Efficient Service Discovery in Wide Area Networks

A Thesis

submitted to the

University of Stirling

for the Degree of

Doctor of Philosophy

by

Alan Brown

Department of Computing Science and Mathematics

University of Stirling

Stirling, FK9 4LA

Scotland

2008

Abstract

Living in an increasingly networked world, with an abundant number of services available to consumers, the consumer electronics market is enjoying a boom. The average consumer in the developed world may own several networked devices such as games consoles, mobile phones, PDAs, laptops and desktops, wireless picture frames and printers to name but a few. With this growing number of networked devices comes a growing demand for services, defined here as functions requested by a client and provided by a networked node. For example, a client may wish to download and share music or pictures, find and use printer services, or lookup information (e.g. train times, cinema bookings).

It is notable that a significant proportion of networked devices are now mobile. Mobile devices introduce a new dynamic to the service discovery problem, such as lower battery and processing power and more expensive bandwidth. Device owners expect to access services not only in their immediate proximity, but further afield (e.g. in their homes and offices). Solving these problems is the focus of this research.

This Thesis offers two alternative approaches to service discovery in Wide Area Networks (WANs). Firstly, a unique combination of the Session Initiation Protocol (SIP) and the OSGi middleware technology is presented to provide both mobility and service discovery capability in WANs. Through experimentation, this technique is shown to be successful where the number of operating domains is small, but it does not scale well.

To address the issue of scalability, this Thesis proposes the use of Peer-to-Peer (P2P) service overlays as a medium for service discovery in WANs. To confirm that P2P overlays can in fact support service discovery, a technique to utilise the Distributed Hash Table (DHT) functionality of distributed systems is used to store and retrieve service advertisements. Through simulation, this is shown to be both a scalable and a flexible service discovery technique. However, the problems associated with P2P networks with respect to efficiency are well documented.

In a novel approach to reduce messaging costs in P2P networks, multi-destination multicast is used. Two well known P2P overlays are extended using the Explicit Multi-Unicast (XCAST) protocol. The resulting analysis of this extension provides a strong argument for multiple P2P maintenance algorithms co-existing in a single P2P overlay to provide adaptable performance. A novel multi-tier P2P overlay system is presented, which is tailored for service rich mobile devices and which provides an efficient platform for service discovery.

Acknowledgements

This Thesis is the culmination of four years of long days and even longer nights. However, without the help of a number of important people this work would not have been possible. Firstly, and most importantly, my supervisor Doctor Mario Kolberg. Mario not only helped with many of the technical aspects of my PhD, but provided a wealth of knowledge and advice which supported me through the "difficult" days. I enjoyed our discussions, especially over a beer, and appreciate the valuable time he invested into my work.

I would also like to thank Doctor John Buford, who was my industrial sponsor throughout the course of my research. My research relationship with John and Panasonic was both enjoyable and fruitful. John's expertise in the field of Peer-to-Peer helped me gain knowledge and experience in what was an entirely new area of research for me. I also have John to thank for my introduction to the worlds most frustrating game - golf.

A word of thanks is also due to my second supervisor, Professor Evan Magill, and also to Professor Ken Turner. Both Evan and Ken have helped focus my

PhD, by reading my annual reports and attending my mini-vivas. Their advice and contributions are appreciated and certainly helped me ‘take a step back’.

Outwith the academic staff at the University, I would also like to extend a thank you to my fellow PhD students, who formed a supportive network of friends and colleagues. I’m sure Lloyd enjoyed my ramblings on P2P network simulations as much as I enjoyed his ramblings on particle swarm optimisation. But having someone to listen to you really makes a difference.

Finally, I would like to give a very special thank you to my wonderful wife Vikkie. My promises of ‘it will be worth it in the end’ and ‘it’s not forever’ will have seemed like little comfort on those endless nights in front of my computer. However, you supported me 100%, proof reading my work and motivating me when I was down.

.

. . . for my wife

Vikkie

Contents

List of Figures

List of Tables

Abbreviations

3G Third Generation network. High bandwidth radio network.

ALM Application Layer Multicast. Multicasting independent of the IP layer. An example is overlay multicasting.

CPU Central Processing Unit. The part of a computer where programs and calculations are executed.

DHCP Dynamic Host Configuration Protocol. Automates the assignment of IP addresses to networked machines.

DHT Distributed Hash Table. A class of decentralized distributed systems.

DNS Domain Name System. Associates hostnames and other information with IP addresses.

DoS Denial of Service. An attack designed to overload a networked node with requests to impede its ability to provide its designated service.

DVMRP Distance Vector Multicast Routing Protocol. A multicast routing protocol based on reverse path forwarding.

EDRA Event Detection and Reporting Algorithm. An algorithm for detecting and reporting node join and leave events in a P2P system.

EDRA* An extension to the originally specified EDRA algorithm designed to improve stability.

GPRS General Packet Radio Service. Cellular radio network.

HTML Hypertext markup language. A language used to give structure to a web document by applying markup in the form of tags to text.

HTTP Hypertext Transfer Protocol. A request/response protocol used in the World Wide Web to transmit/receive web pages.

IEEE Institute of Electrical and Electronic Engineers.

IETF Internet Engineering Task Force.

JXTA A decentralised P2P protocol designed by Microsoft.

IP Internet Protocol. A packet switched protocol used over the Internet.

LAN Local Area Network. A network confined to a local area, e.g. an office or home.

LDAP Lightweight Directory Access Protocol. An application protocol for querying and modifying directory services running over TCP/IP.

MANET Mobile Ad-hoc Network. A self-configuring network of mobile routers (and associated hosts) connected by wireless links.

MD5 A cryptographic hash function which returns a 128-bit hash value.

MSD Meta Service Discovery. A method to discover services using metadata.

NA Networked Appliance. A (household) appliance with a network interface.

OASIS Organization for the Advancement of Structured Information Standards.

OSGi An open standards organization who specified a Java-based service platform that can be remotely managed.

OSPF Open Shortest Path First. A routing protocol used in the Internet by routers to decide the next hop of a packet traversing a network.

PDF Probability Density Function. A mathematical function which represents the probability distribution in terms of integrals, where the sum of the integrals is 1.

PIM Protocol Independent Multicast. A collection of independent multicast protocols (sparse, dense and bi-directional).

P2P Peer-to-Peer. An ad-hoc network of peers.

PAN Personal Area Network. A local network of devices connected close to one person.

RFC Request for Comments. A (proposed) IETF standard.

RTT Round Trip Time. In two-way communication, the total time taken for a message to reach its destination, and the response message to reach the originator.

SDM Service Discovery Mechanism. A mechanism for discovering networked services.

SHA-1 Secure Hashing Algorithm. The first of five cryptographic one-way hashing algorithms designed by the National Security Agency (NSA). Produces a 160-bit message digest.

SIP Session Initiation Protocol. An IETF standard for voice over IP.

SLP Service Location Protocol. A service discovery protocol which allows devices to discover services without prior configuration.

SOAP Simple Object Access Protocol. A protocol for exchanging XML data using HTTP.

SSDP Simple Service Discovery Protocol. A protocol used by UPnP devices to discover services.

TCP Transmission Control Protocol. A core protocol used in the Internet to provide reliable delivery of data between two nodes.

UDDI Universal Description Discovery and Integration. An XML based registry for businesses to list themselves on the Internet.

UPnP Universal Plug and Play. A technical standard for the interworking of networked appliances.

UPnP A/V Universal Plug and Play Audio/Video. An extension to the UPnP technical standard to define Audio/Video usage.

URI Uniform Resource Identifier. A set of characters used to identify a location on the Internet.

URL Uniform Resource Location. Similar to URI. A set of characters used to identify a location on the Internet.

WAN Wide Area Network. A network spread over a large area, typically over multiple cities or even countries/continents.

WSDL Web Services Description Language. An XML-based language that provides a model for describing Web service interfaces.

XCAST Explicit Multi-Unicast. A multi-destination routing protocol used to send multicast messages to a list of destinations.

XML Extensible Markup Language. Markup language which allows users to define their own tags to describe data in a structured format.

WWW Word Wide Web. A system of interlinked, hypertext documents accessed via the Internet

X.10 A powerline protocol for networking IPAs.

List of Publications

The work reported in this Thesis has produced the following publications to date:

1. A. Brown, M. Kolberg, J.Buford, *Chameleon: An adaptable 2-tier variable hop overlay*, 6th IEEE Consumer Communications and Networking Conference (CCNC) - P2P and Content Delivery, Las Vegas, USA, 2009.

2. J. Buford, A. Brown and M. Kolberg, *Exploiting Parallelism in the Design of Peer-to-Peer Overlays*, Computer Communications Journal, Elsevier Science, Volume 31, Issue 3, pages 452-463, February 2008.

3. A. Brown, M. Kolberg and J. Buford, *An adaptable service overlay for wide area network service discovery*, IEEE Globecom 2007, Workshop Enabling the Future Service-Oriented Internet, Washington DC, USA, December 2007.

4. J. Buford, A. Brown and M. Kolberg, *Analysis of an Active Maintenance Algorithm for an O(1)-Hop Overlay*, IEEE Globecom 2007, Washington DC, USA, December 2007.

5. M. Kolberg, F. Kolberg, A. Brown and J. Buford, *A Markov Model for the EpiChord Peer-to-Peer Overlay in an XCAST enabled Network*, A Markov Model for the EpiChord Peer-to-Peer Overlay in an XCAST enabled Network, IEEE International Conference on Communications (ICC), Glasgow, Scotland, June 2007.

6. A. Brown, J. Buford and M. Kolberg, *Tork: A Variable-Hop Overlay for Heterogeneous Networks. 5th IEEE International Conference on Pervasive Computing and Communications (PerCom)*, 4th IEEE International Workshop on Mobile Peer-to-Peer Computing (MP2P'07), White Plains, NY, USA, 2007.

7. J. Buford, A. Brown and M. Kolberg, *Parallelizing Peer-to-Peer Overlays with Multi-Destination Routing*, 4th IEEE Consumer Communications and Networking Conference (CCNC), Las Vegas, USA, 2007.

8. J. Buford, A. Brown and M. Kolberg, *Multi-Destination Routing and the Design of Peer-to-Peer Overlays*, 4th IEEE Consumer Communications and Networking Conference (CCNC), 1st International Workshop on Peer-to-Peer Multicasting (P2PM'07), Las Vegas, USA, 2007.

9. J. Buford, A. Brown and M. Kolberg, *Meta Service Discovery*, 4th IEEE International Conference on Pervasive Computing and Communications (Per-

Com), 3rd IEEE International Workshop on Mobile Peer-to-Peer Computing (MP2P'06), Pisa, Italy, 2006, pp. 124-129.

10. A. Brown, M. Kolberg, D. Bushmitch, G. Lomako and M. Ma, *SIP-based OSGi Device Communication Service for Mobile Personal Area Networks*, 3rd IEEE Consumer Communications and Networking Conference (CCNC), Las Vegas, USA, 2006.

11. A. Brown, M. Kolberg, D. Bushmitch, G. Lomako and M. Ma, *Offering service mobility to home appliances using a Personal Area Network*, PG Net, Liverpool, UK, 2005.

Chapter 1

Introduction

1.1 Overview

In the developed world, devices with multiple modes of network connectivity are proliferating in the consumer market. From the mobile telephone or MP3 player in your pocket to the PDA or laptop in your bag, the choice of networked device has never been so broad. With this increasing number of highly capable devices comes increasing user expectations. Devices must be able to connect to the Internet or some other network and share pictures or music, look up information on the World Wide Web (WWW), discover services offered by other devices, and more. In this context, a service is defined as a function, process or procedure requested by a client and accessible over a network.

How to discover such services and meet the user's growing expectations is a complex problem with a diverse set of requirements, dependent on the category of service discovery system in use. It is these numerous service discovery systems that are the focus of this document. Due to the large number of available service discovery techniques, incompatibility among them is common.

For example, a device fitted with a Bluetooth network adapter and Universal Plug and Play (UPnP) protocol stack may wish to discover a resource to download music. There are no guarantees that, if such a resource exists in the local domain, it will be compatible with the searching device. Alternatively, the device may wish to search a foreign domain for such a resource. How to search multiple administrative domains for services not only compatible but suitable is a complex problem and is addressed in this document.

It is also notable that the devices mentioned in this Chapter are predominantly mobile. As well as being increasingly connected, devices are also becoming smaller and more portable. With mobility comes the need for efficiency.

Mobile devices typically:

- Have less battery power. To preserve battery power, applications are required to become more lightweight and to consume less resources.
- Have less available bandwidth. For example, mobile devices frequently connect to the WWW via GPRS or 3G, and bandwidth is charged per kilobyte. Thus, reducing bandwidth consumption is very important.

- Have less processing power. Smaller, mobile devices have less processing power and memory than larger desktop machines, and again require applications to have a smaller memory footprint and to use less resources.

Taking these requirements into consideration, the aims of this Thesis are described next.

1.2 Aims of this work

The aims of this work are as follows:

1. Design an approach to service discovery which tackles the increasing number of incompatible service discovery techniques.
2. This approach must be resource efficient and suitable for mobile devices.
3. This approach must be capable of operating in a Wide Area Network.

1.3 Contributions

In order to understand the inherent problems associated with service discovery, this Thesis examines the requirements of a good service discovery technique. In addition to this, a survey of existing service discovery solutions is undertaken to identify which of the identified problems have yet to be addressed.

Two approaches are presented to tackle the problem of multiple incompatible service discovery techniques. The first approach utilises a services middleware to provide an adapter capable of translating between incompatible discovery protocols. This approach is defined in Section 3.3 and is shown not only to operate in Local Area Networks (LANs) but to also enable the bridging of LANs and Wide Area Networks (WANs) to offer mobility to normally static devices. This addresses two of the key goals of this Thesis, to tackle multiple incompatible service discovery techniques and to provide an approach suitable to WANs.

The second approach allows devices to discover services by context without relying on a particular service discovery technique. For example, a user may wish to discover a service by location, or by service type or provider. This is achieved using P2P overlays, and differs from previous work in that nodes are only required to support basic Distributed Hash Table (DHT) functionality. Members of a P2P network are likely to meet this criteria.

Because devices are typically becoming more mobile, resource efficiency is an important factor to enable service discovery for mobile devices. This Thesis offers two P2P centric approaches. Firstly, multi-destination routing is applied to two P2P routing algorithms and the design criteria are described. This approach replaces inherently parallel operations with multi-destination multicast to offer significant bandwidth savings. The second approach combines two types of P2P routing table maintenance algorithms in a single P2P system, to give variable performance to peers depending on their bandwidth capability.

1.4 Terminology

In the context of service orientation, a *service* is a set of networked functions. Alternatively, the Organisation for the Advancement of Structured Information Standards (OASIS) defines a service as

> "a mechanism to enable access to one or more capabilities, where the access is provided using a prescribed interface and is exercised consistent with constraints and policies as specified by the service description" [14].

For the purpose of this Thesis, we shall use the latter.

Services are discovered by *devices* and other *services*, and are invoked by *users*. textitService discovery is the process of locating a service of interest. A *device* is any peripheral or part of a computer system that can send or receive data. A *user* is any person who requires a computer for the performance of a task or recreational activity, often referred to as an end-user.

Peer-to-peer networks consist of *nodes*, referred to as *peers* interchangeably in this Thesis. Nodes in a P2P system are referred to as peers because they typically have equal standing within the network. *Nodes* are "devices on a network that demand or supply services or where transmission paths are connected" [15].

1.5 Structure of the Thesis

This document firstly presents a literature review in the area of service discovery. Using a well defined set of requirements for service discovery techniques, a set of diverse problems is described. Designed to tackle this set of problems, a centralised approach to service discovery is presented in Chapter 3 using the Session Initiation Protocol (SIP). This solution, named the SIP Service, is shown to facilitate importing devices and services from one domain to another. Additionally, the SIP Service demonstrates protocol bridging between SIP and the UPnP protocol. After a considered analysis of this approach, the SIP Service is found to scale inefficiently. Consequently, in Chapter 4, an alternative and more suitable solution to the service discovery problem is identified using P2P service overlays.

To address the problem of multiple incompatible service discovery mechanisms existing in a single network, this Thesis next presents Meta Service Discovery (MSD). MSD provides the facility to insert service descriptions containing contextual information about service discovery mechanisms into a P2P overlay network. This is successfully implemented in three Distributed Hash Table (DHT)-based P2P systems.

Having established that P2P overlays are capable of supporting WAN service discovery, Chapter 5 presents a novel method to improve the bandwidth efficiency of service overlays. Using multi-destination multicast, a reduction in excess of 30% bandwidth usage is shown to be achievable in two 1-hop P2P overlays: EpiChord and the Event Detection and Reporting Algorithm (EDRA). EpiChord and EDRA utilise two distinct approaches to routing table maintenance, and these are compared in Chapter 6.

Comparison of active and opportunistic maintenance techniques, used by EpiChord and EDRA respectively, shows complementary performance. EpiChord nodes perform using less bandwidth than EDRA nodes, but with poorer hop counts. Using these results as motivation, this Thesis presents a variable-hop P2P system in Chapter 7, which combines both algorithms used interchangeably by participating nodes. Finally, Chapter 8 concludes this Thesis.

1.6 Summary

This Chapter introduced the concept of the modern networked device which is feature rich, highly connected and mobile. Some of the problems encountered by these modern devices, such as expensive bandwidth, incompatibility of services and inferior processing power have been described. Solutions to these problems define the goals of this Thesis.

Chapter 2

Service Discovery: Context and Issues

In this chapter, a literature review in the area of Service Discovery is presented.

2.1 Introduction

In the developed world, devices with multiple modes of network connectivity are proliferating in the consumer market. Using the TCP/IP protocol stack, these devices can connect not only to the Internet, but to other logical overlay networks such as the Word Wide Web (WWW), content sharing P2P systems, and the Domain Name Resolution System (DNS) amongst many others. In addition to being connected, devices are increasingly mobile: cellular phones, PDAs and laptops introduce a large range of new and exciting services to end users. How to discover such services is a complex problem with a diverse set of requirements, depending on the category of service discovery system in use. These requirements are described in this Chapter, followed by a critique of several service discovery techniques.

2.1.1 Centralised or Decentralised?

Service discovery can be categorised into centralised or decentralised systems. Centralised systems typically comprise of a central server which collects service advertisements and provides a facility for service users to search for the advertised services. Examples of such systems are Google [16] and Yahoo [17] which create a set of *web crawlers* or *bots* to traverse web pages, gathering information to store in their central search repositories. For use in the Word Wide Web (WWW), centralised techniques have proved popular and successful, mainly due to the relatively static population of nodes which makeup the WWW and the context-rich searching facility that can be provided. It is also true that content stored in the WWW (e.g. Web Pages) is refreshed fairly irregularly in comparison to more volatile networks such as P2P content sharing systems. It is due to this volatility that systems such as P2P content sharing often choose a decentralised method of service discovery.

The largest strength of centralised service discovery is possibly its biggest weakness. As already mentioned, a context-rich searching facility is available when service advertisements are stored in a central server. However, this central server introduces a single point of failure in the system - a target for malicious nodes. Should the central server be removed from the system, no services can be discovered/used. In addition to this, as a system grows in size this central server may become a bottleneck, and searching efficiency is vastly reduced. Thus, there is strong motivation for larger systems to become more decentralised.

To address scalability, a hybrid approach to service discovery exists where super nodes/peers assume a larger responsibility for storage or processing than less powerful peers. For example, Google and Yahoo have created large server farms, where the servers interconnect to form a large searching and indexing facility. The P2P content sharing system Kazaa [18] employs a similar approach using super peers.

Unlike centralised and to a lesser extent hybrid systems, decentralised systems do not rely on any form of centralised server as a method of storing service advertisements or routing service requests, thus removing the single point of failure. The indexing method of service advertisements is distributed, often equally, amongst the participating nodes of the system. Decentralised systems can be further sub-categorised as *distributed* or *replicated*. The most common case is distributed, where directory information is partitioned amongst dedicated entities (e.g. SLP [19], Jini [20]) or cached locally in the system by service providers [21]. Replicated systems use directory entities to store entire directories of information [22].

2.1.2 Requirements

To achieve effective service discovery, a number of requirements are imposed on a system. These include: scalability, fault tolerance, efficiency, a comprehensive searching facility, security and load distribution (for decentralised systems). This section describes each of these in more detail.

Scalability: The performance of any service discovery task (advertise, discover, use) or general system maintenance should not degrade as the overall system size increases. As mentioned previously, this is a common problem with centralised servers dedicated to storage of service advertisements. Decentralised systems however tend to cope better with increasing network size because the load on each node (storage of advertisements, number of requests received, etc.) is often distributed equally amongst all nodes.

Fault Tolerance: In large scale autonomous systems, the population of nodes is often transient and volatile. Nodes join and leave the system without administration, and commonly nodes do not notify the system of their departure. Therefore, systems which can handle high levels of churn (joins, leaves) are desirable. For centralised systems, faults can often be catastrophic where a central server is disconnected without warning. Decentralised systems often cope better in this situation, as content is often distributed evenly amongst nodes; thus minimal data is lost. However, persistence of volatile data is a difficult requirement for decentralised systems and often replication of data is required, where multiple nodes store the same piece of data. If one node dies, one of the other nodes can offer its data on its behalf. In addition to preserving the persistence of data, nodes in a centralised system must maintain routing behaviour in the presence of high churn. This is commonly achieved using structured routing algorithms (described later).

Comprehensive Searching: Regardless of the size of or the level of churn in the system (nodes joining and leaving), it should be able to return a full list of results for a given search for a service. The system should also guarantee that if a service exists, it should be able to find it. Various decentralised systems do not offer such a guarantee, e.g. unstructured P2P overlays, although typically structured systems do. Centralised systems do offer a thorough searching facility and provide the facility to perform comprehensive keyword searches.

Efficiency: In large scale systems, the load on each individual node should not be excessive. Load is defined as resource usage, which amongst others could include bandwidth, CPU and storage. Depending on the type of system, load can be induced by service advertisements (inserts), service discovery (lookups), maintenance (structured systems) and storage of data.

Systems which operate within the bandwidth and storage capabilities of the participating nodes are therefore desirable.

Security: Any system which stores valuable information must have a high level of security. Centralised systems are normally centrally administered and participants are forced to register to use the system. This provides a level of control over participating users. In addition, administrators may apply privileges to users restricting their access to certain data/services. Decentralised systems are more difficult to manage, and security is a major research topic. Reputation schemes and implied trust mechanisms are used frequently, though there is no defacto standard for securing decentralised systems.

Load Distribution: The distribution of services to nodes in centralised systems is not normally an issue, though it is significantly imbalanced. Nodes which are rich in resources are chosen to store service information so that excess stress on weaker nodes is avoided. For systems such as Google, high powered back-end servers are used for such a purpose. In decentralised systems, nodes share equal responsibility for indexed services. This is generally achieved using a hashing function of the indexed file name, which generates a random key. This is common in key-based architectures, e.g. structured P2P systems. In hybrid systems, super peers are elected to assume greater load than weaker peers.

Finding service discovery techniques which meet all of the above requirements is most unlikely. Much research has gone into finding such a system, and each of these individual problems has been addressed by researchers worldwide. Until such a system is designed, application programmers and developers will design systems which are suited to the goal of the system. For example, if the system requires a very strong keyword searching technique where security is vital, they would opt for a centralised system. Meantime, improving the methods of addressing each of the described requirements is ongoing work.

The remainder of this section describes several existing service discovery techniques divided into two groups: techniques designed for Wide Area Networks and those designed for Local Area Networks.

2.2 Wide Area Network Service Discovery

A Wide Area Network (WAN) is classed as a computer network which covers a broad geographical area. WANs are often used to connect multiple Local Area Networks (LANs) to provide a much larger network. The largest known WAN is the Internet. Therefore, in this section we describe service discovery approaches which can be achieved in Internet scale networks.

2.2.1 Web Services

A Web Service is a software system connected over a network to provide interoperable machine to machine interaction. Web services have interfaces which are described in a machine-processable format such as WSDL [2]. Other systems which wish to use a web service may do so using SOAP [23] messages in a manner prescribed by the service interface. SOAP messages are written in XML and may be transported using HTTP.

2.2.1.1 Architecture

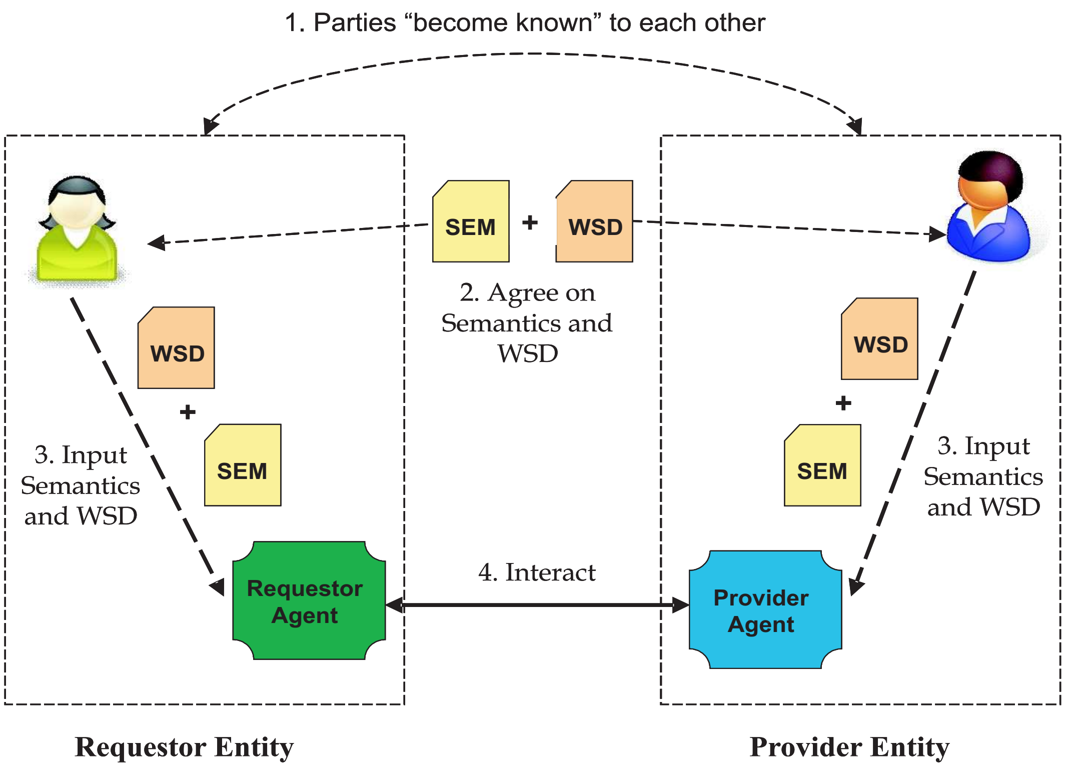

1. Parties "become known" to each other
SEM
+
WSD
2. Agree on Semantics and WSD
WSD
+
SEM
3. Input Semantics and WSD
WSD
+
SEM
3. Input Semantics and WSD
Requestor Agent
4. Interact
Provider Agent
Requestor Entity
Provider Entity

Figure 2.1: Web Services architecture showing a requestor/provider interaction [1]

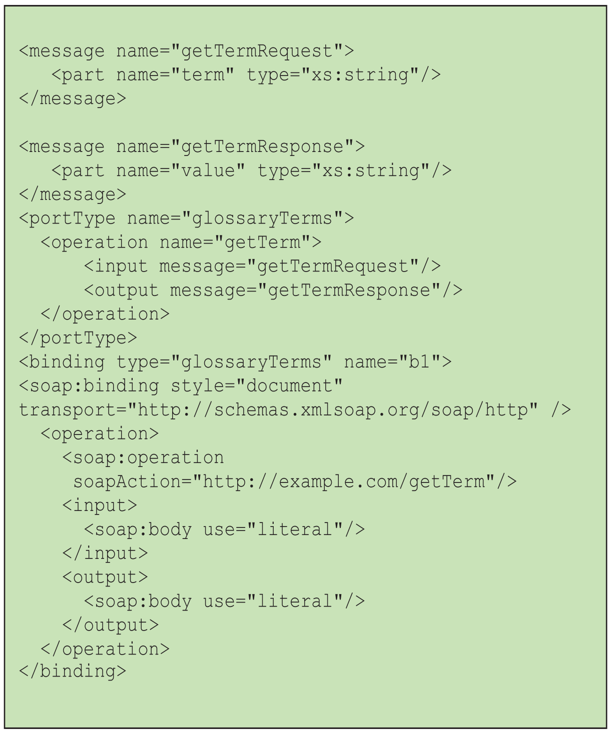
2.2.1.2 Agents, Requesters and Receivers

Web services provide functionality on behalf of their owner. A *provider* entity is a person or organisation that provides an agent which implements a particular service. A *requestor* entity is a person or organisation who requests to use a service provided by a *provider* entity.

An agent is a piece of software or hardware which physically sends or receives messages to provide or request a service. As shown in Figure 2.1, a *requestor* entity uses a requestor agent to exchange messages with a provider agent. Agents are independent of services, and their implementation has no bearing on the service. For example, two agents written in different programming languages can offer identical services.

2.2.1.3 Service Descriptions and Semantics

Information about the message exchange between requesters and receivers is stored in a Web Service Description (WSD). This file describes transport protocols, transport serialisation formats, datatypes and message formats for the interaction between receivers and requesters. It is typically written in WSDL. It also specifies one or more network locations at which a provider agent can be invoked, and may provide some information about the message exchange pattern that is expected. The information stored in the WSD describes mainly the mechanics of the service but sometimes also the semantics. An extract of a WSD is shown in Figure 2.2.



```
<message name="getTermRequest">
   <part name="term" type="xs:string"/>
</message>

<message name="getTermResponse">
   <part name="value" type="xs:string"/>
</message>
<portType name="glossaryTerms">
  <operation name="getTerm">
      <input message="getTermRequest"/>
      <output message="getTermResponse"/>
  </operation>
</portType>
<binding type="glossaryTerms" name="b1">
<soap:binding style="document"
transport="http://schemas.xmlsoap.org/soap/http" />
  <operation>
    <soap:operation
     soapAction="http://example.com/getTerm"/>
    <input>
      <soap:body use="literal"/>
    </input>
    <output>
      <soap:body use="literal"/>
    </output>
  </operation>
</binding>
```

Figure 2.2: An example WSD [2]

Essentially the semantics of a Web Service is the expectations of what the Web Service will do. This may be agreed orally or in the written form of a contract. In addition, as languages become more semantically rich, these may be incorporated into the WSD, and more of the agreement between requester and receiver may be automated.

2.2.1.4 How to Discover Web Services

As their name suggests, Web Services are typically deployed over the Internet and are therefore suitable for WAN Service Discovery. There are many methods in which Web Services are discovered and these are described here.

Registry Approach Using the registry approach to service discovery, service providers are required to publish their service descriptions on a central registry. The registry approach to Web Service discovery is authoritative and centrally controlled. That is, the registry owner has the power to decide who can publish or update services in the central registry of services. Registry owners also have the power to delegate permissions to provider entities to publish their own descriptions but third party providers cannot publish descriptions on behalf of another provider.

The registry approach typically uses Universal Description Discovery & Integration (UDDI) to describe web services in a standard format [24]. UDDI is a standard set of XML schemas designed to be interrogated using SOAP to provide access to WSDL documents. These schemas define the web service providers, the web services they make available and the interfaces used to access these services. Using UDDI, more accurate searches can be executed on the web service because the UDDI description contains more information than is stored in the WSDL document.

Index Approach The index approach to service discovery exhibits very little control in comparison to the registry approach. Essentially, an index is a guide to services which are published elsewhere on the web. Index owners can trawl the web looking for published services to add to their own index. Service providers do not know that their service has been indexed and do not give permission. This lack of control allows third-party services to be included in the index. It should be noted that individual indexes can also be implemented using UDDI. Google [16] is often cited as the most common example of the index approach.

Peer-to-Peer In contrast to the registry and index approaches of Web Service Discovery, Peer-to-Peer (P2P) provides a decentralised solution allowing nodes to discover services dynamically. To discover a service in P2P, a node asks another node in the peer system if they have a service of interest. Because nodes possess both client and server functionality, if the node is responsible for the requested service, it will respond. If not, it will propagate the request amongst its peers until either the service is found or some termination criteria is met.

2.2.2 CORBA

The Common Object Request Broker Architecture (CORBA), developed by the Object Management Group, allows computer applications to work together over a network. Using the Internet Inter-Orb Protocol (IIOP), a CORBA-based program from any vendor, on almost any computer, operating system, programming language, and network, can interoperate with a CORBA-based program from the same or another vendor, on almost any other computer, operating system, programming language and network [25].

In the CORBA architecture, each object specifies its interface using the Interface Definition Language (IDL). Clients invoke the services offered by server objects using the interface specified. The IDL is completely independent of all programming languages, but maps to most popular programming languages using OMG standards, e.g. C, C++, Java, Smalltalk and COBOL. By separating implementation from interface, CORBA provides complete interoperability.

The CORBA architecture works by compiling the IDL into client stubs and server skeletons. Next, an object is created with its respective client. Clients invoke the object implementation via the Object Request Broker (ORB). Due to the well defined nature of the IDL, the client and object may have been compiled using different programming languages, bur remain interoperable. The IDL stubs act as proxies, making and receiving requests for client and object independently.

2.2.3 LDAP

The Lightweight Directory Access Protocol (LDAP) is an application protocol used to query or modify directory services running over TCP/IP [26]. As an example, companies can store thousands of e-mail addresses on a server, and e-mail clients can use LDAP to browse these servers to lookup e-mail address information of contacts within the company. LDAP is not restricted to just contact information though, and can be used to retrieve a number of categories of information from a server. For example, LDAP can be used to lookup information about printer locations and encryption certificates.

Similar to LDAP is Active Directory [27], created by Microsoft. Active Directory not only provides directory services, but allows administrators to assign policies and deploy software and critical updates within an organisation.

2.2.4 Peer-to-Peer Networks

2.2.4.1 Introduction

In a P2P system, each peer possesses both Client and Server functionality. Participating network peers typically contribute resources such as bandwidth, storage, processing power or knowledge to the peer system, allowing the searching, retrieval, sharing or use of such resources. P2P systems form a network of peer applications above the network layer and are either 1-hop or multi-hop.

For more than 30 years, industrial and academic organisations have been utilising P2P architectures. In the late 1960s the original Advanced Research Projects Agency Network(ARPANET) connected a number of American university institutes as equal peers and was the world's first packet switched network.

In 1979 a P2P system named Usenet was developed by graduates from Duke University and the University of North Carolina. Usenet was based on the Unix-to-Unix-copy protocol (UUCP). In a network with multiple peers, one Unix machine would automatically dial another and exchange files. Students at these Universities could post and exchange news and messages using Usenet. Usenet grew significantly from two sites to in excess of one hundred thousand sites, maintaining its highly scalable P2P architecture.

P2P networks were not simply restricted to file and information sharing. As early as 1990, Intel utilised the parallel processing power of ten thousand Intel machines to run distributed simulations. The system Intel developed was called NetBatch, and drastically cut the cost of chip design. It removed the requirement for large mainframe computers within two years and saved Intel $500 million over the lifespan of the system [28].

In the late 1990s, the Internet became a popular mechanism of sharing music and files. Systems such as Napster [29] attracted millions of users to download media from other peers. However, the Recording Industry Association of America (RIAA) began a lawsuit against Napster accusing the designer, Shawn Fanning, of facilitating the theft of copyright material using the Napster system. Napster was closed in July 2001, but not before a number of smaller P2P systems called Kazaa [18] and Gnutella [30] were introduced. Systems such as Kazaa are still popular today, having facilitated the download of billions of files since its introduction.

