inter-HAN service interoperability solutions [31, 51, 139], combining it with a user metaphor that is appropriate for HAN users as applied with SIP in [51, 52]. It takes the novel step of adding to this a standard secure communication mechanism using IM&P and XMPP. This is then extended it with support for an additional protocol and an extensible framework and event model that enables supporting more HAN service protocol in the future. While [10, 51] claim their solutions support similar extensibility goals, no evidence or evaluation of designs for such extensibility are presented in the literature. In the area of intra-HAN service interoperability, the contribution of the *Krox* system architecture is that it facilitates the interoperability and composability of both local and remote services using a common service model. The application of web services as a common service model for HAN service interoperability has been suggested in literature [102, 2], and similarly the composition of services in the HAN has been investigated by a number of authors [16, 17, 52, 126]. However the *Krox* architecture enables, through integration with the inter-HAN components, the seamless interaction and composition of local and remote services, which is not addressed by these existing solutions. Equally however, while some of these authors attempt to define a process for dynamically matching and composing services, this is out of the scope of the *Krox*, architecture, such that it is limited to *enabling* the composition of local and remote service through the representation via a common service model, and their further sharing and reuse via representation of composite services as UPnP devices.

3) Validate the architecture through implementation with two established HAN service protocols - This thesis presents a prototype implementation of *Krox* system architecture and design with plug-ins for UPnP and Jini that shows the *utility* and *feasibility* of the design, and provides an end-to-end system for the purpose of performance evaluation and security analysis.
4) Evaluate the performance and security of the system implementation – The purpose of the performance evaluation and security analysis is to demonstrate the *quality* and *efficacy* of the design artefact, which is *Krox* system architecture and design. The performance evaluation performed as part of this thesis specified a number of evaluation parameters that affect the performance of the integrated intra-HAN and inter-HAN service interoperability. The performance evaluation showed that *Krox* system implementation has linear scalability (sections 6.2-6.3) with the scope defined for

sharing with households in an immediate social circle. The result indicate that on average, the delays introduced by the interaction with remote HANs are dominated by the network latency, however in stress conditions or in the context of nested composite services, the accumulation of the different delays can become apparent. Another aspect of performance that has been explored in the evaluation is the processing power and RAM that are required for the system to execute. As discussed in the previous chapter the evaluation was performed on desktop machines, which are more powerful than home gateways, however with the increased demand for the home gateway as an application platform, its appropriateness as a deployment platform for the *Krox* system is likely to improve. Finally this thesis includes a comprehensive security analysis (section 6.4) that identified the potential threats and attacks that can affect inter-HAN service interoperability, through the communication subsystem, and through attacks made on a specific service protocol. The security analysis is given in the form of attack trees followed by an analysis of the potential risk associated with the attack and with a recommendation for defence mechanism. While it is clear that when supporting inter-HAN service interoperability, new threats are introduced, the purpose of the security analysis is to identify this threats and present mechanisms to defend against them from spreading beyond the scope of a single HAN.

The next section describes the main contributions of this thesis and how they were achieved.

7.2. Contributions of this thesis

This section briefly summarises the contributions of this thesis, as presented in the previous chapters.

This thesis provides an in depth study of solutions for integrated sharing resources (devices, services, and content) between multiple home area networks leading to an analysis of the existing IP based application level service protocols and their appropriateness for service sharing. The various service protocols are analysed and compared to identify the type of service protocols that can support seamless inter-HAN service interoperability. This thesis discussed the requirements from a service for sharing devices, services, and content from the HAN with remote HANs. This resulted in a comprehensive specification of these

requirements that extends the state of the art, in that it integrates intra-HAN and inter-HAN service interoperability requirements.

In accordance with the design science research methodology, the *major contribution* of this thesis is the design of *Krox* system architecture that addresses the unsolved question of integrated intra-HAN and inter-HAN service interoperability and addresses the set of requirements identified in the state of the art study. The approach taken by *Krox* system architecture is suitable for service protocols with a parsable service interface and support for communication mechanism for service invocation. This service interface can lend itself to virtualisation in remote HANs and enable seamless integration with remote HANs. In addition, such a parable service interface enables the service protocol plug-in to map the service interface to web service thereby enabling its intra-HAN interoperability. The plug-in event model defines how service virtualisation can be achieved for multiple service protocols, as demonstrated with the plug-in implementations for Jini and UPnP. A key aspect of the *Krox* system architecture is its use of standard based IM&P communication, to provides a simple and commonly accepted user metaphor for managing sharing and a scalable and secure communication channel. This architecture includes well-defined interfaces, through which extensions for additional protocols can be implemented.

While vendors and manufacturers have adopted SOC as a useful abstraction for supporting device interoperability, service composition was not fully addressed by these implementations. Service composition is an important part of SOC that can enable the realisation of more of the potential of the HAN and can enable 3rd party service providers to offer innovative services based on existing HAN devices. With the growing interest and demand for supporting service composition, and with the availability of more services in the HAN, through sharing from remote HANs, the importance of service composition increases. More services available in the local HAN lead to more potential for innovative composite services that can reuse and leverage the value of these atomic services. *Krox* system architecture supports service interoperability and service composability using a common service interface and a mapping between the service protocol and the common service interface. The plug-in based approach taken by *Krox* system architecture enables the support of mapping from the service protocol to web services, and by that increasing the service interoperability in the HAN, and enabling services to be composed using standard web service orchestration techniques. While service composition for HAN has been suggested in literature, the contribution of this thesis is in its integrated approach that enables to

seamlessly compose local and remote services, and even share the composite services.

