UNIVERSITY OF LONDON
UNIVERSITY COLLEGE LONDON
Faculty of Engineering
Department of Electronic & Electrical Engineering

A Comparative Study of Ad Hoc & Peer to Peer Networks

by

Joseph Borg

Supervised by

Dr. Hermann De Meer

A dissertation submitted in partial fulfilment
of the requirements for the degree of

Master of Science in Telecommunications at the
University College London
AUGUST 2003

Abstract

The dramatic growth in mobile and Internet usage during the last decade has fundamentally changed the dynamics and inter-relations of the telecommunications industry. Two concepts that hold a similar *disruptive* potential on the telecommunications industry within the near future are Ad Hoc networking and Peer to Peer (P2P) networking.

Both Ad Hoc and P2P networking are disruptive technologies in that they 'threaten' to change the structure of the telecommunications industry fundamentally. The underlying concept by which they threaten to do this is common to both; namely decentralisation.

Despite this common concept, research and development has so far tended to treat the fields of Ad Hoc and P2P networking in isolation. The goal of this research work is, therefore, to identify commonalities between the fields of Ad Hoc and P2P networking, as a result of which, convergence of research work would be beneficial. This work also serves to identify areas where the divergence of research work would be beneficial.

The rest of this dissertation is structured as follows: Chapter 1 introduces this comparative study. Chapter 2 provides overviews of Ad Hoc and P2P networking. Chapter 3 provides a comparison of node/peer discovery in Ad Hoc and P2P networks, respectively. A comparison of the IETF Ad Hoc network routing protocols is provided in Chapter 4. Chapter 5 provides a comparison between content discovery and dissemination in P2P networks and routing in Ad Hoc networks. Identity, security, and anonymity in both network types are compared in Chapter 6. Similarly, Chapter 7 compares multicasting in both network types, whilst Chapter 8 compares network management and quality of service. In Chapter 9, a modification to the Chord content discovery technique is proposed, and results are presented. Chapter 10 then concludes this dissertation with a summary of the key conclusions from each chapter, and recommendations for future work.

Acknowledgements

First and foremost, I wish to express my sincere gratitude to my supervisor, Dr. Hermann De Meer, for having introduced me to the field and for his constant encouragement, guidance and support during the course of this project.

Secondly, I would like to thank Dr. Joseph Attard for being my mentor throughout the past year and a half. His guidance has been instrumental for me in pursuing this Masters degree.

I would also like to acknowledge the help of Mr. Geoffrey Cauchi, Dr. Gerd Kortuem, Dr. Sung-Ju Lee, Mr. Jian Li, Mr. Richard Ogier, Mr. Charles E. Perkins and Dr. Lidong Zhou, for answering various questions in relation to their work in the fields of Ad Hoc and Peer to Peer networking, and in relation to communications networks in general.

I am grateful to the British Council and the Chevening scholarship programme for providing the financial support which has enabled me to pursue this Masters degree.

I would like to thank my friends, Antoine, David and Melanie, together with all my colleagues at UCL, for having provided me with a pleasant working and living environment, and for their support throughout my stay in London.

Last but not least, I would like to wholeheartedly thank my parents, Joe and Isabelle, and my sister Maria Clara, for their unconditional support, encouragement and commitment throughout this Masters degree.

Joseph Borg
August 2003

"Anyone who has lost track of time when using a computer knows
the propensity to dream,
the urge to make dreams come true
and the tendency to miss lunch."

Tim Berners-Lee,
Inventor of the World Wide Web

List of Acronyms

ALAN:	Application Layer Active Networking.
ALM:	Application Layer Multicasting.
AMRoute:	Ad Hoc Multicast Routing Protocol.
ANMP:	Ad Hoc Network Management Protocol.
AODV:	Ad Hoc On-demand Distance Vector.
AOL:	Application Optimisation Layer.
ARPANET:	Advanced Research Project Agency Network.
AS:	Autonomous System.
AVP:	Active Virtual Peer.
BER:	Bit Error Rate.
BRAN:	Broadband Radio Access Networks.
BSS:	Base Station Subsystem.
CA:	Certifying Authority.
CAN:	Content-Addressable Network.
CERN:	Conseil Européen pour la Recherche Nucléaire.
CRC:	Cyclic Redundancy Check.
DCP2P:	Data-Centric Peer to Peer.
DDM:	Differential Destination Multicast.
DES:	Data Encryption Standard.
DHCP:	Dynamic Host Configuration Protocol.
DHT:	Distributed Hash Table.
DNS:	Domain Name System.
DoS:	Denial of Service.
DSR:	Dynamic Source Routing.
ETSI:	European Telecommunications Standards Institute.

FTP:	File Transfer Protocol.
GloMo:	Global Mobile Information Systems.
GSM:	Global System for Mobile Communications.
HIPERLAN:	High Performance Radio Local Area Network.
ICQ:	I 'Seek You'.
ICV:	Integrity Check Value.
IEEE:	Institute of Electrical and Electronics Engineers.
IESG:	Internet Engineering Steering Group.
IETF:	Internet Engineering Task Force.
IP:	Internet Protocol.
ISM:	Industrial Scientific Medical.
ISO:	International Standards Organisation.
IV:	Initialisation Vector.
JXTA:	Juxtapose.
LAN:	Local Area Network.
MAC:	Medium Access Control (in the context of OSI layering).
MAC:	Message Authentication Code (in the context of security).
MACT:	Multicast Activation.
MANET:	Mobile Ad Hoc Network.
MAODV:	Multicast AODV.
MD-5:	Message Digest-5.
MIB:	Management Information Block.
ML:	Member List.
MP3:	MPEG Audio Layer 3.
MPEG:	Motion Pictures Expert Group.
MPR:	Multi-Point Relay.
MSN:	Microsoft Network.
MT:	Mobile Terminal.
NAT:	Network Address Translation.
NOL:	Network Optimisation Layer.
NSFNET:	National Science Foundation Network.

NTDR:	Near-Term Digital Radio.
ODMRP:	On-Demand Multicast Routing Protocol.
OLSR:	Optimised Link State Routing.
OSI:	Open Systems Interconnection.
P2P:	Peer to Peer, Peer 2 Peer, Peer-to-Peer.
PAN:	Personal Area Network.
PDA:	Personal Digital Assistant.
PKC:	Public Key Cryptography.
PRNET:	Packet Radio Network.
QoS:	Quality of Service.
RC-4:	Ron's Code-4.
RERR:	Route Error.
RFC:	Request For Comments.
RP:	Rendezvous Point.
RREP:	Route Reply.
RREQ:	Route Request.
RSA:	Rivest Shamir Adelman.
SETI:	Search for Extra Terrestrial Intelligence.
SHA-1:	Secure Hash Algorithm-1.
SNMP:	Simple Network Management Protocol.
SSL:	Secure Sockets Layer.
SURAN:	Survivable Adaptive Radio Networks.
TBRPF:	Topology dissemination Based on Reverse Path Forwarding.
TC:	Topology Control.
TCP:	Transmission Control Protocol.
TLS:	Transport Layer Security.
TTL:	Time-To-Live.
UCP2P:	User-Centric Peer to Peer.
UDP:	User Datagram Protocol.
VCC:	Virtual Control Cache.
VoD:	Video on Demand.

WEP:	Wired Equivalent Privacy.
WiFi:	Wireless Fidelity.
WLAN:	Wireless LAN.
WPA:	WiFi Protected Access.
WWW:	World Wide Web.
XML:	Extensible Mark-up Language.
XMPP:	Extensible Messaging and Presence Protocol.

List of Equations

List of Figures

List of Tables

Table of Contents

1. Introduction

1.1. Introduction

The dramatic growth in mobile and Internet usage during the last decade has fundamentally changed the dynamics and inter-relations of the telecommunications industry. Two concepts that hold a similar *disruptive* potential on the telecommunications industry within the near future are Ad Hoc networking and Peer to Peer (P2P) networking.

The origins of Ad Hoc networking lie in a 1972 US Department of Defence project known as Packet Radio Network (PRNET) [1], while the Internet as originally conceived in the late 1960s, was a P2P network [2].

Progress in Mobile Ad Hoc Networks (MANETs) has, until recently, been hindered by the lack of a number of necessary enabling technologies, such as mobile computing and digital wireless communication. The advances made throughout the 1990s in these enabling technologies have, however, refuelled interest in MANETs, as a result of which the field is now attracting much attention [3]. Large-scale systems have recently been fielded for US military applications, whilst commercial MANET devices, based on standards such as Bluetooth and Institute of Electrical and Electronic Engineers (IEEE) 802.11, have also been introduced in the market.

On the other hand, throughout its history, the popularity of P2P networking has been dictated by whether a centralised or a decentralised network architecture was the architecture of choice at the time. Since P2P networking is inherently decentralised in nature, it lost popularity whenever centralised architectures were preferred. In particular, the use of client/server networking as the architecture of choice for the Internet's application layer diminished the popularity of P2P networking throughout the 1990s.

More recently, a shift in the opposite direction has been attributed to Napster; a music file sharing P2P Internet application launched in 1999. The popularity that Napster gained within a short time span stimulated the development of several other P2P Internet applications which are, to date, primarily used for content sharing or instant messaging. Research interest in P2P networking has also been stimulated as a result of this shift. In fact P2P networking is now thought of as one of the main technologies that will shape the Internet's future [2].

Both Ad Hoc and P2P networking are considered disruptive technologies in that they 'threaten' to change the structure of the telecommunications industry in a fundamental manner. The underlying concept by which they threaten to do this is common to both; namely decentralisation. In P2P networking, this concept is achieved by virtue of the fact that P2P network applications are designed in such a way so that no distinction in role between peers exists; thereby reducing the role of the 'man in the middle' to that of a bit carrier. Ad Hoc networking goes further by eliminating the need for a network operator in the middle altogether. This is because all nodes in a MANET rely on each other for the delivery of bits to other nodes in the MANET.

Despite this common underlying concept, research and development has so far tended to treat the fields of Ad Hoc and P2P networking in isolation, and surprisingly little may be found in the literature that builds on their similarities. In [4], a comparison that is limited to routing in Ad Hoc networks and content discovery in P2P networks, is provided. In [5] and [6], work is presented which builds on noted similarities between Ad Hoc and P2P networking. The work presented in [5] deals with mobile Ad Hoc information systems, whilst that in [6] deals with content discovery in Ad Hoc networks.

The primary goal of this research work is to identify commonalities between the fields of Ad Hoc and P2P networking, as a result of which, convergence of research work and the cross-pollination of ideas would be beneficial. This work also serves to identify areas where the divergence of research work would be beneficial. The applicability of techniques used in one network type, to the other network type, has also been analysed.

1.2. Contributions

The main contributions of this work are summarised below:

- The main aspects that constitute Ad Hoc and P2P networking, respectively, have been identified. The novelty in this work was due to the emphasis placed on the identification of the similarities and differences between the two types of networks.
- The aspects of node discovery in Ad Hoc networks and peer discovery in P2P networks were compared. The comparison identifies fundamental differences between the two respective aspects, suggesting that the divergence of research work is beneficial.
- We have also compared the four Internet Engineering Task Force (IETF) MANET routing protocols currently undergoing standardisation.
- Similarities between routing in Ad Hoc networks and content discovery in P2P networks have been investigated. An analysis of the applicability of content discovery techniques used in P2P networks, to Ad Hoc networks, has also been carried out. This analysis identifies that some of the content discovery techniques can be applied to Ad Hoc networks.
- We have compared identity, security and anonymity in Ad Hoc and P2P networks. In this comparison, we have also analysed the applicability of techniques, used in one network type, to the other and it was noted that a number of techniques are directly applicable, due to the similarity in these aspects for both network types.
- We have compared the aspects of multicasting, network management, and Quality of Service (QoS) in Ad Hoc and P2P networks. We have also analysed the applicability of techniques, used in one network type, to the other and, in particular, noted that multicasting algorithms used in Ad Hoc networks are applicable to Application Layer Multicasting (ALM) in P2P networks.
- A modification to Chord; a content discovery technique for P2P networks is proposed, in order to improve its efficiency. A mathematical model of this modification was derived and the efficiency of Chord with and without the modification was compared. The results show that the modification improves Chord's efficiency in the vast majority of usage scenarios.