2.2.4.2 Simulating P2P Networks

Due to the extremely large scale of P2P overlay networks, complexity has become a major issue in P2P networks. With the number of connected nodes reaching into the millions, a platform which can accurately simulate an overlay network is important. Many freely available network simulators exist, ranging from modelling networks at the packet level to concentrating purely on the overlay network. To determine which network simulators are suitable for P2P overlay networks, a number of tools were reviewed. This work is described in [31].

For the experiments documented in this Thesis, two methods of testing P2P technologies were used. Firstly, for simulation, the SSFNet network simulator was used [32]. Secondly PlanetLab [33], a global scale research network of 920 nodes, was utilised for realistic experiments. These are described next.

SSFNet SSFNet is a collection of Java and C++ based components and uses the Domain Modelling Language (DML) to configure networks as a series of configuration files. The principal classes in the SSF.OS Framework (for modelling of the host and OS components) and the SSF.Net Framework (for modelling the network connectivity) can be used to construct virtually any Internet model. The frameworks SSF.OS and SSF.Net hide all details of the discrete event simulator SSF API, which allows implementation of the protocols similar to a real operating system. This was the chosen simulator for Epichord [32].

SSFNet simulations can be executed independently or in parallel with other simulations running on Linux, Windows or SUN Solaris alike. Regarding maximum network size, a network of 33,000 nodes is achievable (on a powerful machine with 2GB of RAM). SSFNet also has the ability to run a distributed simulation.

PlanetLab PlanetLab is a global scale research network of 920 real nodes. Typically, these nodes are provided by academic institutions and private research labs. Members of the PlanetLab network may distribute software onto each of the member PlanetLab machines. Subsequently, P2P software can be installed on a number of machines and constructed into a P2P network. Similarly, distributed simulators can be installed onto a number of machines and connected to create widely deployed, large scale network simulations. Currently, OpenDHT [34] has been deployed on 200 PlanetLab nodes using the Bamboo P2P algorithm.

2.3 Local Area Network Service Discovery

2.3.1 OSGi Alliance

The OSGi Alliance defines a Java-based standardised, component-orientated environment for networked services [35]. Founded in 1999 by Ericsson, IBM, Oracle and Sun Microsystems, the OSGi Alliance is a non-profit organisation consisting of 35 companies (May 2007) operating in a number of different technology industries. The OSGi mission statement [35] is

> "... to create a market for universal middleware. The OSGi Alliance, therefore, promotes widespread adoption of the OSGi Service Platform to assure interoperability of applications and services delivered and managed via networks."

Initially, the OSGi Alliance focused on providing solutions for home networks. However, in recent years the Alliance has spread into automotive and mobile research, providing solutions for both car and mobile networks. Some examples of OSGi deployments in these sectors are the BMW 5 Series cars and Nokia mobile devices. Combining these successes with the progress made in the networked home suggests the OSGi model is a viable solution for a services middleware.

Currently, the OSGi Alliance is on the fourth release of the OSGi standard specification [36]. This specification consists of two sections: *standard service definitions* and the *OSGi framework*.

2.3.1.1 OSGi Framework

The OSGi architecture shown in Figure 2.3 contains a framework which provides a managed environment in which services can be deployed and executed. As mentioned previously, OSGi is a Java-based environment and is inherently modular.

OSGi applications are deployed in the form of *bundles* that are uploaded onto the framework. Each bundle can access standard Java libraries, but can also share OSGi-specific libraries offered by the framework or other OSGi bundles.

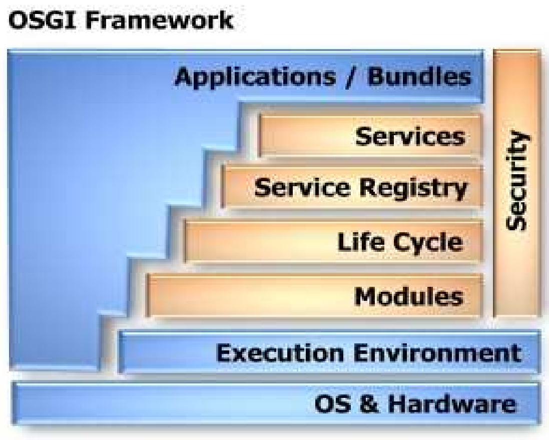


Figure 2.3: OSGi Architecture Overview [3]

2.3.1.2 Bundles

Physically, an OSGi bundle is a Java Archive file (JAR file) consisting of Java classes and perhaps some resources, e.g. images used by the bundle. JAR files are uploaded and made available to other bundles within the framework using an available deployment technique. An OSGi bundle may simply be a Java package designed to be used by other bundles, or more interestingly may provide *services.*

When an OSGi bundle provides a service, the service is registered with the OSGi service registry. OSGi services can query the service registry for other services they may require. For example, an air conditioning service may query a thermometer service to determine whether to switch itself on in order to regulate room temperature. The ability of small services to communicate allows developers to create quite complex systems from a large number of smaller, more manageable services.

2.3.1.3 Bundle Lifecycle

To allow network administrators and users to manage the OSGi framework, each bundle has an associated number of states. Figure 2.4 demonstrates the different states in which a bundle can operate to form its life-cycle. Initially, when a bundle is uploaded to the framework it is in *installed* state. The framework then tries to resolve any dependencies the bundle has on other Java packages provided in the framework. If this is successful, the bundle moves to *resolved* state. From the resolved state a bundle can be uninstalled (removed), updated or started.

Bundles which are updated (a user changes the content of a bundle and uploads the new version) are returned to installed state. Bundles which are started are moved into starting state and then to active state. The bundle then remains in active state until it is stopped, when it returns to resolved state.

2.3.1.4 Service Registry and Service Discovery

The service registry allows services to be discovered through queries formulated in an LDAP [26] syntax. Additionally, service requesters can receive events sig-

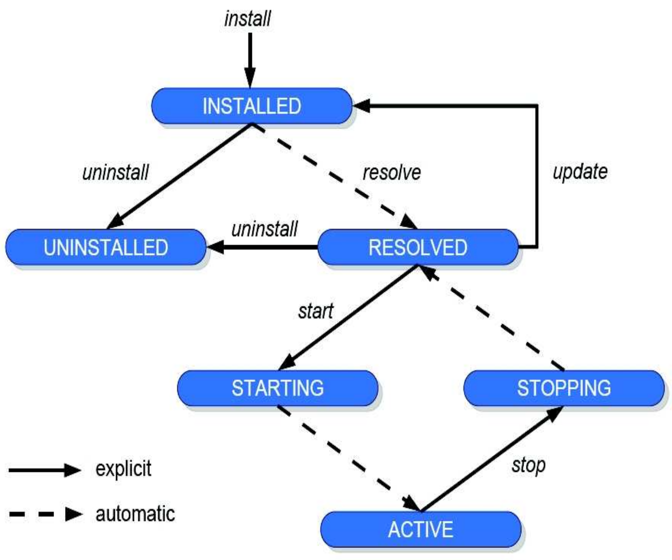


Figure 2.4: OSGi bundle lifecycle

nalling changes in the service registry; these changes include the publication or retrieval of a particular service. Service interfaces are implemented by objects created by the providing bundle. In standard OSGi, the bundle is responsible for run-time service dependency management activities. These include publication, discovery and binding, as well as adapting to changes resulting from dynamic availability (arrival or departure) of services that are bound to the bundle.

In general, the OSGi service specification does not define which properties must be registered with a service unless the service has been defined by OSGi. For example, in recent releases of the service specification, the OSGi Alliance has included specifications for UPnP and Jini services. Therefore, when a UPnP or Jini device is added to the framework, specific properties must be registered along with the services. By providing specifications for UPnP and Jini devices, OSGi can therefore be used as a service platform and discovery technique for multiple service discovery protocols, which is a major strength.

2.3.1.5 Standard OSGi Services

As mentioned in the previous section, the OSGi framework defines many standard OSGi Services. Examples of such services are Jini, UPnP, Bluetooth, HTTP, XML, Security and many more. Any OSGi bundle can leverage these services to enhance its functionality. For example, one may define a bundle which utilises the HTTP service to create a Servlet used for accessing/controlling the bundle from an external, web-based interface.

2.3.1.6 OSGi for Service Discovery

Due to the vast array of pre-defined service bundles in the OSGi framework, OSGi provides a rich service discovery platform for service creators and users. The OSGi service registry provides a reasonable searching technique using LDAP which is universal to all service types. The modular nature of the OSGi platform, coupled with the rich array of standard services provides developers with the unique opportunity to implement cross-protocol service discovery and utilisation. That is, devices which operate using one service discovery protocol may be discovered and executed using another service discovery protocol. An example of this is described in Section 3.3. In addition to this, bundles can be created which add Jini and UPnP services to the service registry for use by other services.

2.3.1.7 Federated OSGi

It is typical that only one OSGi framework exists in a user domain. However, it may be desirable to connect multiple OSGi frameworks to distribute services. For example, a device connected to machine A could use a service provided by a device connected to machine B. If both machines A and B exist in a single network domain, R-OSGi [37] can be used. R-OSGi uses the Service Location Protocol (SLP) [19] to discover services marked for remote invocation. Remote invocation of services is achieved using a proprietary solution.

Alternatively, it may be desirable to connect multiple OSGi frameworks in separate network domains. Connecting multiple domains using OSGi would allow services available only in LANs to be discoverable and executed in WANs. This Thesis presents a solution to connecting multiple OSGi frameworks in separate domains in Chapter 3.5.3. Regardless of whether federating OSGi frameworks occurs locally or globally, the same scalability issues are present. For example, there is a question how best to mitigate the overhead of building up a network of OSGi frameworks.

2.3.2 Jini

Jini [20] is a Java-based Object Oriented (OO) service discovery technology, developed by Sun Microsystems. The purpose of the Jini architecture is to federate groups of devices and software components into a single, dynamic distributed system. Each resulting federation is administered individually. Due to the platform-independent nature of Java, Jini is a very flexible service discovery technology and promotes *code mobility*. Code mobility allows executable code to be moved around a network and executed arbitrarily. For example, Jini clients search for services by interface and thus know which operations the service can perform, but may lack the Java class files for the service implementation. Code mobility would allow this device to download the required service implementation to access the service.

In Jini there exists three entities: **client**, **service** and **lookup service**. Lookup services are a special category of services; clients use well defined protocols to discover these. Lookup services catalogue available services; all Jini interactions between services and clients involve one. To discover a lookup service, nodes multicast a request for lookup services to identify themselves in a LAN. If a lookup service is present, the node will upload a service object to that lookup service. Nodes looking to discover a service search for the *lookup service*, download the service object and invoke it.

It is also possible in Jini for a service to also be a client. For example, a digital camera may provide the service to take pictures and offer them to a mobile device as a service but also require a printer service to print these pictures.

2.3.3 UPnP

The Universal Plug and Play (UPnP) forum [21] was established in 1999 by Microsoft and Intel. The architecture describes a peer-to-peer network where intelligent networked devices are discovered and configured automatically within a LAN [38]. As its name suggests, UPnP is *plug and play*, where device configuration and installation are transparent to the user, providing a simple, user-friendly architecture for networked devices.

Usability is key to UPnP, it is designed to make device use as simple as possible. Many popular devices already exist which follow a plug and play architecture e.g. USB data storage keys, USB mice and keyboards and many more. The success of existing plug and play devices suggests that UPnP would be a popular choice for networked devices. UPnP has already made strides in penetrating the networked device market with devices such as the popular Xbox 360 and Nokia N series mobile phones using UPnP technology.

The UPnP architecture consists of two components: the device and the control point. Devices in UPnP offer services: a printer may offer a print service or a lamp may offer a lighting service. The control point is the component which discovers and makes use of the services offered by devices. Devices cannot control other devices unless they are also control points. A typical UPnP interaction is shown in Figure 2.5, where the UPnP camera has a control point to control the UPnP printer but also offers a service to take photographs.

There are six important functions of the UPnP specificiation and these are briefly described here: Addressing, Discovery, Description, Control, Eventing and Presentation.

Addressing: The first step for a new UPnP device when it joins a network is to get an IP address. If DHCP is not available, devices must use auto-IP to randomly choose an IP address in the 169.254.0.0/16 range.

Discovery: After a device has obtained an IP address, it may now advertise its presence in the network. Devices use SSDP to broadcast an ssdp:alive message to 239.255.255.250, a reserved IP broadcast address. Control points in the network will be listening to this address and will add this new device to their list of known devices. Conversely, if a new control point joins a network it must locate all existing devices. To do so, the control point broadcasts a marginally different SSDP message called an M-SEARCH or an ssdp:discover message. Assuming devices then want to be found, they will respond to this message.

Description: After a device has announced its description on the network a control point will use an HTTP GET message to get the device description. When devices reply to search messages from control points, they specify the address of their device description. This device description is written in XML and describes the device type, serial number, manufacturer and the device services.

Control: After discovering a device and its description, a control point may now control a device. To do so, a control point invokes actions listed in the service description file using SOAP messages.

Eventing: In addition to controlling a device, a control point may also subscribe to a service. After subscribing to a service, a control point is notified of any changes to that service state. For example, a control point may subscribe to a togglepower service of a lightbulb, which has two states: off and on. If a the lightbulb state is changed manually from off to on (not via the control point), the control point would be notified of this state change.

Presentation: UPnP devices may provide a presentation page, which can be HTML. Presentation pages allow devices to be indirectly configured via a web interface. Presentation pages can also show device state and allow the device to be controlled via the web.

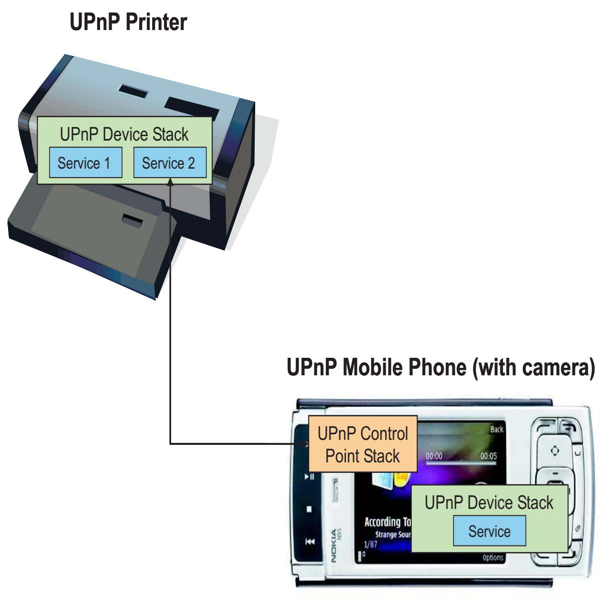


Figure 2.5: UPnP

2.3.3.1 UPnP A/V

UPnP A/V follows the standard UPnP technology model, using a Control Point to discover and control logical UPnP devices. Although the nature of UPnP device and service discovery is inherently distributed, the UPnP A/V protocol has centralised aspects in that media is stored on a central server. The UPnP A/V architecture is described by a three pronged model containing entities referred to as the Control Point, Media Server and Media Renderer. This model is shown in Figure 2.6. Definitions of these entities are as follows:

Control Point: The UPnP A/V Control Point discovers Media Renderers and Media Servers on the network.

Media Server: The Media Server is analogous to a media storage device holding content which can be played on a rendering device.

Media Renderer: The Media Renderer completes the UPnP A/V architecture and is used to playback or render content provided by the Media Server. Media Renderers may include media adapters, networked speakers, televisions and audio systems.

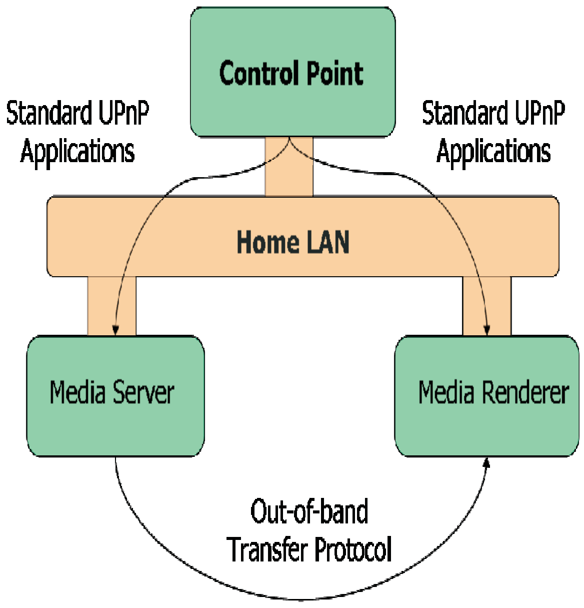

Control Point
Standard UPnP Applications
Standard UPnP Applications
Home LAN
Media Server
Media Renderer
Out-of-band Transfer Protocol

Figure 2.6: UPnP A/V Architecture [4]

2.3.4 Service Location Protocol (SLP)

SLP is a service discovery protocol for Local Area Networks which allows devices to discover services without prior configuration. To be discovered using SLP, a service must first have a Uniform Resource Locator (URL), which is the location of the advertised service. The SLP protocol defines three agents: User, Directory and Service. User Agents (UAs) search for services, Service Agents (SAs) advertise services and, Directory Agents (DAs) list advertised services.

To discover a service, a UA performs one of the following tasks depending on the presence of a DA. If a DA exists, SAs and UAs may discover it using the Dynamic Host Configuration Protocol (DHCP) [39]. DHCP servers distribute the addresses of DAs to UAs and SAs which request them. Alternatively, SAs and UAs can send a request to the multicast group address 239.255.255.253. DAs listening on this address can respond using unicast, thus identifying themselves and their location. Once the address of a DA is known, SAs can advertise their services and UAs can request services from the directory.

In the absence of a Directory Agent, UAs repeatedly send their service requests to the SLP multicast address. SAs listening on this multicast address will respond to service requests if they provide the service being sought.

2.3.5 Mobile Ad-hoc Networks

Mobile ad-hoc networks (MANETs) are essentially wireless ad-hoc networks. Nodes in a MANET, typically mobile routers, are connected by various wireless protocols such as Bluetooth [40] and 802.11x and are entirely self-configuring. Because nodes are free to move arbitrarily the MANET topology changes frequently, introducing unpredictability. As new nodes join MANETs they introduce new

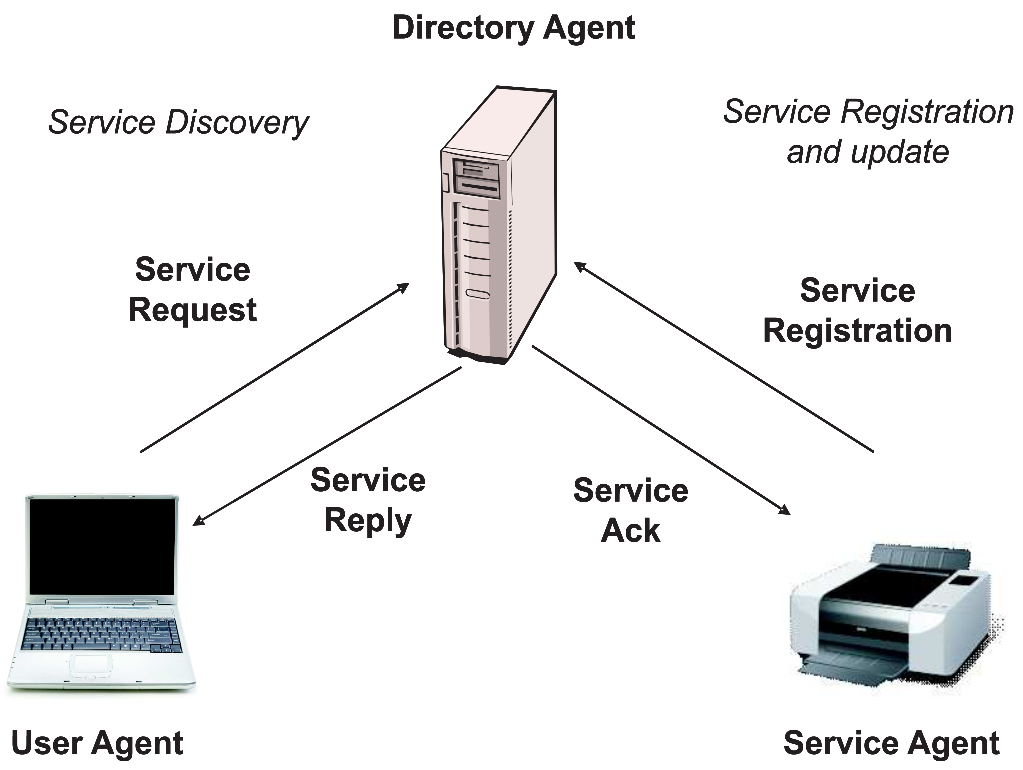


Figure 2.7: SLP Registration and Discovery

services, thus nodes in MANETs frequently search for new services in their proximity.

As nodes in MANETs typically use short range frequencies, service discovery in MANETs is restricted to LANs. For example, nodes equipped with a Bluetooth network adapter will be limited to finding services within a range dictated by the strength of their Bluetooth radio.

2.3.5.1 Bluetooth, IrDA and 802.11x

Technologies such as Bluetooth [41], IrDA [42] and 802.11x [43] are used to connect devices in close proximity to form a MANET which in turn is considered a Personal Area Network (PAN). A PAN is a collection of devices close to one person, which interconnect to form a network in an ad-hoc manner. These devices may not necessarily belong to that person but collectively form his/her PAN. Examples of devices which may be found in a PAN are PDAs, mobile phones, wireless headsets, laptops and portable printers. Device and Service discovery is achieved by broadcasting a signal in the LAN. The distance between devices in a PAN is constrained by the protocols they use to connect to each another. All are restricted, to discovery in LANs. As an example, a typical Bluetooth module will have a connectivity range of 10 meters (25 feet). An example PAN is shown in Fig. 2.8.

2.4 WAN vs. LAN Service Discovery

Almost all of the service discovery techniques described in this chapter are intended for LAN usage. However, with increased mobility in networked devices it is becoming popular for devices to discover services which are located in a different operating domain. For example, mobile phone users whilst away from their home may want to connect to their home media server to download tracks to

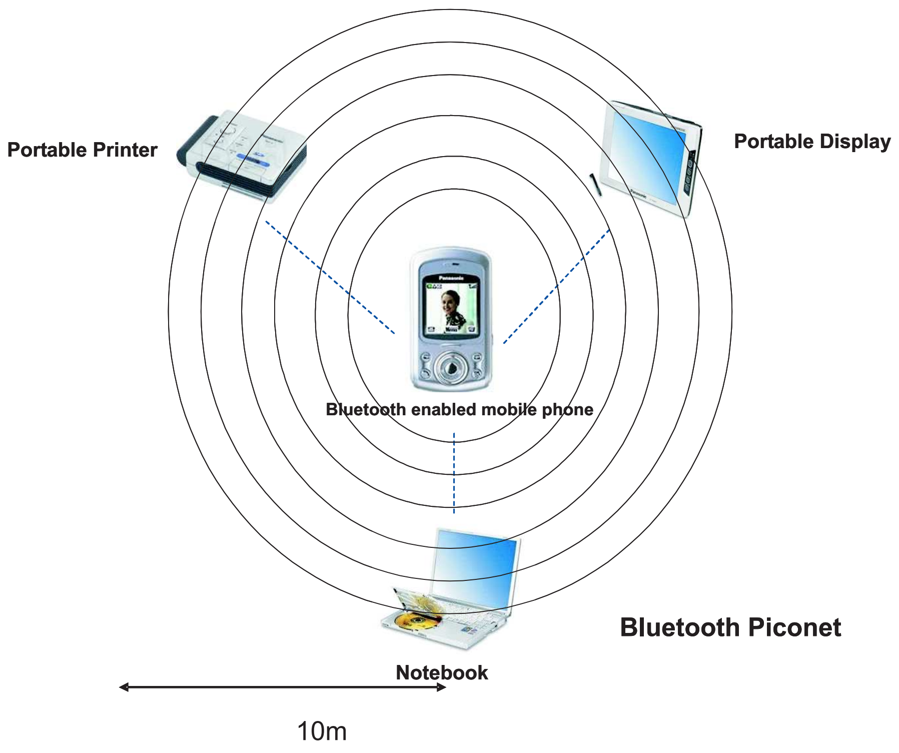


Figure 2.8: Bluetooth Connected PAN [4]

their MP3 player. Existing LAN service discovery techniques, of which there are many, cannot support this model. Therefore, a framework or bridging technology is required to enable LAN service discovery in a WAN.

Utilising P2P networks for WAN service discovery is one possible solution to discovering LAN services in a WAN and is described in more detail in Chapter 4 onwards.

2.5 Summary

This chapter described the service discovery problem domain. The strengths and weaknesses of both centralised and decentralised service discovery techniques are discussed in detail, as were a number of service discovery protocols. This was followed by a comprehensive overview of a selection of service discovery techniques.

Throughout this chapter, the trend of highly networked, mobile devices proliferating in the Internet for the purpose of service discovery was clear. Significant numbers of users look to the Internet to connect, discover and use services. However, many of the services they seek are hidden in local subnetworks, served by protocols such as UPnP, Jini, JXTA, SLP and Bluetooth SDP which are localised and unavailable for use on a global scale. To provide a platform for WAN service discovery of LAN services, new approaches must be considered.

Chapter 3

A Service Discovery Approach using OSGi and SIP

3.1 Introduction

To address the problem of discovering local services on a global scale, this chapter presents a SIP based service.

This service utilises a well known call signalling protocol (SIP) to bridge multiple user domains, offering services in each domain to the other. The restrictions imposed on LAN networked devices is removed and their services become available to devices in foreign domains. The SIP Service also helps address another service discovery problem associated with interoperability amongst the many service discovery techniques used by today's devices. This work is based on our own results in [4]. SIP is briefly summarised in the following section.

3.2 SIP

SIP is a WAN call signaling protocol used both for setting up and for managing multimedia sessions between two or more participants. Multimedia sessions may include Voice over IP, video conferencing and Instant Messaging [44]. As a text-based extentible request-response protocol that supports event management, SIP works seamlessly with Internet protocols like TCP/UDP and is highly suitable for Internet applications. For this reason, SIP features heavily in 3G (3rd Generation) devices [45].

Offering SIP functionality to devices as we do in the SIP Service gives SIP entities known as User Agents (UAs) personal and terminal mobility over WANs [46]. By leveraging the SIP Service, devices using protocols which are normally restricted to a single domain can operate in a WAN environment. Unlike protocols designed for use in a single domain that suffer from less than stringent security features, SIP offers extended authentication and encryption features. Additionally, SIP provides the ability to control devices in both static and mobile domains from a remote, mobile device [47]. Finally, devices using SIP may receive notifications of state changes of remote devices by utilising SIP eventing.

3.2.1 Architecture and Components

Within a SIP network there are two basic types of devices: end devices (user agents) and servers. User agents are the devices used by end users to place or receive calls. These may be SIP phones, or so-called soft phones which are software implementations to be run on a PC. Note that user agents do not necessarily interface directly with a user: an answering machine is also a user agent. User agents are distinguished according to their role in a call: the user agent client places the call, and the user agent server receives the call. User agents initiate and respond to signalling, and send and receive media. User agents are aware of the call state and may provide a number of services, such as Call Waiting, Call Forwarding or Call Screening.

Servers handle the application level control and routing of SIP messages. There are three different kinds of servers defined in SIP: Registrar, Redirect and Proxy servers. If a user is to be invited to join a session (call), there is the question of where the invitation should be sent to as users may be located at different IP addresses. Users are addressed by email-like addresses, e.g. sip:abr@cs.stir.ac.uk. This is a public address. However, at present the user may in fact be located at a computer with the name d25.cs.stir.ac.uk and be logged on as user ab0123. The SIP address for this location would be sip:ab0123@d25.cs.stir.ac.uk. To link the two addresses users need to registrar with a registrar server. Registrar servers work very closely with redirect and proxy servers.

Invitations are sent from the user agent client via a number of redirect or proxy servers to the user agent server. If a redirect server receives an invitation for a user, it checks with the database of the local registrar server and returns to the originator the address where it believes the user to be invited can be found. If the user is not actually located at that address, more information on the user's location may be available from that address.

Proxy servers are similar to redirect servers in that they help to find the location of a user. However, a proxy server does not return the found address but forwards the invitation on to that address. In the path that an invitation follows from the user agent client to the user agent server there may be a number of redirect *and* proxy servers.

Both proxy and redirect servers can host and execute call control services in that they can direct, block, or alter call signalling messages. Consequently, SIP offers the potential for truly distributed service provisioning. Services may be deployed on user agents, redirect servers and proxy servers. Therefore, SIP allows a degree of programmability which is unknown in the PSTN.

3.2.2 SIP Messages

SIP distinguishes two message types: *requests* and *responses*. Requests are messages sent from the user agent client to the user agent server. Responses are messages sent from the server to the client. SIP supports six request messages:

REGISTER is used to register contact information with a register server.

INVITE is used to establish a SIP session.

ACK is used in a three-way handshake after an INVITE to confirm the establishment of a session.

CANCEL is used to cancel the previous request.

BYE is used to terminate an existing session.

OPTIONS is used to query a server about its capabilities, e.g. supported SIP extensions.

SIP responses are based on HTTP responses and are distinguished by a three-digit status code. The first digit specifies the class of the response.

1** Provisional: The server has received the request and continues to process it.

2** Success: The request was successfully received, understood and accepted.

3** Redirection: A further action needs to be taken to complete the request. Usually the reply contains an address to which client should direct subsequent requests. This reply is commonly used by redirect servers.

4** Client Error: The request contains bad syntax or cannot be fulfilled at this server.

5** Server Error: The server failed in fulfilling an apparently valid request.

6** Global Failure: The request cannot be fulfilled at any server.

Responses of the 1** class are said to be *provisional*, whereas all other responses are *final*. Provisional responses need to be followed by a further final response.

Furthermore, SIP defines a *transaction*, which comprises all messages from the first request sent to the server up to the final response sent back to the client. INVITE requests use a three-way-handshake which consists of the INVITE message, possibly provisional responses, a final response and an ACK request. All other requests are answered by a final response which may be preceded by provisional responses.

SIP messages contain a number of *headers* providing more detail on the request, response and transaction. While there are only a few SIP messages, SIP is quite complex. This complexity is largely introduced by headers. There are a number of headers specified in the SIP standard, and furthermore, additional headers may be defined. If a server or user agent does not understand a header it simply ignores it. The OPTIONS request can help to establish what extensions the other party supports. The list below provides an overview of the most common headers.

Header	**Meaning**
To	Destination address
From	Originator address
Call-Id	Session ID
Cseq	Sequential Number of request within a session
Contact	User Agent Client address
Via	Addresses of nodes message has passed through
Content-Type	Type of payload in the message
Content-Length	Length of payload in the message
Allow	Requests understood by client
Supported	SIP extensions supported by client

Table 3.1: SIP Headers.

3.3 SIP Service - Overview

As shown in Figure 3.1, the SIP Service is an OSGi bundle that provides SIP support to other OSGi bundles and OSGi-registered devices. Included in the SIP Service feature set are service registrations, messaging, eventing and full SIP proxy and server functionality. When the SIP Service is deployed, it is registered as an OSGi bundle with the framework service registry, and is subsequently discoverable by any other bundle. Once the Service has been started, it offers to applications three primary objects: SIPServer, SIPDevice and SIPUserAgent.

3.4 Architecture

SIPServer: This object represents an instance of the SIP Service server (proxy/registrar) for the OSGi gateway. As a standard proxy, the server accepts registrations from SIP devices or bundles and proxies SIP messages to these registered SIP entities. Only one instance of the SIP server can operate within the domain. Using the SIPServer class, bundles can perform the following functionality:

- Request a SIPUserAgent object that represents a virtual SIP UA. A virtual SIP UA holds full SIP functionality but is *not* a physical SIP device. Upon creation of the SIPUserAgent, it is automatically registered with the SIP server.
- Obtain an object handle (a reference) to a SIPDevice, both virtual and physical which is registered on the OSGi framework. The SIP server is responsible for maintaining entries to the OSGi service registry for SIP devices registered with its SIP server.
- Receive notification of events relevant to the SIP Service occurring on the framework. Examples of these are devices registering/unregistering with the SIP Service server.

SIPDevice: Represents a reference to both virtual and physical SIP devices. A virtual SIP device is simply a bundle that supports SIPUserAgent function-

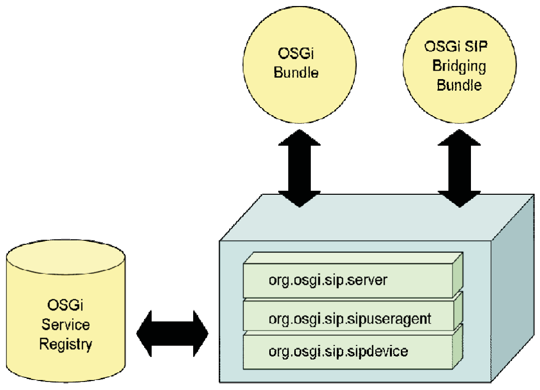


Figure 3.1: SIP Service Architecture [4]

ality. In contrast, a physical SIP device is a native SIP device registered with the SIP Service. Using this interface a bundle can:

- Request information about the given SIP device, e.g. its physical address or device type.
- Address the referenced SIP device in SIP-based communication using the SIPUserAgent interface.

SIPUserAgent: Permits bundles to use SIP methods to engage in communication with real or virtual SIP devices. Full SIP message set functionality is offered, including SUBSCRIBE, NOTIFY, MESSAGE and INVITE. Should a bundle wish to communicate with a SIP UA, it must instantiate at least one instance of the SIPUserAgent object via the SIPServer interface.

3.5 Functionality

As described above, the SIP Service can offer SIP functionality to application bundles. Consequently, an array of important services can be provided within OSGi. As an example, an OSGi bundle which has the ability to communicate with a UPnP device may also request SIP functionality and possess the ability to communicate using both UPnP and SIP. Such a bundle (known as a bridging bundle) can then provide protocol bridging (translation between two or more protocols) within the framework. Additionally, the SIP Service can offer application bundles the mobility features of SIP including personal and terminal mobility.

3.5.1 Protocol Bridging

To translate between devices which use different protocols, a form of application-layer proxying ("bridging") is required, e.g. SIP to/from UPnP. The SIP Service provides this feature by granting SIP UA functionality to a specialised UPnP-SIP bridging bundle. The bridging bundle, as its name implies, is an OSGi bundle that requests SIP UA functionality from the SIP Service and UPnP functionality from the UPnP OSGi Service. Note that the bridging bundle is not a component of the SIP Service but simply utilises it. Devices using different protocols can communicate or control one another via bridging bundles of different sorts. In addition, device representations (e.g. OSGi device bundles and SIP clients) may request information from the OSGi framework via bridging bundles regarding other registered devices, and then subsequently subscribe to their events.

An example of protocol bridging is a SIP mobile light control device which can discover, control and observe the state of a set of UPnP lights (on, off, dim). This can be achieved from anywhere within a WAN. The SIP lighting controller can request information regarding devices (lights) from the UPnP network via the bridging bundle. Note that the UPnP lights are not normally visible to the SIP lighting controller; it relies on the bridging bundle to interrogate the UPnP service and relate this information over SIP using the SIP Service. For each light, the lighting controller can subscribe to its events (changes of state).