A *minor contribution* of this thesis is the comprehensive performance evaluation of the system design that identified the key performance indicators that affect sharing of services from the HAN with remote HANs. While performance is an important requirement for systems for inter-HAN service interoperability, only little information is given in the literature regarding the performance aspects of the relevant systems. Wegner [139] described a benchmark to evaluate the delay introduced by invocation of remote SOAP action, however this evaluation was made in isolation with a single device and on a powerful desktop machine (Intel Core Duo @ 2GHz) and did not include full UPnP implementation. Other systems reviewed in the literature do not present relevant benchmark information, especially in the context of their scalability to multiple HANs and the relevant key performance indicators. The evaluation of *Krox* system provided a benchmark baseline both for intra-HAN and inter-HAN service interoperability that can be compared against by future work in this area.

Finally, a *minor contribution* of this thesis is a comprehensive analysis of the security aspects of sharing devices, services, and content from the HAN with remote HANs. This analysis in the form of attack trees identified potential attacks on HAN service protocols, their impact on the service protocols, on the system for sharing, potential impact on remote HANs with which it is shared, and potential defence mechanisms. While security is an important requirement from systems for sharing devices and services from the HAN, most attention has been given to standard security mechanisms such as authentication, access control, and encryption. Seikkinen et al. [119] Chowdhury et al. [31] focus on authentication, encryption, and access control. Such mechanisms are also discussed in [51, 55, 76, 77, 84, 97, 121, 137, 139]. However the problem of defending against attacks using the vulnerabilities of relevant protocols for spreading beyond the scope of a single HAN is not addressed in the literature. The security analysis given in this thesis gives a reusable baseline in the form of attack trees that can be extended and referenced by future works in inter-HAN service sharing.

7.3. Further work

This section described a number of related research topics and a number of open research

topics that have not been fully researched as part of this thesis.

7.3.1. Additional *Krox* plug-ins

This thesis demonstrated the feasibility of the *Krox* plug-in framework and extensible event model using two service protocols. Additional service protocols could be supported, such as DPWS, OSGi, and HAVi. Moreover, while *Krox* system architecture is restricted to those service protocols with a parsable service interface, it leaves out a number of important protocols for the HAN such as ZeroConf and Bluetooth. A further direction could be to explore how much additional information, in the form of additional annotations to service descriptions, is needed to enable full support for these service protocols with *Krox* system architecture for both intra-HAN and inter-HAN service interoperability.

7.3.2. Capability Sharing Management

Krox system architecture as described in chapter 4 relies on a fine-grained definition of permission per resources. Though the specification of the sharing policies is out of the scope of this thesis, the utility and efficacy of potential solutions should be explored. For example, the Federal Relation Manager [23] was originally suggested for management of capability sharing between large telecoms providers with heterogeneous technical platforms. The FRM represents managed services and resources using a hierarchical capability authority model, which can be dynamically modified in order to create new aggregations of the basic capabilities made available by the underlying devices. By reusing this approach for HAN resources, users can assign different remote HANs different sets of capabilities that correspond to policies applied on the local HAN resources. The integration between *Krox* system and the FRM is on-going research involving the author, which will be completed after the submission of this thesis.

In addition, while the configuration of the sharing policies is out of the scope for this thesis, the integration with FRM will require user studies to determine if the level of sharing control suits the level of technical management skill of the HAN user.

Finally, the integration with the FRM would enable the evaluation of the impact of capability sharing management on the performance of *Krox* system. While the performance

evaluation presented in chapter 6 assumed all devices are shared in order to increase the load on the *Krox* system, the integration with FRM can enable to evaluate the relation between the number of shared and non shared devices and the performance of *Krox* system, with different types of sharing policies of variable complexity.

7.3.3. Lightweight service composition for HAN

One drawback of using XML/SOAP web services and BPEL for service orchestration is their performance. RESTful web services present a more lightweight and scalable approach for web services. However current BPEL specification does not allow composition of RESTFul web services, which is more typically implemented in mashups. A number of recent publications suggest supporting RESTful web services with BPEL service orchestration, such as BPEL for REST [99], and Apache ODE extension for RESTful services [6]. A potential direction could be representation of the composite service with a subset of BPEL constructs and compile the service into a more compact implementation. Bohn et al. [17] suggested promising approach using BPEL-to-Java compiler that could be extended to support RESTful services and result in a more compact representation of BPEL services that is application server free and requires less resources for execution.

7.3.4. Service composition tools for HAN

While the *Krox* system architecture provides the mechanisms needed for interoperability and service composition, it does not define a process for how services are composed. A complementary mechanism is required to exist in the HAN for identifying resources that are needed for specific composite services and just-in-time instrumentation of a composite service that reuses this resource. The *Krox* system architecture provides capabilities that can be used as part of such as system with its service protocol plug-in's support for discovery and mapping to web services. Composite services could be defined using a template, and additional description of the type of constituent services that are required for the instrumentation of the composite service's template. When all of the requirements for a composite service are met, it can be suggested to the user and automatically instrumented and deployed in the local HAN. The significance of service composition will increase in the following years with more devices and services available in the local HAN, therefore tools that allow service providers to offer innovative services to home users will take on increasing importance.

7.3.5. Rate limiting for communication subsystem

As discussed in the security analysis, rate limiting is an important mechanism for increasing the robustness of the *Krox* system architecture to a various types of attacks. There are a number of directions for supporting rate limiting. One direction is to support rate limiting as an XMPP plug-in such that it will allow only a configured amount of messages per time interval to be sent and received between a pair of users or from a single user. Another option is to support this at the communication subsystem level, such that rate limiting will be applied within the *Krox* system instance. The drawback of this approach is that it may have implications on performance.

References

[1] ABI Research. Home networking end-user snapshot: Consumer adoption of home and media networking, June 2008. [online]. Available: http://www.abiresearch.com/research/1007478-Home+Networking+End-User+Snapshot. 26 April 2011 [date accessed].