1.3. Dissertation Structure

The rest of this dissertation is structured as follows. The next chapter provides overviews of Ad Hoc and P2P networking. Each overview includes a definition of the network type, an account of its evolution, and the identification of aspects that constitute the network type. The chapter then concludes with an analysis of the relationship between both network types, and a review of key implementations and standardisation initiatives.

Chapter 3 deals with neighbouring node discovery in Ad Hoc networks and peer discovery in P2P networks. The chapter is concluded with a comparison of these respective aspects.

Chapters 4 and 5 deal with routing in Ad Hoc networks and content discovery in P2P networks, respectively. Chapter 4 compares the four IETF MANET routing protocols. Chapter 5 provides a comparison between Ad Hoc routing and P2P content discovery following which, content dissemination in P2P networks is reviewed, and the applicability of content discovery techniques used in P2P networks, to Ad Hoc networks, is analysed.

The aspects of identity, security and anonymity are dealt with in Chapter 6 for both network types. A comparison is provided in the chapter, which also analyses the applicability of techniques, used in one network type, to the other. In a similar manner, Chapters 7 and 8 provide the equivalent for the aspects of multicasting, and network management and QoS, respectively.

In Chapter 9, a modification to Chord; a content discovery technique for P2P networks, is proposed. Following an explanation of the problem statement that the modification addresses, a description of the modification is given. In turn, this is followed by the derivation of equations, which are then used to compare the relative performance of the modification with the original. The chapter is then concluded with a comparison of the modification's advantages and disadvantages, and an outline of possible future work thereon.

Finally, Chapter 10 concludes this dissertation with a summary of the key conclusions from the comparisons and an abstract of future work that may follow from this study.

2. Overviews of Ad Hoc & P2P Networks

2.1. Introduction

The concepts behind Ad Hoc and P2P networking both date back circa 30 years and find their origins in US Department of Defence funded projects, whose main objective was the construction of resilient military networks. A lot of progress has been achieved since then, both in research and development, as well as in the application of the two network types.

This chapter provides the necessary overviews for both network types, so that a better understanding of the subsequent chapters is possible. In the following six sub-sections, for Ad Hoc networking and subsequently for P2P networking, a definition of the network type and an account of its evolution are provided, and the aspects that constitute the network type are identified. Section 2.8 then reviews the relation between the fields of Ad Hoc and P2P networking following which; Section 2.9 reviews the respective standardisation initiatives.

2.2. Definition of Ad Hoc Networking

As understood presently, *an Ad Hoc network may be defined as a self-organising, self-healing wireless network in which mobile nodes are responsible for discovery of each other and subsequent cooperation so that communication is possible* [1], [5].

The word ‘Ad Hoc’ is defined as: Improvised and often impromptu [7]. For this reason when the term is applied to an Ad Hoc network, it generally implies that the network, unlike a fixed network or a cellular one, cannot rely on the presence of fixed infrastructure and/or human intervention. The network should therefore be capable of self-organisation for communication to be possible and should be self-healing in view of network failures.

2.2.1. Comparison with Cellular Networks

In order to evaluate the properties of an Ad Hoc network, it is useful to benchmark it with a commercial Global System for Mobile Communications (GSM) type cellular network consisting of fixed Base Station Subsystems (BSSs) and Mobile Terminals (MTs). In a cellular network it is necessary to plan the coverage and capacity a priori. In Ad Hoc networks coverage is entirely dependant on the amount of nodes present and the distribution of these nodes; both of which are dynamic in nature. Resources such as computational power, bandwidth, storage capacity and battery life are scarce at all nodes in an Ad Hoc network, unlike cellular networks where this is the case only at the MTs. Security is also a bigger issue since trust is not inherent to nodes in an Ad Hoc network, as it is to a BSS in a commercial cellular network.

On the other hand, an Ad Hoc network has several advantages. Since an Ad Hoc network does not rely on a fixed infrastructure, no single point of failure is introduced in the network, thereby making the network more resilient. Moreover, it can be rapidly deployed in areas where a fixed infrastructure is not available, is not trusted or cannot be relied on [1]. Finally, a densely populated Ad Hoc network also inherently achieves redundancy of routes, thereby further improving resiliency.

2.2.2. Current and Future Applications

The applications of Ad Hoc networks can be categorised as follows:

1) *Military applications*: Ad Hoc networks are particularly suited to the battlefield scenario where soldiers require a portable, instantaneous and resilient communications system operating in a hostile environment. As noted in Chapter 1, such networks have recently been fielded by the US military.
2) *Commercial applications*: The current application of commercially available Ad Hoc networking devices is for the creation of Personal Area Networks (PANs). In a PAN, a user's electronic devices, such as a laptop, mobile phone, and Personal Digital Assistant (PDA) collaborate amongst each other for data exchange and synchronisation. Future applications of commercial Ad Hoc networks will include networking in temporary offices, in classrooms or at conferences [1]. To some

extent, these applications are already possible; albeit restricted to communication between neighbouring nodes. This restriction arises because Ad Hoc routing protocols have not yet been standardised.

3) *Emergency and rescue applications*: Ad Hoc networks in the future could be deployed in emergency and rescue situations where the fixed infrastructure may have been destroyed due to an earthquake or other catastrophe.
4) *Sensor networks*: Ad Hoc networks may also connect sensor networks. A typical application of such networks is for the collection of environmental data, such as pollution measurements. Nodes for this application would have to be small, low cost and low power, so that large numbers could be deployed into an area of interest.

2.3. Evolution of Ad Hoc Networks

The 1972 US Department of Defence sponsored project PRNET evolved into the Survivable Adaptive Radio Networks (SURAN) program in the early 1980s. In both cases, the scope for the projects was military in nature; that of providing packet-switched networking in a battlefield, without relying on any fixed infrastructure.

The advent of mobile computing and viable wireless digital communication, approximately a decade later, rekindled interest in such networks. The European Telecommunications Standards Institute (ETSI) started standardisation of the High Performance Radio Local Area Network/1 (HIPERLAN/1) in 1991 [8]; an Ad Hoc networks standard. The IEEE 802.11 Wireless Local Area Network (WLAN) working group adopted the term 'Ad Hoc' for such networks and included them in its standardisation initiatives. In turn, the US Department of Defence funded other projects related to Ad Hoc networks, including the Global Mobile Information Systems (GloMo) and Near-Term Digital Radio (NTDR) [1].

In 1996, ETSI released the first edition of the HIPERLAN/1 standard. 1997 was a year of considerable activity for commercial Ad Hoc networking. The IEEE 802.11 working group released its first standard, IEEE 802.11 [9]. ETSI formed the Broadband Radio Access Networks (BRAN) working group to work on HIPERLAN/2. The IETF formed the MANET charter with the intent of investigating and standardising routing protocols for

MANETs. In 1998, the Bluetooth special interest group was founded to develop a cable replacement (PAN) technology for electronic devices, based on Ad Hoc networking. Since then, standardisation has progressed under these standardisation bodies as follows:

- The Bluetooth special interest group released v. 1.0 of the standard in 1999 [10]. Since then, Bluetooth standardisation has been carried out in close cooperation with the IEEE 802.15 PAN working group, resulting in the release of the IEEE 802.15.1 standard in 2002, this being based on the Bluetooth specification v. 1.1.
- The ETSI BRAN working group released the first edition of the HIPERLAN/2 standard in 2000 [11].
- The IEEE 802.11 working group released two other standards in 1999, IEEE 802.11a and IEEE 802.11b [9], and another one this year, IEEE 802.11g.
- The IETF MANET charter has submitted four Internet drafts to the Internet Engineering Steering Group (IESG) this year for the publication of Request For Comments (RFC) documents. To date, one document has been published.

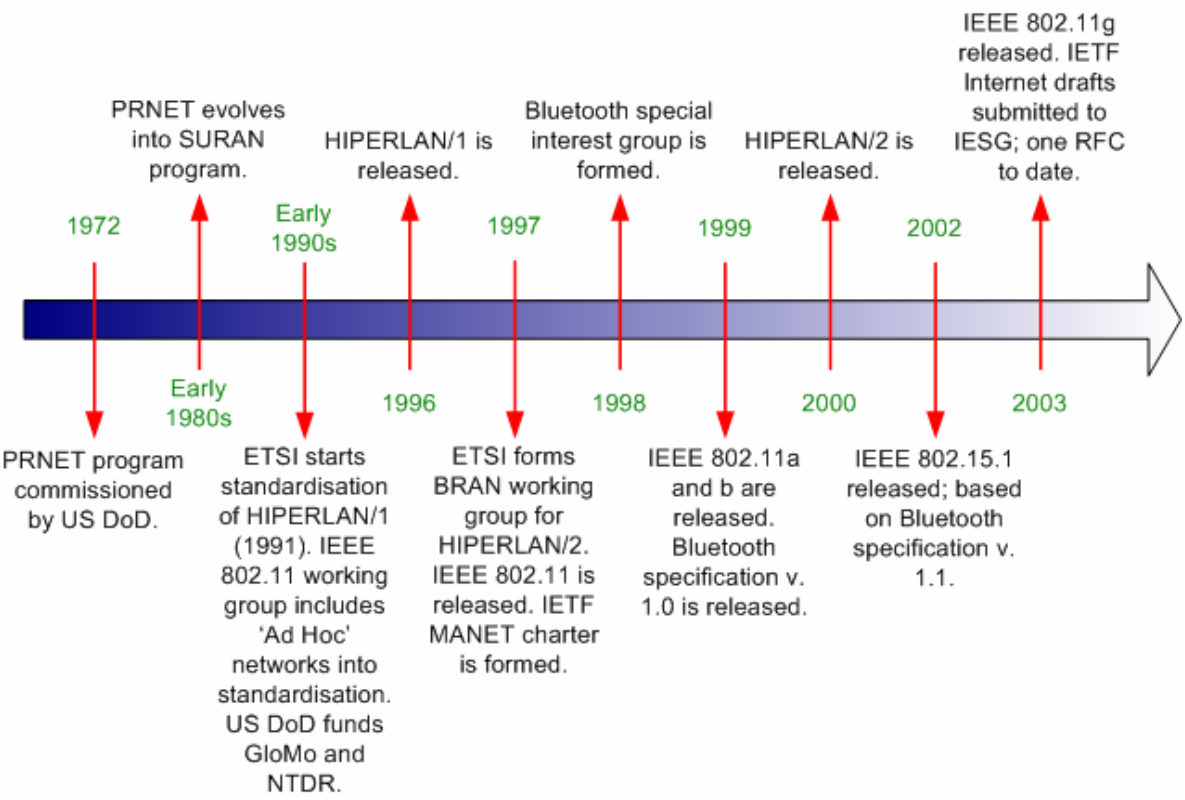

Figure 2.1: Timeline of Ad Hoc networking evolution.

Figure 2.1 summarises the evolution of Ad Hoc networking. The advent of commercial Ad Hoc networking is marked by the release of the IEEE 802.11b and Bluetooth v. 1.0 standards in 1999, due to the introduction of devices based on these standards to the market and the widespread popularity that these devices gained.