In turn the bridging bundle listens to all UPnP event messages, filters and moderates events and forwards the relevant messages to the lighting controller via a notify SIP message and a specially-developed SIP event package. Control of the UPnP lights can be achieved by sending a SIP MESSAGE method to the bridging bundle, which translates the SIP content into a UPnP Service API function call, which ultimately results in a SOAP control action message for the UPnP light. The actual mapping between SIP and UPnP is very simple. The body of a UPnP API call (contents of the SOAP envelope) can be embedded into the body of the SIP message. The bridging bundle then extracts this message and creates the SOAP control message. An example of such a transition is shown later in Figure 3.4.

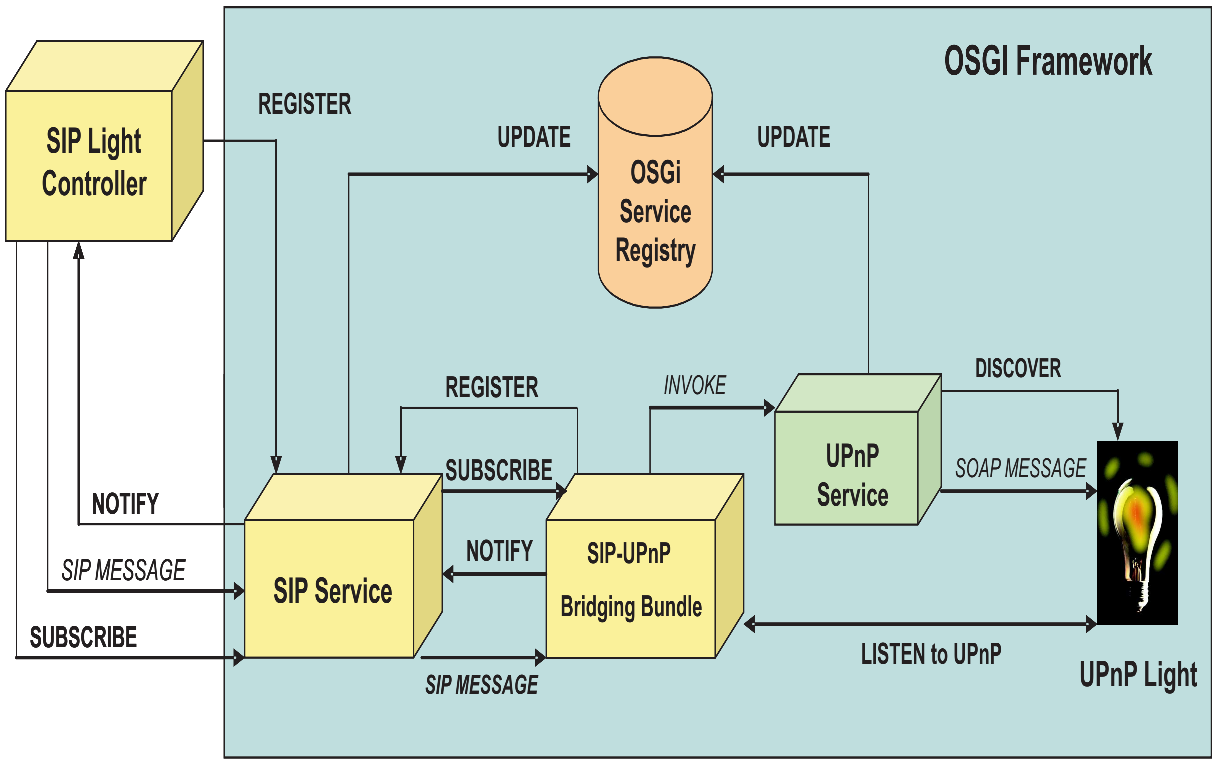


Figure 3.2: Protocol Bridging [4]

This example of protocol bridging is described in Figure 3.2. This form of bridging is achieved within a single OSGi framework. However, it is possible to bridge communications between devices occupying two or more frameworks as discussed in Section 3.5.3.

3.5.2 Application Layer Mobility

As mentioned in Section 3.4, a bundle can instantiate SIP UA functionality from the SIPServer class and behave as a native SIP device. This provides the bundle with personal mobility which allows a bundle service to be addressable and reachable at a single URL regardless of its location. This form of mobility is beneficial for numerous devices including phones, pagers and tracking devices. Personal mobility is achieved using the SIP REGISTER method, which allows a device or application bundle to change its current IP address by binding its public address with a contact Unique Resource Identifier (URI). As the device/application bundle moves throughout a network, it will send a REGISTER message to its original server from each new address. The server would then bind this address (which is the device's new Contact URI) to its public address.

3.5.3 Inter-Gateway Bridging

As discussed earlier, a mobile device user's PAN is restricted to include only devices located within the distance determined by the protocols they use to connect. By implementing inter-gateway bridging, the logical connectivity of a PAN can be increased to include not only local devices but also those registered on another gateway (e.g. Home Gateway). To implement this, devices and their services present on one gateway can be exported to a second gateway using the SIP Service and bridging bundles. Exporting of device information is achieved by packaging device-specific data into the payload of a SIP MESSAGE to be sent to the bridging bundle in the foreign domain. Upon receipt of this device information, the bridging bundle creates a virtual representation of that device which will exist on the foreign gateway and offer the services of the physical device.

3.5.3.1 Example

In the multimedia industry, media content is delivered via various methods including CDs and DVDs. More recently, customers are now downloading films and music from the Internet to various devices, e.g. MP3 players, mobile phones and Media Servers. As an example, suppose a person named John is travelling on business but wishes to record a TV program on his UPnP DVD recorder, which is connected to his home OSGi gateway. By using his SIP Service enabled mobile phone to connect to his home gateway, he can discover his UPnP DVD recorder and browse its Electronic Program Guide (EPG). By selecting the relevant program from the EPG, the Control Point creates a recording task on the DVD recorder using its Scheduled Recording Service (SRS) and the program is recorded upon its broadcast. This example displays UPnP messages over a WAN using inter-gateway bridging.

By bridging the home and mobile gateways, John is able to utilise the services offered by the UPnP DVD recorder in the home from the Control Point located on his mobile device. Without the use of inter-gateway bridging, this is not possible. In effect, when the bridging of the gateways occurs, the user's PAN is logically expanded to include not only the devices in his close proximity but also all interesting devices in his home. A diagram describing the steps taken to import the UPnP DVD recorder from the home gateway can be seen in Figure 3.3.

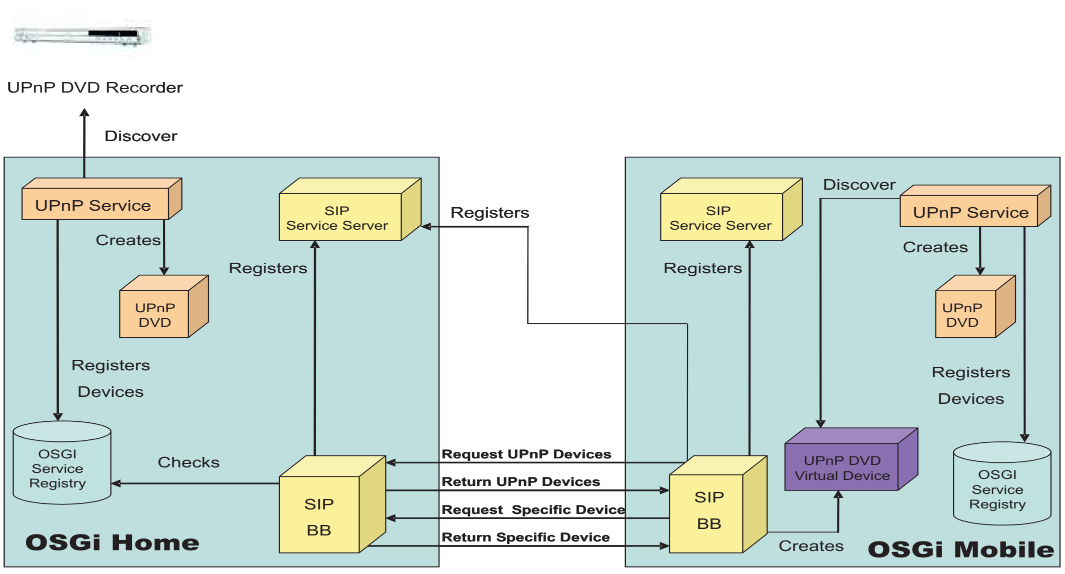


Figure 3.3: Inter-Gateway Bridging: Importing a UPnP device from Home to Mobile gateway [4]

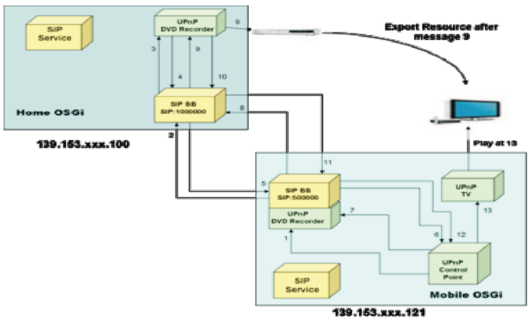
In the diagram there are two gateways (home and mobile). The bridging bundle on the mobile gateway registers with both the SIP Service server on the home gateway and its local server. After registration, it sends a SIP MESSAGE to the home bridging bundle requesting a list of all UPnP devices. The information is returned in the payload of a SIP MESSAGE. After perusing the device information, the mobile bridging bundle chooses a specific device of interest. In this example, the interesting device is the UPnP DVD recorder. After choosing a device, the bridging bundle then requests the specified device's details (services, state variables, actions). Using this device information the bridging bundle creates a virtual UPnP device, which is discovered by the UPnP service on the gateway. The UPnP Service creates a representation of the device and registers it with the framework service registry. The UPnP DVD recorder is now discoverable and controllable by UPnP Control Points on the mobile gateway. Next, a description is given of how control of the UPnP DVD recorder is achieved from a mobile location.

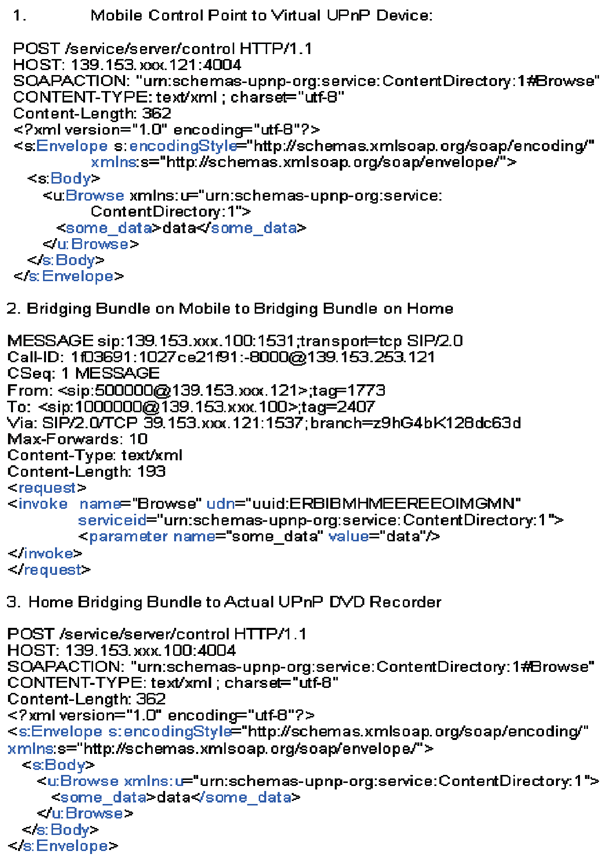
In this example a UPnP Control Point is required. The UPnP Control Point searches for UPnP devices to control. After finding the virtual UPnP DVD recorder, it may begin initiating control commands to begin browsing the EPG. The content of the EPG is exposed to the control point via the Content Directory Service (CDS), a required component of a Media Server. To control the UPnP DVD recorder from a mobile domain, the UPnP messages must be translated into a SIP message and sent to the mobile domain. This is achieved by extracting important data from the UPnP SOAP message, constructing an XML description of that message and sending it across the network to the bridging bundle in the mobile domain.

The bridging bundles are responsible for translation between SIP Service and UPnP Service API calls. When the mobile bridging bundle creates a virtual UPnP device, it must also listen for actions or queries executed on that device. Upon receipt of an event action on a virtual device, the bridging bundle translates the UPnP SOAP message intended for that device into XML and inserts it into the payload of a SIP MESSAGE. This message is then dispatched to the bridging bundle on the home gateway. When the bridging bundle in the home receives this message, it creates a new SOAP message using the received SIP MESSAGE payload and sends it to the physical device. The SOAP message is sent using HTTP, which ensues a response is always generated. The response should be repackaged as a SIP MESSAGE and returned to the bridging bundle on the mobile gateway where this communication is converted back to SOAP and forwarded to the Control Point. This process is completed for the full dialogue required to schedule a recording service. An example of bridging UPnP and SIP messaging can be found in Figure 3.4. These messages carry no data and should be treated as an illustration of bridging UPnP SOAP and SIP messages only.

3.6 Problems with WAN service discovery

By utilising the SIP Service, the static home appliance gains mobility and its services become available in a WAN. Clearly advantageous, the SIP Service bridges the invisible gap between the home and mobile domain. However, it is also clear that in order to access devices and services in multiple domains, the user must initiate bridging and have access to all domains. For security purposes, this is an excellent feature and would ideally suit users in private domains. However, there exists a scenario where devices and services should be publicly accessible and discoverable in a WAN. An example may be traffic monitoring devices on major motorways. To administer this using the SIP Service and a large number




```
1.          Mobile Control Point to Virtual UPnP Device:

POST /service/server/control HTTP/1.1
HOST: 139.153.xxx.121:4004
SOAPACTION: "urn:schemas-upnp-org:service:ContentDirectory:1#Browse"
CONTENT-TYPE: text/xml ; charset="utf-8"
Content-Length: 362
<?xml version="1.0" encoding="utf-8"?>
<s:Envelope s:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
          xmlns:s="http://schemas.xmlsoap.org/soap/envelope/">
  <s:Body>
    <u:Browse xmlns:u="urn:schemas-upnp-org:service:
          ContentDirectory:1">
      <some_data>data</some_data>
    </u:Browse>
  </s:Body>
</s:Envelope>

2. Bridging Bundle on Mobile to Bridging Bundle on Home

MESSAGE sip:139.153.xxx.100:1531;transport=tcp SIP/2.0
Call-ID: 1f03691:1027ce21f91:-8000@139.153.253.121
CSeq: 1 MESSAGE
From: <sip:500000@139.153.xxx.121>;tag=1773
To: <sip:1000000@139.153.xxx.100>;tag=2407
Via: SIP/2.0/TCP 39.153.xxx.121:1537;branch=z9hG4bK128dc63d
Max-Forwards: 10
Content-Type: text/xml
Content-Length: 193
<request>
<invoke  name="Browse" udn="uuid:ERBIBMHMEEREEOIMGMN"
         serviceid="urn:schemas-upnp-org:service:ContentDirectory:1">
         <parameter name="some_data" value="data"/>
</invoke>
</request>

3. Home Bridging Bundle to Actual UPnP DVD Recorder

POST /service/server/control HTTP/1.1
HOST: 139.153.xxx.100:4004
SOAPACTION: "urn:schemas-upnp-org:service:ContentDirectory:1#Browse"
CONTENT-TYPE: text/xml ; charset="utf-8"
Content-Length: 362
<?xml version="1.0" encoding="utf-8"?>
<s:Envelope s:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:s="http://schemas.xmlsoap.org/soap/envelope/">
  <s:Body>
    <u:Browse xmlns:u="urn:schemas-upnp-org:service:ContentDirectory:1">
      <some_data>data</some_data>
    </u:Browse>
  </s:Body>
</s:Envelope>
```

Figure 3.4: Inter-Gateway Bridging: Message flow [4]

of OSGi frameworks would be difficult and intricate. Additionally, there exists no method of 'discovering' other OSGi frameworks. Consequently, an OSGi framework must have knowledge of the remote OSGi frameworks it wishes to connect to. Alternative techniques for publicly accessible WAN service discovery should therefore be available.

3.7 Summary

This chapter described the SIP Service, which offers three primary services to devices and users in an OSGi enabled network: Inter-gateway bridging, protocol bridging and application layer mobility.

The gap between multiple domains is bridged by the SIP Service to facilitate WAN device and service discovery. However, the SIP Service does not scale well, and connecting a significant number of OSGi enabled domains is costly with regards to time. For example, in order to connect to another domain, a bridging bundle must obtain SIP user agent functionality then register with the registrar in both local and foreign domains. Next the bridging bundle must initiate communication with a bridging bundle in the foreign domain to share the descriptions of each device and service. Repeating these steps for large numbers of domains is possible, but time consuming. Therefore, the SIP Service is not an optimal solution for pervasive WAN service discovery and other solutions should be considered.

Chapter 4

P2P Service Overlays

4.1 Introduction

This chapter discusses P2P Service overlays and considers their use as a WAN service discovery technique.

Due to the relatively low barrier of entry in creating and deploying peer-to-peer systems, it seems likely that multiple wide area peer-to-peer service overlays will emerge with different service discovery and advertisement methods. This can create a number of potential issues for Service overlays: security, inefficiency and multiple service discovery types as discussed here.

4.2 Infrastructure

This Section describes the infrastructure of P2P systems and gives examples of the many types of P2P overlays which exist. As shown in Figure 4.1, many P2P overlays have been developed and used commercially, including the eMule [48] P2P client which is implemented on a Kademlia network, Kazaa [18], Gnutella [30] and BitTorrent [49]. These P2P systems can be classified at a high level as centralised or decentralised, as described in detail in Chapter 2. P2P systems can be further sub-categorised as structured, unstructured, multi-hop, one-hop, variable-hop and hybrid as discussed in this Chapter.

4.2.1 Multi-Hop

In a P2P system, operations are carried out between two nodes: the source node and the destination node. The number of hops taken to reach the destination node gives us our definition of a multi- or single-hop system. In a multi-hop system, a message is routed through several hops in the overlay network, with each intermittent node in the source-destination path contributing to the guidance of the message to its destination. The intermediate nodes can either be informed (in the case of a structured system) or uninformed (unstructured system). If the number of hops achievable is greater than 1 then the system is referred to as multi-hop. In a structured multi-hop system, messages are typically routed in

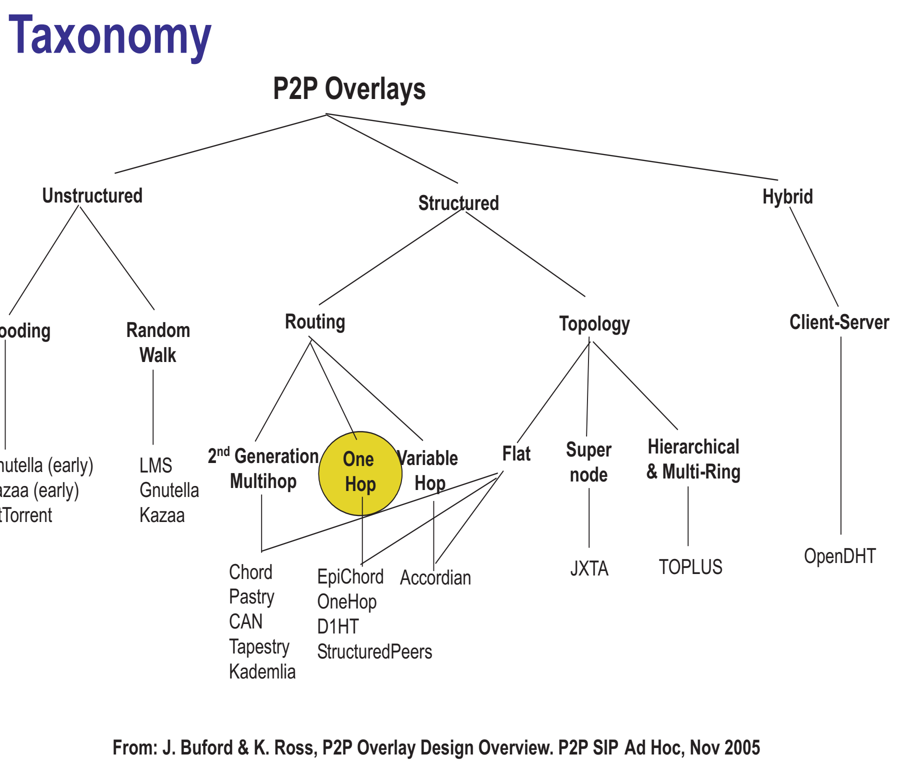


Figure 4.1: Taxonomy of P2P systems

O(log N) hops, where N is the number of nodes. Examples of such systems are Chord [5], Pastry [50] and Tapestry.

4.2.2 One-Hop

A 1-hop system (often referred to as single hop) aims to achieve lookup operations within O(1) hops. To do so, each node maintains a routing table containing the state of *all* other nodes in the system. 1-hop systems do not always achieve success in the first hop due to the difficulty of keeping the routing table up to date. The single most significant factor for this is churn, where nodes join and leave the system frequently. Whenever a node joins or leaves the network, all other nodes must be informed of this event and this takes time.

1-hop systems are an ideal case for P2P. However, until recently it been argued that it was unreasonable to expect small nodes to support the memory requirements associated with a full routing table for a large network. With modern devices now commonly equipped with high capacity, affordable persistent storage the arguments against 1-hop systems are receding. Even the smallest nodes have large storage capacities which are ultimately capable of sustaining a full routing table. For this reason, systems which can obtain 1-hop routing have become a very attractive option.

4.2.3 Structured

Nodes in a structured P2P system each employ a protocol which allows them to route a search throughout the network for some file or information. A structured P2P system guarantees that if a file exists in a P2P overlay it can be found. The most common structured overlay system is the Distributed Hash Table (DHT), described in Section 4.2.6. A DHT works by employing a lookup(key), insert(key) interface. Some well known DHTs are Chord [5], Pastry [50], Tapestry [50] and Tulip [51].

4.2.4 Unstructured

In an unstructured overlay, overlay links (connections between peers) are established arbitrarily. This makes it easy to construct new networks in an ad-hoc manner as a joining peer can simply copy a neighbour's links. Unstructured overlays are rather inefficient, with lookup messages having to be flooded throughout the network to find desired information (but no guarantees). To increase the likelihood of a successful lookup, popular content is likely to be available at several peers. If a peer is looking for rare data shared by only a few other peers, then it is highly unlikely that the search will be successful. In unstructured systems, high volumes of unsuccessful lookup traffic is generated. In spite of this, most of the popular P2P networks such as Gnutella [30] are unstructured.

4.2.5 Hybrid

Hybrid P2P systems have an element of both centralised and decentralised functionality. For example, in Napster [29] the indexing is centralised but file sharing is decentralised. Hybrid systems typically offer good performance due to tasks such as searching being centralised.

4.2.6 Distributed Hash Tables

Distributed Hash Tables (DHTs) are a class of distributed systems which provide storage and retrieval functionality for name and value pairs. Like a traditional hash table, a user may perform a lookup for a name, e.g. lookup(name), to retrieve a value. In addition to this, a user may also store data in the DHT, e.g. put(name, value). The responsibility of the mappings of names to values is distributed, often evenly amongst nodes though other variations exist. For example, some DHTs are proximity aware and store data on peers local to the content. A more even distribution of keys to nodes allows DHTs to scale to very large sizes.

In order for DHTs to work effectively they must remain consistent. That is, should a node which is responsible for some value x leave the system, the DHT must handle reassigning a node responsible for this value. This is known as consistent hashing.

4.2.7 Multi-hop P2P Systems

4.2.7.1 Chord

Chord is a distributed peer-to-peer (P2P) routing infrastructure which performs a mapping between file identifiers (fileId) and node identifiers (nodeId) [5]. With a high degree of accuracy, Chord identifies the node in a P2P system which correctly maps to the file being inserted or located. Chord also adapts efficiently as nodes join and leave the system and can answer queries even if the system is continuously changing. It is widely accepted that Chord is both scalable and reliable.

Each Chord node is assigned a nodeId, which is obtained by creating a hash (e.g., SHA1 or MD5) of the IP address of the given node. In addition, files stored in the system are assigned a fileId, which is obtained by hashing some value associated with the data, e.g. filename and a secret. Chord effectively maps data items onto nodes which have the numerically closest ID. To perform this task, all nodes are ordered in an 'identifier circle' (see Figure 4.2) modulo 2m. Key k is assigned to the fist node whose identifier is equal to, or follows k in the identifier space. This node is called the successor node of key k.

Nodes only require knowledge of their successor nodes, and these are held in a finger table. The finger table consists of at most m nodes. When a node receives a routing message, it checks its finger table for its successor node of the given key

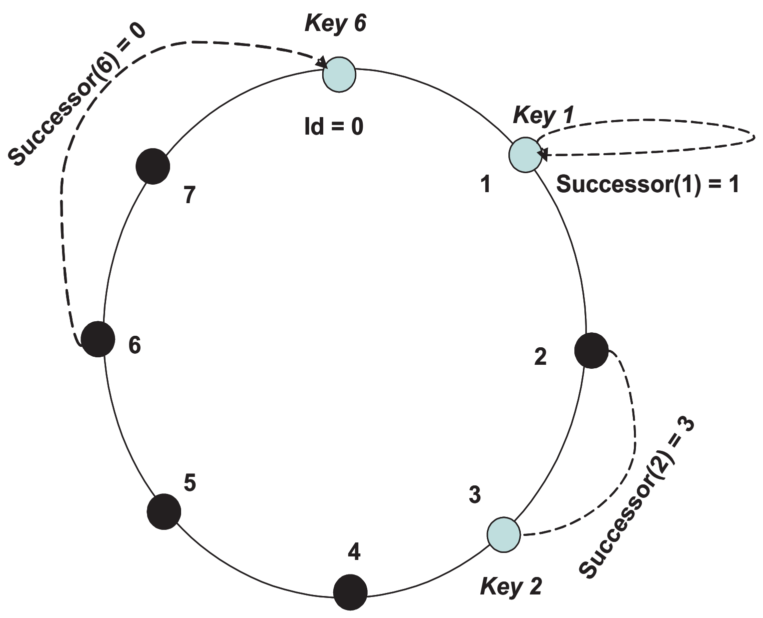


Figure 4.2: A Chord identifier circle consisting of three nodes (0, 1, 3) [5]

and routes the message to that node. If the node does not know the successor for the given key it will route the message to a node that is numerically closer than itself, and the process will continue thereafter until the closest node is found.

4.2.7.2 Pastry

Like Chord, Pastry is a distributed P2P routing algorithm which performs a mapping between nodeIds and keyIds [50]. Each Pastry node is allocated a nodeId using a secure hash of the node's IP address (e.g. SHA-1) to provide uniformity across the node space. For the purposes of routing, a node holds information in a routing table and a leaf set.

Firstly, nodeIds and keyIds are represented as a sequence of digits with base 2^b where b is a configuration parameter, typically 4. A node's leaf set consists of the $l/2$ numerically closest smaller nodeIds (l is typically 16) and $l/2$ numerically closest larger nodes. A node's routing table is organised into $\lceil log_2 b(N) \rceil$ rows with $2^b - 1$ entries in each row. The $2^b - 1$ entries in row n of the routing table each refer to a node whose nodeId matches the present node's nodeId in the first n digits. For example, nodes in row 2 will match the present node in the first 2 digits. The $n + 1th$ digit of each nodeId in node n will not match.

When a Pastry node receives a routing message, it initially checks its leaf set to see if the message key falls within the range covered by those nodes. If so, the message is passed to the relevant node. If not, the node then uses its routing table to forward to a node that shares a common prefix with the key by at least one more digit. Using this mechanism in a Pastry network, routing can be achieved in O(log N) steps where N is the number of nodes in the network.

4.2.8 1-Hop P2P Systems

4.2.8.1 EpiChord

In Epichord, peers maintain a full-routing table and approach 1-hop performance on DHT operations compared to the O(log N) hop performance of multi-hop overlays, at the cost of the increased routing table updates and storage.

In Epichord, as in Chord, peers are organised in a concentric circle in the identifier space, with keys stored at the first node succeeding them in the identifier space. For example, in Figure 4.3, key 2 is stored at node 3, etc. An Epichord peer's routing table is initialised when the peer joins the overlay by getting copies of the successor and predecessor peer's routing table. Thereafter, the peer adds new entries when a request comes from a peer not presently in the routing table, and removes entries which are considered dead. If the churn rate is sufficiently high compared to the rate at which lookups add new entries to the routing table, the peer sends probe messages to segments of the address space called slices. As in Figure 4.4, slices are organised in exponentially increasing size as the address range moves away from the current peer's position. This leads to a concentration of routing table entries around the peer, ensuring a node has accurate knowledge of its neighbours.

To improve the success rate of lookups, Epichord uses p-way parallel requests directed to peers nearest to the destination node who should be responsible for some value *val*. A p-way parallel request is the sending of p unicast messages in parallel, replacing the use of a single unicast request message. A peer which does not have the key of interest replies with additional entries from its routing table that are closer to the key of interest. Due to the cache maintenance mechanism described above, these local peers should be accurate. The requesting peer issues additional lookup requests if any entries are closer to the key than its previous

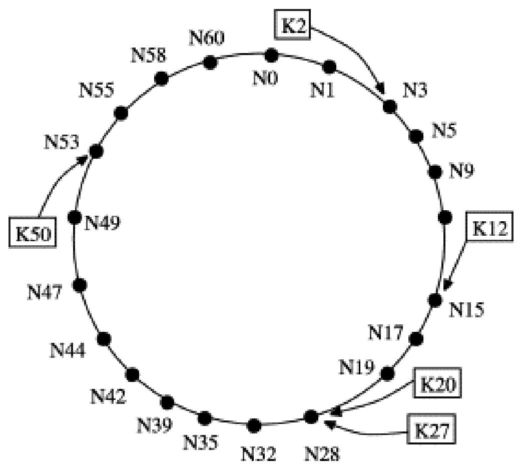


Figure 4.3: Circular identifier space, showing keys and nodes organised with IDs increasing clockwise [6].

set of requests. For example, a peer issues 5 parallel unicast requests to peers around a key of interest. If none of the 5 contain the key, then any replies are used to issue additional parallel unicast requests.

During periods of high churn, a peer maintains at least l active entries in each slice of its routing table, typically 2. When the number of entries in a slice falls below l, the peer issues parallel lookup messages to ids in that slice. Responses to these ids are used to add entries to that slice in the routing table. All parallel lookups in Epichord are carried in separate unicast messages.

4.2.8.2 D1HT

As per EpiChord, D1HT nodes maintain a routing table containing all nodes in the peer system. To join the D1HT system, a node p_{any} must know one of the

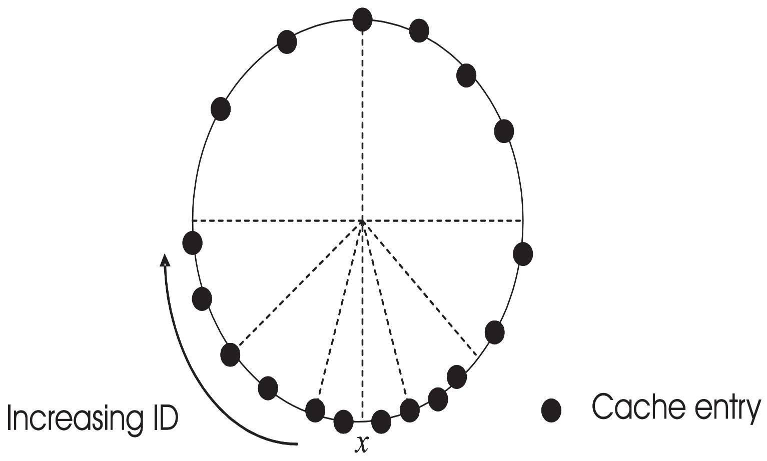


Figure 4.4: Division of address space into exponentially smaller slices with respect to node x [6].

existing member nodes already in the system. Assuming this, the joining node p hashes its own IP address and asks p_{any} to route a lookup for p, which will return the successor of p. Node p will now contact its successor node p_{succ} to be inserted into the ring and gain information about IDs it is responsible for. After this, p will then receive a copy of its successor node's routing table. This method is common throughout 1-hop systems and is consistent with EpiChord.

The only events which affect the accuracy of the routing table in peer systems are joins and leaves. Consequently, if all nodes in the peer system are aware of all joins and leaves, the routing table remains up-to-date. Subsequently, the D1HT maintenance algorithm concentrates on an efficient, effective technique to propagate join/leave information throughout the system. Inefficient methods to distribute this information, e.g. broadcast, are avoided. D1HT proposes the Event Detection and Report Algorithm (EDRA).

4.2.8.3 EDRA

To disseminate information about joins and leaves (events), each peer p sends up to ρ propagation messages at each Θ secs time interval, where $p = \lceil log_2(n) \rceil$ and Θ is based on the system dynamics. Every message will have a Time To Live (TTL) value associated with it in the range from (0 - ρ) and is sent to $succ(p, 2^l)$, which is the nodes at positions modulo 2 in the successor path. Also, p will include in its messages all events that occurred in time interval Θ.

Figure 4.5 illustrates how EDRA disseminates an event in a D1HT peer system with only 11 peers. The following rules formally define the EDRA algorithm and are taken from [7].

Rule 1: Every peer will send at least one and up to ρ messages at the end of each Θ secs interval, where $p = \lceil log_2(n) \rceil$.

Rule 2: Each message will have a Time To Live counter (TTL) in the range 0 to ρ - 1, and carry a number of events. All events brought by a message with TTL = l will be acknowledged with TTL = l by the receiving peer.

Rule 3: A message will only contain events acknowledged during the ending interval. An event acknowledged with TTL = l, l > 0, will be included in all messages with TTL $< l$ sent at the end of the current Θ interval. Events acknowledged with TTL $= 0$ will not be included in any message.

Rule 4: The message with TTL $= 0$ will be sent even if there is no event to report. Messages with TTL > 0 will only be sent if there are events to be reported.

Rule 5: If a peer does not receive any message from its predecessor for T_{detect} secs, it assumes that the predecessor has left the system.

Rule 6: When a peer detects an event in its predecessor (it has joined or left the system), this event is considered to have been acknowledged with TTL $= \rho$, and so is reported through ρ messages according to rule 3.

Rule 7: A peer p will send all messages with TTL $= l$ to $succ(p, 2^l)$.

Rule 8: Before sending a message to $succ(p, k)$, p will discharge all events related to any peer in $stretch(p, k)$.

4.2.9 Overlay Networks

An overlay network is a group of peers connected via *virtual links*. Each virtual link in the Overlay can represent many physical links in the underlay network as shown in Figure 4.6.

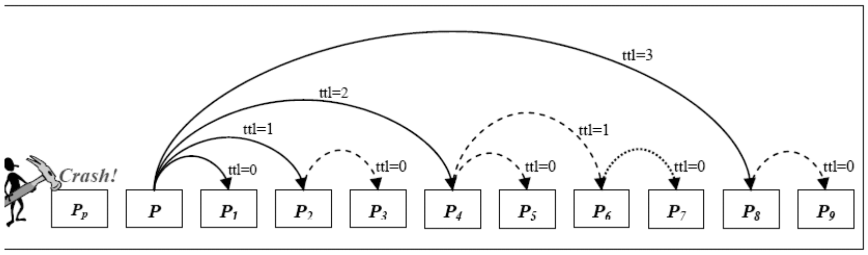


Figure 4.5: A D1HT system with 11 peers (p = 4), where a peer crashes and this event is detected and reported by its successor nodes. Each peer should only receive one single notification of the event [7].

4.2.10 Service Overlays

Although typically associated with Web Services, service orientation in a P2P architecture means that peers can offer and use services from any other peer without relying on centralised resources. By utilising P2P networks for service discovery, a highly scalable and generally fault tolerant service discovery mechanism is possible, achieving global scale service advertisement and discovery. Combining service advertisement and discovery with a P2P overlay creates what is referred to as a service overlay [52]. The construction of a service overlay involves mapping service descriptions, typically complex structured documents, to the overlay index mechanism to provide an efficient index and lookup facility.