[2] Aiello, M., "The Role of Web Services at Home," *Telecommunications, 2006. AICT-ICIW '06. International Conference on Internet and Web Applications and Services/Advanced International Conference on*, pp. 164-164, 19-25 Feb. 2006

[3] Alexander, J., D. Box, L.F. Cabrera, D. Chappell, G. Daniels, R. Janecek, C. Kaler, B. Lovering, R. McCollum, D. Orchard, S. Parastatidis, J. Schlimmer, I. Sedukhin, and J. Shewchuk, "Web Service Transfer (WS-Transfer)," September, 2006. [online]. Available: http://specs.xmlsoap.org/ws/2004/09/transfer/WS-Transfer.pdf. 26 April 2011 [date accessed].

[4] Allard, J., V. Chinta, S. Gundala, and G. G Richard III,. 2003. "Jini Meets UPnP: An Architecture for Jini/UPnP Interoperability," In *Proceedings of the 2003 Symposium on Applications and the Internet* (January 27 - 31, 2003). SAINT. IEEE Computer Society, Washington, DC, 268.

[5] Andrews, T., F. Curbera, H. Dholakia, Y. Goland, J. Klein, F. Leymann, K. Liu, D. Roller, D. Smith, S. Thatte, I. Trickovic, and S. Weerawarana, S. Business Process Execution Language for Web Services, Version 1.1. OASIS, 2003.

[6] Apache ODE Project. (2010) [online]. RESTful BPEL part II (http://ode.apache.org/restful-bpel-part-ii.html) 25 April 2011 [date accessed].

[7] Arnold, K., R. W. Scheifler, J. Waldo, A. Wollrath, B. O'Sullivan, *The Jini Specification*, Addison-Wesley Longman Publishing Co., Inc., Boston, MA, 1999.

[8] Beatty, J., G. Kakivaya, D. Kemp, T. Kuehnel, B. Lovering, B. Roe, C. St. John, J. Schlimmer, G. Simonnet, D. Walter, J. Weast, Y. Yarmosh, and P. Yendluri, "Web

Services Dynamic Discovery (WS-Discovery)," April 2005. [Online]. Available: http://schemas.xmlsoap.org/ws/2005/04/discovery/. 26 April 2011 [date accessed].

[9] Belimpasakis P., A. Saaranen, and R. Walsh, "Home DNS: Experiences with Seamless Remote Access to Home Services," *World of Wireless, Mobile and Multimedia Networks, 2007. WoWMoM 2007. IEEE International Symposium on a*, 2007, pp. 1-8.

[10] Belimpasakis, P. and V. Stirbu, "Remote Access to Universal Plug and Play (UPnP) Devices Utilizing the Atom Publishing Protocol," In *proceedings of the Third International Conference Networking and Services* (ICNS 2007) pp. 59-59, Athens, Greece, June 2007.

[11] Belimpasakis, P. and A. Saaranen, "Seamless User-Oriented Content Sharing,". Tampere *University of Technology. Department of Communications Engineering*. Research Report 2009:1, 23 p., Tampered 2009.

[12] Belimpasakis, P., S. Moloney, V. Stirbu, and J. Costa-Requena, J., "Home media atomizer: remote sharing of home content - without semi-trusted proxies," *Consumer Electronics, IEEE Transactions on*, vol.54, no.3, pp.1114-1122, August 2008.

[13] Belimpasakis, P., "Seamless User-Generated Content Sharing in the Extended Home," Doctoral Thesis, Tampere University of Technology, June 2009.

[14] Berners-Lee, T., J. Hendler and O. Lassila (May 17, 2001). "The Semantic Web," *Scientific American Magazine*, May 2001, pp. 28-37.

[15] Bluetooth, "Specification of the Bluetooth system: core specification v2.0 + EDR," Nov. 2004. [online]. Available: https://www.bluetooth.org/Technical/Specifications/adopted.htm. 26 April 2011 [date accessed].

[16] Bobek, A., H. Bohn, F. Golatowski, G. Kachel, and A. Spreen. "Enabling Workflow in UPnP Networks, " in *INDIN'05*, Perth, Australia, 2005.

[17] Bohn, H., F. Golatowski, and D. Timmermann, "Dynamic device and service discovery extensions for WS-BPEL," *Service Systems and Service Management, 2008 International Conference on*,pp.1-6, June 30 2008-July 2 2008.

[18] Bottaro, A., A. Gérodolle, and P. Lalanda, "Pervasive Service Composition in the Home Network," *AINA '07: Proc. 21st Int. Conf. Advanced Networking and Applications*, Washington, DC, USA, May 21–23, pp. 596–603. IEEE Computer Society.

[19] Bottaro, A. and A. Gérodolle, 2008. "Home SOA : facing protocol heterogeneity in

pervasive applications," *In Proceedings of the 5th international Conference on Pervasive Services* (Sorrento, Italy, July 06 - 10, 2008). ICPS '08. ACM, New York, NY, 73-80.

[20] Box, D., D. Ehnebuske, G. Kakivaya, A. Layman, A. Mendelsohn, H.F. Nielsen, S. Thatte, and D. WinerSimple, "Object Access Protocol (SOAP), " Version 1.1. W3C Note, W3C, May 2000. [online]. Available: http://www.w3.org/TR/soap/ 25 April 2011 [date accessed].

[21] Box, D., L.F. Cabrera, C. Critchley, F. Curbera, D. Ferguson, S. Graham, D. Hull, G. Kakivaya, A. Lewis, B. Lovering, P. Niblett, D. Orchard, S. Samdarshi, J. Schlimmer, I. Sedkhin, J. Shewchuk, S. Weerawarana, and D. Wortendike, "Web Services Eventing (WS-Eventing)," March 2006. [online]. Available: www.w3.org/Submission/WS-Eventing/. 26 April 2011 [date accessed].