2.4. Aspects of Ad Hoc Networking

Figure 2.2 illustrates the aspects that constitute Ad Hoc networking. As may be observed, there are a total of eight aspects. The aspect of node discovery is in turn subdivided into two sub-aspects, these being neighbouring and non-neighbouring node discovery.

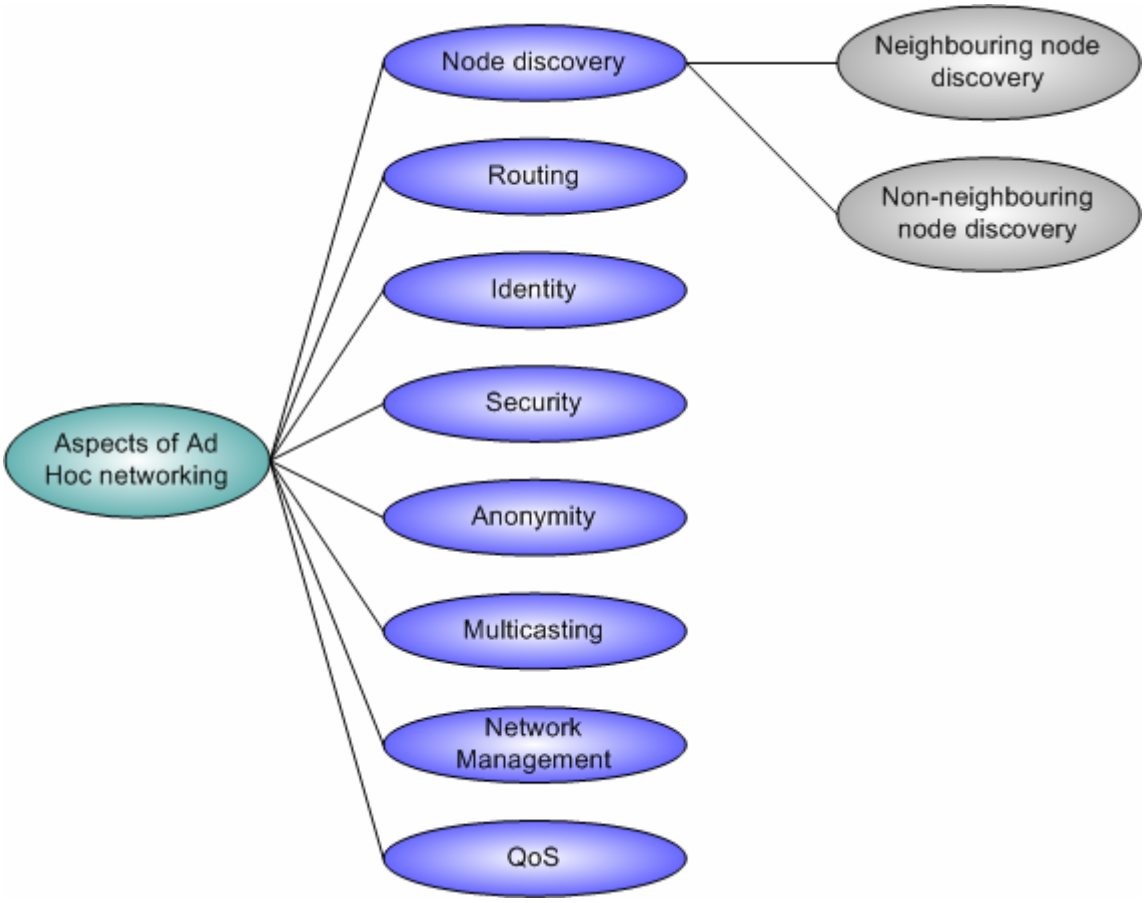

Figure 2.2: Aspects of Ad Hoc networking.

2.5. Definition of P2P Networking

As understood presently, *a P2P network may be defined as an application layer overlay network in which all entities are equal and all contribute some of their resources, thus giving rise to a network in which each entity (peer) is both a content requestor and a content provider* [5], [12].

The word 'peer' is defined as: To be, or to assume to be, equal [7]. In essence, therefore, P2P means: Equal communicating with equal [13]. The importance of the definition lies in the word 'equal', as it implies that no distinction theoretically exists between the entities that make up the network. Each peer is therefore analogous to both a client (as it can request and obtain content or services from other peers), and a server (as it should provide content or services to all other peers). The term *servent* (server/client) is used in literature to denote this fact [12].

In practice, P2P networks are divided into two main types: *Pure* and *hybrid*. A pure P2P network is one which fully conforms to the definition just given and, from a theoretical perspective, is the only true P2P network [13]. Within a pure P2P network, any arbitrarily chosen node may be removed from the network, without effecting the operation of the network [12]. A pure P2P network is also referred to as an *atomistic* P2P network, implying that each node is self contained and has all the necessary features required to participate.

A hybrid P2P network is one that conforms to the above definition of P2P only for data dissemination, and not for the discovery of peers or content. This is because discovery of peers or content is performed with the use of a server. For this reason, a hybrid P2P network is also referred to as a *server-mediated* P2P network [14]. The server typically consists of a database that can be queried by peers and is classified as either User-Centric P2P (UCP2P) or Data-Centric P2P (DCP2P) [13].

A UCP2P server maintains a database of all peers. This database is used by peers to search for other peers, inquire about their current status, and obtain their connection parameters. A DCP2P server maintains a database of content and the corresponding peers from which the content may be obtained. This database is used by peers to search for content and obtain

connection identifiers for the peers that store the desired content, once the required content is found. As may be expected, UCP2P servers are typically used in instant messaging P2P networks, whilst DCP2P servers are used in content sharing P2P networks.

2.5.1. Comparison with Client/Server Networks

In order to evaluate the properties of a P2P network, it is useful to benchmark it with a client/server network, this being the most established application layer network architecture over the Internet.

Within a client/server network, a server provides all the necessary content or services [12]. For it to do so, it is generally equipped with a relatively high degree of resources, such as processing power, network bandwidth and storage capacity. The clients on the other hand need not have such a high degree of resources since they only need to obtain content or request services from the server, without sharing any resources of their own with other clients, or with the server [12].

An important attribute of client/server networks is that there is a one-to-many relationship between server and clients, thus resulting in a strict two level hierarchy in which the server is the only entity at the top level. The advantage derived from using this highly centralised network architecture is that of having entire control of the network at the server. Thus, from an operations and management perspective, the client/server network architecture is generally the architecture of choice.

On the other hand, when compared to P2P networks, client/server networks have the following drawbacks:

- The system's 'intelligence' resides at one entity (i.e. the server). A single point of failure is therefore introduced.
- The one-to-many relationship between server and clients reduces scalability since the server has to be capable of handling all the requests received from clients per unit time. If the amount of requests per unit time exceeds the server's capabilities, a loss of service is experienced.

- When used over the Internet, client/server networks are bandwidth inefficient, since this strict two-level hierarchy does not efficiently map to the decentralised architecture of the Internet's network layer. Therefore, when client/server application layer traffic is mapped onto the Internet's network layer, a significant amount of network layer links may be underutilised.

It is important to note that, all of these disadvantages ultimately translate into additional financial cost, due to the redundancy/resiliency features required, the server capacity required, and due to bandwidth underutilisation, respectively.

2.5.2. Current and Future Applications

The current applications of P2P networks can be categorised as follows:

1) *Content sharing*: Several content sharing P2P networks have been launched since Napster. In these types of applications, a user typically shares some locally stored content, such as music files and video files, with other peers on the P2P network. In turn, the user may search for and discover desired content at other peers. Given that the desired content is found, it may then be downloaded from these peers.
2) *Instant messaging*: A number of instant messaging P2P networks are also widely used over the Internet. In this type of application, a user maintains a contact list of other users (i.e. peers). When any of these users are online, the given user may chat with them. Group conversations are typically also supported.
3) *Distributed processing*: In such applications a server typically makes use of peers registered with it to process raw data with otherwise idle resources [13]. In our opinion, although peers do give up some of their resources for the application at hand, this type of application does not adhere to the definition of P2P[1], since the server is the sole provider of the raw data. This application is perhaps more accurately classified under Grid computing, in which a number of data centres collaborate together in order to perform large scale computations [14].

[1] This type of application has been included here since it is often classified as P2P in literature.

Several applications may benefit from P2P networking in the future. It is important to note that, besides as yet unthought-of applications, most of the current applications that use client/server networking may also be applied to P2P networking. The following is a list of some of the applications that may benefit from P2P networking in the future:

- The World Wide Web (WWW), in its current form is client/server in nature, since web servers are used to host websites, whilst web browsers (i.e. clients) are used to browse through these websites. Having said this, the Web was originally intended to be P2P in nature [13], and may reassume this form in the future. In this case, each peer would host the user's website, whilst also providing the facility to browse through the websites of other peers.
- Web search engines can be designed to be distributed across peers in a P2P network, instead of having a server provide this facility. A key advantage that would be gained from doing this is that scalability is improved, since the task of storing entries for websites and retrieving entries that match the request would be distributed. This would allow for more entries to be maintained [2].
- Several applications that require a large degree of fault tolerance could benefit from the resilient nature of P2P networking. One such application is the Domain Name System (DNS). Instead of using the hierarchy of servers that DNS is currently based on, DNS could be distributed across a P2P network, and could operate in a similar manner to a content sharing P2P network.
- The concept of offering services, as opposed to content, could also be applied to P2P networks in the future. In this case, a peer would search for the desired service at other peers in the P2P network, and would be able to make use of it upon locating it. Alternatively, a number of peers could collaborate in providing the desired service. Examples of the services offered could include web hosting, Video on Demand (VoD), as well as the equivalent of Web services for P2P networks.
- Multiplayer gaming applications may also be adapted for P2P networking. To some extent, multiplayer gaming applications based on P2P networking are already available. These applications are, however, generally restricted to a Local Area Network (LAN) environment. Internet based multiplayer games usually rely on the use of a known server.

2.6. Evolution of P2P Networks

The Internet bears its origins to a 1966 US Department of Defence funded project called Advanced Research Project Agency Network (ARPANET). The project saw initial completion three years later and was intended to address the following two needs [15]:

1) A need to share expensive computing and communications resources amongst university researchers.
2) The US military's desire to have a communications infrastructure that was robust against a large-scale nuclear attack by the former Soviet Union.

The first need was met with the adoption of wide area packet-switching as the technology of choice and hence, the development of routing. As the ARPANET grew, the need to internetwork different data-link layer technologies with a standardised protocol suite was recognized, and addressed throughout the 1970s, culminating in the standardisation of the Transmission Control Protocol/Internet Protocol (TCP/IP) suite in 1980 [16]. Interestingly, the history of Ad Hoc networks and that of the Internet cross paths since the first formal use of TCP was in 1976 for internetworking between PRNET and ARPANET [15], [17].

More importantly, the second need was met by designing the network to be decentralised, thus ensuring that no single point of failure is present, and that the existence of multiple paths to a destination is possible. This design constraint held true at the network layer and at the application layer of the ARPANET, and is the reason why the Internet's precursor may be thought of as the first P2P network.

Eventually, several other networks were connected to the ARPANET throughout the 1980s, after the latter adopted TCP/IP officially in 1983 [16]. This period essentially marks the beginning of the Internet. ARPANET itself was officially disbanded in 1990, since the National Science Foundation Network (NSFNET) was to replace the ARPANET as the Internet's backbone [18]. Throughout this period of time, the Internet remained primarily a P2P network.