Peers in a service overlay require protocol support for the overlay operations: join, leave, lookup, insert and finally routing table maintenance. Service overlays are constructed by nodes self-organising into a P2P overlay network using a join protocol. How a node learns about existing nodes in the overlay is system de-

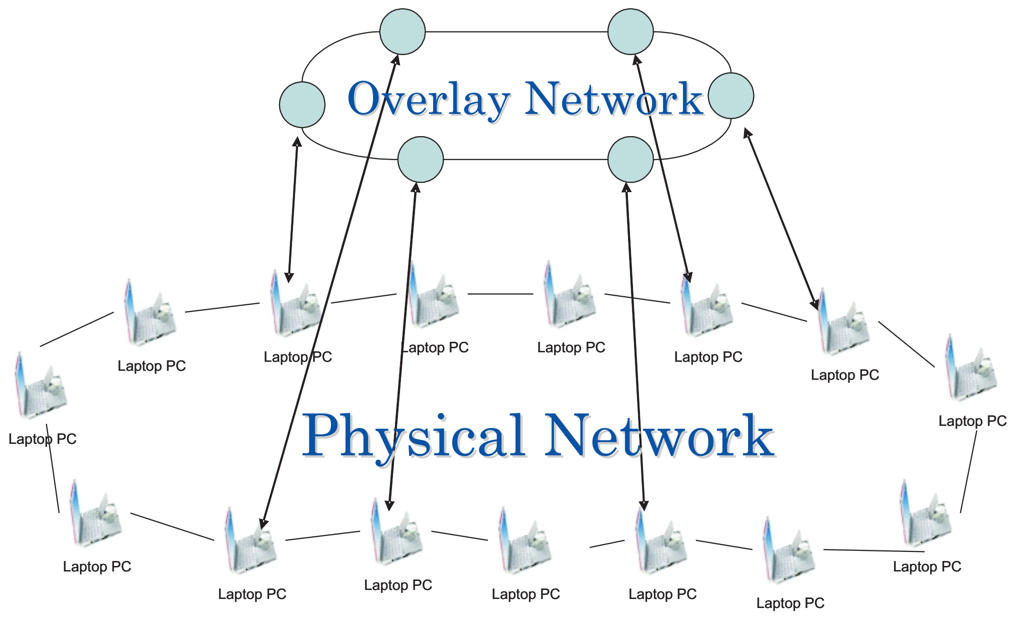


Figure 4.6: Example Overlay Network

pendent. First nodes are assigned a unique node ID, after which they connect to their neighbouring peers. In the case of a structured overlay, nodes create their own routing table. The majority of service overlays use a Distributed Hash Table (DHT) for lookup and storage functionality. Entries to be stored in the overlay are hashed using a hashing function, (e.g. SHA1) and stored at nodes whose ID is the closest match to the ID of the entry to be stored. Therefore when lookups are issued to the overlay in the form lookup(id), the overlay routing algorithm resolves which node is responsible for that entry by determining which node ID is closest to the lookup ID.

4.3 Problems

4.3.1 Multiple Service Discovery Types

Before examining the inherent problems associated with discovering services over a WAN, it is important to address the problems associated with choosing one of the many Service Discovery Mechanisms (SDMs) available. As multiple SDMs proliferate across various administrative domains, mobile devices will require a method to locate and select the appropriate mechanism according to the context of the mobile device, such as network domain, location, protocol, and application. One possible solution to this problem is Meta Service Discovery, described in Section 4.5.

4.3.2 Inefficiency

Many multi-hop peer-to-peer structured overlays have been proposed for peer-to-peer applications, but are characterised by O(log N) hop count. In addition, because each overlay hop is routed in the underlay in potentially many native hops, multi-hop overlays have poor latency characteristic for connecting large numbers of peers. Consequently several systems have been developed to trade-off latency for larger routing tables. Larger, more accurate routing tables allow nodes to lower hop counts to around O (1) hops. However these designs lead to increased network traffic for managing the larger routing tables.

Devices which want to participate using peer-to-peer applications are required to decide between the cost of keeping routing tables accurate or the cost of having to send increased numbers of messages for DHT operations. For mobile nodes or nodes which have restricted bandwidth and processing power, this is not an attractive choice as both designs are potentially expensive. One solution to this problem is described in Section 5.

4.3.3 Security

It is accepted that all nodes in a P2P network pose a potential threat to their peers. Denial of Service (DoS) attacks, where nodes bombard a targeted node with requests with the intent to overwhelm it, are amongst the most common threats. In addition to these, exposing confidential information, sharing inaccurate data and administering viruses and trojan horses are also common. Such obvious threats will inevitably hinder the progress of many P2P applications, and measures to tackle these issues are required.

One method to reduce the threat posed by peers is to authenticate them. That is, if a node's identity is known then it is less likely that the node will misbehave. However, authentication of P2P systems is non-trivial for several reasons. Most modern P2P systems are decentralised and decentralisation of a network removes the ability to administer security centrally. As an example, a centrally administered network may have external vetting techniques (registered details) to ensure nodes are trustworthy before they are allowed access to network services. The central administrator can subsequently ask nodes to identify themselves when present on the network. When central servers are removed from this infrastructure, nodes must assume a level of trust amongst their peers and methods to enhance this trust are required.

It is clear that security is a key research area for P2P networks. However, this Thesis does not further address security as it is beyond the scope of its work. Instead this Thesis focuses on problems with service discovery and searching inefficiencies.

4.4 WAN Service Usage

Devices connect to the Internet and discover services offered by service providers. We refer to this pattern as wide-area service discovery. Many commercial services are already offered to devices through web services. Using wide-area service discovery, location-specific discovery can be performed outside the immediate area. For example, a roaming device might discover services along a planned route. Wide-area service discovery does not require devices to offer services to all other devices on the Internet. Devices connect to the Internet, offer services, and discover services offered by other devices connected to the Internet. We refer to this pattern as wide-area peer-to-peer service discovery. Compared to wide-area service discovery, it means that all classes of users can participate as both service providers and service users. It permits a range of use not available to the previous patterns:

- Users can be content publishers as well as consumers.
- Collections of home networks and PANs can interoperate. For example, we can use peer-to-peer discovery mechanism to share PVRs (Personal Video Recorders) between different home networks, so that a program can be recorded on a remote PVR if the local one is scheduled for a conflicting time.
- Multiple users in a social network can share device resources regardless of geographic proximity. Users can form online communities and interact directly using the personal devices.

A greater scale of resource sharing can be achieved, and resources that might not be available in the local area can be discovered and used. For example, a network of devices forms a personal grid, and can be used to distribute computation load to implement algorithms for content-based retrieval or image recognition.

4.5 Meta-Service Discovery

4.5.1 Introduction

This Thesis presents Meta Service Discovery, a technique used to find and select a Service Discovery Mechanism (SDM) by context. There are many SDMs including network services (such as SLP), integrated device network adapters (such as Bluetooth) and those integrated into middleware platforms (such as Web services, Jini, and JXTA):

- Bluetooth Service Discovery Protocol (SDP) [40]
- Corba Trading Object Service [53]
- Service Location Protocol (SLP) [19]
- Simple Service Discovery (SSDP) in UPnP [38]
- Web services and UDDI [1, 54, 55]
- Salutation [56]
- Jini [57]
- JXTA [58]
- Wide-area extensions to SLP: WASRV [59], wide-area SLP [60].
- Secure Service Discovery Service (SSDS) [61].

It could be argued that multiple SDMs, both per domain and in the public Internet, will co-exist in the future for several reasons:

- Service discovery in pervasive computing is an extension of web search, and there will be economic incentives for multiple SDMs just as there are multiple web search sites today.

- Application-specific SDMs will be designed and managed for the benefit of specific user communities, somewhat analogous to the multitude of web portals today.

- Administrative control is argued in [62] as a basis for hierarchical and multi-ring overlay architectures. The same arguments apply to the use of multiple SDMs.

- Experience has shown that standardisation does not guarantee a single solution in the market place, the large number of incompatible instant messaging and presence protocols are one example of this.

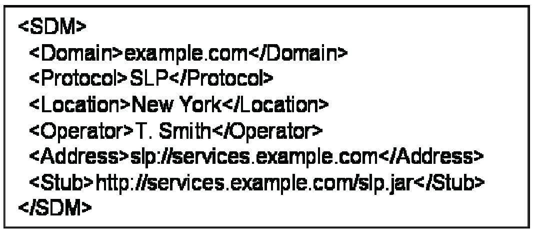
- The low barrier of entry to create both application-specific, (e.g., file sharing) and general purpose (e.g. Open DHT) peer-to-peer overlays is leading to the proliferation of P2P overlays, and service-oriented overlays will follow this trend.

- From the various efforts to develop complex query languages and semantic lookup over Distributed Hash Tables (DHTs), the trade-off between query language power, efficiency and overhead could lead to a spectrum of service overlays with query languages matched to the needs of the application domain.

Consequently, a method is needed to identify and select SDMs. For example, a mobile node may roam to a new location, and applications on the node may wish to discover services available in the new context. The node must first determine which SDMs are available to use in this context. The node can uses Meta Service Discovery to identify SDMs by context. In this step, it may obtain a client stub to download and use for accessing the SDM. It can then communicate directly with each SDM of interest, issuing discovery requests and obtaining service descriptions.

4.5.2 Design

In essence, Meta Service Discovery is a SDM for discovering other SDMs. By treating SDM discovery as a type of discovery problem, the approach of creating a searchable description and a method for SDMs can be followed. Figure 4.7 shows a document describing a SDM. Each tag contains an attribute of the SDM, such as domain, protocol, and location. Meta Service Discovery then becomes the searching, advertising, and querying of collections of such descriptions. In the wide-area approach, SDM descriptions are stored in a structured overlay.



```
<SDM>
  <Domain>example.com</Domain>
  <Protocol>SLP</Protocol>
  <Location>New York</Location>
  <Operator>T. Smith</Operator>
  <Address>slp://services.example.com</Address>
  <Stub>http://services.example.com/slp.jar</Stub>
</SDM>
```

Figure 4.7: SDM Description

4.5.2.1 Meta Service Discovery in a Structured Overlay

In this scenario, SDM descriptions are stored in a DHT implemented using a structured overlay. There are several techniques for storing structured documents in a DHT so that the queries can be made against fields of the document. These techniques involve partitioning a structured document into strands and breaking up strings into substrings which can be separately indexed. As discussed later, this leads to multiple index entries per SDM description.

Figure 4.8 illustrates Meta Service Discovery on a structured overlay using a DHT. For simplicity, the wide-area DHT is shown as a single ring, but it could also be a hierarchical or multi-ring overlay. Entries in the DHT are depicted at the bottom of the diagram, and represent various SDM repositories, servers, and specialised service overlays. Each SDM repository, server or service overlay is indexed by its SDM description. Queries for SDM entries may be by any combination of fields in the SDM description, such as domain, location, size, and operator.

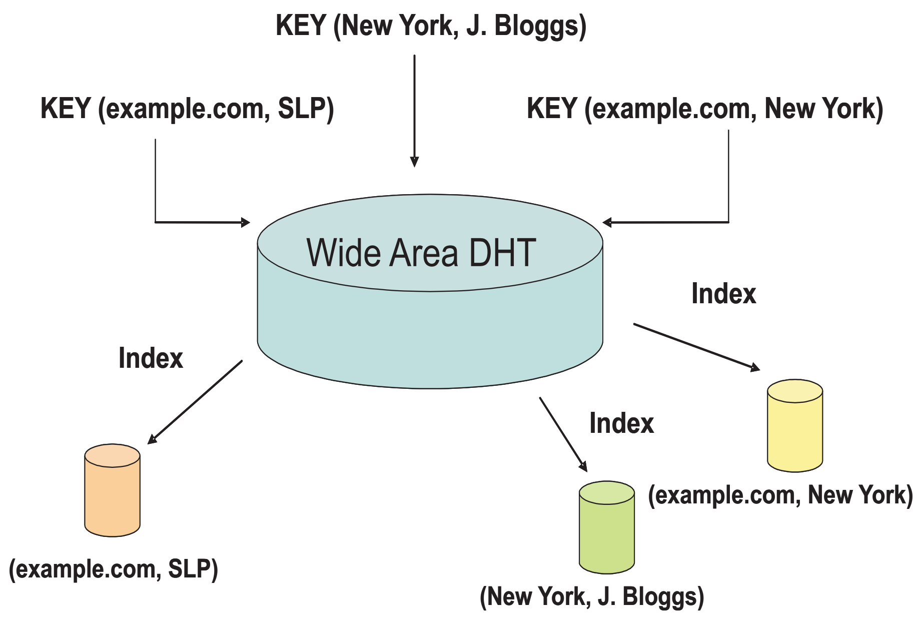


Figure 4.8: Meta Service Discovery global index entries and lookup by various types, for discovery of various SDMs [8]

Since SDM descriptions are significantly less complex than most service descriptions (for example, those found in WSDL, SLP, or UPnP), the mapping of SDM descriptions to the DHT hash indexing is straightforward, and queries are simpler than if full service descriptions were directly hashed in to the DHT index. In Section 4.5.3, some example mappings implemented in three existing structured overlays (INS-Twine [22], FreePastry [50, 62] and OpenDHT [34]) are described. The following section describes each overlay network and the underlying DHTs used by each.

4.5.3 Mapping SDM Descriptions to Existing P2P Overlays

4.5.3.1 Experimentation

To test Meta Service Discovery in existing P2P overlay networks it was necessary to create a large data set of SDMs. The dataset was then inserted into three available structured overlay indexing systems (FreePastry, OpenDHT, and INS/Twine). For FreePastry and OpenDHT an interface was created to decompose an SDM description into combinations of fields to ensure a wildcard lookup could be made by any combination of [domain, SDM protocol, location, and a free text field]. In these two systems, each SDM resulted in 15 index entries. As described in the previous section, INS/Twine provides an interface to map structured documents into index entries in Chord. For INS/Twine the set of SDMs from the generated dataset were encoded into the format used by INS/Twine (INS).

For FreePastry, 20 SDMs were stored in a 220 (virtual) node network. The FreePastry file storage mechanism (PAST) was implemented to provide persistent storage. Storing 20 SDMs resulted in 300 total entries in the DHT (20 x 15). 300 queries were then issued to match the 300 entries, all of which returned a result. Due to a synchronisation problem with the FreePastry implementation, a larger test was not carried out. Another limitation of FreePastry is that duplicate keys are overwritten rather than appended. Since the SDM is decomposed into separate index entries for each field for wildcard purposes, collisions on protocol name, for example, were common. This means that duplicate keys are replaced, and consequently cannot be retrieved.

At the time of writing, OpenDHT is deployed on about 200 nodes on PlanetLab. PlanetLab is described in detail in Section 2.2.4.2. 72,019 SDMs were entered into the PlanetLab network, which led to 1,072,014 total entries, for which 1,008,140 queries were issued. Unlike FreePastry, OpenDHT does permit multiple duplicate keys, and a lookup returns all matching keys. During the querying there were 7,334 errors (non responses) (0.73%). OpenDHT errors are non-responses and are attributed to node or network conditions. For example, a congested network, failed link or router. OpenDHT successfully handled the SDM mapping and queries needed for these experiments. For INS/Twine 200 nodes were created and 200 SDMs stored upon them leading to 2200 total entries. 2200 queries were then issued to validate the data entries. INS/Twine successfully handled the SDM mapping and queries needed for these experiments.

4.5.3.2 Sizing and Key Distribution Analysis

If Meta Service Discovery were widely deployed, it would be useful to estimate the number of SDM entries that would be required. It is assumed that each domain that has type A DNS Resource Record will advertise at least one SDM.

The most recent measurement is 71,723,098 type A DNS entries [63]. Type A DNS entries are Host addresses.

For estimating sizing and measuring key distribution, it was assumed that SDM locations would correspond to population densities. Two data sets were obtained, one containing the LL position for the 2555 largest cities [64] and another containing the square area for the 40 largest cities [65]. Both are worldwide datasets. For the 40 largest cities, assuming a grid spacing of 1 city block (about 200 meters), there are about 3.5M grid points.

4.5.3.3 Representing Service Discovery Mechanisms by location

Each SDM may also be referenced by location, e.g. street address, landmark, or latitude-longitude (LL). Representing location in structured P2P systems is difficult and costly. A location may cover a specific range, e.g. city block, town or state. When searching for data by location, a user must know the specific location value which was indexed when the data was entered. This is an unreasonable assumption. Many approaches to tackle this problem have been developed, including range queries. A range query searches the DHT for values in a range defined by the user. For example, if a service were located at the LL (10 10 42.10) and the user defined the searchable range as (10 10 42.0) to (10 10 43.0), queries would be issued for all values between this range until the entry was discovered. This method is rather crude and costly.

To address this problem, this Thesis presents a novel technique to allow street addresses and landmarks to be converted to a corresponding LL. The LL can be normalised to decimal format and aligned to the nearest grid point on a map. As an example, if a service location fell within a grid section, it could be aligned to the centre point of that grid section. That resulting grid point can then be directly indexed into the DHT. By implementing this solution, lookup is considerably simplified as only a single query is required to search for a SDM by location. However, it is fair to wonder if the grid alignment solution could possibly affect load balance in the DHT and cause 'clumping' of data in nodes which represent popular service locations. Tests in Section 4.5.3.4 discard this concern.

4.5.3.4 Key Distribution

As described previously, each SDM is decomposed into separate fields and indexed into combinations of fields. Key distribution of the domain name and location entities was evaluated separately. For the distribution of domains, a list of 405,097 domains registered in 1997 was obtained from [66]. After hashing each domain using SHA1, keys were inserted into a network consisting of 10,000 nodes. Figure 4.9 shows the Probability Density Function (PDF) of the key distribution using real domain names. These results are similar to those measured in [5], and demonstrate that by indexing domain names almost no penalty is incurred compared to randomly generating keys. The slight deviation in the uniformity

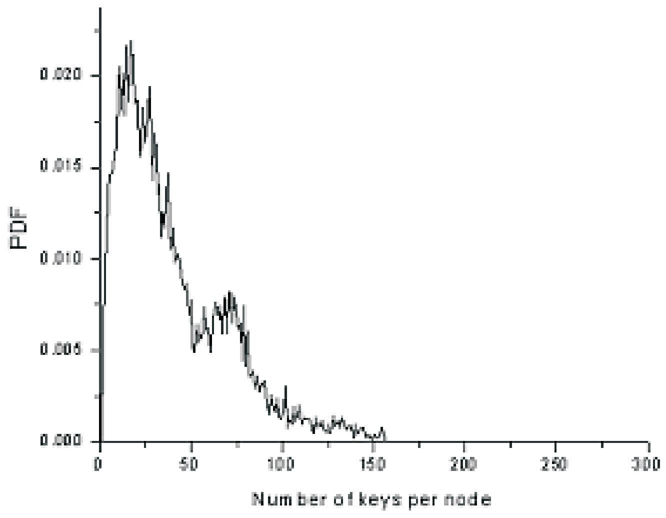


Figure 4.9: PDF of key distribution of 405,097 real domain names in a 10,000 node network [8]

is likely caused by the non-uniform dataset, but this is marginal. For example, with regard to domain names, there exists a disproportionate number of '.com' top level domains.

As described previously, location was represented in a normalised LL format. LL positions were randomly generated and hashed using SHA1 in 2555 cities on a 200 meter grid spacing. Figure 4.10 shows the PDF for 0.5 million location entries over 10,000 nodes. Figure 4.11 shows the 1st and 99th percentile and mean keys per node for a range of keys. These results, like those collated for the distribution of domain names, are almost as uniform as those measured in [5] for randomly generated keys.

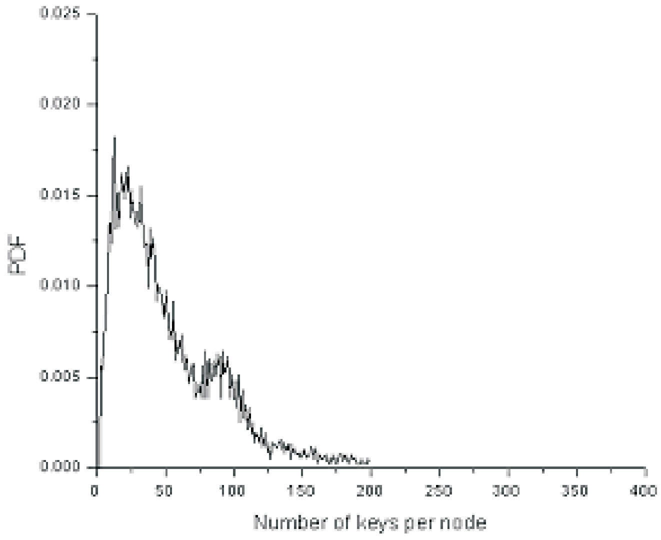


Figure 4.10: PDF for location based key distribution of 500,000 keys over 10,000 nodes [8]

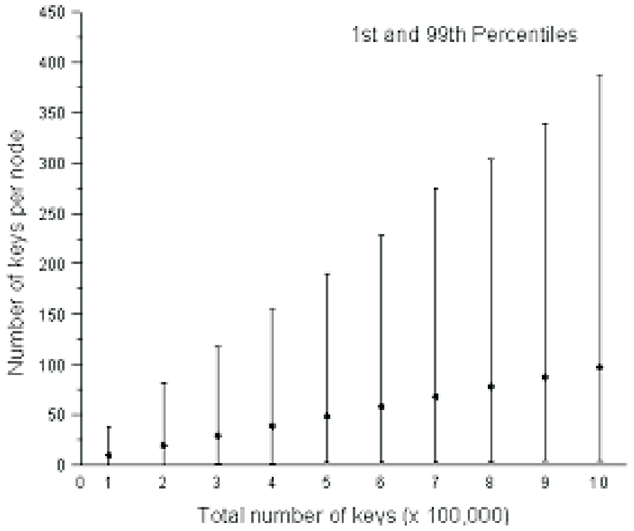


Figure 4.11: Number of keys per node vs. total numbers of keys for 10,000 nodes [8]

4.6 Summary

In this chapter, P2P service overlays were presented as a potential solution to the WAN service discovery problem. However, P2P overlays suffer from inefficiencies, security problems and lack a clear method to address the large number of available service discovery mechanisms.

To address the latter of these issues, Meta Service Discovery was presented. Meta Service Discovery is successfully implemented using three well known P2P systems, confirming P2P systems can be used to search for and store SDM descriptions. Key distribution for MSD is uniform.

A new location-based searching mechanism was presented as an extension to Meta Service Discovery. Latitude-Longitude pairs can be normalised to decimal format and aligned to the nearest grid point on a map. This also results in a near uniform key distribution.

Chapter 5

Improving the efficiency of P2P Systems

5.1 Introduction

This Chapter presents a novel method to improve the efficiency of all categories of P2P systems. Using multi-destination multicast, it is shown that it is possible for users of P2P systems to reduce their bandwidth consumption by 30%. For mobile device users, this is a significant figure.

5.2 Why improve efficiency?

With the proliferation of mobile networked devices flooding the consumer market, device users are presented with a vast choice of methods to access networked services. It is not uncommon for mobile phone users to browse the Internet, engage in Instant Messenger chats, and download pictures almost simultaneously. It would therefore not be unreasonable to suggest that P2P applications would be an interesting application scenario for mobile devices, offering device users a new method to access already popular content. However, P2P Service overlays are inefficient. Systems which offer reasonable performance, e.g. 1-hop overlays, consume large amounts of bandwidth to maintain accurate routing tables. For mobile devices, users are charged per kilobyte for the data they download. Therefore designers of P2P systems for mobile devices have to realise that every packet counts.

5.3 Multicast

It is well known that messages that are multicast can make significant savings over similar messages that are sent using unicast. The quantitative benefits of multicasting have been formulated in the Chuang-Sirbu scaling law [67] which shows that the ratio of the number of edges in the multicast versus the average unicast path length equals $m^{0.8}$ (where m is the multicast group size). The per link reduction in message traffic from multicast is then:

$$1 - m^{-0.2} \tag{5.1}$$

The exponent ε = -0.2 represents the multicast efficiency. The assumptions of the Chuang-Sirbu law are:

1. Receivers are randomly distributed throughout the network.
2. Multicast trees are shortest-path trees constructed using Dijkstra's algorithm and extend only to the edge router.
3. There is no control overhead, or control overhead is $< 1\%$ of total traffic.
4. It applies to networks that are representative of inter-domain routing topologies of the Internet, with average node degree in the range of 3 to 4, and a range of sizes from 47 routers to 5000 routers.
5. The multicast routing at the network layer does not support any reliability mechanism.

The first assumption relates to the placement of peers in the network and the distribution of the set of peers in a parallelised overlay operation. In overlay analysis, it is generally assumed that peers are randomly distributed throughout the network. With the exception of proximity-aware overlays, neighbours in the overlay are not close to each other in the network and are unlikely to cluster on the same subnet. As a result, parallel overlay operations which are directed at a set of adjacent overlay nodes, such as a parallel lookups, are sent to a random set of network locations. In the case of proximity-aware overlays, the multicast savings can be no worse than predicted by Chuang-Sirbu and may be better due to the sharing of the tree path. The remaining assumptions relate to the network and the multicast mechanism. Using multi-destination routing satisfies assumptions 2, 3, and 5. The networks of interest satisfy assumption 4.

Further evaluation of Chuang-Sirbu has been completed in [68], which derives a similar expression, and [69] which finds some shortcomings of Chuang-Sirbu with respect to large groups and provides a revised formulation. Chalmers and Almeroth [70] used actual multicast data sets on the Internet and synthesised multicast trees, finding a slightly lower multicast efficiency ε in the range $-0.34 < \varepsilon < -0.30$. The results are based on the actual multicast infrastructure in place at the time of the data collection, which may constrain multicast branching points more so than in synthetic topologies. Further, their analysis includes multicast trees extended to the end points, which produces increased savings from Chuang-Sirbu if there is clustering of end points at subnets.

In the case of the parallelised overlay operations discussed here, the size of the multicast group is within the Chuang-Sirbu formulation, but is frequently at the lower end of the formulation ($m < 20$). In this range, the multicast efficiency exponent derived by Chalmers and Almeroth is also applicable. For brevity, multicast savings $1-m^{\varepsilon}$ with $-0.34 < \varepsilon < -0.20$ are referred to as the Chuang-Sirbu law. In comparison, Fahmy and Kown [71] give a range of $-0.4 < \varepsilon < -0.2$ for IP multicast efficiency, but do not present experimental results to substantiate

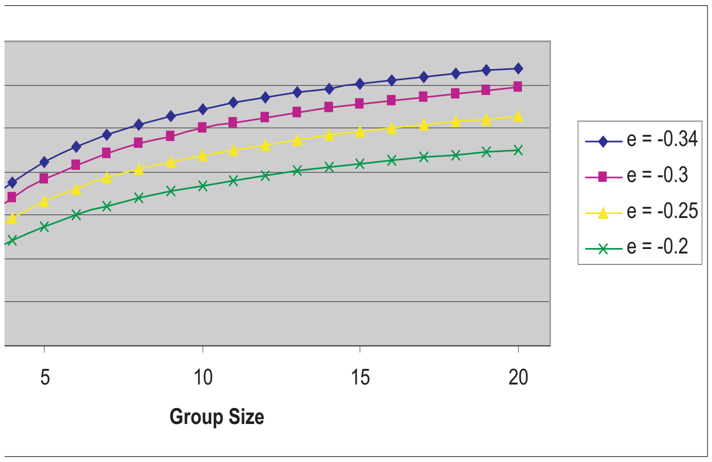


Figure 5.1: Message savings versus group size for Chuang-Sirbu, varying ε

the larger range. Figure 5.1 shows message savings versus group size for a set of values for ε.

For mobile P2P applications, achieving message savings in excess of 27% is interesting. One would expect, however, that a trivial solution such as multicasting would have already been considered for P2P systems. To explain why this is not the case, this Thesis present two different methods of IP multicasting: the host-group model and multi-destination multicasting, whose suitability for P2P is discussed.

5.3.1 Traditional Host Group Model

The prevailing host-group multicast protocols including PIM, DVMRP, and CBT create a single group address per multicast tree, and each router stores state for each active group address. Routers advertise multicast groups to leaf nodes, and nodes which wish to join the multicast tree inform the router. The state in the router grows with the number of simultaneous multicast groups. Further, there is a delay when creating a multicast group, and the network may only support a limited number of group addresses. Therefore, for a large overlay it is impractical for each node to have a group address for each set of other nodes it sends multicast messages to, as the number of multicast groups is large and the group sizes are typically small.

Worsening the problem is that the average peer session time is as low as 60 minutes in some overlays, meaning that group state is replaced relatively frequently. Host group multicast is designed for relatively small number of very large sets of recipients. So host group multicast is not a good choice for P2P overlay operations where there are many simultaneous small groups of peers involved in a message, and the groups are short-lived.

5.3.2 Multi-Destination Multicast

The concept of multi-destination routing was proposed in the early years of multicast protocol design [72] but, as Ammar observes [73], subsequent protocol design focused on enabling large multicast groups. However in the past several years, there has been recognition of multi-destination routing as a complementary multicast technology that has advantages for applications which feature large numbers of small groups. In multi-destination multicast routing, instead of two or more unicast messages sent to separate destinations, a single message is sent containing the list of the destinations and the message content from the original messages. Multicast-enabled routers route the message until a split point is reached (according to unicast routing decisions). At each such point, duplicate messages containing the subset of destinations for each forwarding path are created and routed. This continues until a message contains only a single address in which case it is converted to a unicast message and is routed to its destination. If only a subset of all routers is multicast-enabled, these routers forward multicast packets to other multicast routers using tunnels through unicast routers. Alternately, the network contains hosts which implement the multi-destination multicast routing.

Overlay nodes sending a multi-destination multicast message send the message to one such host. The host routes the multi-destination messages to other hosts according to the network routing rules, e.g. OSPF. At each such host, duplicate messages containing the subset of destinations for each forwarding path are created and routed. This continues until a message contains only a single address, in which case it is converted to a unicast message and is routed to its destination. Figure 5.2 illustrates this. Here node A is sending a multi-destination multicast message to 4 nodes: B, C, D and E. Routers where a split occurs are shown by filled circles. The numbers indicate the count of destination addresses in the message.

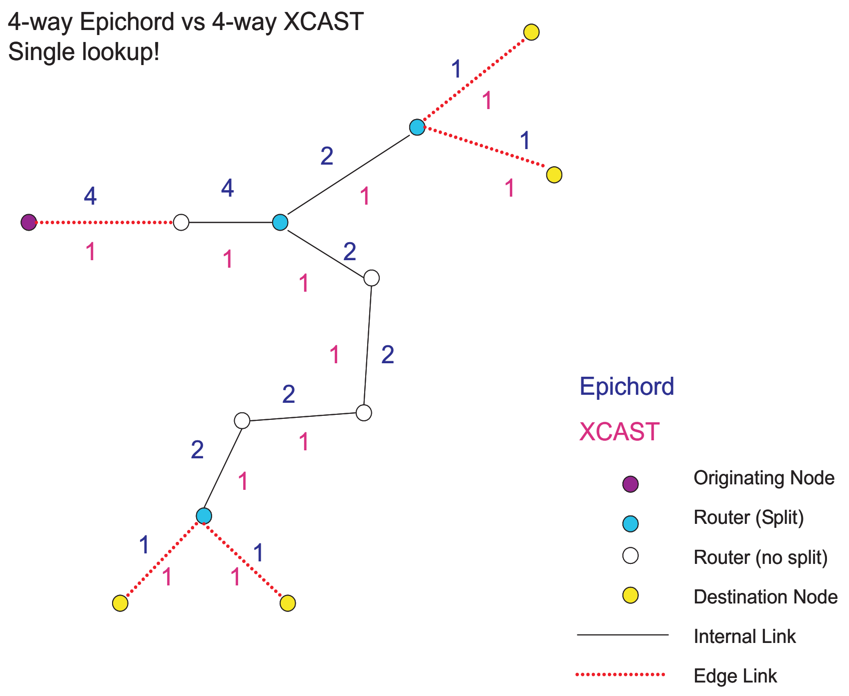


Figure 5.2: Progress of a multi-destination multicast message through a network

Multi-destination multicast routing does not require special state in routers. Thus there is no router state constraint on the number of groups. However there is additional processing overhead at each router for the forwarding algorithm to process the list of addresses. Whereas host-group multicast routers have forwarding state for each group address, for a multi-destination packet with N destinations, there is O(N) work at each router to process the list of addresses and to make a forwarding decision for each destination. Packet duplication work for multi-destination routing is similar to that of host-group multicast.

Multi-destination multicast imposes a maximum group size v. This is restricted by the Maximum Transfer Unit (MTU) of the network. Currently for Ethernet this is 1500 bytes as an upper bound. For IPv4, this gives a maximum size of 127 addresses [74]. Since peers in the overlay maintain routing tables or addresses of other peers, there is no group join overhead when peers are directly reachable in the overlay. Thus the time to create a new multicast group is not a factor. It can therefore be concluded that multi-destination routing is generally applicable to networks where group sizes are small and membership is changeable, e.g. P2P networks. Recently an experimental IP protocol for multi-destination multicast called explicit multicast (XCAST) has been specified [74] and several XCAST testbeds have been deployed. [75] analyse the performance of XCAST combined with host-group multicasting. For brevity in the remainder of this Thesis, XCAST is used to mean only multicast protocol.

5.4 Design Criteria

The typical use of a peer-to-peer overlay is to provide widely available end-to-end network services that would be difficult to deploy inside the network, or to share computing resources among a large set of users. A variety of peer-to-peer overlays have been developed for applications such as file sharing, instant messaging, and telephony. With reference to Figure 4.1, examples of file/content sharing systems used commercially today are the eMule [48] P2P client which works on a Kademlia network (structured), Kazaa [18], Gnutella [30] and BitTorrent [49] (unstructured). Instant messaging and telephony systems, e.g. Skype [76], possibly use P2P technologies though this is unconfirmed due to the proprietary nature of the systems. However, the work of P2P SIP [77] certainly aims to provide these systems using P2P networks.

To explain the potential for using multicast messaging in an overlay, first consider the various types of messages that are possible:

1. A node sends a discovery message to another node during overlay bootstrap.
2. A node announces it is joining or leaving the overlay.
3. A node sends routing tables or routing table excerpts to another node.
4. A node sends node metrics and/or overlay measurements to another node.
5. A node sends a DHT lookup request or response to another node.
6. A node sends an event to one or more subscriber nodes.
7. A node sends its application state to another node, for example, for replication for load distribution and reliability.
8. A node acting as an inter-overlay gateway translates a message from one overlay to another.
9. A node sends a measurement probe, heart-beat or gossip message to another node.

10. A node sends messages which combine or concatenate other types of messages, such as exchanging routing table updates with DHT request or response messages.

11. A node sends messages to create, join, use, manage, and leave using application layer multicast (ALM) tree using an overlay routing algorithm.

12. A node sends an acknowledgement to any of the above messages.

Let P be the set of peers in the overlay during some interval T, where $|P| = n$. Let M be the set of overlay protocol message types for the overlay. $M_t \in M$ is one of the above message types, and m_j is a message instance of a given type M_t, with j as a unique identifier for each message instance in interval T. Define F_i as the set of all combinations of P of sizes i = 2, 3 .. n. Distinguish multicast messages where the destinations all receive the same message type M_t from multiplexed multicast messages where destinations receive multiple possibly overlapping message types (such as probe and lookup messages). In general, the former offer the most efficiency gains since they frequently share message payload. The cases identified later for specific existing overlay designs are single-type multicast messages.