[22] Bray, T., J. Paoli, C.M. Sperberg McQueen, Eve Maler, F. Yergeau, and J. Cowan, "Extensible Markup Language (XML) 1.1," W3C Recommendation, August 2006. [online]. Available: http://www.w3.org/TR/xml11/. 26 April 2011 [date accessed].

[23] Brennan, R., K. Feeney, J. Keeney, D. O'Sullivan, J.J. Fleck, S. Foley, and S. v.der Meer. 2010. "Multi-Domain IT Architectures for Next Generation Communications Providers," IEEE Communications Mag. vol. 48, no. 6, pp 110- 117, Aug 2010.

[24] Brennan, R., D. Lewis, J. Keeney, Z. Etzioni, K. Feeney, D. O'Sullivan, J.A. Lozano, B. Jennings, "Policy-based Integration of Multi-Provider Digital Home Services", *IEEE Network Magazine, special issue on Digital Home Services*, 23, (6), 2009, p50 – 55.

[25] Brown N., and C. Kindel, "Distributed Component Object Model Protocol – DCOM/1.0," Internet Draft, November 1996. [online]. Available: http://tools.ietf.org/html/draft-brown-dcom-v1-spec-00. 26 April 2011 [date accessed].

[26] Bull, P.M., P.R. Benyon, and P.R. Limb, "Residential Gateways," *BT Tech. J.*, Apr. *2002*. pp. *73-81*.

[27] Campbell, C.S., E. Kandogan, A. November, R. Barrett, and P.P. Maglio. "Policity: an experimental evaluation of policy-based administration in a city simulation," IEEE Workshop on Policies for Distributed Systems (Policy 2005), Stockholm, Sweden, 6-8 June 2005.

[28] Chan, S., D. Conti, C. Kaler, T. Kuehnel, A. Regnier, B. Roe, D. Sather, J. Schlimmer, H. Sekine, J. Thelin, D. Walter, J. Weast, D. Whitehead, D. Wright, and Y. Yarmosh, "Devices Profile for Web Services," Feb. 2006. [Online]. Available:

http://schemas.xmlsoap.org/ws/2006/02/devprof/ 25 April 2011 [date accessed].

[29] Cheshire, S., and D. H. Steinberg, *Zero Configuration Networking, the Definitive Guide* O'Reilly, 2005.

[30] Cheshire, S., B. Aboba, and E. Guttman, "Dynamic Configuration of IPv4 Link-Local Addresses," Internet Engineering Task Force, RFC 3927, May 2005. [online]. http://tools.ietf.org/html/rfc3927. 26 April 2011 [date accessed].

[31] Chowdhury, R., A. Arjona, J. Lindqvist, and A. Yla-Jaaski, "Interconnecting multiple home networks services," *Telecommunications, 2008. ICT 2008. International Conference on*, pp.1-7, 16-19 June 2008.

[32] Christensen, E., F. Curbera, G. Meredith, and S. Weerawarana, "Web Services Description Language (WSDL) 1.1", World Wide Web Consortium, Note NOTE-wsdl-20010315, March 2001. [online]. Available: http://www.w3.org/TR/wsdl. 26 April 2011 [date accessed].

[33] Czarnecki, K. and U.W. Eisenecker, "Aspect Oriented Programming", in *Generative programming: methods, tools, and applications*. Addison-Wesley. 2000.

[34] Digital Living Network Alliance, "DLNA Interoperability Guidelines v1.0, " 2004.

[35] DNS-SD.org. [online]. DNS-SRV (RFC 2782) Service Types: http://www.dns-sd.org/ServiceTypes.html. 26 April 2011 [date accessed].

[36] Droms, R., "Dynamic Host Configuration Protocol," Internet Engineering Task Force, RFC2663, March 1997. [online]. Available: http://www.ietf.org/rfc/rfc2131.txt. 26 April 2011 [date accessed].

[37] EEye Digital Security website. [online]. UPNP - Multiple Remote Windows XP/ME/98 Vulnerabilities http://research.eeye.com/html/advisories/published/AD20011220.html 26 April 2011 [date accessed].

[38] Egevang k. and P. Francis, "The IP Network Address Translator," Internet Engineering Task Force, RFC 1631, May 1994. www.ietf.org/rfc/rfc1631.txt 25 April 2011 [date accessed].

[39] Fielding, R., "Architectural Styles and The Design of Network-based Software Architectures," Ph.D. thesis, University of California, Irvine, 2000.

[40] Foley, S. N., W. M. Fitzgerald, "An Approach to Security Policy Configuration using Semantic Threat Graphs", 23rd Annual IFIP WG 11.3 Working Conference on Data and Applications Security (DBSec), Concordia University, Montreal, Canada, July 12-15, 2009.

[41] Freed, N. "Behaviour Of and Requirements for Internet Firewall," Internet

Engineering Task Force, RFC 2979, Oct. 2000. Available: www.ietf.org/rfc/rfc2979.txt. 25 April 2011 [date accessed].

[42] Gamma, E., R. Helm, R. Johnson, and J. Vlissides, *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison Wesley. 1994.

[43] Gamma, E., and K. Beck, *Contributing to Eclipse*, Addison-Wesley, 2003.

[44] Gaw, J., "worldwide home networking 2007-2011 forecast, " IDC, October 2007.

[45] GNU Citizen website. [online]. http://www.gnucitizen.org/blog/hacking-the-interwebs/ 26 April 2011 [date accessed].