From a network layer perspective, the only relevant modification since this period was the use of Autonomous Systems (ASs), after the concept was introduced in 1986 in order to

improve the scalability of network layer routing. The concept of an AS introduced a two-level hierarchy in routing; creating a distinction between intra-AS routing and inter-AS routing. Although the degree of decentralisation was reduced with the introduction of ASs, decentralisation is still maintained since the existence of multiple paths is nonetheless possible, and since an AS may make use of multiple inter-AS gateways, hence avoiding the creation of single points of failure. In conclusion, at the network layer, the Internet fundamentally remains a P2P network to date, albeit one with a hierarchy.

On the other hand, the application layer went through considerable changes with the commercialisation of the Internet in 1994 [2]. Prior to this, most of the applications in use at the time can be considered as P2P. Usenet newsgroups, for example, operated in a P2P manner. The WWW, released in 1992 by the Conseil Européen pour la Recherche Nucléaire (CERN) [18], was also intended as a P2P application. Other applications such as Telnet and File Transfer Protocol (FTP) applications, although designed to operate in a client/server manner, could also be classified as P2P since every connected computer typically ran both server and client software, thereby creating no disparity between the amount of servers and clients [2].

The commercialisation of the Internet brought about with it the dominance of client/server networking at the Internet's application layer. Amongst the several technical and commercial reasons cited for this radical shift are the following [2], [13]:

- The shortage in IP addresses led to the use of protocols, such as the Dynamic Host Configuration Protocol (DHCP) for dynamic IP address assignment. This implied that the IP address no longer represented a permanent identity, and therefore, that server software could no longer be run on every computer.
- The release of the Mosaic browser as a standalone web client as opposed to an integrated web client/server application.
- The use of dial-up modems for connecting users and a non-flat rate tariffing scheme implied that the connectivity of the average user was transient in nature; an added reason for the inability to run server software.

- The use of technologies, such as firewalls and Network Address Translation (NAT) further compounded the problem of running server software.
- The operations and management requirement of Internet-based business ventures implied that client/server networking provided a better solution.

Client/server networking remained the de facto standard, throughout the rest of the 1990s. Popular P2P networks, such as I 'Seek You' (ICQ) (launched in 1996), were also used throughout this period, however, they failed to attract the attention that Napster did in 2000; a year after its launch. Since then, several P2P networks have been launched to date. These networks include Gnutella and FastTrack for content sharing, Yahoo! Messenger and Microsoft Network (MSN) Messenger for instant messaging, and Search for Extra Terrestrial Intelligence (SETI) @home for distributed processing. More importantly, these networks embody what is currently understood by P2P networking: An application layer overlay network, generally using the Internet as the underlying network. Although not as numerous and active as in Ad Hoc networking, standardisation initiatives have also started since the year 2000. These are Jabber for instant messaging and the Juxtapose (JXTA) project for a P2P framework. Figure 2.3 summarises the evolution of P2P networking.

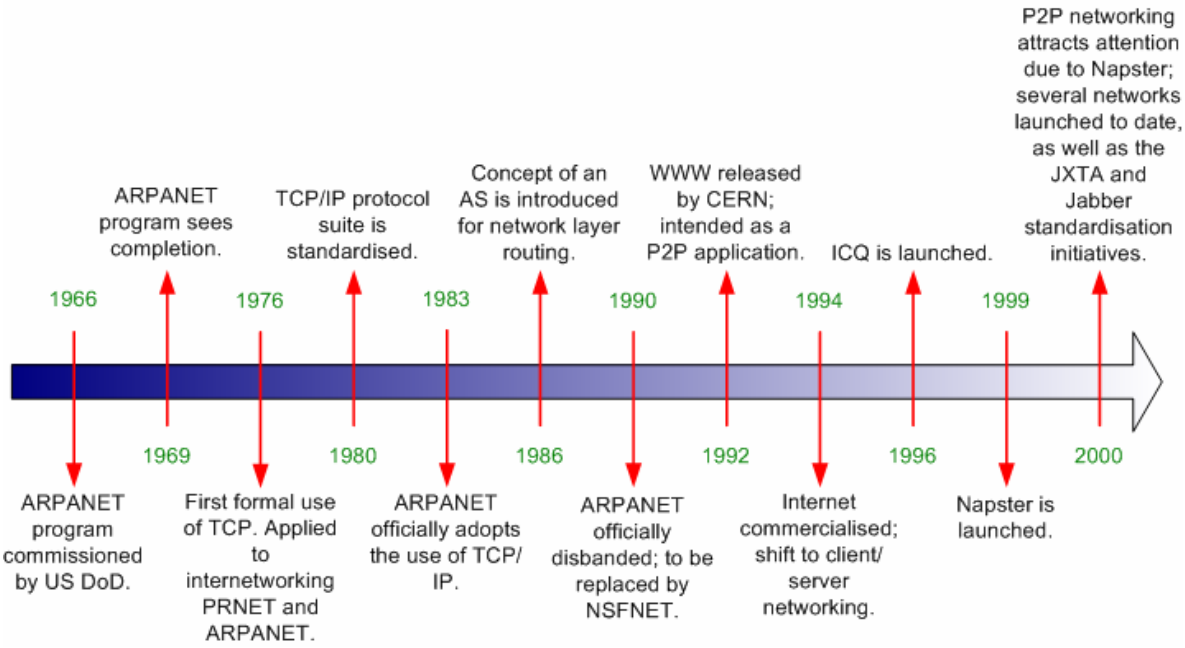

Figure 2.3: Timeline of P2P networking evolution.

2.7. Aspects of P2P Networking

Figure 2.4 illustrates the aspects that constitute P2P networking. As may be observed, there are also a total of eight aspects. In this case, however, three of the aspects are subdivided into two sub-aspects each. Content discovery and dissemination subdivides into content discovery and content dissemination. The aspect of identity subdivides into user identification and content identification. Finally, the aspect of network management subdivides into monitoring and control, and free riding.

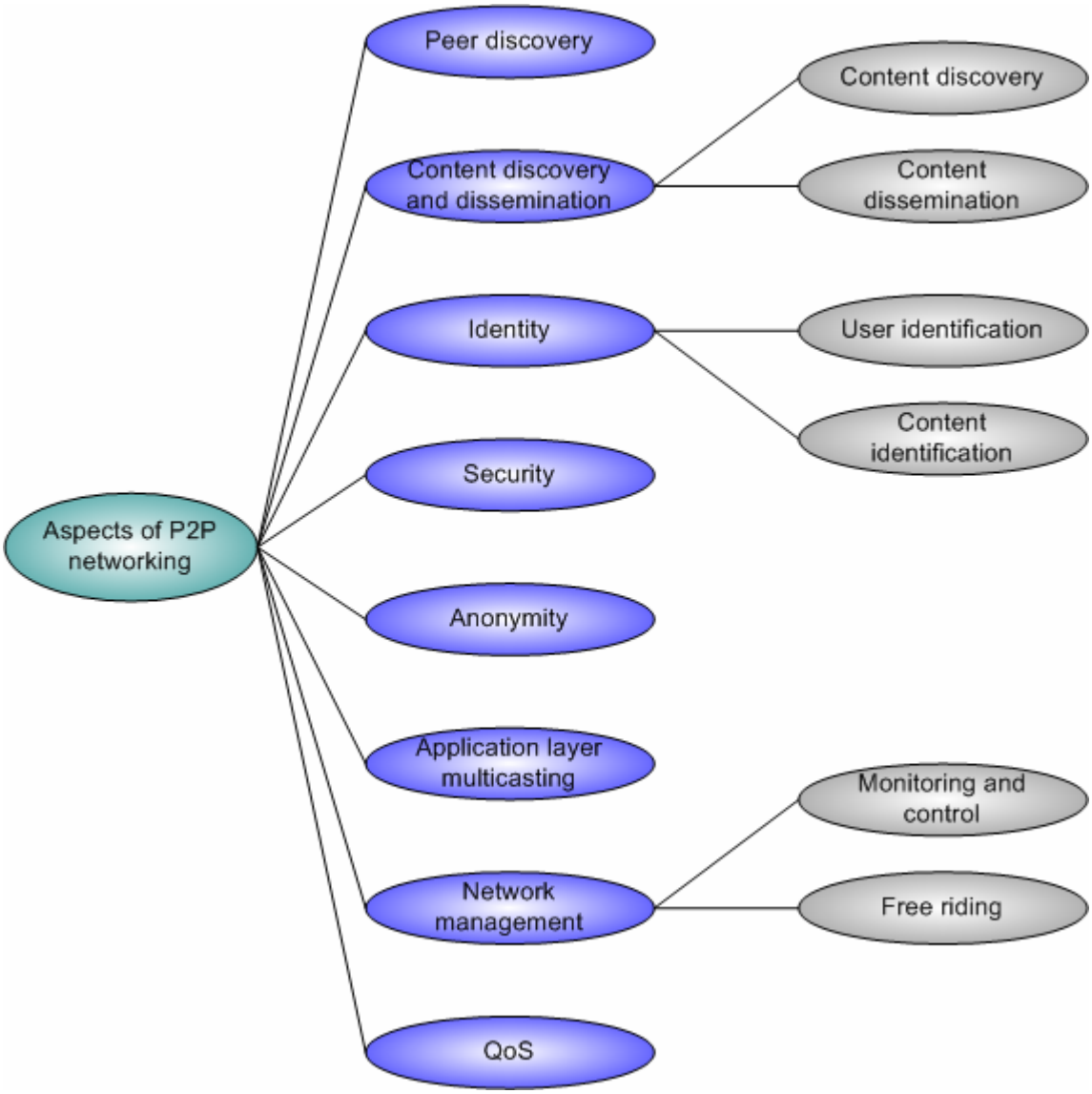

Figure 2.4: Aspects of P2P networking.

2.8. Relationship of Ad Hoc and P2P Networking

A closer look at the overviews of Ad Hoc and P2P networking in the previous sections of this chapter reveals a number of commonalities and differences between the two network types. In view of the subsequent chapters, it is important to note these fundamental commonalities between Ad Hoc and P2P networking:

- *Decentralised architectures*: In both network types, the network architecture is inherently decentralised in nature.
- *Transient connectivity*: The connectivity of nodes or peers in Ad Hoc or P2P networks respectively, is transient. In Ad Hoc networks, this is due to node mobility. In current P2P networks, this is primarily due to the lack of permanent Internet connectivity or a static IP address, at most peers.
- *Heterogeneity of resources*: The resources available to nodes or peers are not equal. Several different electronic devices, such as a laptop, mobile phone or PDA, may form part of an Ad Hoc network. All of these devices typically have differences in computational power, storage capacity and battery life. Similarly, computers running the same P2P application may vary significantly in specification.

On the other hand, the following differences should also be noted:

- *Network size*: Ad Hoc networks are generally limited to a few hundred nodes in size, since scalability is limited due to resource constraints. In contrast, current P2P networks run over the Internet and therefore, potentially have to scale so as to accommodate several millions of peers.
- *Relevance in Open Systems Interconnection (OSI) model*: Current research and development in the field of Ad Hoc networking is focused on the lower three layers of the OSI model; the network layer in particular. Conversely, research and development in P2P networking is presently concerned with application layer P2P overlay networks. Therefore, the top three layers of the OSI model are relevant in P2P networking. In summary Ad Hoc networking generally deals with application-independent network issues, whilst P2P networking generally deals with application-oriented network issues. This is illustrated in Figure 2.5.