During interval T, the accumulated count of messages sent by all peers in P is A = Σ a_t where each a_t is the count of those messages of type M_t. These messages are sent over network links L. For unicast transmission, the average number of messages per link is then $A/|L|$. The goal is to replace sets of unicast messages to multiple destinations from the same peer with a multicast message to the same destinations, reducing the per link message traffic to $A'/|L|, A' < A$. The number of unicast messages that can be combined depends on the message type, the degree of parallelism of that message type in a given overlay, and their temporal locality.

To identify temporal locality needed for using a multicast message in place of unicast messages, an outgoing message queue is constructed at each peer. Each peer p has a queue Q which has pending messages m_j to send. After adding a message m_j to Q, the peer examines Q and may combine a set u of messages in Q to create a new multicast message m_c to group g_k where m_c contains the contents of the individual u messages, $p \ni g_k$, $|g_k| = |u|$, $g_k \in F_{|u|}$, and k is a unique group identifier. The peer may flush one or more messages from Q, combine other unicast and multicast messages in the queue, or wait for further messages. The peer acts to maintain the maximum queueing delay of any message below a threshhold d_q. Assume peers agree on the rules for combining and extracting unicast messages to and from multicast messages. Assume further that the decision process by which messages are combined considers that the benefit of multicast for network efficiency is proportional to the amount of overlap of the content of the combined unicast messages.

The number of multicast groups used in the overlay in the interval T is then $N_G = |G|$ where $G = \{g_i : \forall$ i$\}$. The maximum group size is $|g_{max}|$ such that $\forall$ g_k :

Table 5.1: Multicast parameters for two O(1) overlays [10]

$|g_k| \leq |g_{max}|$ and $\exists\ g_k : |g_k| = |g_{max}|\}$. The rate of group formation is r = N$_G$/T and the frequency of group use f(g$_k$) = $|m_c|/T$, the number of multicast messages m$_c$ to the group g$_k$ in interval T.

For example, Table 5.1 shows example values for two one-hop overlays which are described earlier in this Thesis. EpiChord (see Section 4.2.8.1) uses p-way lookups for both application-level and maintenance messages, and the number of groups needed during T and the rate of group formation are directly related to the lookup rate r$_L$. Although group size is small, for a high-churn overlay the rate of group formation might be as high as 1 group per second per peer, and the groups have low frequency of re-use due to the random distribution of access in the overlay.

D1HT together with EDRA (see section 4.2.8.2) uses up to (log n)-way messages, and its rate of group formation is directly related to the churn rate. A high churn network is considered to have a mean node lifetime of one hour. The frequency of group use is related to the EDRA event maintenance interval θ, which decreases as the churn rate increases.

Multicast routing offers efficiency and concurrency to overlay designers. It may improve response time for operations in which parallelism locates a shorter path more quickly. Reliability may be affected, since a lost multicast packet effects multiple messages.

5.5 Related Work

A large number of peer-to-peer overlays have been proposed, and overlay design continues to be an active area of research. Lua et al. [78] survey many contemporary overlays but do not discuss multicast-enabled overlays. Aberer et al. [79] present an overlay design space but also do not address multicast-enabled overlays. Identifying and analysing parallelism in overlay messaging has not been generally addressed. Use of an overlay for application layer multicasting (ALM) is a separate and different issue, and existing ALMs use unicast messaging at the network layer.

Kelips [80] is a O(1)-hop overlay which uses an epidemic multicast protocol for exchanging overlay membership and other soft state between nodes. As described in [81], such a protocol consists of two sub-protocols: a multicast data dissemination protocol, and a gossip protocol to exchange message history for reliability purposes. In [81] the multicast data dissemination protocol can be either IP multicast if available or random spanning trees over the multicast group formed by unicast connections. The Kelips epidemic multicast heartbeat protocol, which maintains the soft state in each node, gossips to nodes in its own group and to contacts in other groups. Each node has two multicast groups in a given gossip round. Intra-group targets for gossip are weighted towards neighbors, so intra-group multicast groups are repeated in subsequent rounds until neighbors change. Inter-group targets vary round by round, so inter-group multicast group membership changes round by round at each node. Consequently, Kelips requires in a given gossip round O(n) number multicast groups, and group membership for each group changes each round. The epidemic protocol based on [81] used by Kelips does not scale in terms of router state if IP multicast is used, and has substantial group membership maintenance overhead.

Oh-ishi et al. have considered the use of Protocol Independent Multicast (PIM) [82] in sparse mode (PIM-SM) and source specific mode (PIM-SSM) [83] to reduce message traffic in peer-to-peer systems. Their analysis focuses on using multicast routes between peers in different ISP networks. He and Ammar [75] analyse the performance of XCAST combined with host-group multicasting, where XCAST is used for small groups and host-group multicasting is used for large groups. For XCAST sessions they use a dynamic tunneling mechanism between routers corresponding to XCAST branch points in a given session. Since most routers in a multicast path are non-branching, the XCAST routing processing in each router is significantly reduced. However this mechanism is session-oriented and would not be useful for parallelising overlay operations which use short-lived groups.

5.6 XCAST in 1-hop Overlays

To test the theory that multicasting parallel operations in Overlay networks can achieve message savings according to Chuang-Sirbu, the XCAST multi-destination routing protocol was implemented in the SSFNet network simulator [32]. Many network simulators currently exist and [31] describes the steps taken to decide upon SSFNet as my choice network simulator. After implementing the XCAST multi-destination multicast protocol on SSFNet, two 0(1)-hop algorithms were implemneted; EpiChord [6] and EDRA [7]. EpiChord is described earlier in Section 4.2.8.1 and EDRA in Section 4.2.8.3.

All simulations in this section were carried out using a 10,450 node network on the SSFNet simulation environment. The underlay network consisted of 25 autonomous systems, each containing 13 routers and 405 hosts.

5.6.1 EpiChord

According to the rules described in Section 5.4, the EpiChord algorithm clearly exhibits operations that are inherently parallel. Two such operations in EpiChord which exhibit parallel behaviour are DHT lookups and probe messages. To improve the success rate of lookup messages, EpiChord uses parallel *p-way* lookup requests directed to peers nearest to the node which it believes is responsible for a lookup value. To test multi-destination routing, these *p* parallel unicast requests

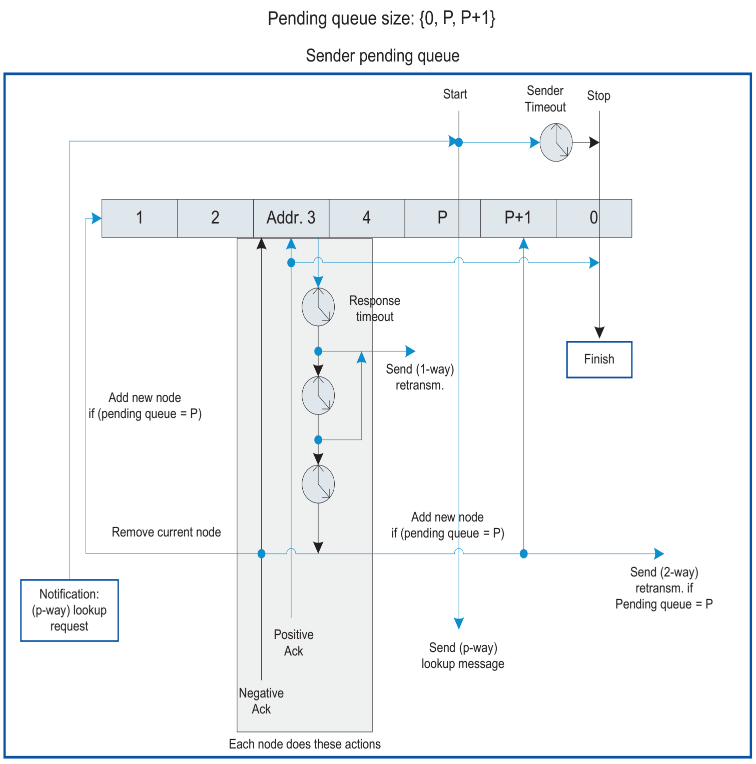

Pending queue size: {0, P, P+1}
Sender pending queue
Start
Sender Timeout
Stop
1
2
Addr. 3
4
P
P+1
0
Response timeout
Send (1-way) retransm.
Finish
Add new node if (pending queue = P)
Remove current node
Add new node if (pending queue = P)
Send (2-way) retransm. if Pending queue = P
Notification: (p-way) lookup request
Positive Ack
Send (p-way) lookup message
Negative Ack
Each node does these actions

Figure 5.3: EpiChord pending request queue states and actions [9]

were replaced with a single XCAST request sent to the p destinations. Figure 5.3 describes the EpiChord lookup and probe mechanism in greater detail.

In EpiChord, for a p-way request, p requests are initially sent and are placed in the pending queue. For any peer, after a first or second timeout, a unicast request is resent to that peer. After the third timeout, or if a peer responds with a NAK, the peer is removed from the pending queue. If the pending queue is of size p-1, two new peers are sent a 2-way request and are added to the pending queue, making its size p+1. Lookup terminates when an ACK is received. One consequence of this algorithm is that the amount of parallelism in practice is not simply p due to the unicast retransmissions and to the 2-way messages used by EpiChord to recover from timeouts and negative responses. Further, the proportion of unicast retransmissions and 2-way messages is dependent on the churn rate and the level of routing table accuracy.

Therefore, if parallel unicast lookup messages and slice refresh messages are replaced with a single multi-destination packet [11], this can reduce the number of lookup messages by up to 32% for edge links and 31% for internal links over 5-way unicast. Alternately, for a given message load, a higher routing table accuracy can be obtained. Note that p-way EpiChord results in parallel message traffic that is on average less than p-way due to invalid routing table entries, retransmissions, and negative acknowledgements [11]. In addition, probe lookups for slice refresh can be aggregated into p-way XCAST messages. That is, during a stabilisation cycle, there could be 10 slices that need lookups. These can all be combined in one XCAST message with 10*p addresses. The following compares unicast EpiChord with XCAST-enabled EpiChord for lookup-intensive and churn-intensive workloads.

5.6.1.1 XCAST enabled EpiChord - Lookup-Intensive workload

In this workload, nodes join the network at a rate of 2 per second and issue on average 2 lookups per second. The overlay network grows until it reaches 1200 EpiChord nodes. XCAST-enabled EpiChord performed equivalently to unicast

EpiChord for average hop count, lookup latency and the success rates of lookups across all degrees of parallelism, thus retaining EpiChords performance advantages over Chord as described in [6].

XCAST significantly reduced message traffic for both overlay maintenance and DHT lookups. This reduction was evaluated for both edge and internal links. An edge link is defined as a duplex link connecting a host and a router, and an internal link as a router-router connection. In this simulated network, there exists 40% more edge links than internal links, and internal links naturally carry more traffic. During the simulation, the network grows from 200 nodes to 1200 nodes. All message traffic was counted on each link in regular fixed sample intervals. Lookup messages are used by EpiChord for three purposes: joins, maintenance and application lookups. Join messages are sent when a new node wishes to join the network and issues a p-way lookup message to its successor node and p-1 predecessor nodes. Maintenance lookup messages are sent when the routing table does not satisfying the required number of nodes per slice. Application lookups are standard lookups for some value in the DHT, issued twice per second per node.

As shown in Figure 5.4, XCAST-enabled EpiChord reduces the number of application lookups per internal link by up to 30% for a 5-way mode versus unicast EpiChord. Similarly, Figure 5.5 shows that using XCAST reduces the number of messages on the edge link by up to 31%. In general, for a request-response

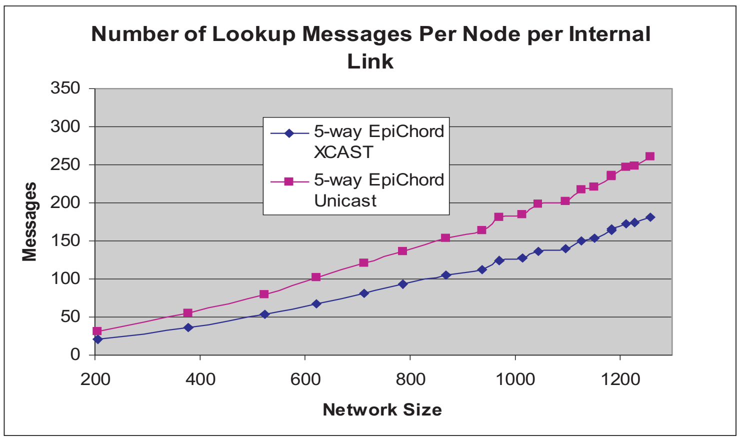


Figure 5.4: Average number of lookup messages counted per internal link for a lookup-intensive workload [10]

protocol, replacing p unicast requests with 1 XCAST packet leads to a savings rate of

$$(p-1)/(2*p) \tag{5.2}$$

for an edge link, assuming all responses are returned as separate unicast packets. For p=5, the expected savings rate is 40%. Three factors account for the reduced savings.

Causes of reduced savings As shown, the saving on the internal link is potentially much greater than that on the edge link. However, these results do not follow such a theory. The following reasons explain such behaviour:

Invalid routing table entries: In a large scale overlay, the routing table is likely to be out-of-date due to churn. Should a lookup message be sent to a

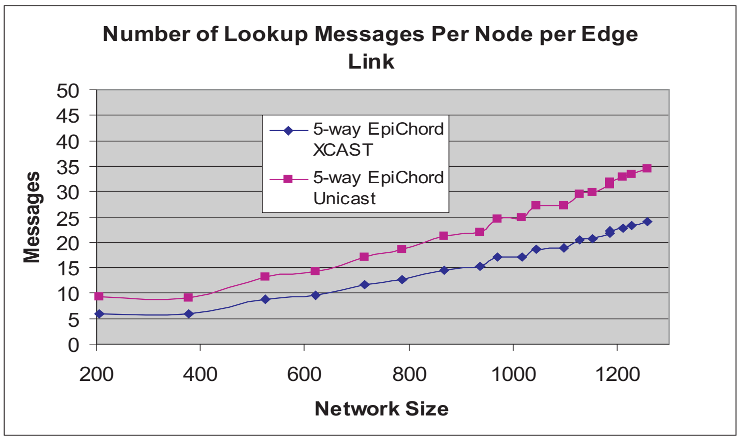


Figure 5.5: Average number of lookup messages counted per edge link for a lookup-intensive workload [10]

node that is offline, the message will never reach the destination edge link (thus improving the edge link saving) and will traverse the internal network until its TTL expires (thus reducing the internal link saving). As shown in [6], the percentage of stale entries in the cache is around 13% for a steady state network of 1200 nodes. This suggests that 13% of all lookups should fail to reach their destination, thus demonstrating the above behaviour. If this figure could be reduced, an improvement in the savings per link could be achieved.

Re-transmissions: When an individual message from a p-way lookup reaches timeout, the node will check the cache to determine if this node has reached timeout enough times to be considered dead. If not, it will retransmit a single UDP lookup message. After re-transmission, if the number of responses the sending node is waiting for x is (> 0 && $\leq p$), then it will issue a (p-x)+1 way lookup to new nodes. Subsequently, for a p-way simulation, many lookups may be issued that demonstrate less than the allowed p degree of parallelism.

Negative responses: When a node receives a lookup message, it will check to see if it is responsible for the requested item. If not, it shall respond with the l most likely nodes to try next. When the originating node receives this negative response, it shall add the l nodes to its routing table and then issues a new lookup message to the (p - x)+1 next best nodes where x is the number of responses it currently awaits. Subsequently, for a 4-way lookup with 3 responses awaited, a node may send a further 2-way lookup. This behaviour explains why a 1-way XCAST enabled EpiChord outperforms unicast EpiChord as 2-way lookup can be sent.

For a 5-way XCAST lookup in a lookup-intensive workload, only 21.5% of packets on the edge link and 18.2% on the internal link are actually XCAST packets. All other packets are UDP. This reduces the possible saving per link. Also, from Figure 5.6, one can see that of these messages, between 55% - 62% were in actual fact 2-way messages. Again, this will reduce the potential message saving per link. Large numbers of XCAST packets carrying two destinations stem from the number of re-transmissions and negative responses. Also, as an XCAST message can carry least two destinations, it is natural that this is the most commonly found message.

This behaviour is further validated in Section 5.6.3.

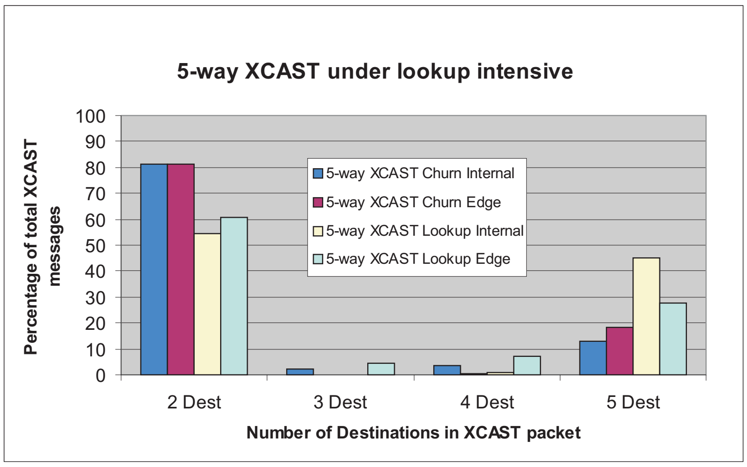


Figure 5.6: Numbers of XCAST messages per link which contain 2, 3, 4, and 5 destination addresses in a 5-way XCAST enabled EpiChord setup with a lookup intensive configuration. [11]

5.6.1.2 XCAST enabled EpiChord - Churn-Intensive workload

For the churn-intensive workload, on average 15 nodes join the overlay network per second and issue on average one lookup every 100 seconds. Average node lifespan is 550 seconds and the network grows in size continuously to 9000 nodes. All measurements in [6] were repeated for the churn-intensive workload, with measurements taken for the lookup messages per link. As before, the XCAST-enabled EpiChord results for the average hop count, lookup latency, failure and timeout rates were consistent with unicast EpiChord.

In a churn-intensive workload, the savings on both the internal and edge links are somewhat reduced. The primary reason for this is that increased churn causes lower routing table accuracy, leading to a higher percentage of unicast retransmissions and 2-way requests, reducing the amount of parallelism compared to the lookup-intensive workload. As shown in Figure 5.7, XCAST-enabled EpiChord shows a reduction of up to 24 % on edge links and in Figure 5.8 a 23% reduction on internal links for a 5-way simulation.

Maintenance lookups show a reduction of 25% on edge links and 24% on internal links. Finally, the reduction for application lookup messages on edge links for join messages is 23% and 22% on internal links, again for a 5-way simulation.

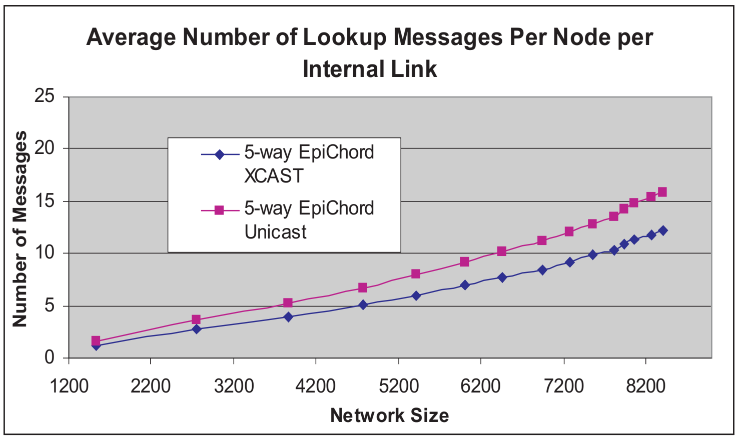


Figure 5.7: Average number of lookup messages per node on internal links for a churn-intensive workload [10]

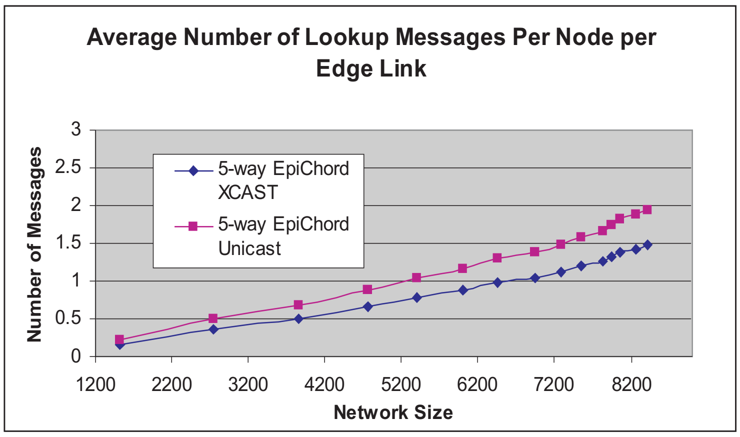


Figure 5.8: Average number of lookup messages per node on edge links for a churn-intensive workload [10]

5.6.2 D1HT EDRA

D1HT [7], described in more detail in Section 4.2.8.2 exhibits many parallel operations, particularly in the Event Detection and Reporting Algorithm (EDRA). In EDRA, each peer gathers information about events (node joins and leaves) over a period of θ and reports this information to its $\log_2 n$ successor nodes, where n is the size of the network. Each of these messages sent to the $\log_2 n$ peers is identical with the exception of the destination and thus is a prime example of the type of operation that is suitable for multi-destination multicast. This process is described next.

Replacing unicast propagation messages with a single multi-destination packet observes propagation message savings of around 35% per link. Note that when p = $\lceil$ $\log_2 n$ $\rceil$, p is the maximum degree of parallelism used for propagation messages. However, the full savings calculated using the Chuang-Sirbu law for size can not be achieved because each node which receives a propagation message will send a propagation message with TTL-1. So for each propagation message with TTL= p, further propagation messages will be created with TTL=$(p \ldots 0)$, i.e. with a lower level of parallelism. Hence the saving is reduced. Next is a description of the simulation environment and results when unicast propagation messages are replaced by multi-destination propagation messages.

Like EpiChord, the EDRA maintenance algorithm was implemented using the SSFNet simulation environment. However, upon testing the implemented EDRA algorithm, it was clear it had some serious design issues. These issues were identified and corrected, as described in detail in Section 6.2.3. The improved version of EDRA will be referred to as EDRA*.

Simulations were carried out on a 10,450 node underlay network consisting of 25 autonomous systems, each containing 13 routers and 405 hosts. The overlay network grows to 1,200 nodes in steady state. Node lifespans are distributed using a heavy-tailed Pareto distribution of α=1, β=1800, and lookups are issued on average every 10 minutes per node. A median lifespan of one hour is achieved using this distribution, and is consistent with past studies as described in [84]. The EDRA* algorithm can be dynamically tuned to calculate the value required to satisfy the maximum percentage of routing failures f. Table 5.2 shows the

Table 5.2: Percentage Saving for XCAST over Unicast EDRA* [10]

Parameters	Edge Link	Internal Link
EDRA*, f=10	33.10%	35.10%
EDRA*, f=30	33.50%	35.90%
EDRA*,f=50	33.60%	37.00%
EDRA*, f=70	33.10%	35.70%

percentage message saving for multi-destination propagation messages on edge and internal links vs. unicast.

Figure 5.9 shows the bandwidth required to satisfy various levels of f. When calculating bandwidth usage, all messages are considered which directly result in routing table updates. That is, for EDRA* lookup messages, propagation messages and all replies are counted. Also, proxy messages for new nodes in the join interval and any probe message which carries a routing table entry are considered. Figure 5.9 shows that unicast EDRA* operates in the range of 35-55 bytes per second per node to maintain levels of f=10-70% . XCAST EDRA* operates using significantly less bandwidth, in the region of 10-18 bytes per second per node. These bandwidth requirements are further shown in Figure 5.10 versus hop counts for both Unicast and XCAST versions of EDRA*.

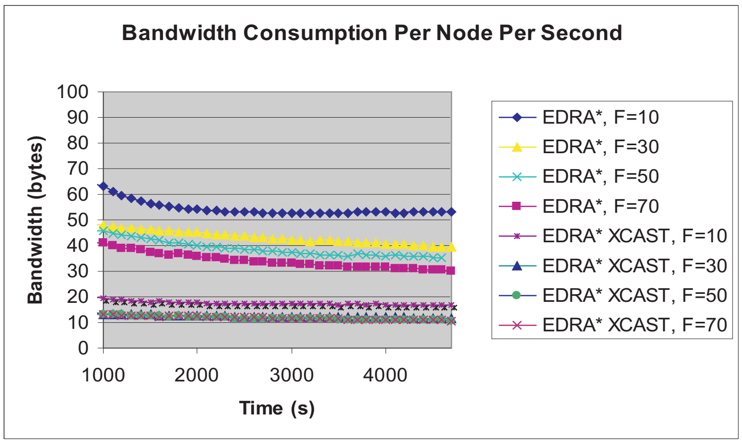


Figure 5.9: Bandwidth consumption for EDRA* Unicast vs. EDRA* XCAST for varying degrees of f [10]

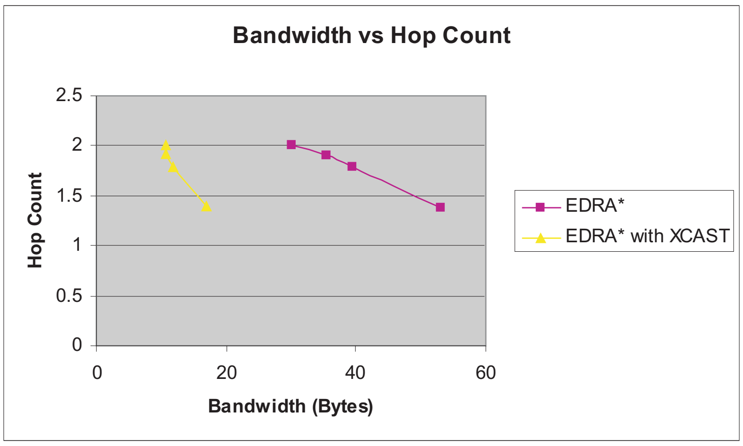


Figure 5.10: Bandwidth vs. Hop Count measurements for XCAST enabled EDRA* vs. Unicast EDRA* and Unicast EpiChord [10]

5.6.3 Validation of Results

As shown for both EpiChord and D1HT EDRA, significant savings can be achieved when applying XCAST to operations which are inherently parallel. These savings have been demonstrated through simulations in networks of up to 9000 overlay nodes. To add extra credence to these results, this section offers a formalisation of the EpiChord retransmission model, and uses a Markov model to evaluate the potential for parallelism in EpiChord. To confirm the savings obtained simulating EpiChord using the SSFNet network simulator, the results of the Markov model are correlated with both these results and the predicted savings of the Chuang-Sirbu scaling law. This work is further described in [9].

5.6.3.1 Estimating the XCAST gain

To quantify multicast efficiency against unicast, the message hop count can be used as a metric. As in [85], this saving can be defined as:

$$\delta = 1 - \frac{\text{average multicast hops}}{\text{average unicast hops}} \tag{5.3}$$

The value of δ is in the range [0,1]. When δ equals 0, the use of XCAST has no advantages over unicast. However, as δ increases towards 1, so does the benefit of using multicast.

[86] reformulates equation (5.3) as:

$$\delta = 1 - \frac{g_N(m)}{f_N(m)} \tag{5.4}$$

m represents the multicast group size, N is the number of nodes in the overlay network, $g_N(m)$ is the average number of hops in the multicast distribution tree and $f_N(m)$ is the average number of hops in the unicast case. $f_N(m) = m.E[H]$, where $E[H]$ is the average number of hops in the physical topology from one overlay source node to a destination overlay node.

5.6.3.2 Multicast Gain without considering re-transmissions

The Chuang-Sirbu Law [67] states that the number of hops in the multicast tree is a function of the multicast group size and a scaling factor k with a value between 0 and 1. Indeed, they approximate $g_N(m)$ by:

$$g_N(m) = E[H] * m^k \text{ where } k = 0.8. \tag{5.5}$$

However, considering the work of Chalmers and Almeroth [70] this can be amended to consider that k is not a fixed value but in actual fact a range.

Thus, equation 5.5 can be redefined as:

$$g_N(m) = E[H] * m^k \text{ where } k = 0.66 < k < 0.8 \tag{5.6}$$

and the multicast metric can be defined as :

$$\delta = 1 - \frac{E[H] * m^{0.66<k<0.8}}{E[H] * m} = 1 - m^{-0.34<k<-0.2} \tag{5.7}$$

where δ returns the value of the savings that multicast offers when compared to unicast transmissions of m lookups. For example, if $m = 5$, the saving obtained by employing multicast rather than unicast is $1-(5)^{-0.34<k<-0.2} \approx 27.5\% - 42.1\%$.

5.6.3.3 Retransmissions in the EpiChord simulation

As described in Section 5.6.1, when an EpiChord node is sending a request message, two queues are employed: the pending queue and the tried queue. The pending queue is the queue of all nodes from which a response is awaited. The tried queue is the queue of nodes that a request has been sent to for this lookup. For a p-way request, p nodes are sent a lookup message, and these p nodes are put into pending and tried queues. Hence initially, both queues hold the same entries. This queue was previously described in Figure 5.3.

The following discussion concentrates on the pending queue. The tried queue simply keeps track of nodes to which a message has been sent to, to avoid contacting nodes repeatedly for a single request.

The setup of the pending queue may change during a request, depending on node timeouts and negative responses received. P is the level of parallelism. As stated above, initially a P-way request is sent to the nodes in the queue. If a node times out, the node is tried a further two times using single UDP messages. After the third timeout, the node is removed from the pending queue. If the queue length equals P, two new nodes are added to the queue and a two-way XCAST request is sent to them. Hence the queue length is now P+1. If the queue length is already P+1, no new message is sent. An example of these cases is shown in Fig 5.11.

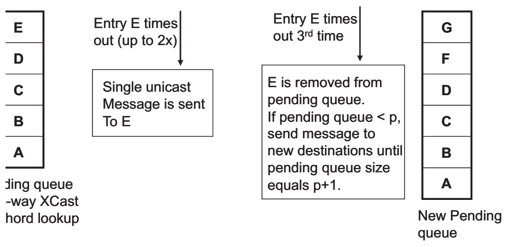

E
D
C
B
A
ding queue
-way XCast
hord lookup
Entry E times out (up to 2x)
Single unicast Message is sent To E
Entry E times out 3rd time
E is removed from pending queue. If pending queue < p, send message to new destinations until pending queue size equals p+1.
G
F
D
C
B
A
New Pending queue

Figure 5.11: Timeout scenario for a 5-way XCAST EpiChord request. [9]

If a node responds with a negative message (does not have the data), it is also removed from the queue, and again depending on the current length of the queue, a new two-way request is sent to new nodes. This process continues until a node responds with the data looked for (a positive response). As above, XCAST is used for this request. An example scenario is shown in Figure 5.12. Thus the behaviour after a negative response and a third node timeout is identical.

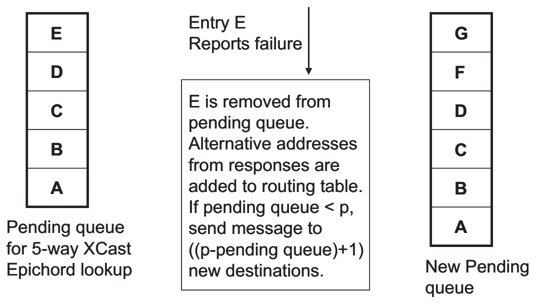

E
D
C
B
A
Pending queue for 5-way XCast Epichord lookup
Entry E Reports failure
E is removed from pending queue. Alternative addresses from responses are added to routing table. If pending queue < p, send message to ((p-pending queue)+1) new destinations.
G
F
D
C
B
A
New Pending queue

Figure 5.12: Failure scenario for a 5-way XCAST EpiChord request. [9]

Retransmissions stop, as soon as a success message from any tried node is received. In the following section it is shown how equation (5.7) must be modified to take retransmissions into account.

5.6.3.4 Multicast Gain when considering re-transmissions

To consider retransmissions, the cost of unicast and the cost of multicast, C_u and C_m, respectively, can be defined with the equations (5.8) and (5.9). The cost of multicast can be derived as the sum of the cost of sending m-way lookups, (m-1)-way lookups, ..., 1-way lookups because the multicast group size varies according to the size of the pending queue.

$$\begin{aligned} C_u &= f_N(m) + f_N(re - transmissions) \\ &= E(H) * m + E(H) * E(X) \end{aligned} \tag{5.8}$$

where X is the number of retransmissions and $E(X)$ is the average number of retransmissions for a request.

$$\begin{aligned} C_m &= g_N(m) + E(m-1) * g_N(m-1) + \\ &\quad \ldots + E(1) * g_N(1) \\ &= E(H) * m^k + E(m-1) * E(H) * (m-1)^k + \\ &\quad \ldots + E(1) * E(H) \end{aligned} \tag{5.9}$$

where $E(m-1)$ is the average number of (m-1)-way retransmissions.

In the simulation of EpiChord, only certain types of messages can be sent: a single p-way request initially, 2-way XCAST messages after a negative response from a tried node or the third timeout of a node, and a single UDP message after a node times out once or twice. Thus equation (5.8) and (5.9) can be simplified as shown in equations (5.10) and (5.11). The Chuang-Sirbu law would suggest a value of 0.8 for k.

$$C_u = E(H) * m + E(2) * 2 * E(H) + E(1) * E(H) \tag{5.10}$$

$$C_m = E(H) * m^k + E(2) * E(H) * 2^k + E(1) * E(H) \tag{5.11}$$

Here E(2) is the expected number of 2-way retransmissions per lookup, and E(1) is the expected number of unicast retransmissions per lookup. Thus only the cost of the initial request plus two-way retransmissions plus UDP retransmissions, until the requesting node has obtained a positive acknowledgment, are considered.

The pending queue size changes depending on the type of response the sender receives. The probabilities of receiving each type of response is known from our simulations, hence the pending queue size can be calculated, and thus the average number of 2-way and 1-way retransmissions the sender issues for each lookup. To do so, the behaviour of the pending queue has been modelled as a Markov chain.

5.6.3.5 Markov Model

For the model it is assumed that a node which sent out requests to other nodes in its pending queue can receive three different types of responses: positive response, negative, and timeout. This is shown in Fig. 5.13.

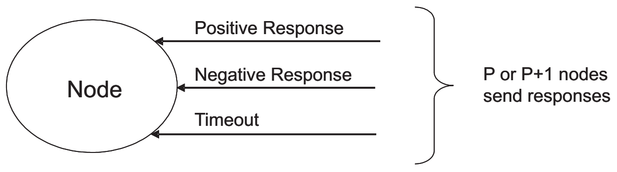


Figure 5.13: Responses received by a node

The probabilities for a positive response (p+), negative response (p-), and timeout (pt) can be calculated as show in equation (5.12). Initially the number of messages as observed in our simulations can be used. However, other values can be inserted, allowing for *what-if* analysis.

$$p+ = \frac{\text{number of positive responses}}{\text{total number of responses}}$$

$$p- = \frac{\text{number of negative responses}}{\text{total number of responses}}$$

$$pt = \frac{\text{number of timeouts}}{\text{total number of responses}} \tag{5.12}$$

Figure 5.14 shows the resulting Markov chain for a request with P=3. In the diagram all transient states are shown, however, the absorbing final state representing the reception of a positive response {0,0,0} has been omitted for clarity. The absorbing state can be reached from any state when a positive response is received.