[46] Gulbrandsen, A., P. Vixie, and L. Esibov, "A DNS RR for specifying the location of services (DNS SRV)," RFC 2782, Feb. 2000. [online]. Available at http://www.rfc-editor.org/rfc/rfc2782.txt. 25 April 2011 [date accessed].

[47] Guttman, E., C. Perkins, J. Veizades, and M. Day, "Service Location Protocol, Version 2," RFC 2608, June 1999. [online]. Available: http://www.ietf.org/rfc/rfc2608.txt. 25 April 2011 [date accessed].

[48] Guttman, E. and J. Kempf, "Automatic Discovery of Thin Servers: SLP, Jini and the SLP-Jini Bridge, " *Proc. 25th Ann. Conf. IEEE Industrial Electronics Soc.* (IECON 99), IEEE Press, Piscataway, N.J., 1999.

[49] H.A. Simon, *The sciences of the artificial*, 2nd ed. Cambridge: MIT Press, 1981.

[50] Häber A., M. Gerdes, F. Reichert, A. Fasbender, and R. Kumar, "Using SIP Presence for Remote Service Awareness," *Norsk Informatikkonferanse 2008 (NIK'08)*, Kristiansand, Norway, 2008.

[51] Häber, A., J.G.R. De Mier, and F. Reichert, "Virtualization of Remote Devices and Services in Residential Networks," In *Next Generation Mobile Applications, Services and Technologies, 2009. NGMAST '09. Third International Conference on* (2009), pp.182-186, 15-18 Sept. 2009

[52] Hackmann, G., M. Haitjema, C. Gill, and G-C. Roman, G.-C., "Sliver: A BPEL Workflow Process Execution Engine for Mobile Devices," *Proc. Service-Oriented Computing –ICSOC 2006*, Springer, Berlin / Heidelberg, 2006, pp. 503-508.

[53] Handley, M., and E. Rescorla, "Internet Denial-of-Service Considerations," Internet Engineering Task Force, RFC 4732, November 2006. [online]. Available: http://tools.ietf.org/html/rfc4732. 25 April 2011 [date accessed].

[54] Harrenstien, K., "NAME/FINGER protocol," Internet Engineering Task Force, RFC 742, December 1977. [online]. Available: http://tools.ietf.org/html/rfc742. 26 April 2011 [date accessed].

[55] Haruyama, T., S. Mizuno, M. Kawashima, and O. Mizuno, "Dial-to-Connect VPN

System for Remote DLNA Communication," *Consumer Communications and Networking Conference, 2008. CCNC 2008. 5th IEEE*, pp.1224-1225, 10-12 Jan. 2008.

[56] Henry, D., [online]. The Future of your wireless home network. http://mashable.com/2011/02/23/future-home-networking-tips/. 26 April 2011. [date accessed].

[57] Hevner, A., S. March, J. Park, and S. Ram, "Design Science in Information Systems Research," *MIS Quarterly* (28:1) 2004, pp. 75-105.

[58] HomePNA, "Interface Specification for HomePNA 2.0: 10M8 Technology, " Englewood Cliffs, NJ, Dec. 1999.

[59] IANA.org. [online]. Port number assignment http://www.iana.org/assignments/port-numbers. Last updated: April, 2011. 26 April 2011 [date accessed].

[60] IBM. [online]. Java Theory and Practice: Decorating with Dynamic Proxies. http://www.ibm.com/developerworks/java/library/j-jtp08305.html 26 April 2011. [date accessed].

[61] Ibrahim, A., and M. ÓFoghlú, "New Role of Policy-based Management in Home Area Network – Concepts, Constraints and Challenges," In *proceeding of 3rd International Conference on New Technologies, Mobility and Security (NTMS)*, December 2009, Cairo, Egypt.

[62] IEEE, "IEEE Standard for Information technology-Telecommunications and information exchange between systems-Local and metropolitan area networks-Specific requirements - Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications," *IEEE Std 802.11 – 2007 (Revision of IEEE Std 802.11-1999)*, pp. C1-1184, 12 2007.

[63] IEEE Std. 1394-1995, "IEEE Standard for a High Performance Serial Bus," IEEE Press, Piscataway, N.J. Aug. 1996.

[64] IEEE P802.15. [online]. Working Group for WPAN Task Group 1 (http://ieee802.org/15/pub/TG1.html) 25 April 2011 [date accessed].

[65] Intel. [online]. DeviceRelay, http://software.intel.com/en-us/articles/intel-software-for-upnp-technology-technology-overview. 25 April 2011 [date accessed].

[66] ITU-T FG IPTV, Working Document: IPTV Architecture, FG IPTV-DOC-0084, 2007.

[67] Järvinen, P. (2000) "Research Questions Guiding Selection of an Appropriate Research Method," *Proceedings of the 8th European Conference on Information Systems (ECIS 2000)*, Vienna, Austria July 3-5.

[68] Java.net. [online]. JXTA Community Projects, "JXTA protocol specifications," https://jxta-spec.dev.java.net. 25 April 2011 [date accessed].

[69] Jordan, D. and J. Evdemon, "Web Services Business Process Execution Language v2.0," OASIS, 2007. [online]. http://docs.oasis-open.org/wsbpel/2.0/OS/wsbpel-v2.0-OS.html. 26 April 2011 [date accessed].

[70] Kawsar, F., T. Nakajima, and K. Fujinami, "A document centric approach for supporting incremental deployment of pervasive applications". *In Proceedings of the 5th Annual international Conference on Mobile and Ubiquitous Systems: Computing, Networking, and Services*, (MOBIQUITOUS2008) Dublin, Ireland, July 21-25, 2008.

[71] Kent, S. and K. Seo, "Security Architecture for the Internet Protocol, " Internet Engineering Task Force, RFC 4301, Dec. 2005. [online]. Available: http://www.ietf.org/rfc/rfc4301.txt 25 April 2011 [date accessed].