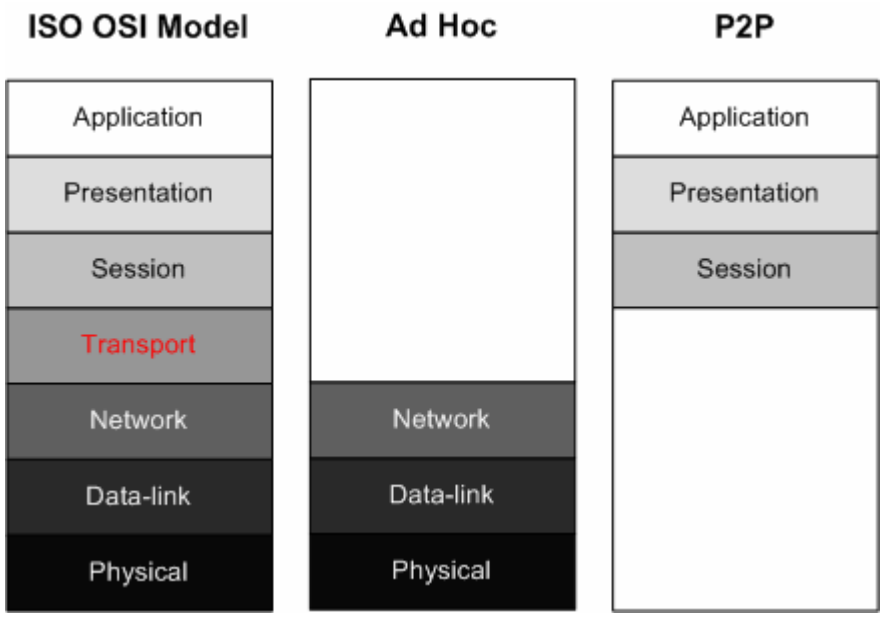

Figure 2.5: OSI layers relevant to Ad Hoc and P2P networking.

Figure 2.6 shows how the aspects (and sub-aspects) of Ad Hoc and P2P networking, presented in Sections 2.4 and 2.7 respectively, are related for the purposes of this comparative study. As can be expected, most of the aspects are present in both network types and therefore, are related directly.

An exception to this is the sub-aspect of non-neighbouring node discovery in Ad Hoc networks that is in actual fact performed with the use of routing protocols and hence, dealt with under the aspect of routing. Secondly, the sub-aspects of content dissemination, user identification, content identification, and free riding in P2P networks have no equivalent in Ad Hoc networks. Regardless, they have been included in this comparative study due to the relevance they may gain in Ad Hoc networking in the future.

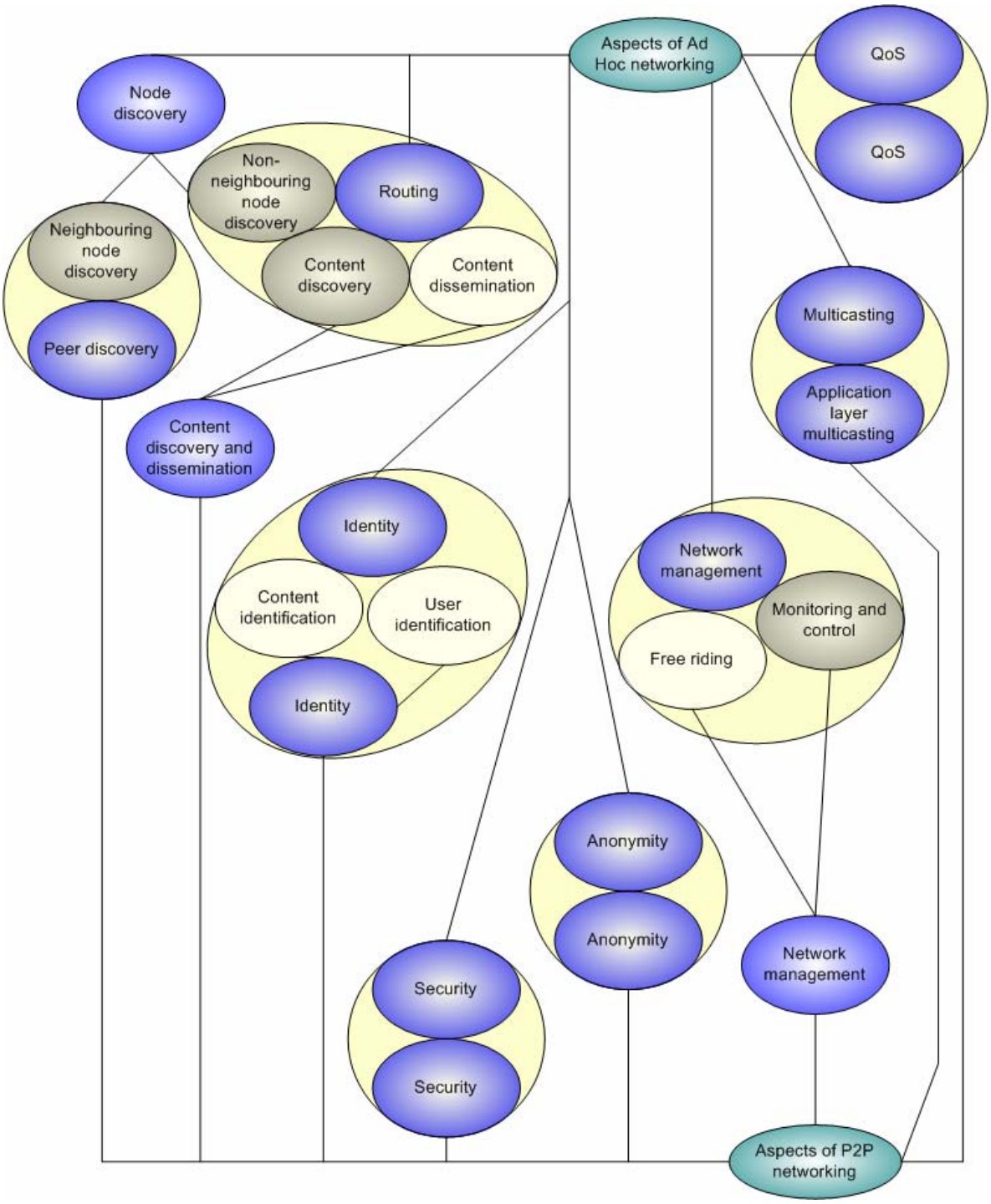

Figure 2.6: Aspect comparison between Ad Hoc and P2P networking.

2.9. Standardisation Initiatives and Implementations

In conclusion to this chapter, this section provides a brief description of some of the standardisation initiatives relevant to Ad Hoc and P2P networking, since these are referred to in subsequent chapters. Additionally, this section lists some of the P2P overlay network implementations in use over the Internet, since these are also used as examples in subsequent chapters.

2.9.1. Ad Hoc Networking Standardisation Initiatives

The following are the standardisation initiatives relevant to Ad Hoc networking:

- *Bluetooth/IEEE 802.15*: The Bluetooth special interest group has standardised Bluetooth; a cable replacement/PAN standard that operates in an Ad Hoc manner. The standard operates in the 2.4 GHz Industrial Scientific Medical (ISM) frequency band and supports a maximum bit rate of 1 Mbps. The standard has also served as the basis for the IEEE 802.15.1 standard, standardised by the IEEE 802.15 working group. Further information about Bluetooth may be found in [19] and [20].
- *ETSI HIPERLAN*: ETSI has also released two standards for WLANs. HIPERLAN/1 was the first standard and supports a maximum bit rate of 20 Mbps by operating in the 5 GHz frequency band. The second standard was subsequently standardised by the BRAN working group of ETSI to operate in the 5 GHz frequency band at a maximum bit rate of 54 Mbps. HIPERLAN/2 has also been harmonised with the IEEE 802.11a standard. Both standards also support an Ad Hoc mode of operation. Further information about the HIPERLAN standards may be found in [8] and [11].
- *IEEE 802.11*: The IEEE 802.11 working group has released a set of standards addressing the physical and data-link layers of WLANs. These standards are:
    - IEEE 802.11 supporting a maximum bit rate of 2 Mbps and operating in the 2.4 GHz ISM frequency band.
    - IEEE 802.11a supporting a maximum bit rate of 54 Mbps and operating in the 5 GHz frequency band.
    - IEEE 802.11b supporting a maximum bit rate of 11 Mbps and operating in the 2.4 GHz ISM frequency band.

  - IEEE 802.11g supporting a maximum bit rate of 54 Mbps and operating in the 2.4 GHz ISM frequency band.

  All standards support an infrastructure-less (i.e. Ad Hoc) mode of operation. Further information about the IEEE 802.11 set of standards may be found in [9] and [21].

- *IETF MANET*: The IETF MANET charter is currently standardising the following four routing protocols for MANETs:
  - Ad Hoc On-demand Distance Vector (AODV) routing, documented in RFC 3561 with status experimental.
  - Dynamic Source Routing (DSR) protocol, documented in Internet draft draft-ietf-manet-dsr-09.txt.
  - Optimised Link State Routing (OLSR) protocol, documented in Internet draft draft-ietf-manet-olsr-11.txt.
  - Topology dissemination Based on Reverse Path Forwarding (TBRPF), documented in Internet draft draft-ietf-manet-tbrpf-10.txt.

  Further information about the IETF MANET routing protocols may be found in [22], [23], [24] and [25].
- *IETF Mobile IP*: The IETF Mobile IP charter has standardised IP mobility support for wireless networks, including Ad Hoc networks. This is documented in RFC 3344. Mobile IP allows nodes in a wireless network to retain their IP address between different fixed network interconnection points. Further information about Mobile IP may be found in [26].

2.9.2. P2P Networking Standardisation Initiatives and Implementations

The following are the standardisation initiatives relevant to P2P networking:

- *Jabber*: Jabber is an open-source Extensible Mark-up Language (XML) protocol for instant messaging applications. Its most notable feature is that it is interoperable with other proprietary instant messaging P2P networks through the use of gateway Jabber servers. The Jabber protocol is currently being standardised under the IETF Extensible Messaging and Presence Protocol (XMPP) charter. Further information about Jabber can be found in [27] and [28].

- *JXTA*: JXTA is an open-source framework for P2P networks. Project JXTA was initiated by Sun Microsystems and has three main objectives:
    - Interoperability: The framework is designed so that different P2P networks using the JXTA framework are interoperable.
    - Platform independence: The framework is designed so that it can be implemented in any programming language, run on any operating system, and over any underlying transport layer protocol.
    - Ubiquity: The platform is designed so that it can run on a wide range of electronic devices, including computers, mobile phones and PDAs.

  Further information about project JXTA can be found in [29].

Table 2.1 lists the current P2P network implementations used over the Internet that are used as examples in subsequent chapters[2]. The only exception to this is Napster, since it is no longer operational due to legal issues. It has nonetheless been used in this comparative study as an example of a P2P network that makes use of a DCP2P server.

P2P Application/ Network	Category	P2P Network Type	Website
Napster	Content sharing	DCP2P (Hybrid)	http://www.napster.com
Gnutella	Content sharing	Atomistic (Pure)	http://gnutella.wego.com
Kazaa/FastTrack	Content sharing	Atomistic (Pure)	http://www.kazaa.com
Freenet	Content sharing	Atomistic (Pure)	http://freenet.sourceforge.net
ICQ	Instant messaging	UCP2P (Hybrid)	http://web.icq.com
MSN Messenger	Instant messaging	UCP2P (Hybrid)	http://messenger.msn.com
Yahoo! Messenger	Instant messaging	UCP2P (Hybrid)	http://messenger.yahoo.com

Table 2.1: Current P2P networks used over the Internet.

[2] The list itself is not exhaustive of all the P2P network implementations currently used over the Internet.

3. Node/Peer Discovery

3.1. Introduction

Since both Ad Hoc and P2P networks have the commonalities of decentralisation and transient connectivity, they share another commonality in that a given node/peer has to be capable of performing discovery of other nodes/peers in order to join the network. Techniques for performing node or peer discovery in Ad Hoc or P2P networks respectively are therefore reviewed in this chapter following which, a comparison of node/peer discovery in the two network types is presented.