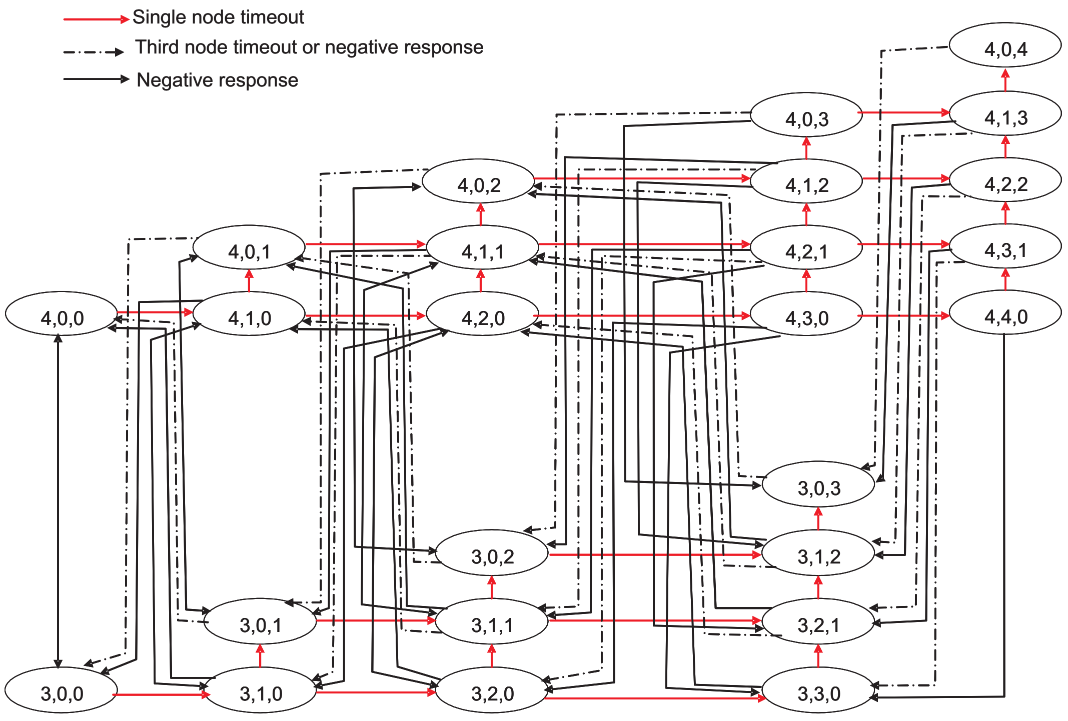


Figure 5.14: Markov Chain Model - State Diagram for the Pending Queue [9]

The name of each state contains three numbers $\{k, i, j\}$: k is the current length of the queue, i is the number nodes in the queue which have timed out once, and j is the number of nodes in the queue which have timed out twice. The initial state of the queue is {P,0,0}, where the pending queue has a size P and 0 timeouts have occur. The retransmission process ends when the sender receives a positive response, thus the pending queue state becomes {0,0,0} (not shown).

When designing the model, three assumptions have been made to achieve a good balance between accuracy and complexity of the model.

Assumption 1: An embedded discrete (homogeneous) time Markov Chain model was used to describe the pending queue operation. This means the transition probabilities do not change over time (homogeneous) and the model is expressed in terms of transition probabilities rather than rates. The time the queue is in each state is ignored. The probabilities of making transitions from each state to all other states when a transition occurs are the only concerns.

Assumption 2: A transition occurs after one and only one response is received by the queue. Hence transitions consider only the change of state of any one node in the pending queue.

Assumption 3: It is equally likely for a node to timeout once, twice, or three times. The probability of timing out or to receive a negative response is independent of the state.

The transitions between states indicate the reception of a negative response or a node timeout. Taking a closer look at the model, it can be seen that it is split into two halves. The bottom half describes the states for the queue length = P (3 in this example), whereas the top half includes the states for queue length = P+1 (4 in this example). Moving from the start state to the right, increasing number of first node timeouts are modeled (up to P). Moving up line by line until the middle of the diagram, the second timeouts of nodes are modeled (up to P). The same is repeated in the top half of the model, however for up to P+1 nodes. The two halves are joined by transitions indicating third node timeouts and negative responses. Negative responses and the third timeout of a node trigger the same transition in the diagram, hence arrows indicating these are combined for brevity. The transition probabilities of the Markov model are shown separately in Table 5.3.

State	# Timeouts	Prob. Type	Event	Next State	Prob. Transition
$\{k,i,j\}$	0	$(k-i-j)/k$	neg. response	$\{\hat{k},i,j\}$	$(k-i-j)/k \times p_-$
$0 \le i+j < k$					
$1 \le i,\ 1 \le j$					
	0	$(k-i-j)/k$	timeout	$\{k,i+1,j\}$	$(k-i-j)/k \times p_t$
	1	i/k	neg. response	$\{\hat{k},i-1,j\}$	$(i/k) \times p_-$
	1	i/k	timeout	$\{k,i-1,j+1\}$	$(i/k) \times p_t$
	2	j/k	neg. response/timeout	$\{\hat{k},i,j-1\}$	$(j/k) \times (p_- + p_t)$
$\{k,i,j\}$	1	i/k	neg. response	$\{\hat{k},i-1,j\}$	$(i/k) \times p_-$
$0 \le i+j = k$	1	i/k	timeout	$\{k,i-1,j+1\}$	$(i/k) \times p_t$
$1 \le i,\ 1 \le j$	2	j/k	neg. response/timeout	$\{\hat{k},i,j-1\}$	$(j/k) \times (p_- + p_t)$
$\{k,i,j\}$	1	1	neg. response	$\{\hat{k},i-1,j\}$	p_-
$i = k$	1	1	timeout	$\{k,i-1,j+1\}$	p_t
$\{k,i,j\}$	2	1	neg. response/ timeout	$\{\hat{k},i,j-1\}$	$(p_- + p_t)$
$j = k$					
$\{k,i,j\}$	0	$(k-i)/k$	neg. response	$\{\hat{k},i,j\}$	$(k-i)/k \times p_-$
$0 \le i+j < k$					
$1 \le i,\ j = 0$	0	$(k-i)/k$	timeout	$\{k,i+1,j\}$	$(k-i)/k \times p_t$
	1	i/k	neg. response	$\{\hat{k},i-1,j\}$	$(i/k) \times p_-$
	1	i/k	timeout	$\{k,i-1,j+1\}$	$(i/k) \times p_t$
$\{k,i,j\}$	0	$(k-j)/k$	neg. response	$\{\hat{k},i,j\}$	$(k-j)/k \times p_-$
$0 \le i+j < k$					
$i = 0,\ 1 \le j$	0	$(k-j)/k$	timeout	$\{k,i+1,j\}$	$(k-j)/k \times p_t$
	2	j/k	neg. response/timeout	$\{\hat{k},i,j-1\}$	$(j/k) \times (p_- + p_t)$
$\{k,i,j\}$	0	1	neg. response	$\{\hat{k},i,j\}$	p_-
$0 \le i+j < k$					
$i = 0,\ j = 0$	0	1	timeout	$\{k,i+1,j\}$	p_t

Table 5.3: Transition Probabilities with $\hat{k} = P + 1$ if k = P and P if k = P + 1

The number of transient states n in the Markov Chain can be calculated using equation (5.13). The first half calculates the number of states for queue length equal to P, whereas the second half calculates the number of states when the pending queue size equals P+1.

$$n = (P+1) * (P+2) * 0.5 + (P+2) * (P+3) * 0.5 \tag{5.13}$$

The probabilities for timeouts and negative responses have been obtained using the SSFNet XCAST-EpiChord simulator [11].

The Markov state diagram above is then used to calculate the expected number of retransmissions per lookup. For this, the transition matrix needs to be derived. Let T be the transition matrix. T is composed of the transient states and the one absorbing state.

$$T = \begin{pmatrix} Q & R \\ 0 & I \end{pmatrix} \tag{5.14}$$

Here the transition matrix for the transient states, Q, is of size $(n \times n)$. n as calculated above is the number of transient states in the model. The absorption matrix R is of size $(n \times 1)$, the matrix of zeros O is of size $(1 \times n)$, and finally the Identity matrix I is of size (1×1). So in total, T is of size $(n + 1 \times n + 1)$.

Q can now be used to calculate the fundamental matrix N. The entry N(i,j) shows the expected number of visits to transient state j before absorption, starting in state i.

N depends only on the square sub-transition matrix Q. Hence the total number of visits is the sum of the expected number of visits on the different steps. The expected number of visits to j on the n-th step, starting i is Q(i,j). Or written differently: $N = I + Q + Q^2 + Q^3 + Q^4 + \ldots$. Here I is the Identity matrix.

Now N can be calculated using the relatively simple Equation (5.15). If both the left and the right side are multiplied by $(I - Q)$, this results in $N * (I - Q) = I$. Hence N can be calculated as (inv is the matrix inverse function):

$$N = inv(I - Q) \tag{5.15}$$

Using N, the expected number of 2-way retransmissions can be derived. It represents the sum of N(i,j) when the queue state goes from P to P+1. The expected number of unicast retransmissions can be calculated in a similar fashion. It is the sum of N(i,j) when a first or second timeout occurs in queue state P and P+1. Matlab was used to calculate the fundamental matrix and the values of the expected number of retransmissions. These can then be used in (5.10) and (5.11). For the following experimentation k=0.8 according to the Chuang Sirbu law has been assumed. With these two values the saving of using XCAST can be calculated using equation (5.16).

$$saving = 1 - \frac{C_m}{C_u} \tag{5.16}$$

5.6.4 Experimentation

XCAST reduces the number of messages sent per lookup and hence the number of messages per link. This achieves a bandwidth saving on each link used by the messages.

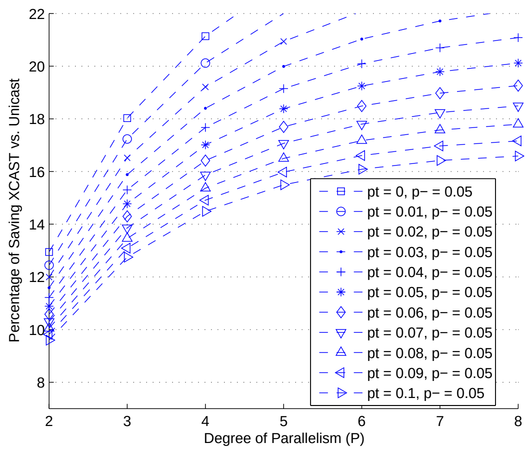


Figure 5.15: Saving achieved using XCAST for increasing levels of node timeouts for different degrees of parallelism (lookup-intensive). [9]

The following figures illustrate the saving that can be achieved for varying probabilities of receiving a negative acknowledgment (p-) or timeout response (pt). The values chosen were guided by the values from our EpiChord simulations. Two sets of graphs are presented, one with p- and pt modelled on a lookup-intensive behaviour in the overlay (nodes join the network at a rate of 2 per second and issue on average 2 lookups per second), and the other on a churn-intensive behaviour (15 nodes join the overlay network per second and issue one lookup every 10 seconds). This setup follows the original EpiChord simulations by MIT.

Figure 5.15 shows the XCAST savings which can be achieved for varying levels of node timeouts at different degrees of parallelism. The probability of negative responses was taken from our simulations under a lookup-intensive workload. As can be seen the same saving of about 18% can be achieved by a 3-way lookup with p-=0.05 and no timeouts, and a 6-way request with the same value for p- and pt=0.07, or a 8-way request with the same p- and pt=0.08.

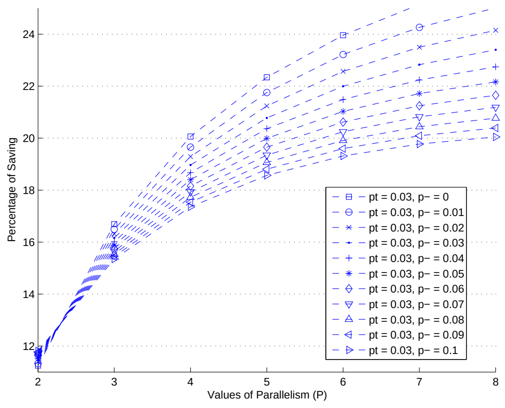


Figure 5.16: Saving achieved using XCAST for increasing levels of negative responses for different degrees of parallelism (lookup-intensive). [9]

Figure 5.16 shows the XCAST savings which can be achieved for varying levels of negative responses at different degrees of parallelism. The probability of timeouts was taken from our simulations under a lookup-intensive workload.

Figure 5.17 shows the XCAST savings which can be achieved for varying levels of node timeouts at different degrees of parallelism. Here the probability of negative responses was taken from our simulations under a high churn workload.

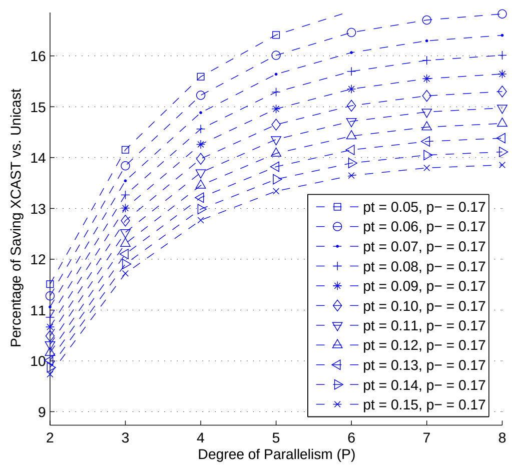


Figure 5.17: Saving achieved using XCAST for increasing levels of node timeouts for different degrees of parallelism (churn workload) [9]

Figure 5.18 shows the XCAST savings which can be achieved for varying levels of negative responses at different degrees of parallelism. The probability of timeouts was taken from our simulations under a high churn workload.

The figures show that for increasing numbers of node timeouts and negative responses (and hence retransmissions) the XCAST saving decreases. However, the saving decreases faster for increasing node timeouts than for increasing negative responses. This is because node timeouts result in UDP messages, whereas negative responses result in 2-way XCAST messages. In fact for 2-way EpiChord, the saving goes up slighty when considering negative responses. This is because the initial request is also only sent 2-way.

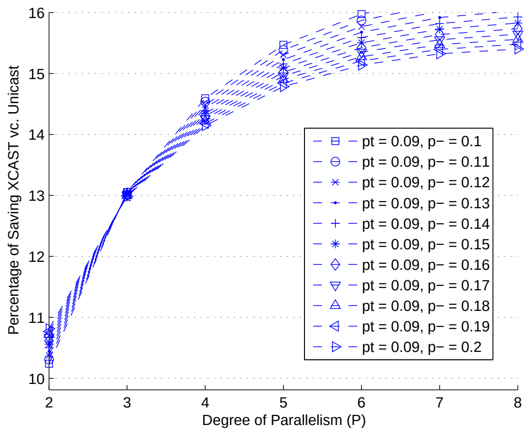


Figure 5.18: Saving achieved using XCAST for increasing levels of negative responses for different degrees of parallelism (churn workload) [9]

5.6.5 Comparison: analytical and simulation results

5.6.5.1 Discussion of the number of retransmissions

As stated above, the probability of having a transition due to a negative response or timeout in the Markov chain model have been calculated using results from the SSFNet simulations. The simulations returned the average number of negative and timeout responses before absorption per lookup. By dividing these values by the number of states in the Markov chain, the probability of having a negative response or a timeout can be calculated. Indeed, starting from any state in the pending queue, the probability that any node replies with a negative response is equally likely to occur. Similarly, the probability that any node times out, when the pending queue is in any state, is equally likely to occur.

In the simulations, values of 2.54 for the expected number of negative responses and 1.77 for the expected number of timeouts per request were obtained for P = 5 (degree of parallelism). Using equation (5.13) the number of transient states in the Markov chain model for P=5 can be calculated as 49. Thus, the probability of receiving a negative response starting from any state is $\frac{2.54}{49} = 0.052$, and the probability of timeout starting from any state is $\frac{1.77}{49} = 0.036$.

The results from the Markov chain model for P=5 return values of 1.35 for the expected number of 2-way messages and 1.18 for the expected number of unicast retransmissions. The results obtained by simulation were 1.22 for the expected number of 2-way messages and 1.20 for the expected number of unicast retransmissions. Thus the model yields results quite close to the simulation values. Table 5.4 shows the values for P = 3, 4, and 5 for a simulated network of 1200 nodes under a lookup-intensive workload (join rate of 2 nodes per second and two lookups per second).

P	Neg Resp. per lookup	Timeouts per lookup	Xcast (model)	Xcast (simul)	unicast (model)	unicast (simul)
3	1.44	1.30	0.77	0.75	0.87	0.90
4	1.98	1.54	1.06	1.00	1.02	1.05
5	2.54	1.77	1.35	1.22	1.18	1.20

Table 5.4: Lookup-Intensive Results for Parallelism 3, 4 and 5 [9]

Table 5.5 show the comparison of the results obtained from the Markov model vs. simulation results for a network of 9000 nodes under high churn (join rate of 15 nodes per second and one lookup every ten seconds).

P	Neg Resp. per lookup	Timeouts per lookup	Xcast (model)	Xcast (simul)	unicast (model)	unicast (simul)
3	6.10	3.16	3.19	3.05	2.20	2.22
4	7.27	3.67	3.81	3.45	2.52	2.60
5	8.49	4.23	4.49	3.92	2.88	3.03

Table 5.5: Churn-Intensive Results for Parallelism 3, 4 and 5

The slight deviation between the model results and simulation results can be explained by the three assumptions discussed earlier in Section 5.6.3.5. The main reason for this deviation is the use of an embedded discrete time Markov Chain to describe the pending queue operation. This means that over time, the transition probabilities do not change, unlike a real network. The time the queue spends in each state is ignored.

A more accurate model could be achieved using a transition rate matrix, rather than a transition matrix. By doing so, transitions probabilities are replaced by transitional intensities that capture the time dependency between the reception of responses and the network state (e.g. a timeout occurs because a node has disappeared or because of congestion). In the current model, the probabilities are modeled on the average length of a request, i.e. the average time until absorption. However, taking into account transitional intensities to model probabilities will make the model more accurate. The three probabilities for the transitions pt, $p-$, and $p+$ would change with an increasing numbers of timeouts or negative responses. For instance, if many timeouts are encountered, the probability for $p-$ will decrease. Such a model would not only capture single requests, but also many requests. For this a new transition indicating a new request, from the state $0, 0, 0$ (absorbing state) to $P, 0, 0$ (initial state for a request), needs to be included. This would change the Markov Chain to be ergodic.

5.6.5.2 Comparison of the achieved Multicast Saving

The simulation model used in this Thesis uses a network model of the American academic network identical to that used for the original MIT EpiChord simulations. Clearly, this is not a random network and simulation results show a saving of 29% for a lookup-intensive workload and 25% for a churn-intensive workload for $m = 5$. These results indicate a savings factor k of just below 0.7. This matches the results cited by the Chuang-Sirbu model of $0.66 < k < 0.8$. To verify this we used varying values for k in the Markov model.

Figure 5.19 shows the saving for EpiChord without any retransmissions (solid lines), with retransmissions as expected under a lookup-intensive workload (dotted-dashed lines), and with retransmissions under a churn-intensive workload (dashed lines), for varying values of k. The figure shows that for $m = 5$, $pt = 0.036$, $p- = 0.052$ (lookup-intensive), and $k = 0.7$ the saving is about 27%. For $m = 5$

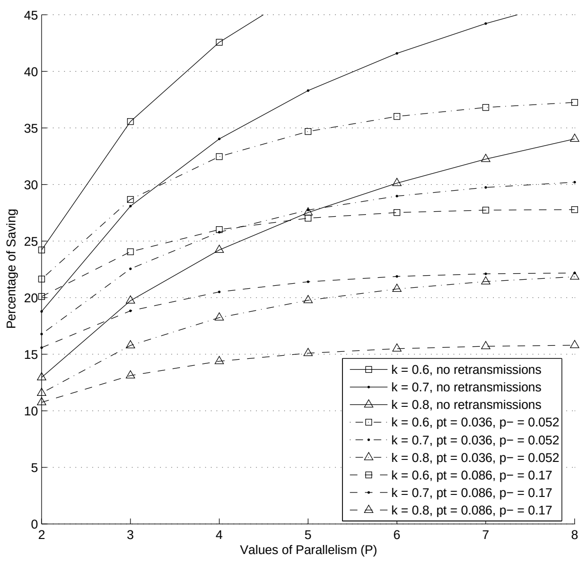


Figure 5.19: Saving achieved using XCAST for varying factor k. [9]

and $pt = 0.086$ and $p- = 0.17$ (churn intensive), and $k = 0.7$ the saving is about 22%. This matches our simulation results and proves that significant savings can be achieved overlay operations using XCAST.

5.6.6 XCAST Deployment

As previously stated, if only a subset of all routers are multicast-enabled, these routers forward multicast packets to other multicast routers using tunnels through unicast routers. Should this be the case, multicast savings would naturally be reduced. At the time of writing, multicast deployment in the Internet is slow. There are a number of reasons why this is so, e.g. Internet Service Providers (ISPs) are reluctant to introduce measures to reduce bandwidth consumption when they charge per kilobyte of bandwidth used. For XCAST to be accepted and deployed wide-scale, there not only has to be interesting usage scenarios such as parallelising P2P overlays, but deployment also has to be in the best interests of consumers, ISPs and network administrators. Thus, there is no guarantee that XCAST will ever be fully deployed in the Internet.

5.7 Summary

This Chapter has presented multi-destination multicast and has formulated a design criteria for its use in P2P networks. The designs of two 1-hop P2P algorithms were altered to utilise multi-destination multicast and these were successfully simulated in a network simulator. Bandwidth savings of $> 30\%$ were shown to be achievable and results have been confirmed using a Markov model.

Chapter 6

Active vs. Opportunistic Maintenance

In the previous chapter, multi-destination multicast was implemented using two 1-hop P2P overlays. This chapter compares the two techniques employed by these algorithms for maintaining a node's routing table: active stabilisation and opportunistic maintenance.

6.1 Introduction

Although multi-destination multicast is a viable and worthwhile solution to inefficiency in P2P networks, wide scale deployment may never actually occur. Therefore, other methods to address efficiency in P2P overlays are required. Many multi-hop peer-to-peer structured overlays have been proposed for peer-to-peer applications which are characterised by O(log N) hop count [78]. Because each overlay hop is routed in the underlay in potentially many native hops, multi-hop overlays have a poor latency characteristic for connecting large numbers of peers. Consequently several systems have been developed to trade-off latency for larger routing tables. However these designs lead to increased network traffic for managing the larger routing tables. Thus efficient overlay maintenance in O(1)-hop (hereafter, one-hop) overlays is an important research question. Two techniques have emerged [87] for maintaining routing tables in one-hop overlays: active stabilisation where peers use fixed rates of communication to maintain a target routing table accuracy, and opportunistic maintenance where routing table updates depend on lookup load and available bandwidth.

An example active stabilisation algorithm is EDRA (Event Detection and Reporting Algorithm) used in the D1HT one-hop overlay [7] and described earlier in 4.2.8.2. EDRA has been proposed to give reasonable message rate for high levels of routing table accuracy. For example, D1HT has up to an order of magnitude lower maintenance bandwidth usage compared to OneHop [88], another active stabilisation one-hop overlay. Examples of opportunistic overlay maintenance include EpiChord [6] and Accordion [89].

This chapter evaluates active stabilisation and opportunistic maintenance to compare the cost-performance tradeoffs between these techniques and shows that active stabilisation can achieve lower hop-count compared to opportunistic methods, whereas opportunistic maintenance is more bandwidth efficient for lower levels of routing table accuracy. This behaviour indicates that an adaptive combination of the two techniques can provide a broader operating range than each individual technique provides, and offers a basis for a variable-hop design called Chameleon which is described later.

6.2 Active stabilisation : D1HT EDRA

D1HT EDRA, described in detail in section 4.2.8.2 is a one-hop P2P overlay which claims to offer 99% routing table accuracy to member peers. In Section 5.6.2, we showed through our simulated implementation of EDRA that using multi-destination multicast for parallel maintenance operations can achieve significant bandwidth savings. However, whilst implementing the EDRA algorithm defined in [7], we found several significant flaws in the design. Before evaluating EDRA's true performance, we addressed each of the original design flaws and provided improvements. These are described next:

6.2.1 EDRA Design Flaws

An error in a peer routing table leads in general to propagation of errors in other peers routing tables. We identify five cases in which EDRA as currently specified can lead to errors:

1. Propagation delay for routing tables to be updated when a node joins or leaves the overlay. We call this the join interval.
2. Different path lengths in EDRA event propagation trees, which can lead to peer routing tables being out of sync
3. Handling of unacknowledged events
4. Handling of duplicate events
5. Masking of concurrent events

These cases are described next.

6.2.1.1 Explicit Join Interval

Using the notation described in Section 4.2.8.3, the time to propagate an event around an EDRA ring of size n nodes is $\theta.\rho$ where $\rho = \lceil log_2 n \rceil$. For $n = 1M$ nodes, $\rho = 20$, so the time to propagate a join event is up to 20 event intervals. After locating its successor node p$_{succ}$, the joining node p obtains a list of other peers from p$_{succ}$ and obtains the routing table from the closest such peer p$_{RT}$ as measured in round-trip time. This routing table will be missing node join and leave events that occurred in the last $\theta.\rho$ interval and that have not yet reached p$_{RT}$. If the average session length of a peer is S$_{avg}$, then the event rate is $r = 2.n/S_{avg}$. At any time, on average there will be $(\theta.\rho.n/S_{avg})$ join events and

$(\theta.\rho.n/S_{avg})$ leave events that have not yet propagated to p_{RT}. Further, because it takes $\theta.\rho$ time for other nodes to update their routing table to include the new node p, p's predecessor nodes will not send it event messages until the $\theta.\rho$ interval is complete (these nodes are within one message hop of p$_{RT}$). Thus p's initial routing table will be missing events from a time window $\theta.\rho$ before joining, and up to $\theta.\rho$ after p has joined.

6.2.1.2 Different Event Propagation Path Lengths

For a single event detected by peer p, EDRA propagation in stretch (p, x) leads to delay differences up to $log_2(x) - 1$ hops or time $(log_2(x) - 1)$ which can lead to peer routing tables being out of sync. For example in Figure 6.1, "A" marks two nodes in the stretch $(p, 16)$ where adjacent nodes have a delay difference of three intervals, and "B" marks two nodes in the stretch$(p, 32)$ where adjacent nodes have delay difference of four invervals. This means that of the adjacent nodes mentioned in, e.g. "A", one will receive a propagated update message in θ interval one, and its neighbour at interval 4. For the three intermediate intervals, their routing tables are out of sync. Thus, for a single event, routing tables at adjacent nodes in an overlay of size n can be inconsistent for an interval of up to $(log_2(n) - 1)$. During this interval, other events propagated by these adjacent nodes can be incorrectly propagated, leading to downstream routing table errors which can lead to duplicate events arriving at a peer.

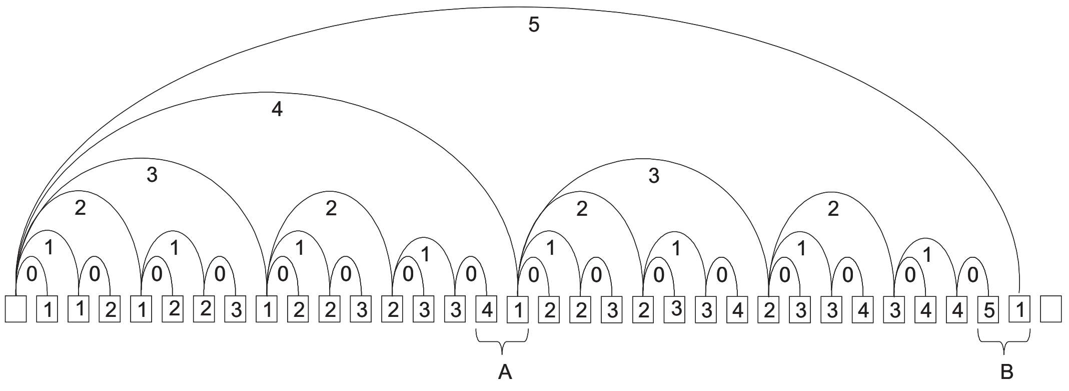


Figure 6.1: Propagation delay for a single event for different paths. Adjacent nodes at A receive messages at intervals 4 and 1 while adjacent nodes B are intervals 5 and 1. [12]

6.2.1.3 Forwarding of un-acknowledged events

During each θ interval, each peer receives up to $log(n)$ messages containing events. According to EDRA rule 2, each event received with TTL=t is acknowledged with a TTL=t response. However D1HT does not prescribe the peer behaviour if an event is not acknowledged. One purpose of the acknowledgement is to trigger the retransmission of an event message if the message is lost due to a link error or network congestion. However, it is also possible the destination peer p_D has left the overlay and the event has not yet propagated to the reporting peer p_R. Here the successors of peer p_D will also not receive this event.

6.2.1.4 Handling duplicate events

As described in 6.2.1.2, routing table errors manifest themselves in EDRA in duplicate event messages being received at a peer. EDRA does not define any mechanism for handling duplicate events. Duplicate events are an indicator that a successor or predecessor node has not received an event and consequently will not have the correct successor path.

6.2.1.5 Masking of concurrent adjacent events

If two adjacent peers leave the overlay simultaneously, the successor peer leaves the overlay before it can distribute the leave event of its predecessor. Likewise, simultaneous joins mask the predecessors join.

6.2.2 EDRA Stability

We implemented the EDRA algorithm according to the specification described in [7]. To evaluate the performance of EDRA, we measured the accuracy of each nodes routing table in a simulated steady state network of 1,200 nodes. By accuracy, we mean the percentage of nodes in the overlay that are present in a node's routing table. This does not consider the percentage of stale entries in the routing table. Monnerat et al analytically show that the EDRA maintenance algorithm can maintain routing table accuracies of 99% when the acceptable maximum fraction of routing failures $f = 1\%$. EDRA can be tuned dynamically to calculate the rate of event propagation θ to satisfy the given value for f.

Figure 6.2 shows EDRA accuracy over time compared to EDRA* (modified EDRA) where $f = 1\%$. EDRA fails to achieve the levels of accuracy to satisfy $f = 1\%$ and accuracy degrades over time to under 70%. With this performance, EDRA is not able to support a one-hop overlay because it has no mechanism to recover from routing table degradation. The causes of this routing table degradation were described earlier in Section 6.2.1. Further, EDRA defines no recovery mechanism to correct routing table errors even if a peer determines that its routing table accuracy is below a given threshold.

The modifications to EDRA, called EDRA* here, to correct these problems cause the routing table accuracy to reach values close to $f = 1\%$, as shown in Figure 6.2. The corrections to EDRA cause it to be virtually stable, though slight degradation of the accuracy is still present. This means that some join and leave events are still being missed. When routing table accuracy is high, it is likely that single routing table errors occur. Some potential solutions to this problem are described in Section 6.2.3.1. The modifications to EDRA cause a modest increase in the maintenance traffic. We do not discuss the bandwidth requirements of EDRA because it is incidental due to its instability. Next the improvements made to the EDRA algorithm are described.

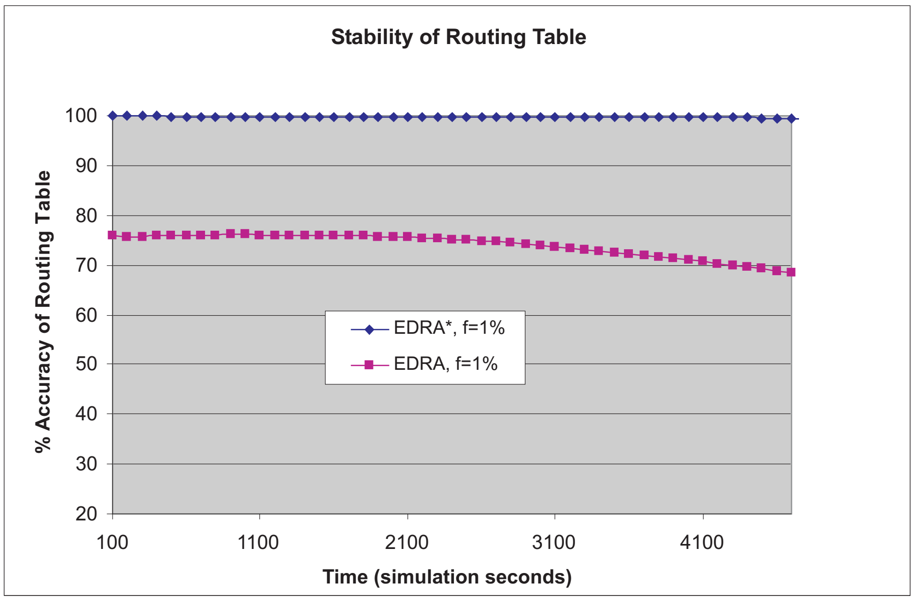


Figure 6.2: EDRA accuracy over time compared to EDRA*, where $f = 1\%$ [12]

6.2.3 Corrections to EDRA

The following corrections were made to the EDRA algorithm.

Explicit join interval: To fix the routing table errors due to the update propagation delay when a node joins the overlay, we define a join interval $\theta.\rho$ in which p_{RT} forwards events it receives after sending its routing table to ρ. If p_{RT} leaves the ring before the join interval is complete, then ρ selects a new p_{RT} to use as its join proxy.

Forwarding of unacknowledged events: When propagating an event, a reporting peer p_R may not get an acknowledgment from the destination peer p_D, which it selected from its propagation path. This may be because p_R

may not have yet received a leave event for peer p_D. In this case, peer p_D's successors will also not receive this event message. To ensure the event is properly propagated, peer p_R can proxy for peer p_D by forwarding the event with TTL=t-1 to those peers that would receive the event directly from p_D if it were still in the ring. It can continue to proxy events during an interval: the maximum time for a leave event for p_D to propagate to it. At the same time it should update its routing table by removing the entry for p_D.

Handling of duplicate events: The peer receiving the duplicate message can send the event with the same TTL to its predecessor as a recovery mechanism. The event should be labeled as "recovery from duplicate message". A peer seeing the event for the first time can process it for forwarding as usual, but not to its successor who sent the message. Otherwise the peer can ignore the event. Forwarding to immediate successor and predecessor peers corrects single routing table errors. It is probable that single routing table errors occur when routing table accuracy is high. A peer can dynamically evaluate its routing table accuracy according to the number of lookup timeouts it receives. If routing table accuracy indicates many routing table errors, then forwarding to immediate successor and predecessor is only a partial solution. Instead of forwarding to a larger neighborhood of peers, we use lookup traffic to exchange routing table updates between peers. This is described later.

Concurrent adjacent events: The probability of concurrent events (join-join, join-leave, leave-join) at adjacent nodes is low. To prevent concurrent events from being missed, when a new node p joins the overlay, it contacts not only its successor node p_{succ}, who is responsible for reporting the new join events, but also its predecessor p_{pred}. Subsequently, when the predecessor of p is notified of p's existence by its own predecessors, it is required to notify p. Thus, we can require that if p is not notified about its own join event in period $\theta.\rho$ it should initiate the reporting of its own join by itself.

6.2.3.1 Additional Improvements to EDRA

To improve the stability of the EDRA* algorithm, the following improvements were implemented.

Maintenance concatenated in lookups: The EDRA algorithm further augmented by carrying routing table updates in lookup requests and responses. That is, any events received in the last θ interval are included in DHT requests and responses. Additionally, we prescribe that when a node receives any message from any other node (maintenance, lookup and join) it tries to add that node to its routing table.