[72] Kosiur, D., *Building and Managing Virtual Private Networks,* John Wiley and Sons, Inc., USA, 1998.

[73] Kreitz G. and F. Niemela. "Spotify - large scale, low latency,p2p music-on-demand streaming," In *Peer-to-Peer Computing*, pp. 1-10. IEEE, 2010.

[74] Laszlo, J., *Home Networking: Seizing Near-Term Opportunities to Extend Connectivity to Every Room*. Jupiter Research (BRB02-V01) (2002).

[75] Lea, R., S. Gibbs, A. Dara-Abrams, and E. Eytchison, "Networking home entertainment devices with HAVi," Computer , vol.33, no.9, pp. 35-43, Sep 2000.

[76] Lee, H.Y., and J.W. Kim, "An Approach for Content Sharing among UPnP Devices in Different Home Networks," *Consumer Electronics, IEEE Transactions on*, vol.53, no.4, pp.1419-1426, Nov. 2007.

[77] Loeser, C., W. Mueller, F. Berger, and H.J. Eikerling, "Peer-to-Peer Networks for Virtual Home Environments," In *Proceedings of the 36th Hawaii International Conference on System Sciences* (HICSS'03), 2003.

[78] Mahy, R., P. Matthews, and J. Rosenberg "Traversal using relays around NAT (TURN): Relay extensions to session traversal utilities for NAT (STUN), " Internet Engineering Task Force, RFC 5766, April 2010. [online]. Available: http://tools.ietf.org/html/rfc5766. 26 April 2011 [date accessed].

[79] McIlraith**,** S.A., T.C. Son, and H. Zeng (March 2001). "Semantic Web Services," *IEEE Intelligent Systems* 16(2) 46-53.

[80] Melnikov, A., and K. Zeilenga, "Simple Authentication and Security Layer (SASL), " Internet Engineering Task Force, RFC 4422. [online]. Available:

http://tools.ietf.org/html/rfc4422. 26 April 2011 [date accessed].

[81] Metcalfe, R.M., D.R. Boggs, "Ethernet: distributed packet switching for local computer networks, " *Communications of the ACM*, v.19 n.7, p.395-404, July 1976.

[82] Miller, B., T. Nixon, Ch. Tai, and M. Wood, "Home networking with universal plug and play, " IEEE Communications Magazine, (Dec. 2001) 104-109.

[83] Moon, K-D., Y-H. Lee, Y-S. Son, and K. Chae-Kyu, "Universal home network middleware guaranteeing seamless interoperability among the heterogeneous home network middleware," *Consumer Electronics, IEEE Transactions on*, vol.49, no.3, pp. 546- 553, Aug. 2003.

[84] Motegi, S., K. Tasaka, A. Idoue, and H. Horiuchi, "Proposal on Wide Area DLNA Communication System," *Consumer Communications and Networking Conference, 2008. CCNC 2008. 5th IEEE*, pp.233-237, 10-12 Jan. 2008.

[85] Motorola Inc., "LonWorks Technology Device Data Book, revision 2," 1996.

[86] Moyer, S., D. Marples, and S. Tsang, "A protocol for wide-area secure networked appliance communication" *IEEE Communications Magazine*, vol. 39, no. 10, pp. 52-59, Oct. 2001.

[87] Moyer, S., and A. Umar, "The impact of network convergence on telecommunications software", *IEEE Communications Magazine*, vol. 39, no. 1, pp. 78-84, Jan. 2001.

[88] Nakazawa, J., H. Tokuda, W. K. Edwards, and U. Ramachandran, "A Bridging Framework for Universal Interoperability in Pervasive Systems, ". In *Proceedings of the 26th IEEE International Conference on Distributed Computing Systems*, p.3, July 04-07, 2006.

[89] Negus, K.J., A.P. Stephens and J. Lansford, "HomeRF: Wireless Networking for the Connected Home," *IEEE Personal Communications* **7** 1 (2000), pp. 20–27.

[90] Nottingham, M., and R. Sayre, "The Atom Syndication Format, ", RFC 4287, IETF, December 2005. [online]. Available: http://www.ietf.org/rfc/rfc4287.txt. 25 April 2011 [date accessed].

[91] Newmarch, J., "UPnP Services and Jini Clients," *Information Systems: New Generations (ISNG 2005)*, Las Vegas 2005.

[92] Object Management Group, *The Common Object Request Broker: Architecture and Specification,* Revision 2.0, 1995.

[93] Oh, H., J. Lim, K. Chae, and J. Nah, "Home gateway with automated real-time intrusion detection for secure home networks," in *Computational science and Its Applications – ICCSA 2006*. Springer Berlin / Heidelberg, 2006, pp. 440-447.

[94] Oh, Y-J., H. Lee, J. Kim, E. Paik, and K. Park, “Design of an extended architecture for sharing DLNA compliant home media from outside the Home,” *IEEE Transactions on Consumer Electronics*, vol. 53, no. 2, May 2007.

[95] Papazoglou, M.P., "Service-Oriented Computing: Concepts, Characteristics and Directions," in Fourth International Conference on Web information Systems Engineering (WISE), pp. 3, 2003.

[96] Papazoglou, M.P., P. Traverso, S. Dustdar, and F. Leymann, "Service-Oriented Computing: State of the Art and Research Challenges," Computer, vol. 40, no. 11, pp. 38-45, November, 2007.

[97] Park, H., E.H. Paik, and N. Kim, "Architecture of Collaboration Platform for Ubiquitous Home Devices," *Mobile Ubiquitous Computing, Systems, Services and Technologies, International Conference on*, pp. 301-304, 2009 Third International Conference on Mobile Ubiquitous Computing, Systems, Services and Technologies, 2009.