3.2. Node Discovery in Ad Hoc Networks

Since Ad Hoc networks are wireless networks, one speaks of a radio range; this being defined by the maximum transmission/reception range across which two nodes can communicate directly. In Ad Hoc networks parlance, nodes within radio range of each other are termed neighbours. Discovery of, and communication with non-neighbouring nodes is also possible, however, this requires the use of a routing protocol. Non-neighbouring node discovery is therefore reviewed as part of routing in Ad Hoc networks; which will be the subject of the next chapter.

Discovery of neighbouring nodes in Ad Hoc networks is considered a data-link layer issue and the technique chosen is standard specific. The techniques used in IEEE 802.11 and Bluetooth are therefore reviewed here so that an appreciation of neighbouring node discovery is possible.

3.2.1. Neighbouring Node Discovery in IEEE 802.11

In the IEEE 802.11 set of standards, there are two possible ways in which a node can learn about the existence of its neighbours:

1) *Passive scanning*: When passive scanning is used, a node simply awaits for the reception of a **Beacon** frame from other nodes [21]. Beacon frames contain clock information that the receiving node may use in order to synchronise its clock with that of the other nodes. As a result, the node also learns of the existence of its neighbours.
2) *Active scanning*: A node may alternatively ask for information about other nodes by transmitting a **Probe Request** [21]. A node receiving this Probe Request would reply with a **Probe Response**. A Probe Response would typically contain identification and capability information.

3.2.2. Neighbouring Node Discovery in Bluetooth

The technique used in Bluetooth is considerably different. A Bluetooth network is generally defined by a *piconet*, this being a collection of up to eight Bluetooth devices. One of the devices in the piconet assumes the role of a *master*, with each of the other devices acting as a *slave* to the master. One of the master's roles is to take care of new nodes that should be added to the piconet. For it to do so, it enters an **Inquiry** state to check if there are other nodes nearby. A new node willing to join the piconet would in turn be in an **Inquiry Scan** state and, upon receiving an **Inquiry** frame from the master, the new node responds to the master by sending its Bluetooth device address [19].

Subsequently, the master unit enters a **Page** state, constructs a **Paging** frame with the new node's Bluetooth device address and sends this frame to the new node, which in turn responds whilst in a **Page Scan** state [19]. The master will then send the new slave synchronisation information, so that the new slave can participate in the piconet. Figure 3.1 illustrates this sequence of operations.

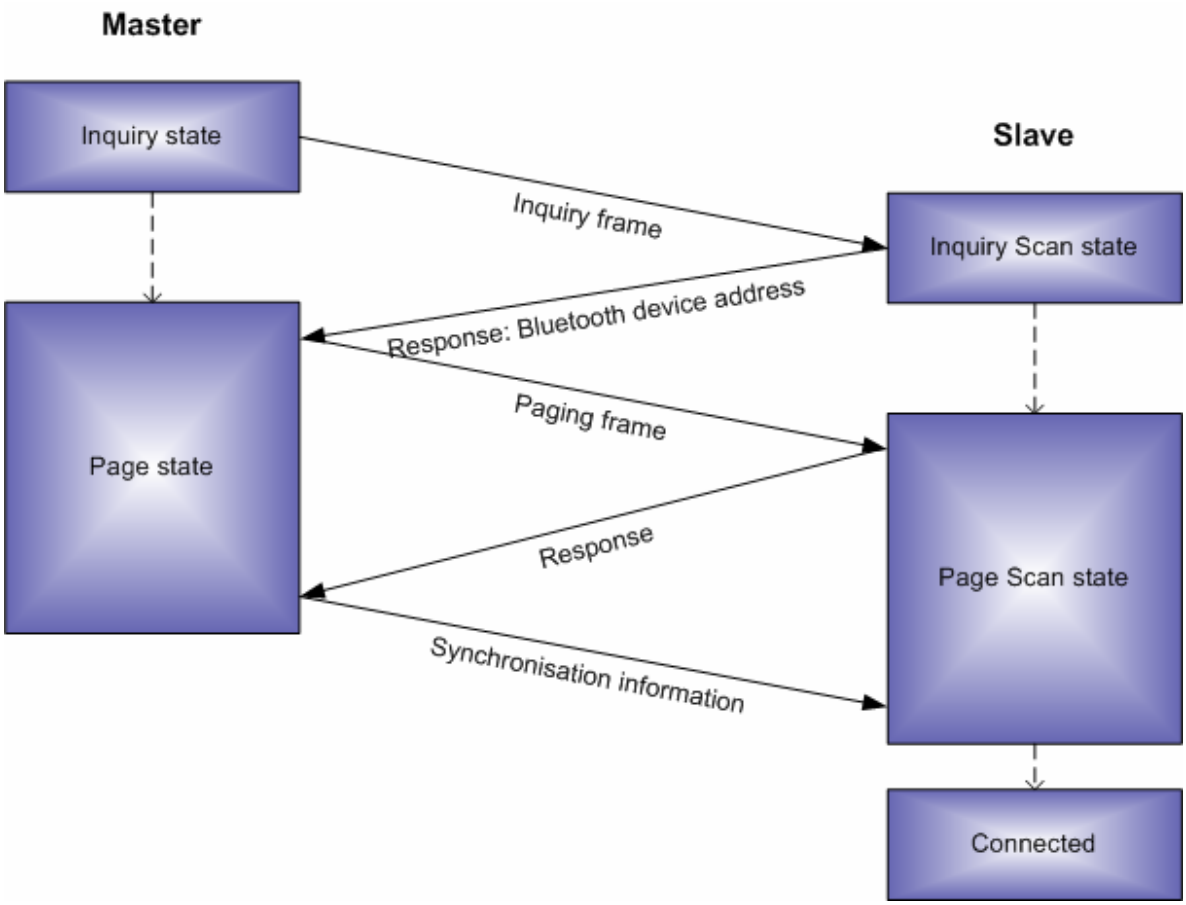

Figure 3.1: Neighbouring node discovery in Bluetooth.

3.3. Peer Discovery in P2P Networks

The techniques used to perform peer discovery in P2P networks are considerably more diverse (and complex) than the equivalent techniques used to perform neighbouring node discovery in Ad Hoc networks. The primary reason for this is that peer discovery has to be performed at the application layer in P2P networks and, therefore, no assumptions can be made about the functionality provided by the underlying network and data-link layers.

This section provides a review of peer discovery in UCP2P networks and atomistic P2P networks. Peer discovery in DCP2P networks is essentially a non-issue, since the discovery of content in these networks also implies the discovery of the corresponding peers at which the desired content is located. Peer discovery in DCP2P networks is therefore reviewed as part of content discovery in P2P networks; which will be the subject of Chapter 5.

3.3.1. Peer Discovery in UCP2P Networks

The solution adopted for peer discovery in UCP2P is essentially a trivial one, as a central server is relied upon to provide this service. As mentioned in Section 2.5, the UCP2P server maintains a database of all registered users. Since the specifications of such a database are generally proprietary and not disclosed, a fictitious example is provided in Table 3.1 for illustrative purposes. As may be observed, the database typically contains an entry for every user[3] identity registered with the system. Each entry is indexed with an index number for searching and sorting purposes. For every entry, the current IP address and TCP port number is maintained, provided that the user is not offline. The user's status is also maintained in this database so that other users may determine whether the given user is currently reachable.

Index Number	User Identity	Current IP Address	Current TCP Port Number	Status
1	joeborg@ucp2p	144.82.214.58	5060	Online
2	claraborg@ucp2p	201.10.156.2	7002	Online
3	melaniemd@ucp2p	144.82.214.156	6023	Busy
4	daveg@ucp2p			Offline
5	antoinegc@ucp2p	120.13.98.152	9321	Away

Table 3.1: Example of a UCP2P database.

Other peers registered with the server that want to connect to a given peer, simply query this database for the peer's connection identifiers, and subsequently establish a direct connection with the peer. For example, if user joeborg@ucp2p wanted to communicate with user claraborg@ucp2p, the user would query the UCP2P server by providing the required user identity (i.e. claraborg@ucp2p). In this case, upon finding the required entry, the UCP2P server would return an IP address of 201.10.156.2 and a TCP port number of 7002. User joeborg@ucp2p may then establish a direct connection with user claraborg@ucp2p.

Although server-mediation provides a simple and elegant solution for peer discovery, the solution is generally frowned upon at it introduces a single point of failure, and hence, one of the advantages of using a P2P network as opposed to using a client/server network is lost. Secondly, although the bandwidth expended at the server is low compared to client/server networking since communication occurs directly between peers, the

[3] Due to the context, the words 'user' and 'peer' are used interchangeably in this sub-section.

informational and computational complexity associated with maintaining an accurate database and providing search capabilities is still relatively high. Regardless, the use of a UCP2P server has proven extremely popular for instant messaging P2P networks and is in fact made use of in all of the three implementations listed in Sub-section 2.9.2.

3.3.2. Bootstrapping in Atomistic P2P

In an atomistic P2P network, a peer wishing to join the P2P network must first bootstrap to at least one other peer that is already part of the P2P network. This may be performed in two ways. When the P2P application is intended for a localised environment such as a LAN, a broadcast may be used to discover other peers on the network. An application layer broadcast in this case would translate itself into a data-link layer broadcast, given that a data-link layer protocol that supports broadcast, such as Ethernet, is used. Although this solution is used in a number of P2P applications, notably multiplayer gaming [13], its main disadvantage is that the application is restricted to a local environment.

The use of broadcast for joining a P2P network, however, is not possible in P2P networks that span multiple IP subnets. This therefore includes all of the P2P overlay networks in use over the Internet. The problem here is one evidenced in several other network applications; namely that of performing broadcast over a switched/routed network. The specific problem for atomistic P2P is that broadcast packets are not forwarded by routers in an IP network.

To overcome this problem, atomistic P2P networks require a peer to know of at least one other peer that is already part of the P2P network. In turn, this implies that certain peers are required to have a greater degree of stability, if they are to be relied upon by other peers wishing to join the P2P network. This degree of stability may be attainted if such a peer has a static IP address and uninterrupted connectivity with the P2P network. This type of peer is referred to as a *host cache* in Gnutella [2] or a *portal* in FastTrack [4]. The term ‘stable peer’ is used throughout the rest of this study to refer to this entity in general.

The use of a stable peer has one important implication: From a theoretical perspective, an atomistic P2P network that spans multiple subnets and that fully conforms to the definition of P2P given in Section 2.5 cannot exist, since a distinction is created between stable peers and all other peers in the P2P network. Nonetheless, P2P networks that make use of stable

peers for the establishment of initial connectivity are still regarded as atomistic in practice, since the stable peer simply mediates initial connectivity; it does not provide the server functionality relied upon in UCP2P or DCP2P.

3.3.3. Peer Discovery in Atomistic P2P – Flooding Technique

Given that a peer has successfully joined the P2P network, the flooding technique may then be used to discover other peers. The review given here is based on the Gnutella implementation of flooding, however, the concept behind the technique should be clear, irrespective of the implementation details.

Two message types are defined in the Gnutella protocol specification for peer discovery, these being **Ping** and **Pong**. Ping is used to discover other peers, whilst Pong is a reply to a Ping. A Pong message contains the IP address and information about the data being shared at the discovered Peer [30]. All Gnutella messages have a Time-To-Live (**TTL**) field that limits the range that the messages propagate. The value of this field is decremented by one at each (application layer) hop and, upon reaching a value of zero, the message is dropped [30]. A **Message ID** field is also used, so as to uniquely identify each message and hence ensure that a peer does not forward the same message more than once.