Routing table recovery using lookups: The extensions to EDRA* described above are preventive. While significantly reducing the routing table error rate compared to EDRA, any missed event not prevented by these extensions will propagate as more routing table errors over time. Thus, like EpiChord, background lookup traffic is added to recover from routing table errors. But, unlike EpiChord, this lookup traffic is randomly distributed in the overlay.

Because all events are propagated to all nodes, it is a reasonable assumption that if one node misses an event, a significant number of other nodes will also miss it. Consequently, if a node leave event is missed this could result in this node remaining in a large number of other nodes routing table indefinitely. Currently, EDRA has only one method to remove stale entries from its routing table. If a lookup to a particular node fails, it will remove that node from its routing table. To reduce the number of stale entries in a routing table, this Thesis proposes, but does not implement, the following solution.

Stale Entry Detection: To improve accuracy of a routing table, nodes can calculate the average time fellow nodes are alive in the network. This can be achieved by recording the time that each node is added and removed to/from its routing table and calculating the average lifespan. Using this knowledge, a node can estimate when a node in their routing table is likely to die. When this period of time expires, a heartbeat message can be sent to that node. If it responds, its timer can be reset. To avoid nodes being bombarded with heartbeat messages when they have been in a network for the average node lifespan, only nodes in the immediate vicinity of the suspected dead node should send a heartbeat. This can be parameterised. If a response to a heartbeat is not received, an EDRA leave event should be initiated for that node. Subsequently, all other nodes can then remove it from their routing table.

6.2.4 Opportunistic Maintenance: EpiChord

The EpiChord maintenance algorithm is described in detail in Section 4.2.8.1. One of the opportunistic aspects of the EpiChord maintenance algorithm is adding new entries to a peer's routing table when a request comes from an unknown peer, and removing entries which are considered dead. In addition to this, if the churn rate is sufficiently high compared to the rate at which lookups add new entries to the routing table, peers can send probe messages to segments of the address space called slices. Slices are organised in exponentially increasing size as the address range moves away from the current peer's position. This leads to a concentration of routing table entries around the peer, which improves convergence of routing. Finally, nodes which receive a lookup request respond either with a positive response or with entries from their routing table that are closer to the ID. These responses are used not only to locate the lookup entry but to update the requesting node's routing table.

6.2.5 Comparison of EDRA* and EpiChord

We implemented the D1HT EDRA maintenance algorithm in the same simulation environment as EpiChord for comparison using an equivalent topology and workloads.

6.2.5.1 Simulation Environment

Simulations were carried out on a 10,450 node underlay network using the SSFNet network simulation tool. The topology of the underlay is identical to previously reported studies of EpiChord [90]. The overlay network size was 1,200 nodes. Node lifespans were distributed using a heavy-tailed Pareto distribution with $\alpha = 1$, $\beta = 1800$ and lookups were issued on average every 10 minutes per node. This distribution yields a mean node lifespan of 1 hour - consistent with past studies [4]. We ran each experiment for 2 hours of simulation time; statistics collection began at 30 minutes after the network reached a stable state. EDRA* was tested for f=10, 30, 50 and 70. We compared these to EpiChord using 3-way and 5-way parallelism. We implemented the EDRA* improvements to EDRA which address the EDRA routing table update inaccuracies and also acknowledgment behaviour.

6.2.5.2 Simulation Results

The performance of EDRA* was measured in terms of routing table accuracy, average hop count and bandwidth consumption per second per node. We also discuss these in relation to the success rate of lookup messages. High routing table accuracy infers that a high number lookups will be resolved in a low number of hops. A low hop count is indicative of lower latency (one less RTT per hop) and fewer lookup messages to resolve a request.

1. **Routing Table Accuracy:** As shown in Figure 6.3, EDRA* with $f = 10$ and $f = 30$ operates very close to the expected routing table accuracy of 90% and 70% respectively. When f was increased to 50 and 70, EDRA* operated with routing table accuracy of $> 50\%$ for both. EpiChord with 3 and 5-way parallelism has significantly lower levels of routing table accuracy. 3-way EpiChord maintains around 10% accuracy, with 5-way EpiChord operating 2-3% above this rate.

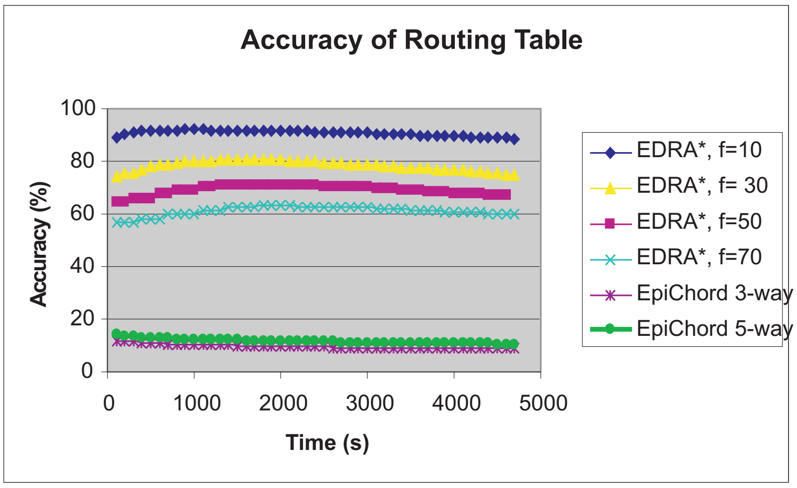


Figure 6.3: Routing table accuracy of EDRA* over varying degrees of f and EpiChord over varying degrees of parallelism [12]

2. **Lookup Hop Count:** In Figure 6.4, the EDRA* hop count is lowest at 1.2 - 1.3 when $f = 10$. When f is raised to 70, EDRA* hop count increases to around 1.9. EpiChord 3-way achieves a hop count of 2.2 for 5-way parallelism and 2.4 for 3-way parallelism.

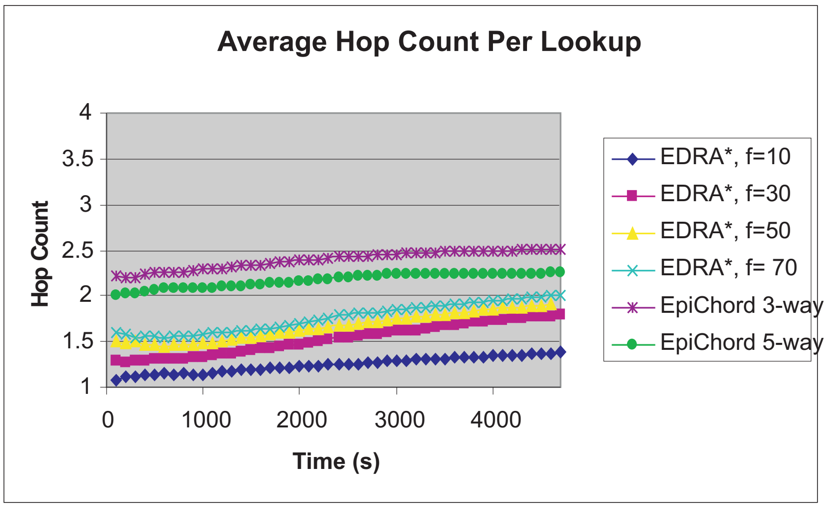


Figure 6.4: Average lookup hop count for EDRA* over varying degrees of f and EpiChord over varying degrees of parallelism. [12]

3. **Bandwidth Requirements:** When calculating bandwidth requirements for both EDRA* and EpiChord, we consider only messages which directly result in routing table updates. For EDRA* this is lookup and propagation messages, replies, proxy messages for new nodes in the join interval, and any probe message which carries a routing table entry. For EpiChord these are application lookups, slice probing maintenance lookups, replies, and probe messages which carry routing table entries.

The marginal decline in accuracy over time in Figure 6.3 and increase in hop count in Figure 6.4 is caused by routing table errors propagating the network - this can be addressed by the solution proposed in 6.2.3.1.

Both EDRA* and EpiChord messages have 20 byte headers plus 16 bytes per node ID and 4 bytes per IP address. EpiChord messages include 4 bytes for port numbers, 4 bytes for control and 4 bytes for timestamps, which are not required in EDRA*. The EpiChord measurements are consistent with those used in [3]. Figure 6.5 compares the bandwidth requirements of EDRA* and EpiChord. EDRA* requires 30 to 50 bytes/sec for target $f = 10 - 70$. EpiChord requires somewhat less bandwidth to maintain its routing accuracy, needing only 18 bytes/sec for 3-way and 20 bytes/sec for 5-way.

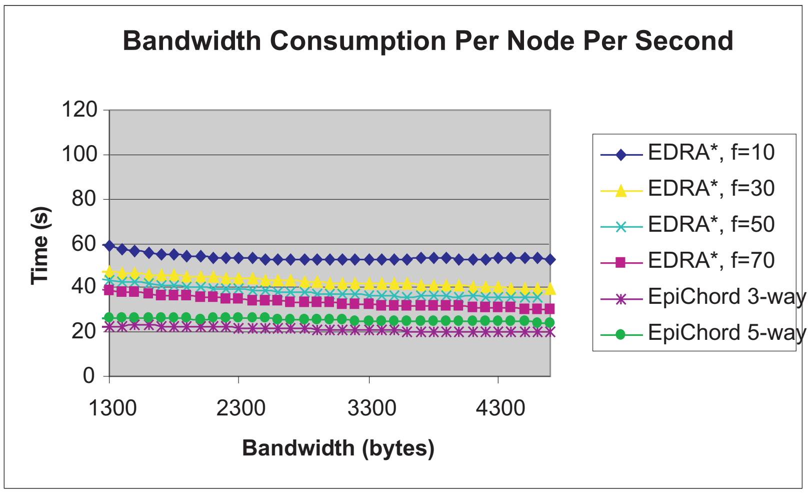


Figure 6.5: Bandwidth consumption of EDRA* over varying degrees of f and EpiChord over varying degrees of parallelism. [12]

6.2.5.3 Discussion

EDRA and EDRA* are active stabilisation algorithms that report every event in the system in a timely manner. To do this, nodes frequently send maintenance messages to other nodes. EpiChord is an opportunistic maintenance algorithm and has a lower routing table accuracy. It uses lookup parallelism to compensate for its lower routing table accuracy. So, in general, compared to EpiChord, EDRA and EDRA* use more bandwidth for maintenance with less bandwidth needed for application lookups. In our simulations for EDRA*, when f=10 (routing table accuracy of 90%) nodes are required to send around 50 bytes per second of maintenance traffic. Conversely, 3-way EpiChord requires 18 bytes per second to maintain 10% routing table accuracy.

An important DHT performance metric is the average hop count of lookups, that is, the number of overlay nodes a query must visit before it is resolved. To justify the higher bandwidth requirements of EDRA*, it must produce low lookup hop counts. Again for f=10, EDRA* performs between 1.2 - 1.3 hops per lookup up to about 1.9 for f=70. EpiChord 3-way and 5-way require on average 2.2 and 2.4 hops per lookup. EpiChord uses parallel lookups to compensate for the low routing table accuracy and enables EpiChord to achieve these hop counts. When the degree of parallelism is increased there exists a greater chance of reaching the desired node in a single hop, and more replies will be received which recommend the next hop. EpiChord concentrates on convergence of routing, thus these replies are more likely to contain the desired destination node. Interestingly, the average hop count of 5-way EpiChord is somewhat close to EDRA* where f=70, yet its bandwidth requirements are 15 bytes per second per node less, suggesting it may be preferable to EDRA* for high f, but it cannot achieve the performance of EDRA* for low f. Further, the maintenance traffic for EDRA* is fairly stable over a large range of routing table accuracies. From our simulations, we extrapolate that EDRA* cannot effectively operate at the low levels of bandwidth witnessed in EpiChord. Equally, it is clear that EpiChord cannot operate effectively at the high levels of routing table accuracy achieved by EDRA*. This suggests that these algorithms offer complementary performance versus cost tradeoffs. Figure 6.6 shows this complementary behaviour more clearly, comparing bandwidth vs. hopcount for EDRA* and EpiChord.

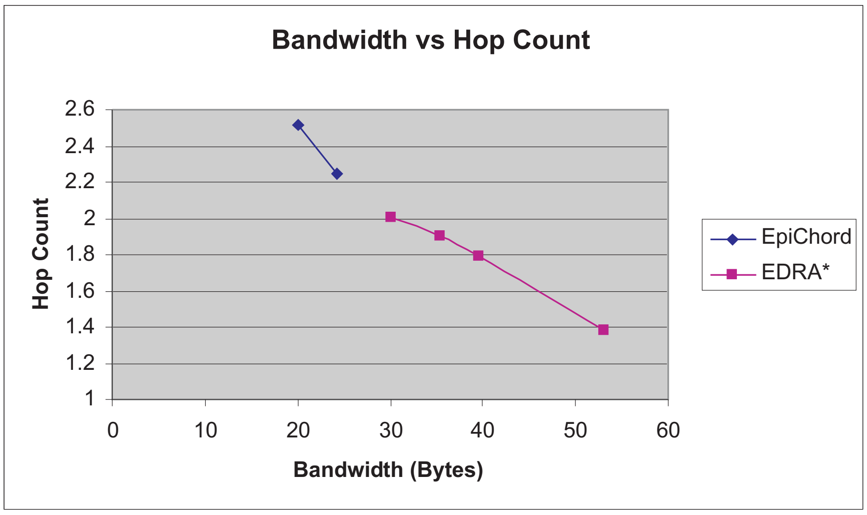


Figure 6.6: Bandwidth vs. Hopcount for EDRA* ($f = 10 - 70$) and EpiChord (3-5 way)

6.3 Summary

Structured Peer-to-Peer (P2P) overlays typically offer either multi-hop or one-hop performance. In this Chapter, it was shown that two overlays offered complementary performance depending on the needs of the system. For systems which require low hop counts, the EDRA* algorithm offer better performance. For systems which require low bandwidth, the EpiChord algorithm is more suitble.

Chapter 7

Chameleon: A Variable Hop Overlay

7.1 Introduction

To determine wether the benefits of both EDRA* and EpiChord algorithms can be captured in a single Overlay, a new system design was created. In this Chapter we present Chameleon, which combines maintenance techniques from both EDRA* and EpiChord into a novel two-tier approach that allows nodes to adapt to variations in their available bandwidth. Chameleon uses opportunistic routing table updates for lower bandwidth nodes and an active stabilisation approach for high bandwidth nodes, offering a performance from O(log n)-hop up to O(1)-hops. In doing so its nodes can operate successfully in large scale (mobile) networks where peers need to adapt to changes in the available bandwidth.

7.2 Motivation

Currently, networked devices exist with a multitude of capabilities. For example, high-powered desktop machines have extensive processing power and plentiful bandwidth. In contrast to this, mobile networked devices often have strict bandwidth limits and limited processing power. These limitations are typically caused by expensive bandwidth costs and poor battery performance. With P2P applications becoming increasingly popular for networked devices, e.g. Instant Messaging and content sharing systems, adaptable P2P systems are required to meet the diverse needs of hosting devices. The ability of an overlay to support peers that dynamically adapt traffic levels to available bandwidth is called a variable-hop overlay. Accordion, an existing example of a variable-hop overlay [89], uses opportunistic updating when bandwidth is plentiful and reverts to multi-hop overlay performance otherwise. By providing adaptive performance to nodes, peers with different bandwidth capacities can co-exist in the same overlay network.

Since a variable-hop overlay is adaptive, peers of different bandwidth capacities can exist in the same overlay. A peer might have high bandwidth capacity in one interval and low bandwidth in another. An example of this is when mobile nodes have 3G and 802.11x connectivity, and switching occurs due to availability of a given connection. This behaviour makes variable hop overlays well suited to heterogeneous networks where a large range of devices with varying network capacities participate. For example, mobile nodes which rely on GPRS connections are severely restricted by available bandwidth. For this reason, being able to participate in the overlay system with minimal bandwidth requirements is desirable.

7.2.1 Accordion

The concept of variable hop overlay is introduced in [89]. Accordion is a variable hop overlay in which a peer limits its routing table update message level based on its available bandwidth. During periods where nodes have low bandwidth capabilities, routing performance may reach that of multi-hop overlays; for higher bandwidth, routing performance reaches one-hop. Accordion uses recursive parallel lookups so as to maintain fresh routing table entries for a node in its neighbourhood of the overlay, reducing the probability of timeout. The peer requesting the lookup selects destinations based both on the key and also gaps in its routing table. Responses to forwarded lookups contain entries which fill these routing table gaps. Note that recursive parallel lookups create more load on the target peer compared to iterative parallel lookups, since the target node receives p messages, where p is the number of parallel unicast messages sent per each request. Excess bandwidth is used for parallel exploratory lookups to obtain routing table entries for the largest scaled gaps in the peer's routing table. The degree of parallelism is dynamically adjusted based on level of lookup traffic and bandwidth budget, up to a maximum configuration such as 6-way.

Compared to the Accordion variable hop overlay, the Chameleon approach presented here targets a wider performance range to achieve lower hop-count for peers with high-bandwidth capacity.

7.3 Chameleon

7.3.1 Overview

In Chameleon, nodes assess their available bandwidth to determine whether they are *high* or *low* bandwidth peers. Chameleon is classified as a two-tier system because nodes operate using different maintenance algorithms depending on their available bandwidth. High bandwidth peers operate using the EDRA* algorithm to maintain routing table accuracy of up to 99%. By 99% routing accuracy, a peer is aware of 99% of the other peers in the system. Low bandwidth peers operate using the EpiChord routing algorithm, offering poorer performance at a reduced cost. Nodes are able to change classification should their circumstances change, e.g. a mobile device finds an 802.11x hotspot. Additionally, nodes may increase/decrease their performance by adjusting system parameters. High and low bandwidth peers co-exist in a single overlay as shown in Figure 7.1.

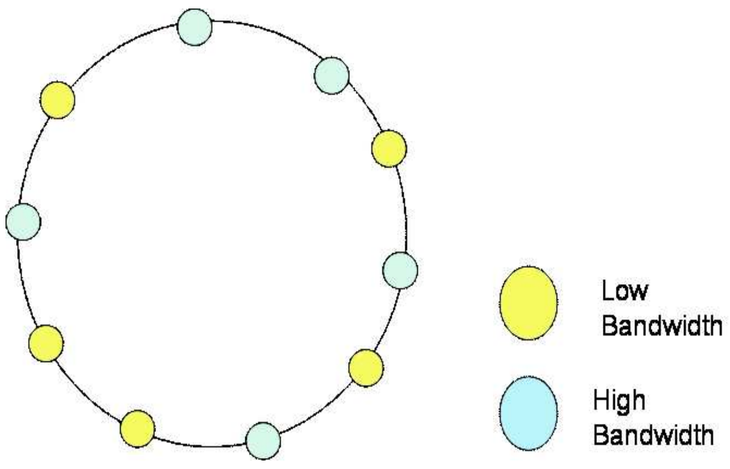


Figure 7.1: Chameleon system featuring high and low bandwidth peers

7.3.2 Design

7.3.2.1 Selection of Maintenance Algorithm

There are two methods to determine the algorithm to be used by a node. One primitive method involves the device user instructing the device to perform in either high or low bandwidth mode. However, a method which does not rely on external knowledge or user input is desirable. Subsequently, a node can test its available bandwidth by retrieving content from another node in the system and testing its download speed (bytes/sec). The node would then select the relevant maintenance algorithm depending on their bandwidth. This Thesis does not define what constitutes high or low bandwidth as this can vary depending on the type of devices using the system and the applications used.

7.3.2.2 High Bandwidth Nodes

As previously stated, high bandwidth nodes (herein H peers) operate using the EDRA* active stabilisation maintenance algorithm. These H peers maintain their routing algorithms independently of low bandwidth peers (herein L peers) to makes up the 2-tier system. This is shown in Figure 7.2.

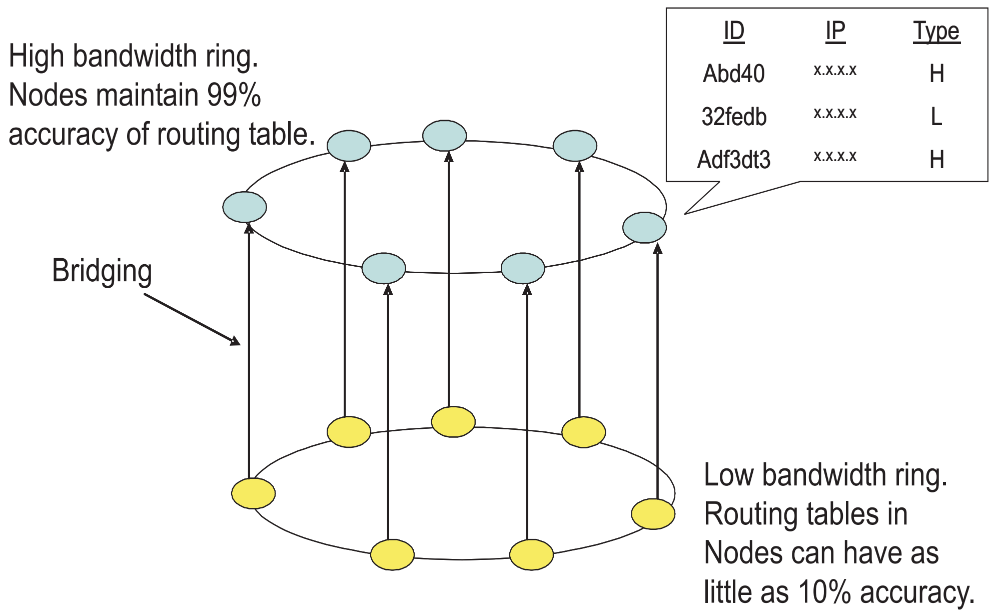


Figure 7.2: Chameleon's 2-tier maintenance system with H and L peers operating independently [13]

For H peers to maintain 99% accuracy of their routing tables, they must be aware of all other H and L peers in the system. When an H peer joins the system, it contacts its nearest existing H peer and initiates a cache transfer to obtain a copy of its routing table. The existing H peer then reports this join event amongst all other H peers. This behaviour ensures all H peer joins are reported in Chameleon. Likewise, should an H peer leave the system, its successor H peer detects this leave event (using standard EDRA rules) and report it to all other H peers. Thus, with very few changes to the system, H peers are still able to report other H peer join and leave events.

As already stated, H peers must have a method of detecting new events relating to L peer joins and leaves in the system. To enable this functionality in Chameleon, the notion of a bridging node is introduced. Each L peer in the Chameleon system must have one bridging node. When an H peer receives a request from an L peer to be its bridging node, it firstly responds to accept this request then reports this join event to all other H peers in the high bandwidth ring. As a bridging node, the H peer is required to contact the L peer at regular intervals (such as every 10 seconds) to ensure it is still alive. If an H peer is observing high churn in the system, it may reduce the length of the interval accordingly. To check that an L peer is alive, it sends a CHECK_ALIVE message. If the L peer does not respond, it is assumed dead and its death is reported amongst the other H peers in the system. This again ensures that all L peer joins and leaves are reported to the high bandwidth nodes, guaranteeing 99% accuracy of the routing table.

Should an H peer not wish to become a bridging node for an L peer it may send a rejection. The L peer would then look at its own routing table, find another suitable candidate H peer and send it a request. One possible extension to the

Chameleon system would be to encourage H peers to be bridging nodes for L peers. For example, H peers may be offered incentives [91],[92],[93], e.g. priority downloads of popular data in exchange to be bridging nodes.

In the event that an H peer dies, it may orphan a number of L peers that it bridges for. The Chameleon system ensures that this death is detected, as the L peers will stop receiving CHECK_ALIVE messages from their now dead bridging node. These L peers can then remove the dead node from their own routing tables and search for a new *bridging node*.

Finally, H peers have the ability to slightly alter their performance and bandwidth usage. H peers in Chameleon, just as in EpiChord, can send p-way lookup messages. Should an H peer wish to reduce its bandwidth consumption it can reduce the degree of parallelism used for lookup messages.

7.3.2.3 Variable Bandwidth

The Chameleon system is a variable-hop overlay. Therefore, at any time nodes may change their bandwidth classification. That is, nodes may change from an H peer to an L peer and vice versa. There are many reasons for nodes altering their bandwidth classification, but primarily a change in environmental circumstances will be the cause, (e.g. a mobile node with WiFi detecting a hot-spot offering increased bandwidth). Conversely, a mobile node may leave a hot-spot and revert to using GPRS or 3G. In both of those circumstances, a node may wish to change its bandwidth classification and alter its performance in the Chameleon system.

To perform an upgrade, L to H, the following process should be followed. Firstly, the L peer sends an UPGRADE message to its bridging node, which responds by transferring its full accurate routing table to the upgrading L peer. The L peer then changes its bandwidth classification to H and asks its bridging node to propagate this UPGRADE amongst its fellow H peers. In doing so, every H peer learns about the L peer's upgrade and will add it to their propagation paths. This new H peer will now learn about all new events in the Chameleon system as it is now part of the highly accurate high-bandwidth tier.

The downgrading process is very simple. Firstly, the downgrading H peer initiate's a DOWNGRADE message to its fellow H peers and waits for the period of time $\theta * p$, which is defined by the EDRA* algorithm as the time taken for an event to propagate the H tier. Upon expiry of this time period, the H peer simply changes its bandwidth classification from H to L and ceases to use the EDRA* algorithm for routing table maintenance. Thereafter, this L peer operates as a standard L peer, using EpiChord techniques to update its routing table.

7.4 Simulations

7.4.1 Setup

Chameleon was simulated using the SSFNet simulation environment. In order to compare our results with those for EDRA* and EpiChord, identical simulation parameters to those described in Section 6.2.5.1 were used. That is, node lifespans were distributed using a heavy-tailed Pareto distribution to achieve a median node lifespan of 1800 seconds. Lookups were issued at a rate of one lookup every 10 minutes per node. Our simulated network consisted of 10,450 physical nodes running 1200 Chameleon nodes. Unless stated otherwise, data was collected after 2100 seconds of simulation time, which was the period of time taken to grow the Chameleon system to a steady size of about 1200 nodes. Data was then collected for the next 2000 seconds.

For these experiments, graceful node deaths were not simulated. When a node dies, it does not inform any other node in the system and thus the algorithm must detect these deaths. Although the Chameleon network was grown to a steady size, the rate of churn in our network is still high with around 1 node joining/leaving the system every 2 seconds. The SSFNet simulator does not simulate link traffic and thus messages are not lost due to network congestion. However, message loss is still an important factor due to nodes exclusively dying ungracefully. In real implementations of P2P systems an ungraceful exit is a worst case scenario for node deaths and thus maximum stress is imposed on Chameleon.

7.4.2 Results

To determine the effect of varying the proportions of H and L peers, Chameleon was simulated using 10%, 50% and 90% H peers. This allows us to test the system in extreme cases such that H peers far outweigh L peers and vice versa. Figure 7.4 shows the average bandwidth consumption of H and L peers for different ratios of H and L peers (10:90, 50:50, 90:10) in the system. The cases with 10% and 90% H peers are considered extreme cases to explore boundary behaviour. As can be seen, regardless of the number of L peers in the system, L peer bandwidth consumption remains fairly constant at around 15-16 bytes per second. However, as the number of H peers decreases, H peer bandwidth use increases. For 50% H peers, bandwidth consumption is around 70 bytes per second and for 10% H peers this rises to around 180 bytes per second. When Chameleon contains 90% H peers, the system performs better than with 50% H peers, with around 50 bytes per second consumed. The reasons for this are explained below in Section 7.4.3. The slight increase in bandwidth consumption is caused by propagation of routing table errors. This is addressed in Section 6.2.3.1.

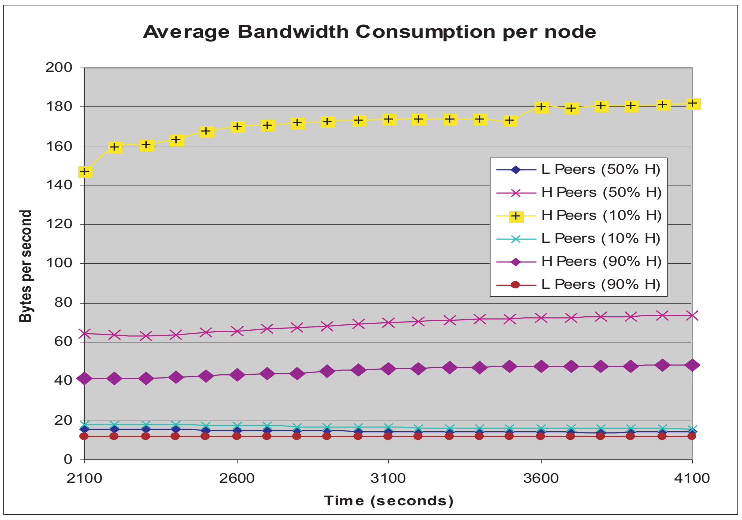


Figure 7.3: Average Bandwidth Consumption for H and L peers with variable H:L ratio

Figure 7.4 shows the average hop count and bandwidth consumption for different ratios of H and L peers. High bandwidth peers obtain almost one-hop performance regardless of the ratio of H:L peers. As discussed above, the lowest bandwidth requirement exists if there is a high proportion of H peers in the system (90%), and the largest amount of bandwidth is used for a low proportion of H peers in the system (10%). Low bandwidth peers perform better when there are a large number of H peers in the system. For 10% H peers, performance is around 1.5 hops, improving to 1.25 hops for 10% L peers.

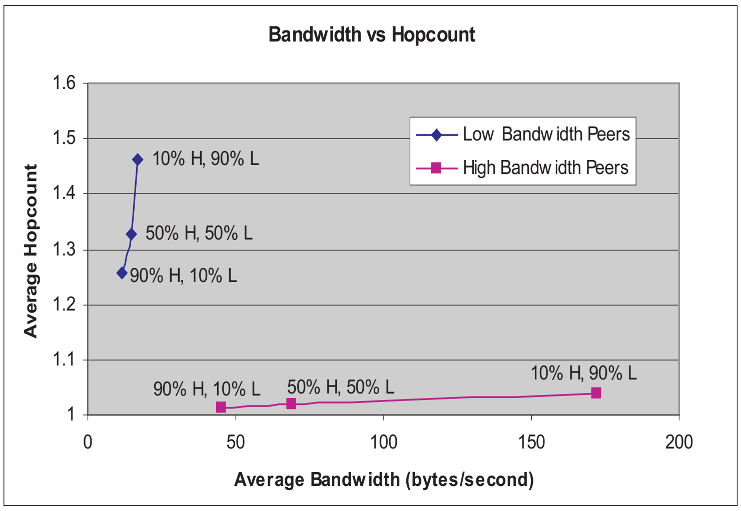


Figure 7.4: Hop count for H and L peers for different ratios of H and L peers.

7.4.3 Discussion

As shown in Figure 7.3, decreasing the number of H peers below 50% increases the bandwidth requirements for these nodes. This increase is caused by an enlarged number of L peers that H peers must bridge for. Considering a system with 100 nodes, if 10% of these are high bandwidth, each H peer needs to bridge, on average, 9 L peers. As the balance of H:L peers realigns, this figure naturally falls and so does the bandwidth requirement of those nodes. This behaviour is expected. Although the bandwidth requirements are high when minimal numbers of H peers exist, performance is still very good at nearly 1 hop. As these nodes have more bandwidth available than L peers, this increase is reasonable and acceptable.

It can also be observed that H peer bandwidth consumption for Chameleon is higher for 50% H than for 90% H peers. This confirms the above theory that increasing the number of H peers reduces the average node bandwidth per second for H peers. As the proportion of H peers increases, so does the number of H peers whose events need to be reported using EDRA*. This increase is compensated however by the reduced bridging requirement on each node, thus significantly reducing peer bandwidth consumption.

It is also clear, however, that certain ratio's of H:L peers will be unsustainable by Chameleon. For example, a system with 1% H peers will not offer a balanced performance. However, we do not envisage that this scenario is likely to occur. A large proportion of networked devices in the Internet are desktop machines, which typically have good resources. If we can then assume in a real deployment that the ratios of H:L peers mirror the ratio of desktop:mobile devices, Chameleon offers excellent performance. Also, as mentioned in Section 7.3.2.2, one extension to Chameleon could be an incentive scheme to encourage more nodes to operate as H peers.

Another interesting observation from Figure 7.3 is that L peers maintain good performance with minimal bandwidth requirements (around 15 bytes per second) regardless of the number of L peers in the system. Referring back to Figure 6.6, EpiChord peers using 5-way parallel lookups are only capable of 2.2 hops per lookup. However, as shown in Figure 7.4, Chameleon L peers using the same algorithm are capable of average lookup hop counts of 1.4 hops whilst using less bandwidth than standalone EpiChord. This phenomenon is caused by a synergy between the H and L peers. Because H peers exist in the Chameleon system with 99% accurate routing tables, it is natural that L peers learn from H peers.

An example of this synergy is as follows. An L peer sends a lookup for some value in the system and the request arrives at an H peer. The H peer has 99% accuracy of its routing table and responds with accurate next hop information. The L peer then uses this information to update its routing table and complete the lookup. As time progresses, this L peer is accumulating accurate Chameleon membership information from H peers, thus improving its own routing table accuracy. Naturally, the higher the ratio of H peers in the Chameleon system the more likely L peers will learn from H peers. This is demonstrated by low-bandwidth peer average hop counts decreasing as the number of H peers increases.

To highlight the significance of the bandwidth savings achieved using the Chameleon algorithm, it is prudent to apply the monetary costs of operating Chameleon on a mobile device. Looking at the major UK mobile phone operators, a charge of £1 per MB is standard for Pay As You Go customers. Node lifespan per day is assumed at 1 hour, consistent with past studies [5]. Assuming this tariff and lifespan, Chameleon L peers using 15 bytes/sec would pay 5.1p per day. However, if a node were performing using similar bandwidth consumption to Chameleon H peers, e.g. up to 180 bytes per second, they would be charged 0.61p for the 1 hour spell. It is understandable why a Chameleon H peer would only operate at H status if they were using a 802.11g hotspot or had access to some other cheap, high speed network. Thus, the motivation to downgrade to L peer status when using GPRS or 3G is clear.

As a 2-tier algorithm, Chameleon exhibits good performance both in terms of bandwidth usage and hop counts. It could therefore be argued that an algorithm with additional tiers would provide even better performance. However, it should be noted that for each additional tier added to a P2P algorithm, the overhead associated with managing the network is increased. For example, transitions between tiers all carry a message overhead, as does maintaining links between nodes in each tier. Consequently, we suggest that part of the success of the Chameleon algorithm is its simplicity and low levels of required infrastructure maintenance.

7.4.4 Comparison with EDRA*, EpiChord and Accordion

This Section discusses the performance of Chameleon compared with the performance of the two individual algorithms, EpiChord and EDRA*, and also with the variable hop algorithm Accordion. To compare these systems, EpiChord and EDRA* were simulated using identical simulation settings and environment described in Section 7.4. Figure 7.5 shows the bandwidth and hop count performance for the Chameleon system with varying ratios of H:L peers, the EDRA* system for varying degrees of f (maximum fraction of allowed stale entries) and the EpiChord system for varying degrees of parallelsim. The hop count achieved by Chamelon H peers is exceptionally low, at virtually 1 hop and bandwidth ranges from a competitive 45 bytes/sec per node when the ratio of H:L peers is high to 170 bytes/sec when the ratio of H:L peers is low.