[98] Parrend, P. and S. Frenot. "A security analysis for home gateway architectures," In *International Conference on Cryptography, Coding & Information Security*, Venice, Italy, November 2006.

[99] Pautasso, C., "BPEL for REST," in *Proc. of BPM’2008, LNCS 5240*, Milan, Italy, 2008, pp. 278–293.

[100] Pavlidou, N., A. J. H.Vinck, J. Yazdani, and B. Honary, "Power line communications: State of the art and future trends, " *IEEE Commun. Mag.*, vol. 41, pp. 34–40, Apr. 2003.

[101] Peltz, C., "Web services orchestration and choreography," *Computer*, vol.36, no.10, pp. 46- 52, Oct. 2003.

[102] Perumal, T., A.R. Ramli, C.Y. Leong, S. Mansor, and K. Samsudin, "Interoperability among Heterogeneous Systems in Smart Home Environment," *Signal Image Technology and Internet Based Systems, 2008. SITIS '08. IEEE International Conference on*, pp.177-186, Nov. 30 2008-Dec. 3 2008.

[103] Plummer. D.C., "An Ethernet Address Resolution Protocol," Internet Engineering Task Force, RFC 826, November 1982. [online]. Available: http://tools.ietf.org/html/rfc826. 26 April 2011 [date accessed].

[104] Rahman, M., C. Akinlar, and I. Kamel, "On Secured End-to-End Appliance Control Using SIP," In *Proc. of the 5-th IEEE International Workshop on Networked Appliances*, pp.24-28. Oct. 2002.

[105] Ramish, A., R. Stewart, and M. Dalal, "Improving TCP's Robustness to Blind In-

Window Attacks", Internet Engineering Task Force, RFC 5961, August 2010, Work in progress. [online]. Available: http://tools.ietf.org/html/rfc5961 25 April 2011 [date accessed].

[106] Redondo Diaz, R.P., A.F. Vilas, M.R. Cabrer, J.J. Pazos Arias, and M. Rey Lopez, "Enhancing Residential Gateways: OSGi Service Composition," *Consumer Electronics, IEEE Transactions on*, vol.53, no.1, pp.87-95, February 2007.

[107] Rellermeyer, J.S. and G. Alonso, "Services everywhere: OSGi in distributed environments," in *EclipseCon 2007*, 2007.

[108] Rellermeyer, J.S., G. Alonso, and T. Roscoe, "R-OSGi: Distributed Applications through Software Modularization," In *Proceedings of the ACM/IFIP/USENIX 8th International Middleware Conference (Middleware 2007)*, Newport Beach, CA, 2007.

[109] Rescorla, E., *SSL and TLS: Designing and Building Secure Systems*. Addison-Wesley, 2000.

[110] Rose, B., "Home networks: a standards perspective," *Communications Magazine, IEEE*, vol.39, no.12, pp.78-85, Dec 2001.

[111] Rosenberg, J. "Interactive connectivity establishment (ICE): A protocol for network address translator (NAT) traversal for offer/answer protocols, " Internet Engineering Task Force, RFC 5766, April 2010. [online]. Available: http://tools.ietf.org/html/rfc5245. 26 April 2011 [date accessed].

[112] Rosenberg, J., H. Schulzrinne, G. Camarillo, A. Johnston, J. Peterson, R. Sparks, M. Handley, and E. Schooler, "SIP: Session Initiation Protocol," RFC 3261, Internet Engineering Task Force (IETF), Jun. 2002. [online]. Available: http://www.ietf.org/rfc/rfc3621.txt. 25 April 2011 [date accessed].

[113] Rosenblum, D., "What Anyone Can Know: The Privacy Risks of Social Networking Sites," *Security & Privacy Magazine, IEEE*, Volume: 5, Issue: 3, June 2007.

[114] Saint-Andre, P., "Extensible messaging and presence protocol (XMPP): Core," Internet Engineering Task Force RFC 3920, October 2004. [Online]. Available: http://www.ietf.org/rfc/rfc3920.txt. 26 April 2011 [date accessed].

[115] Schneier, B., "Attack trees: Modeling security threats," Dr. Dobb's journal, December 1999.

[116] Schulzrinne H., S. Casner, R. Frederick, and V. Jacobson, "RTP: A Transport Protocol for Real-Time Applications," Internet Engineering Task Force, RFC 3550, July 2003 [online]. Available: www.ietf.org/rfc/rfc3550.txt. 25 April 2011 [date accessed].

[117] Schulzrinne, H., A. Rao, and R Lanphier, "Real Time Streaming Protocol (RTSP)," Internet Engineering Task Force, RFC 2326, Feb 1998. Available: www.ietf.org/rfc/rfc2326.txt. 25 April 2011 [date accessed].

[118] Schulzrinne, H., and J. Rosenberg, “The Session Initiation Protocol: Internet centric signalling,” *IEEE Communications Magazine*, vol 38, Oct. 2000.

[119] Siekkinen, M., J. Manner, S. Tarkoma, and A. Ylä-Jääski, 2009. "hBox: connecting homes". In *Proceedings of the 3rd international Conference on New Technologies, Mobility and Security* (Cairo, Egypt, December 20 - 23, 2009). K. AI-Agba, M. Badra, and G. B. Newby, Eds. IEEE Press, Piscataway, NJ, 218-222.

[120] Singh, M.P., and M. Huhns, (2005). *Service-Oriented Computing: Semantics, Processes, Agents*. Hoboken, NJ: John Wiley & Sons.