A typical sequence of events would start with the stable peer receiving a Ping message from the Peer that just connected. The stable peer would then forward this message to all peers that are connected to it, except to the peer which received the Ping [31]. Subsequently, all peers that receive this message do the same operation and, additionally, they reply back to the peer from which the Ping message originated, with a Pong message. The Ping message keeps propagating until the TTL expires. In order to minimise the cost of flooding, Pong messages are only propagated back along the path on which the Ping message was received. If a peer receives a Pong message without having received the corresponding Ping message beforehand, the Pong message is dropped [30]. This Ping/Pong procedure is illustrated in Figure 3.2.

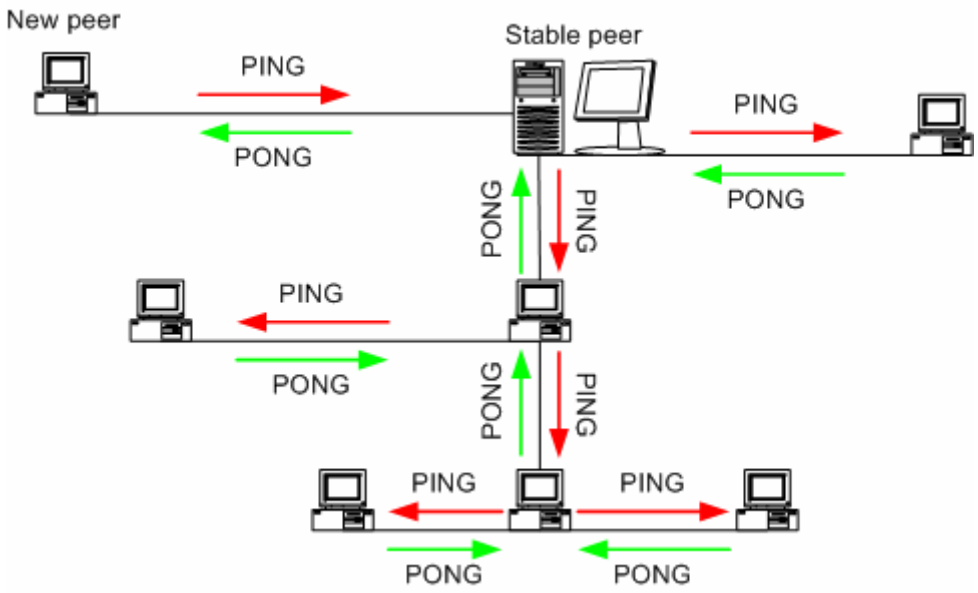

Figure 3.2: Flooding technique in Gnutella.

Upon receiving the Pong messages, the new peer may then connect to any of the discovered peers by sending a **GNUTELLA CONNECT** request. If a **GNUTELLA OK** message is received in response, direct connectivity would have been established. The Ping/Pong process may be initiated by a peer whenever needed.

Typically, Gnutella peers have an average node degree[4] of four [30]. Although such a node degree seems low, in practice this is adequate as content queries are also flooded in this manner, thereby assuring that the Query reaches more than four peers.

With an average node degree of four and the use of a TTL field to limit flooding, one might expect the flooding technique to scale well. Nonetheless, as demonstrated in [30], scalability is a serious drawback of the flooding technique, especially in Gnutella where the same technique is also used for content discovery.

[4] The node degree is the amount of other nodes to which a given node is connected to.

3.3.4. Peer Discovery in Atomistic P2P – Rumour Mongering Technique

The term 'rumour mongering' refers to a class of protocols known as *gossip* protocols [31]. The operation of gossip protocols is based on the random selection of peers to which a given message should be forwarded, as opposed to the approach used in flooding, where a Ping message is forwarded to all peers except to the peer from which it was received. Although this class of protocols has been successfully used in other applications, such as database consistency management, it is a relatively new idea within the context of P2P networks [31].

A specific type of gossip protocol, known as *blind counter rumour mongering*, has been analysed in [31]. The protocol is similar in operation to the flooding technique used in Gnutella, however, two additional parameters are specified:

1) *B*: Specifies the number of peers to which a message should be forwarded. Random selection is employed to select *B* peers out of all peers to which a connection is available.
2) *F*: Specifies the amount of times a given peer should forward the same message. This parameter is necessary since, unlike flooding, all the possible peers are not exhausted the first time a message is forwarded.

In order to minimise cost, the selection of peers to which the message is to be forwarded also takes into account the following:

- It ensures that the message was not received from the potential next-hop peer.
- It ensures that the message was not already sent to the potential next-hop peer by the forwarding peer, in a previous forwarding attempt.

Results obtained in [31] indicate that the rumour mongering technique provides a more scaleable approach when compared to the flooding technique, however, this is traded-off with a reduced reach and an increase in delay. This trade-off is controlled by modifying the parameters *B* and *F*. When *B* is set to the node degree and *F* is set to one, the protocol is identical in operation to flooding.

3.4. Comparison of Node/Peer Discovery

From the review given in this chapter of the aspects of neighbouring node discovery in Ad Hoc networks and peer discovery in P2P networks, two clear differences emerge:

1) A distinction exists between neighbouring and non-neighbouring nodes in Ad Hoc networking. This distinction is notably absent in P2P networking.
2) The use of broadcast frames is relied upon in Ad Hoc networking for the discovery of neighbouring nodes. The use of broadcast in P2P networks is limited to LAN applications. P2P networks that span multiple subnets cannot rely on broadcast for peer discovery and therefore, have to use a server or a stable peer to provide initial connectivity.

The first difference is due to the OSI layers of relevance in Ad Hoc and P2P networking, respectively. In Ad Hoc networks, the aspect of neighbouring node discovery is performed at the data-link layer. Non-neighbouring node discovery is essentially part of routing and therefore performed at the network layer. A distinction between neighbouring and non-neighbouring nodes is therefore necessary in Ad Hoc networking since non-neighbouring nodes cannot be discovered at the data-link layer due to radio range limitations. Above the network layer, however, this distinction need not be created since connectivity to all nodes is provided by the network layer itself. Application layer P2P overlay networks, therefore, need not create this distinction.

The second difference stems out of the transmission medium used. An Ad Hoc network makes use of an inherently broadcast medium in nature: The wireless medium. As a result, an Ad Hoc network's data-link layer protocol generally supports the use of broadcast frames. A P2P network on the other hand cannot assume this of the data-link layer protocol in use, more so because the P2P network may span several underlying data-link layer protocols. Secondly, broadcast cannot be performed at the network layer since IP does not forward broadcast packets across IP routers. For this reason, a P2P application running over the Internet may not use broadcast.

Both of these differences, in our view, are fundamental to the correct operation of node/peer discovery in both network types. It is therefore beneficial that research on node/peer discovery in Ad Hoc and P2P networks respectively, is distinct.

Routing protocols for Ad Hoc networks still have to be developed, trialled and improved upon before they become as established as their fixed network counterparts. In the meantime, the distinction between neighbouring and non-neighbouring nodes is necessary to the field of Ad Hoc networking, since it has to be taken into account in the development of routing protocols. Conversely, connectivity to all nodes in a network is provided as a service to the application layer and, therefore, there is no reason why a distinction between neighbouring and non-neighbouring peers should be created in the field of P2P networking.

From a node discovery perspective, the use of broadcast frames allows an Ad Hoc network to be a self-organising one. In other words, nodes may discover each other without relying on any stable, predetermined nodes to do so. Although self-organisation is also a desirable property for P2P networks, broadcast clearly cannot be used for the reasons given before. The use of a stable peer is therefore justified. On the other hand, since no fixed infrastructure is present in an Ad Hoc network, the use of a stable node to provide initial connectivity is not possible.

In conclusion, although Ad Hoc and P2P networks share the commonality of requiring node/peer discovery to be performed, the techniques used to perform this have to be dissimilar between the two network types, due to the OSI layers at which node/peer discovery is performed in the two network types, and due to the support for broadcast messages in one network type and not the other.

4. Routing in Ad Hoc Networks

4.1. Introduction

Communication between non-neighbouring nodes in an Ad Hoc network requires the use of a routing protocol so that multi-hop paths may be discovered and made use of. In Ad Hoc networking, there are four additional constraints that have to be taken into consideration when evaluating an Ad Hoc routing protocol [32]:

1) *Dynamic topologies*: This includes the ability of nodes to move freely and arbitrarily. The topology therefore changes randomly and rapidly.
2) *Bandwidth and channel constraints*: The bandwidth available is often limited in the wireless domain. Additionally, there are several effects acting on the channel characteristics including path loss, interference, noise and fading. Since these effects are sensitive to motion, their combined result also varies dynamically and unpredictably. Transmitter power and receiver sensitivity also affect the radio range and Bit Error Rate (BER), and since these are not necessarily identical for all nodes in an Ad Hoc network[5], the existence of unidirectional links is also possible.
3) *Power constraints*: Since the nodes are mobile, operation is typically battery dependent and hence the power available is exhaustible. Optimisation for power conservation is therefore necessary.
4) *Security*: A wireless network is clearly more prone to security threats, mainly due to the ease of eavesdropping and spoofing, since an intruder does not need a physically attached network node. Encryption is therefore typically applied, however, other issues, such as router authentication (prior to the exchange of routing protocol information) still have to be addressed in light of the additional complexity posed by the decentralised nature of Ad Hoc networks.

[5] The Bluetooth standard, for example, supports three transmission ranges, these being 1 m, 10 m, and 100 m.

The challenge that these four constraints pose, coupled with the fundamental importance associated with routing protocols for communication between non-neighbouring nodes, has resulted in a situation whereby routing is the single most active area of Ad Hoc networking research [5]. Consequently, several Ad Hoc routing protocols have been proposed until now, all with their own relative advantages and disadvantages [4].

This chapter provides a review of the four IETF MANET routing protocols currently undergoing standardisation in view of the impetus that they are likely to gain from this standardisation initiative. Following these reviews, a comparison of the four routing protocols is provided.

4.2. Proactive vs. Reactive Routing Protocols

A routing protocol used in an Ad Hoc network may be classified as one of two possible types: *Proactive* or *reactive*. Proactive routing protocols (also known as *table-driven* protocols) are similar to the ones used in fixed networks [3], in that each node actively maintains a routing table containing entries to all destinations within the network. The use of proactive routing protocols typically requires each node to send updates to other nodes on a periodic basis and/or whenever a change in connectivity is detected.

Reactive routing protocols (also known as *on-demand*, *source-initiated* or *demand-driven* protocols) on the other hand, are routing protocols that determine the path to be taken only when that path is required. Upon requiring a route to a known destination, a node typically floods the network with a search request for a route to the destination. Any intermediate node with a cached route or the destination itself may reply to the request, which the source may then use for the actual transfer of data [3].

There are obvious differences between proactive and reactive routing protocols, with both categories having their relative merits and drawbacks. It is likely, that no one protocol will meet all the constraints that have to be taken into consideration, as set out in the previous section. Hence, a mixture of routing protocols is likely to be used in practice [32].

Proactive routing protocols have the following advantages over reactive routing protocols:

- Discovery of non-neighbouring nodes is inherent.
- Routes to all the nodes are maintained and hence, ready to be used when required.
- Proactive protocols are adaptive to change as a change in topology generally causes new control information to be sent out.

A significant disadvantage of proactive routing protocols, however, is due to the control traffic that such protocols generate. This control traffic poses a significant traffic burden, especially in a highly dynamic environment such as an Ad Hoc network, as each change is likely to trigger the transmission of control traffic. As a result, resource usage limits the scalability of proactive routing protocols.