The nearest comparison of Chameleon H peers is with EDRA*, and it is evident that although EDRA* consumes $<$ 60 bytes per second when sampling rates of f $>$ 10%, it does not achieve hop counts as low as Chameleon H peers. Therefore, it can be said that the performance of the Chameleon system is superior to what can be achieved using EDRA* or EpiChord at comparable levels of bandwidth. When analysing Chameleon L peers, the closest comparison is to the EpiChord algorithm. As is shown in Figure 7.5, Chameleon L peers for varying ratios of H:L achieve comparable low hop counts, between 1.25 and 1.45 hops per lookup with $<$ 20 bytes/sec per node. EpiChord performs at very similar levels of bandwidth(22 bytes/sec) but due to the synergy described in Section 7.4.3, Chameleon L peers achieve much lower hop counts, with EpiChord nodes performing at $>$ 2.2 hops per lookup. Hence Chameleon L peers can operate at bandwidth levels at least as low as EpiChord whilst significantly outperforming EpiChord on hop counts. Having compared Chameleon with both EpiChord and EDRA*, we have shown that the overall Chameleon system offers a wider range of performance than can be achieved by using either EDRA* or EpiChord individually. Depending on a nodes available bandwidth, Chameleon node's can achieve better performance than the two individual algorithms.

Compared to the Accordion variable hop overlay, Chameleon targets a wider performance range to achieve lower hop count for peers with high-bandwidth capacity. Direct comparisons with Accordion are difficult as the measures are based on network latency, however in [89] the authors report an average round trip time of 178 ms in their network. With this, Accordion achieves a lookup latency of about 200ms. This equals a hop count of 1.12.

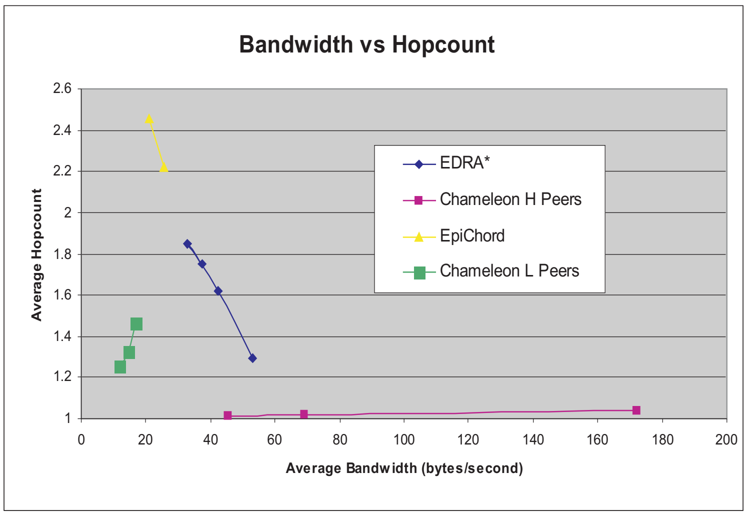


Figure 7.5: Bandwidth vs. Hop Count for Chameleon vs. EpiChord and EDRA* systems

7.5 Service Discovery on Chameleon

To test Chameleon's efficiency and flexibility as a service overlay, we implemented Meta Service Discovery, as described in Section 4.5, using Chameleon.

7.5.1 Data Set

Since future online peer-to-peer service offerings are likely to evolve from existing consumer-oriented services, the YellowPages (YP) [94] classification, which has about 2500 categories was selected. For each category, synthetic service names and service interface names were generated. These services were inserted into each DHT where each service is indexed by 'service_name', 'service_interface' and a concatenation of both service interface and name. This allows MSD to be performed by issuing service lookups for metadata.

In the absence of a global approach for measuring service popularity, web page popularity was used, which is widely suggested to follow a Zipf distribution. Therefore, we assume that the most popular service is requested twice as often as the next and so on.

7.5.2 Simulation

Again, service discovery on Chameleon was simulated using the SSFNet simulation environment. The simulations were carried out on an overlay network consisting of 1200 nodes running on a physical network of 10,450 nodes. Simulation parameters were consistent with Section 7.4. Two lookup loads for Chameleon were simulated using services from the YellowPages, both using service popularity according to a Zipf distribution but with varying frequency of lookups. In our low lookup load, lookups were issued on average every 10 minutes. In our high lookup load, lookups were issued every 20 seconds. Data was collected after 30 minutes of simulation time, the time taken to grow the network to 1200 nodes. Data was then collected for a further 2000 seconds.

7.5.3 Results

Firstly, when using MSD in Chameleon, all Chameleon lookups resolved accurately. That is, if a lookup value existed in the system, it was successfully returned. Figure 7.5 shows the bandwidth vs. hop count when comparing the Chameleon system for varying ratios of H:L peers for two lookup loads. For L peers, increasing the lookup level reduces the hopcount from between 1.25 - 1.5 hops to 1.09 - 1.23 hops. Bandwidth consumption is shown for H peers. Increasing the lookup rate for high ratios of H:L peers reduces hop counts and increases bandwidth.

For 90% H peers, bandwidth increased from 45 bytes/sec to 60 bytes/sec and hop counts dropped from 1.014 to 1.009. However, when a low ratio of H:L peers exists, bandwidth consumption drops slightly from 169 bytes/sec to around 158 bytes/sec. In this scenario, in the low lookup load, L peers have low levels of routing table accuracy and often struggle to find H peers to bridge for them. When this is the case, H peers take longer to detect L peers (as they receive less frequent join requests because L peers have inaccurate routing tables). Their routing tables are therefore out of date, which increases the number of lookups they have to send to locate data in the overlay. This in turn increases their bandwidth requirements. However, when the lookup load is increased, L peers perform significantly better, with much lower hop counts from a more accurate routing table. This in turn impacts on H peers, who now also have more accurate routing tables and thus in turn reduce their own bandwidth requirements to locate lookup values. This explains the reduction of bandwidth for H peers even after increasing the number of lookups.

7.6 Summary

A unique two-tier P2P overlay system called Chameleon was created using a combination of the EDRA* and EpiChord routing algorithms. Nodes which have low available bandwidth can opt to use the EpiChord algorithm, and conversely nodes which have large available bandwidth can opt to use EDRA*. Using varying proportions of high and low bandwidth nodes, Chameleon was shown to achieve close to 1-hop performance for high-bandwidth nodes, and, if the proportion of high bandwidth nodes is high, almost 1-hop performance for low-bandwidth nodes. It was also demonstrated that low bandwidth nodes ‘learn’ from their higher bandwidth peers to achieve improved performance.

Chapter 8

Conclusion

8.1 Introduction

The goal of this Thesis was to design a service discovery approach to tackle the increasing number of incompatible service discovery techniques proliferating the consumer market. Due to the increasingly mobile nature of networked devices, this approach needs to be resource-efficient and capable of operating in a Wide Area Network. With this goal in mind, this chapter presents the key achievements of the Thesis. These include:

- The SIP Service is capable of bridging multiple administrative domains to provide service discovery in a Wide Area Network. In addition to bridging domains, the SIP Service can transfer SIP functionality to adapters (known as bridging bundles) to translate between SIP and other protocols.

- Meta Service Discovery provides a method to locate and select the appropriate service discovery mechanism according to the context of the mobile device, such as network domain, location, protocol and application.

- Multi-destination multicast has been integrated with P2P overlays for the first time. Design criteria has been formulated for its use in P2P networks with two 1-hop P2P algorithms altered to utilise it. Bandwidth savings of $> 30\%$ were shown to be achievable, and the results were confirmed using a Markov model.

- Complementary performance was achieved between two P2P overlay maintenance algorithms such that, depending on the goal of the system (efficiency or performance), each could be suitable. Using this information, a unique 2-tier overlay maintenance algorithm was designed and shown to offer a wider performance range than offered by other variable-hop overlays. Additionally, when combining these two algorithms into a single 2-tier overlay, a synergy between low and high bandwidth nodes was identified. When low-bandwidth and high-bandwidth nodes co-exist in an overlay, the performance of low-bandwidth nodes increases as they 'learn' from their higher-bandwidth neighbours.

These achievements are detailed in this Chapter.

8.1.1 Centralised Service Discovery for WANs

Service discovery techniques can be classified as centralised or decentralised. This Thesis initially attempted to provide WAN service discovery using a combination of SIP and OSGi, as described in Chapter 3. The outcome was the SIP Service which offers three primary services to devices and users in an OSGi- enabled network: Inter-gateway bridging, protocol bridging and application layer mobility.

Inter-gateway bridging, described in Section 3.5.3, allows devices and their services present on one gateway to be exported to a second gateway using the SIP Service and bridging bundles. This is achieved by exporting device information from one domain to the other. Using this information, a bridging bundle on the foreign domain can create a virtual representation of that device which will exist on the foreign gateway and offer the services of the physical device. This gives logical mobility to devices which may not be physically portable.

Protocol bridging provides the translation between different protocols using application-layer proxying. For example, a SIP device can control a native UPnP device by issuing control commands via an intermediate bridging bundle. Implementations of protocol bridging were successful, and control of a UPnP light switch was achieved using a SIP-enabled device.

Application layer mobility can be achieved by non SIP devices or software bundles present in the same domain as the SIP Service. By simply requesting SIP UA functionality, a non-SIP device can behave as a native SIP device, obtaining full SIP functionality and all the benefits that come with it, e.g. personal mobility.

The SIP Service provides a viable solution to the growing problem of incompatible service discovery types using bridging bundles to translate between different protocols. Additionally, by using inter-gateway bridging, the SIP Service is capable of enabling WAN service discovery. However, to do so a connection between bridging bundles in each domain must be established, and this does not scale well. Therefore, an alternative solution was sought.

8.1.2 Decentralised Service Discovery for WANs

To counteract the scalability issue of the SIP Service, decentralised P2P systems, were considered. P2P systems where peers can offer and use services from any other peer without relying on centralised resources, are known as P2P service overlays. By utilising P2P networks for service discovery, a highly scalable and generally fault-tolerant service discovery mechanism is possible, achieving global scale service advertisement and discovery.

Using P2P service overlays as a medium for service discovery, this Thesis presented Meta Service Discovery (MSD). Meta service discovery is used to find and select a service discovery mechanism by context. As multiple service discovery mechanisms (SDMs) proliferate across various administrative domains, mobile devices will require a way to locate and select the appropriate mechanism according to the context of the mobile device, such as network domain, location, protocol, and application.

To test the feasibility of MSD, three structured overlay-based index systems (FreePastry, OpenDHT, INS/Twine) were used to index a SDM data set and wild card queries were issued in each system. We estimated the sizing of the key set if indexing of SDMs were widely deployed. We further analysed the key distribution of SDM index entries, focusing on location and domain attributes. Assuming a SHA1 hash function, MSD obtained key distribution results that are comparable to measurements from hashing random strings.This identified a feasible solution to find and select SDMs by context.

Although P2P service overlays are a scalable solution for service discovery, in many cases they suffer from inefficiencies and many routing algorithms are not suitable for low-bandwidth/low-resource devices. To tackle this issue, this Thesis presented the use of multi-destination multicasting in P2P overlays to reduce bandwidth costs in all categories of P2P systems. By extending the SSFNet network simulator to support the XCAST multicast protocol, we were able to measure the bandwidth savings of two P2P routing algorithms, EpiChord and EDRA, on which parallel unicast operations were replaced with multi-destination multicast. Where the degree of parallelism used was 5, bandwidth savings were $> 30\%$. This is a significant saving, especially when the hosting device of the P2P algorithm is mobile and bandwidth is charged per kilobyte. To further prove the bandwidth savings achievable by utilising the XCAST multicast protocol, this Thesis presented a Markov model of XCAST-enabled EpiChord. This Markov model confirmed that the savings measured by the simulated XCAST-enabled network were both realistic and achievable.

However, deployment of multicast routers in the Internet is slow. If only a subset of all routers are multicast-enabled, these routers forward multicast packets to other multicast routers using tunnels through unicast routers. Should this be the case, multicast savings would be reduced. Therefore, multi-destination routing alone is not a guaranteed method of reducing bandwidth in P2P service overlays.

When comparing the bandwidth consumption of EpiChord and EDRA, the EDRA algorithm as originally specified was not found to be stable and did not offer good performance. This Thesis presented a number of corrections and enhancements to the original EDRA algorithm, and compared the corrected version's performance to that of EpiChord. EpiChord and EDRA use different routing table maintenance algorithms, and a comparison between these algorithms indicated complementary performance. The EDRA algorithm can offer close to 1-hop performance at moderate levels of bandwidth, whereas EpiChord offers moderate performance at much lower bandwidth consumption. Using this information, this Thesis proposed that combining these algorithms in a single two-tier system would achieve a scalable and highly efficient P2P algorithm capable of enabling service discovery.

Thus, the final offering of this Thesis is the Chameleon P2P algorithm. The Chameleon P2P service overlay is a unique 2-tier P2P routing algorithm with the ability to switch routing table maintenance techniques depending on available bandwidth. Where bandwidth is plentiful, Chameleon nodes can elect to utilise the active stabilisation maintenance technique used in EDRA. Conversely, where available bandwidth is minimal, Chameleon nodes can elect to utilise the opportunistic maintenance algorithm employed by EpiChord. Thus, Chameleon nodes can achieve a wider range of performance than can be achieved by any other P2P service overlay to date. For high-bandwidth nodes, performance approaching 1-hop is available. For low-bandwidth nodes, performance close to 1-hop is also achievable but for lower bandwidth.

By combining EpiChord and EDRA into a single overlay, this Thesis identified an improvement in the performance of low-bandwidth EpiChord nodes. This is due to a synergy between high and low bandwidth nodes which co-exist in a single overlay. That is, nodes with low-accuracy routing tables passively learn from nodes in the same system with more accurate routing tables, consequently improving their performance.

8.2 Mapping to Original Objectives

The original objectives of this Thesis were detailed in Section 1.2. The success of this work in meeting these objectives is now considered.

1. **Design an approach to service discovery which tackles the increasing number of incompatible service discovery techniques.** The Chameleon system is capable of tackling the increasing number of incompatible service discovery techniques by successfully implementing Meta Service Discovery.

2. **This approach must be resource efficient and suitable for mobile devices.** Nodes in the Chameleon system can alternate between routing algorithms depending on available bandwidth, making it efficient and ideal for mobile devices. In addition to the Chameleon algorithm, this Thesis also identified multi-destination multicast as a suitable extension to P2P systems, which can reduce bandwidth savings by at least a further 30%.

3. **Finally, this approach must be capable of operating in a Wide Area Network.** The Chameleon P2P service overlay is pervasive and suitable for mobile devices. Therefore, it is ideally suited for operation in a Wide Area Network.

8.3 Limitations

The purpose of this section is to clearly define the scope of this document. Addressing several contributions in turn, the limitations of this work are stated.

- **XCAST**

  This work looked at the usefulness of implementing multi-destination multicast in P2P networks. To implement multi-destination multicast, XCAST routers must be deployed in the Internet. To handle partial deployment of XCAST routers, tunneling is a proposed solution by the Scalable Adaptive Multicast research group [95]. XCAST routers maintain a routing table containing the IP addresses of other XCAST routers in the Internet. If the next hop in the network is not an XCAST router, it would tunnel the message to the next XCAST router available. If an XCAST router does not know the address of another XCAST router, it explodes the XCAST packet into unicast messages and sends them using traditional network routing. The design of tunneling is yet incomplete, and is still part of the XCAST standardisation process. Consequently, partial deployment was not simulated here as the aim of this Thesis was to demonstrate the potential of multi-destination routing in P2P overlays.

  It should be noted that, even with just a small number of XCAST enabled routers in the Internet, notable bandwidth savings are achievable at the edge link. This first link is quite important as it is often charged by volume of data sent. As long as the messages sent by a source node in the network are multi-destination, a useful saving will be achieved.

One limitation of XCAST is the maximum group size. This Thesis did not boundary test the maximum practical size of the destination list. This is restricted by the Maximum Transfer Unit (MTU) of the network. Currently for Ethernet this is 1500 bytes as an upper bound. For IPv4, this gives a maximum size of 127 addresses [74]. Taking the maximum level of parallelism as 127, the maximum size of a Chameleon network of high-bandwidth peers is 1E+38.

This work showed that XCAST can give bandwidth savings of around 30% where the level of parallelism is 5. The optimum level of parallelism was not studied. A maximum level of parallelism of 5 was chosen to allow the results to be compared with existing work, demonstrating accurate savings.

XCAST is not suitable for sensor networks due to the processing power required on each XCAST router.

- **Chameleon**

  The Chameleon algorithm was implemented on the SSFNet network simulation tool and studied with various network loads and variations in proportions of high and low bandwidth nodes. However, deploying Chameleon on a mobile device was not in the scope of this work. This is covered in the further work section.

Also, this Thesis did not research the best method to classify Chameleon nodes as high or low bandwidth. Two suggestions were made, but this Thesis not define what constitutes high or low bandwidth as this can vary depending on the type of devices using the system and the applications used.

8.4 Further Study Questions

This document focuses on several key research questions in significant detail. However, there are a number of secondary research goals which could be further studied. These include but are not limited to:

1. **Effects of partial deployment of the XCAST protocol**. As described in Chapter 5, multicast deployment in the Internet is slow. If only a proportion of routers is multicast-enabled then multicast savings will be reduced. Implementing multi-destination multicast in P2P networks has been shown in this Thesis to yield savings of up to 33% where the degree of parallelism is set to 5. However the effects of varying proportions of multicast enabled routers has not been studied here.

   The participants from the WIDE Project/Fujitsu Lab on behalf of the Scalable Adaptive Multicast (SAM) research group [95] have recently developed a testbed for Application Layer Multicast (ALM) on PlanetLab [33]. PlanetLab is a global research network of in excess of 800 nodes. Their work is described here [96]. Using this network, a simulated P2P overlay can be feasibly deployed globally and tested. The proposed tunneling technique could be tested with varying levels of active XCAST routers to determine the effects of partial deployment.

To create such a simulation, a P2P simulator such as Overlayweaver [97] would need to be deployed globally on PlanetLab and configured with P2P algorithms which are compatible with XCAST, e.g. the extended EpiChord/EDRA and Chameleon algorithms described in this document.

2. **Deployment of Chameleon on a mobile Device**

   The Chameleon algorithm was designed to be suitable for use in devices with variable bandwidth, e.g. mobile phones. To confirm the suitability of Chameleon, a working prototype of the system would need to be developed on a mobile platform. In order to be completely compatible, some challenges would need to be addressed, one of which is changing IP addresses.

   For example, as a mobile device traverses multiple sub-networks, its addressable location may change. In order to locate a device in a 1-hop P2P network, its address must remain fixed and constant to keep churn rates low and to ensure the address is accurate. Thus, if each node had to update its location every time it changed subnet, churn would be intolerable. One possible solution to this is Mobile IP, where a device has a permanent 'care of' address. The suitability of Mobile IP for Chameleon is an interesting research question but raises concerns over triangular routing, where messages are routed in a suboptimal manner. Consequently, a modified mobile IP solution would be required.

For example, it is possible that Chameleon nodes could store their IP address in the DHT as a key-value pair with their 'care of' address. Whenever a node changes IP address, it updates its new IP address in the DHT. Nodes would then try to contact other nodes using the address currently stored in their routing table. If that lookup fails, they can find the new current address by sending a lookup for the nodes 'care of' address in the DHT. They would then update their routing table with the foreign node's new IP address and re-issue the lookup. This mechanism raises a concern about how to control which stored addresses a node can update. Consequently, further research would be required into the suitability of Mobile IP for Chameleon.

3. **Generic P2P functionality**

   The Chameleon algorithm does not address other research questions associated with P2P algorithms such as how to handle load balancing and the persistence of data stored in the network. These are regarded as future work.

8.5 Summary

In this Chapter the key achievements of this work have been highlighted. These include the design of a P2P algorithm called Chameleon which is efficient and suitable for WANs and capable of supporting the increasing number of incompatible service discovery techniques. In addition to this, multi-destination routing has been applied to P2P overlays for the first time, offering significant bandwidth savings.

A number of secondary achievements also arose from this work. Firstly, there is the design of the SIP Service which enables device and service discovery and control in a centralised WAN. In addition to this, the EDRA P2P algorithm was redesigned to ensure that stable performance of a 1-hop algorithm could be achieved, the result of which forms part of the successful implementation of the Chameleon P2P service overlay.

References

[1] Web Services Architecture. W3C Note, Feb 2004. `http://www.w3.org/TR/2004/NOTE-ws-arch-20040211/`.

[2] WSDL W3C Submission Request. W3C Note, 2001. `http://www.w3.org/TR/wsdl`.

[3] OSGi architecture, Sept 2008. `http://www.adhoco.com/technologies/java-osgi`.

[4] A.Brown, M.Kolberg, D. Bushmitch, M. Ma, and G. Lomako. A SIP-based OSGi device communication service for mobile personal area networks. In *IEEE Consumer Communications and Networking Conference*, pages 502–508, 2006.

[5] I. Stoica, R. Morris, D. Liben-Nowell, D. R Karger, M. F. Kaashoek, F. Dabek, and H. Balakrishnan. Chord: A scalable peer-to-peer lookup protocol for internet applications. *IEEE/ACM Trans*, 11(1):17–32, February 2003.

[6] B. Leong, B. Liskov, and E. D. Demaine. Epichord: Parallelizing the Chord lookup algorithm with reactive routing state management. *Computer Communications*, 29:1243–1259, 2006.

[7] L. R. Monnerat and C. L. Amorim. D1HT: A distributed one hop hash table. In *Proceedings of the 20th IEEE International Parallel & Distributed Processing Symposium (IPDPS)*, Rhodes Island, Greece, April 2006.

[8] J. Buford, A. Brown, and M. Kolberg. Meta service discovery. In *3rd International Workshop on Mobile Peer to Peer Computing, Pisa, Italy*, March 2006.

[9] M. Kolberg, F. Touvet, A. Brown, and J. Buford. A Markov model for the Epichord peer-to-peer overlay in an XCAST enabled network. In *IEEE International Conference on Communications (ICC)*, 2007.

[10] J. Buford, A.Brown, and M.Kolberg. Exploiting parallelism in the design of peer-to-peer overlays. *Computer Communications Journal, Elsevier Science*, 31(3):452–463, February 2007.

[11] J. Buford, A. Brown, and M. Kolberg. Parallelizing peer-to-peer overlay with multi-destination routing. In *IEEE Consumer Communications and Networking Conference (CCNC 2007)*, 2007.

[12] J. Buford, A. Brown, and M. Kolberg. Analysis of an active maintenance algorithm for an O(1)-hop overlay. In *IEEE Globecom 2007 General Symposium*, November 2007.

[13] A. Brown, M. Kolberg, and J. Buford. An adaptable service overlay for wide area network service discovery. In *IEEE Globecom 2007 Workshop - Enabling the Future Service-Oriented Internet. USA.*, 2007.

[14] OASIS. `http://www.oasis-open.org`.

[15] WTCS. `http://www.wtcs.org/snmp4tpc/jton.htm`.

[16] Luiz Andre Borroso, Jeffery Dean, and Urs Holze. Web search for a planet: The google cluster architecture. 23(2):22–28, April 2003.

[17] Yahoo. `http://www.yahoo.com/docs/info/faq.html`.

[18] Kazaa. `http://www.kazaa.com`.

[19] E. Guttman, C. Perkins, J. Veizades, and M. Day. `Service Location Protocol Version 2. IETF RFC 2608. June 1999.`

[20] Jini Architecture. Whitepaper, 1999. `http://www.sun.com/software/jini/whitepapers/architecture.pdf`.

[21] UPnP: Universal Plug and Play Forum. Website, Sept 2001. `http://www.w3.org/TR/wsdl`.

[22] M. Balazinska, H. Balakrishnan, and D. Karger. A scalable peer-to-peer architecture for intentional resource discovery. In *Pervasie Computing*, 2002.

[23] SOAP. W3C Recommendation, Apr 2001. `http://www.w3.org/TR/soap12-part0/`.

[24] UDDI. $http : //www.uddi.org/pubs/uddi_v 3.htm.$

[25] OMG. `http://www.omg.org/`.

[26] K. D. Zeilenga. Lightweight Directory Access Protocol LDAP. Internet Draft, Sept 2005. `draft-ietf-ldapbis-roadmap-08.txt`.

[27] Active Directory. `http://www.microsoft.com/windowsserver2003/technologies/directory/activedirectory/default.mspx`.

[28] NetBatch. `http://www.infoworld.com/articles/hn/xml/00/08/24/000824hnpeer.html`.

[29] Napster. `http://http://www.napster.co.uk`.

[30] M. Ripeanu, I. Foster, and A. Iamnitchi. Mapping the Gnutella network: Properties of large-scale peer-to-peer systems and implications for system design. *IEEE Internet Computing Journal*, 6(1), 2002.

[31] A. Brown and M. Kolberg. Tools for peer-to-peer network simulation, draft-irtf-p2prg-core-simulators-00.txt, work in progress. July. 2006.

[32] SSFNet Simulator. http:// http://www.neurogrid.net.

[33] Planetlab. http://www.planet-lab.org/.

[34] S Rhea, B Godfrey, B Karp, J Kubiatowicz, S Ratnasamy, and S. Shenker. Opendht: a public DHT service and its uses. In *2005 Conference on Applications, Technologies, Architectures, and Protocols For Computer Communications (Philadelphia, Pennsylvania, USA)*, August 2005.

[35] OSGi: The Open Services Gateway Initiative. Web site, Feb 2004. http://www.osgi.org.

[36] OSGi R4 Specificiation. Specification, May 2004. http://www.osgi.org/Specifications/HomePage.

[37] Jan S. Rellermeyer, Gustavo Alonso, and Timothy Roscoe. Building, deploying, and monitoring distributed applications with eclipse and r-osgi. In *eclipse '07: Proceedings of the 2007 OOPSLA workshop on eclipse technology eXchange*, pages 50–54, New York, NY, USA, 2007. ACM.

[38] Universal Plug and Play Device Architecture 1.0. Specification, April 2004. http://www.upnp.org/specs/arch/UPnP-arch-DeviceArchitecture-v1.0-20080424.pdf.

[39] R. Droms. Automated configuration of TCP/IP with DHCP. *IEEE Internet Computing*, 3(4):45–53, July 1999.

[40] Bluetooth v2.0. Specification, Nov 2004. http://www.bluetooth.com/Bluetooth/Technology/Building/Specifications/.

[41] Bluetooth. http://www.bluetooth.com.

[42] IrDA. http://www.irda.org.

[43] 802.11 Working Group. http://www.ieee802.org/11/.

[44] SIP. http://www.sipforum.org.

[45] S.M. Faccin-P and Lalwaney-B.Patil. Multimedia services: Analysis of mobile IP and SIP interactions in 3G networks. *IEEE Communications Magazine*, 42(1), January 2004.

[46] H. Schulzrinne and E. Wedlund. Application-layer mobility using SIP. *SIGMOBILE Mobile Computing Communications Review*, 4(3):47–57, 2000.

[47] S.Moyer S. Tsang, D. Marples. Accessing networked appliances using the session initiation protocol. In *International Conference of Communications, Helsinki, Finland*, June 2001.

[48] EMule. http://www.emule-project.net.

[49] Bittorrent. http://www.bittorrent.com.

[50] A. Rowstron and P. Druschel. Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems. In *IFIP/ACM Interna-*

tional Conference on Distributed Systems Platforms (Middleware), Heidelberg, Germany, pages 329–350, November 2001.

[51] I. Abraham, A. Badola, D. Bickson, D. Malkhi, S. Maloo, and S. Ron. Practical locality-awareness for large scale information sharing. In *ICTS*, 2005.

[52] X. Gu, K. Nahrstedt, and B. Yu. Spidernet: An integrated peer-to-peer service composition framework. In *Proceedings of the 13th IEEE international Symposium on High Performance Distributed Computing*, volume 01, pages 110–119, June 2004.

[53] Object Management Group. `http://www.omg.org`.

[54] K. Gottschalk, S. Graham, H. Kreger, and J. Snell. Introduction to web service architecture. *IBM Systems*, 41(2), 2002.

[55] Microsoft. Web Services Dynamic Discovery (WS-Discovery). Specification, Oct 2004. `http://schemas.xmlsoap.org/ws/2004/10/discovery/ws-discovery.pdf`.

[56] Salutation Architecture Specification (part 1) Version 2.0. Specification, 2005. `http://web.archive.org/web/20050627074915/http://www.salutation.org/`.

[57] JINI Network Technology. `http://www.jini.org`.

[58] Bernard Traversat, Ahkil Arora, Mohamed Abdelaziz, Mike Duigou, Carl Haywood, Jean-Christophe Hugly, Eric Pouyoul, and Bill Yeager. Project JXTA 2.0 Super-Peer Virtual Network, May 2003. `http://www.jxta.org`.

[59] J.Rosenberg, H.Schulzrinne, and B.Suter. IETF draft-ietf-svrloc-wasrv-01.txt. In *Work in progress*, November 1997.

[60] An-Cheng Huang and Peter Steenkiste. A flexible architecture for wide-area service discovery. In *The Third IEEE Conference on Open Architectures and Network Programming (OPENARCH 2000)*, March 2000.

[61] S. E Czerwinski, B. Y Zhao, T. D Hodes, A. D Joseph, and R. H. Katz. An architecture for a secure service discovery service. In *5th Annual ACM/IEEE international Conference on Mobile Computing and Networking (United States)*, August 1999.

[62] A. Mislove and P. Druschel. Providing administrative control and autonomy in structured peer-to-peer overlays. In *International Workshop on Peer-to-Peer Systems*, 2004.

[63] Netcraft, September 2005. `http://news.netcraft.com/archives/web_server_survey.html`.

[64] Online GIS Grid-Arendal and Maps Database. `http://www.grida.no/db/gis/prod/html/global_3.htm (29th September 2005)`.

[65] Online GIS Grid-Arendal and Maps Database. http://www.grida.no/db/gis/prod/html/glohuma1.htm (29th September 2005).

[66] Internet Systems Consortium. http://www.isc.org/index.pl?/ops/ds/ (29th September 2005).

[67] John C.-I. Chuang and Marvin A. Sirbu. Pricing multicast communication: A cost-based approach. *Telecommunication Systems*, 17(3):281–297, 2001.

[68] G. Phillips, S. Shenker, and H. Tangmunarunkit. Scaling of multicast trees: Comments on the Chuang-Sirbu scaling law. In *SIGCOMM*, pages 41–51, 1999.

[69] P. Van Mieghem, G. Hooghiemstra, and R. van der Hofstad. On the efficiency of multicast. *IEEE/ACM Trans. Netw.*, Volume 9(6):719–732, December 2001.

[70] Robert C. Chalmers and Kevin C. Almeroth. Modeling the branching characteristics and efficiency gains of global multicast trees. In *INFOCOM*, pages 449–458, 2001.

[71] Sonia Fahmy and Minseok Kwon. Characterizing overlay multicast networks and their costs. *IEEE/ACM Transactions on Networking*, 15(2):373–386, 2007.

[72] Lorenzo Aguilar. Datagram Routing for Internet Multicasting. In *SIGCOMM '84: Proceedings of the ACM SIGCOMM symposium on Communications architectures and protocols*, pages 58–63, New York, NY, USA, 1984. ACM.

[73] M. Ammar. Why Johnny can't multicast: Lessons about the evolution of the Internet. In *Keynote - Network and Operating System Support for Digital Audio and Video*, 2003.

[74] R. Boivie, N. Feldman, Y. Imai, W. Livens, D. Ooms, O. Paridaens, and E. Muramoto. Explicit multicast (XCAST) basic specification, draft-ooms-xcast-basic-spec-13.txt, Internet draft. expires january, 2008. http://tools.ietf.org/html/draft-ooms-xcast-basic-spec-13.

[75] Q. He and M. Ammar. Dynamic host-group multi-destination routing for multicast sessions. *Telecommunication Systems*, 28(3-4), 2005.

[76] Skype. `http://www.skype.com`.

[77] P2PSIP. `http://www.p2psip.org/ietf.php`.

[78] E. Lua, J. Crowcroft, M. Pias, R. Sharma, and S. Lim. A survey and comparison of peer-to-peer overlay network schemes. *IEEE Communications Surveys and Tutorials*, Volume 7(2), March 2005.

[79] K. Aberer, L. Onana Alima, A. Ghodsi, S. Girdzijauskas, M. Hauswirth, and S. Haridi. The essence of P2P: A reference architecture for overlay networks. In *The Fifth IEEE International Conference on Peer-to-Peer Computing*, Konstanz, Greece, 31 Aug - 2 Sep 2005.

[80] I. Gupta, K. Birman, P. Linga, A. Demers, and R. van Renesse. Kelips: building an efficient and stable P2P DHT through increased memory and background overhead. In *Proceedings of the 2nd International Workshop on Peer-to-Peer Systems (IPTPS 03)*, 2003.

[81] K. P. Birman, M. Hayden, O. Ozkasap, Z. Xiao, M. Budiu, and Y. Minsky. Bimodal multicast. *ACM Trans. Computing Systems*, 17(2):41–88, May 1999.

[82] T. Oh-ishi, K. Sakai, H. Matsumura, and A. Kurokawa. Architecture for a peer-to-peer network with IP multicasting. In *18th International Conference on Advanced Information Networking and Applications (AINA'04)*, volume 2, 2004.

[83] T. Oh-ishi, K. Sakai, H. Matsumura, and A. Kurokawa. Study of the relationship between peer-to-peer systems and ip multicasting. *IEEECOMMS*, Jan 2003.

[84] J. Li, J. Stribling, R. Morris, and M. F. Kaashoek. Bandwidth-efficient management of DHT routing tables. In *NSDI*, 2005.

[85] Robert C. Chalmers and Kevin C. Almeroth. Developing a multicast metric. In *IEEE Global Telecommunications Conference(GLOBECOM)*, pages 382–386, 2000.

[86] P. Van Mieghem, G. Hooghiemstra, and R. Van Der Hofstad. On the efficiency of multicast. *IEEE/ACM Transac. on Networking*, Volume 9(6), December 2001.

[87] Jinyang Li, Jeremy Stribling, Robert Morris, M. Frans Kaashoek, and Thomer M. Gil. A performance vs. cost framework for evaluating DHT design tradeoffs under churn. In *Proceedings of the 24th IEEE Infocom*, March 2005.

[88] A. Gupta, B. Liskov, and R. Rodrigues. Efficient routing for peer-to-peer overlays. In *Proceedings of the 1st Symposium on Networked Systems Design and Implementation (NSDI)*, pages 113–116), 2004.

[89] J. Li, J. Stribling, R. Morrris, and M. F. Kaashoek. Bandwidth-efficient management of DHT routing tables. In *2nd Symposium on Network Systems Desing and Implementation*, Boston, May 2005.

[90] B. Leong, B. Liskov, and E. D. Demaine. Epichord: Parallelizing the Chord lookup algorithm with reactive routing state management. In *12th International Conference on Networks (ICON)*, Singapore, November 2004.

[91] P. Antoniadis, C. Courcoubetis, and R. Mason. Comparing economic incentives in peer-to-peer networks. *Computer Networks Journal, Elsevier Science*, 46(1):133–146, 2004.

[92] P. Golle, K. Leyton-Brown, and I. Mironov. Incentives for sharing in peer-to-peer networks. In *Proceedings of 3rd ACM conference on Electronic Commerce*, pages 264–267, 2001.

[93] K. Lai, M. Feldman, I. Stoica, and J. Chuang. Incentives for cooperation in peer-to-peer networks. In *Proceedings of the Workshop on Economics of Peer-to-Peer Systems*, 2003.

[94] YellowPages. `http://www.yellowpages.com`.

[95] Scalable Adaptive Multicast. `http://www.samrg.org/`.

[96] XCAST-Testbed. `http://tools.ietf.org/id/draft-muramoto-irtf-sam-exp-testbed-00.txt (May 2007)` .

[97] Overlayweaver. `http://overlayweaver.sourceforge.net`.