[121] Song, T., Y. Kawahara, and T. Asami, "DAS: An intuitive DLNA content sharing system using SNS access control," *Broadband Network & Multimedia Technology, 2009. IC-BNMT '09. 2nd IEEE International Conference on*, pp.570-574, 18-20 Oct. 2009.

[122] Srisuresh, P., and M. Holdrege, "IP Network Address Translator (NAT) Terminology and Considerations," Internet Engineering Task Force, RFC2663, August 1999. [online]. Available: http://www.ietf.org/rfc/rfc2663.txt. 25 April 2011 [date accessed].

[123] Sun Microsystems. [online]. Java Remote Method Invocation specification. java.sun.com/j2se/1.5/pdf/rmi-spec-1.5.0.pdf. 26 April 2011 [date accessed].

[124] The OSGi Alliance, "OSGi Service Platform, Core Specification r4," Aug. 2005; [online]. Available: http://www.osgi.org 25 April 2011 [date accessed].

[125] The X10 Specification, X-10 (USA) Inc. 91 Ruckman Road Box 420, Closter, NJ 07624 [online]. Available: http://www.smarthome.com/manuals/MAN-1136.pdf. 26 April 2011 [date accessed].

[126] Timm, C., J. Schmutzler, P. Marwedel, P., and C. Wietfeld, 2009. "Dynamic web service orchestration applied to the device profile for web services in hierarchical networks," In *Proceedings of the Fourth international ICST Conference on Communication System Software and middleware* (Dublin, Ireland, June 16 - 19, 2009). COMSWARE '09. ACM, New York, NY, 1-6.

[127] TiVo. [online]. Website: http://www.tivo.com/. 25 April 2011 [date accessed].

[128] Townsley, W., A. Valencia, A. Rubens, G. Pall, G. Zorn, and B. Palter, "Layer Two Tunneling Protocol “L2TP”," Internet Engineering Task Force, RFC 2661, August 1999. [online]. Available: http://www.ietf.org/rfc/rfc2661.txt. 25 April 2011 [date

accessed].

[129] Tsang, S., D. Marples, and S. Moyer, "Accessing networked appliances using the session initiation protocol", *IEEE International Conference on Communications*, no. 1, pp. 1280-1285, Jun. 2001.

[130] UPnP Forum. [online]. UPnP website (http://www.upnp.org/). 26 April 2011 [date accessed].

[131] UPnP Forum, "DeviceSecurity:1 Service Template Version 1.0, " November 2003. [online]. Available: http://upnp.org/specs/sec/UPnP-sec-DeviceSecurity-v1-Service.pdf. 26 April 2011 [date accessed].

[132] UPnP Forum, "SecurityConsole:1 Service Template Version 1.0, " November 2003. [online]. Available: http://upnp.org/specs/sec/UPnP-sec-SecurityConsole-v1-Service.pdf. 26 April 2011 [date accessed].

[133] UPnP Forum, "UPnP Device Architecture 1.0," October 2008. [online]. Available: http://upnp.org/sdcps-and-certification/standards/device-architecture-documents/. 26 April 2011 [date accessed].

[134] UPnP Hacks website. [online]. http://www.upnp-hacks.org/ 26 April 2011 [date accessed].

[135] USB 2.0 Specification, [online]. Available: http://www.usb.org/developers/docs/ 26 April 2011 [date accessed].

[136] Valtchev, D., I. Frankov, "Service gateway architecture for a smart home," *Communications Magazine, IEEE*, vol.40, no.4, pp.126-132, Apr 2002.

[137] Venkitaraman, N., "Wide-Area Media Sharing with UPnP/DLNA," *Consumer Communications and Networking Conference*, 2008. CCNC 2008. 5th IEEE, pp.294-298, 10-12 Jan. 2008.

[138] Weerawarana S., F. Curbera, F. Leymann, T. Storey, and D.F. Ferguson. W*eb Services Platform Architecture*, Prentice Hall, 2005.

[139] Wegner, T., "A Modular UPnP Proxy for Secure Remote Access," *Digital Society, 2010. ICDS '10. Fourth International Conference on*, pp.72-77, 10-16 Feb. 2010

[140] Weiser, M., "The Computer for the 21st Century," Sci. Amer., Sept., 1991, pp. 94–104.

[141] Wolfinger, R., S. Reiter, D. Dhungana, P. Grunbacher, and H. Prahofer, "Supporting Runtime System Adaptation through Product Line Engineering and Plug-in Techniques," *Composition-Based Software Systems, 2008. ICCBSS 2008. Seventh International Conference on*, pp.21-30, 25-29 Feb. 2008.

[142] XMPP standard foundation. [online]. XEP-0205: Best Practices to Discourage

Denial of Service Attacks http://xmpp.org/extensions/xep-0205.html. 26 April 2011 [date accessed].

[143] Zahariadis, T. and K. Pramataris, "Multimedia home networks: standards and interfaces," Comput. Stand. Interfaces 24, 5 (Nov. 2002), 425-435.

[144] Zang, N., M. Beth Rosson and V. Nasser, "Mashups: who? what? why?, " in *Conference on Human Factors in Computing Systems 2008*, Florence, Italy: ACM, 2008, pp. 3171-3176.

[145] Zheng, O., J. Poon, and K. Beznosov, "Application-based TCP hijacking,". In *Proceedings of the Second European Workshop on System Security* (Nuremburg, Germany. March 31 - 31, 2009).

[146] Zhu, F., M.W. Mutka, and L.M. Ni, L.M., "Service discovery in pervasive computing environments," Pervasive Computing, IEEE , vol.4, no.4, pp. 81- 90, Oct.-Dec. 2005.

[147] Zigbee Alliance, "Zigbee specification: Zigbee document 053474r13 Version 1.1," 1 Dec. 2006. Zigbee website: www.zigbee.org. 25 April 2011 [date accessed].