Conversely, the main advantage derived from the use of reactive routing protocols is the reduction in control traffic generated. This should therefore lead to better scalability properties as resources are only expended for routing when there is a need for a route. The significance of this advantage, however, is dependent on the nature of the traffic that is being carried, as this dictates how frequently new routes are requested.

By making use of reactive routing protocols, the advantage derived from minimising control information is traded for the following disadvantages:

- There is a significant delay incurred between the request for a route and the actual transmission of data traffic, as a route has to be found. This may be unacceptable for some applications, especially those with real-time properties.
- The use of flooding for route discovery may cause a significant traffic burden, as well as pose security issues, such as Denial of Service (DoS) attacks [3].
- Caching of routes is problematic, as the reliability of cached routes in such a dynamic environment is hard to ascertain. On the other hand, having little or no caching implies flooding needs to be performed each time a route is required.
- No non-neighbouring node discovery is performed in reactive routing protocols. Rather it is assumed that the node to be contacted is present. The extent to which this disadvantage becomes a drawback is highly context dependant. Applications in

which all the nodes participating in the network are known in advance are not hindered by this drawback. On the other hand, applications in which unplanned communication is made use of are likely to be hindered by this drawback.

The last point deserves particular attention. From a comparative perspective, a solution such as the flooding technique used in P2P networks could be implemented at the application layer. However, such a solution would have to be analysed more in depth as a number of issues may arise. Possibly the biggest issue is whether non-neighbouring node discovery should be user initiated or automated.

If node discovery is user initiated, the solution may be somewhat cumbersome and the user may nonetheless miss out on potential communication opportunities, due to lack of user initiated node discovery. On the other hand, if node discovery is automated, then one would have to look into how often this should be done and whether the benefit of using a reactive routing protocol is still significant.

Besides classification into proactive or reactive, a protocol may generally also be classified as a *link-state* or a *distance-vector* routing protocol[6]. Although a treatise of the relative merits and drawbacks of these two protocol types is beyond the scope of this dissertation[7], the main distinguishing factor is easily summarised. In a link-state routing protocol, a node distributes information about the status of links with its neighbours to all nodes within the network, whilst in a distance-vector routing protocol, a node distributes distance information about all of its routes, to its neighbours [15].

[6] This classification is usually more applicable to proactive routing protocols.
[7] The relative merits have been contended at length for fixed network routing protocols. Further information on link-sate and distance-vector routing may be found in [15].

4.3. IETF MANET Routing Protocols

The four sub-sections within this section provide a review of the operation of each of the four IETF MANET routing protocols, respectively. At the time of writing, RFC 3561 has been published for AODV, whilst the Internet drafts for the other three routing protocols have been submitted to the IESG for approval and subsequent publication as RFCs.

4.3.1. Ad Hoc On-demand Distance Vector

AODV is a reactive routing protocol, implying that routes are only created when required. The protocol is a distance-vector based protocol, it, however, improves over the basic algorithm by using sequence numbers to avoid routing loops and count-to-infinity [22]. The protocol also allows for the creation of IP multicast routes, although the full processing of IP multicast route requests is not specified. AODV employs a flat network architecture.

In AODV, a source node desiring to communicate with a given destination node for which it does not have a valid route, must initiate a *path discovery* process. Once this process is initiated, the source node broadcasts a Route Request (**RREQ**) packet to its neighbours, which in turn forward the packet to their neighbours and so on [33]. This process continues until either the destination, or an intermediate node with a valid route to the destination, is reached. Figure 4.1 illustrates an example for the former case.

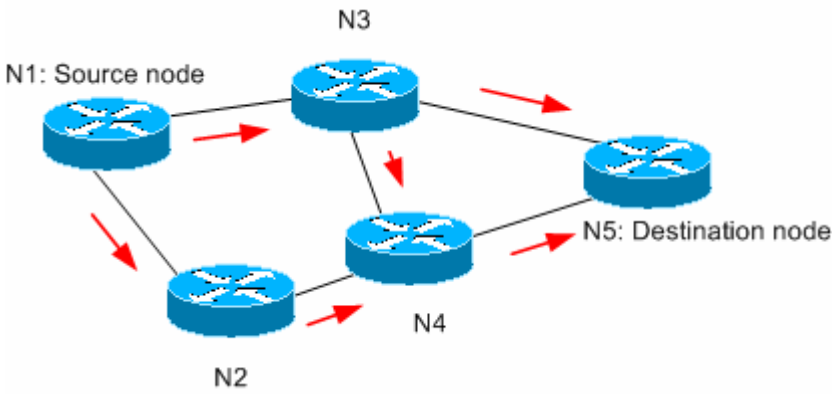

Figure 4.1: Propagation of a RREQ packet in AODV. Adapted from [33].

Each node within an Ad Hoc network making use of AODV is associated with a **Sequence Number** and an **RREQ Identifier**. The RREQ identifier is incremented for every RREQ

that the node initiates and, together with a node's IP address, it uniquely identifies every RREQ [22]. Besides the RREQ identifier, an RREQ packet also contains source and destination IP addresses, and source and destination sequence numbers. The sequence number for the destination in this case is the most recent one that the source is aware of.

Intermediate nodes can reply to RREQs only if they have a route to the destination, the corresponding sequence number of which is greater or equal to the one in the RREQ [33]. During the forwarding process, nodes record in their **Routing Table**, the address of the neighbour from which the first copy of the RREQ broadcast packet was received, thus setting up a reverse route entry. Subsequent copies received are discarded.

If the RREQ packet reaches the destination or intermediate node with a valid route, the node in question replies with a unicast Route Reply (**RREP**) packet back to the neighbour from which it received the RREQ. Thus the RREP packet follows the reverse path of the RREQ and, in doing so, allows nodes on this path to enter a forward route entry into their tables. The propagation of the RREP packet is illustrated in Figure 4.2, for the example presented in Figure 4.1. A **Timer** is associated with each route table entry, upon expiry of which, the route is invalidated.

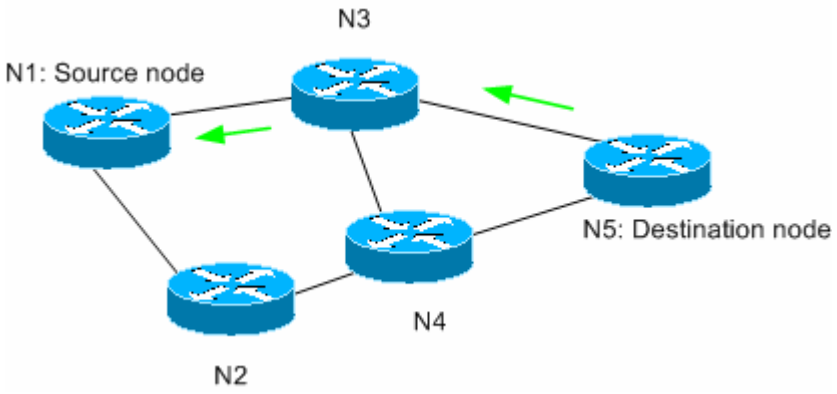

Figure 4.2: Propagation of a RREP packet in AODV. Adapted from [33].

If the source node moves, it can reinitiate path discovery, so as to find a new route to the destination. If an intermediate node along the route moves, its upstream neighbour notices and propagates a Route Error (**RERR**) packet to its active upstream neighbours to inform them of the erasure of that part of the route. This process is repeated until the source node is

reached [33]. The source node may then reinitiate path discovery. The upstream node may alternatively attempt to repair the route. Support for this is, however, optional [22].

Hello packets are used to maintain local connectivity. A node can, however, only make use of such packets if it is part of an active route [22]. A Hello packet consists of an RREP packet with the TTL set to one, thereby restricting it to the neighbouring nodes. Neighbours of a given node should listen for retransmission of packets to ensure that the node is still within reach. If the retransmission is not heard, other techniques, including the reception of a Hello packet, may be used to determine whether the next hop is still within radio range.

4.3.2. Dynamic Source Routing

The DSR protocol is a reactive protocol based on the concept of source routing [33]. As with AODV; the protocols allows for the use of multicast, however, the implementation of multicast routing is not specified in [23]. Nodes in an Ad Hoc network making use of DSR maintain a **Route Cache**, containing routes that the node is aware of. Entries are updated as new routes are learned. DSR also employs a flat network architecture.

The DSR protocol consists of two phases: *Route discovery* and *route maintenance*. A source node wishing to communicate with a given destination node must first check its route cache for an unexpired route. If a route is found, it is made use of for communication. If no route is found, route discovery is initiated using a **Route Request** packet. This packet contains the addresses of the source and destination, and a unique **Identification Number** [33]. The identification number is generated by the source node and, alongside the source node's IP address, uniquely identifies each Route Request packet.

Each node receiving the packet checks whether it knows of a route to the destination. If it does, it may reply to the message; otherwise, it adds its own IP address to the packet and then forwards the packet along its outgoing links. A Route Request packet is only forwarded by a node if the node's own IP address does not appear in the packet and it hasn't previously seen the request [33]. This effectively reduces the amount of duplicate Route Request packets received by other nodes.

The operation of source routing is illustrated in the example given in Figure 4.3. As may be observed, each node forwarding the Route Request packet adds its own IP address (denoted by the node's name in the example) to the packet itself. A node with an unexpired route in its route cache, or the destination node itself, may reply to the Route Request with a **Route Reply** packet. As may be noted from Figure 4.3, the replying node would know all the IP addresses of the nodes through which the Route Request packet traversed the network. Known as a **Route Record**, this list of IP addresses is then placed within the Route Reply packet by the replying node. If the replying node is an intermediate node, it additionally places its cached route within the Route Reply packet.

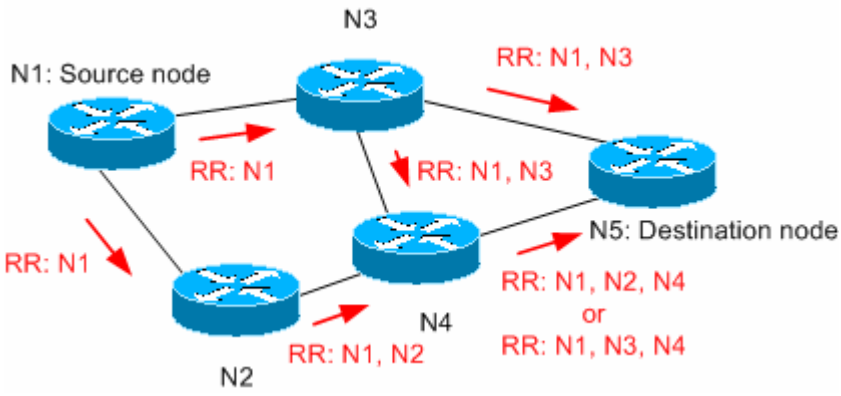

Figure 4.3: Propagation of a Route Request packet in DSR. Adapted from [33].

Since DSR provides support for the use of unidirectional links, the replying node may in turn perform route discovery to discover a reverse link to the source node. In doing so, it piggybacks the Route Reply to the new Route Request packet [33][8]. The use of bidirectional links only is also catered for in the protocol's specification. In this case, the Route Request packet would follow the reverse path indicated by the Route Record.

Route maintenance is achieved through the use of **Route Error** packets. When a link error is encountered by a node, it sends a Route Error packet to all nodes which sent a packet, routed over that link, since the last **Acknowledgement** was received from the node across the link in error [23]. When a Route Error packet is received, the node in error is removed from the route cache and all routes making use of this node are truncated at that point [33].

[8] Not doing so would result in a situation of deadlock.

Acknowledgements are also used to verify the correct operation of the links. Such acknowledgements also include passive ones, where a node is able to hear the next node performing forwarding of the packet along the route.

4.3.3. Optimised Link State Routing

The OLSR protocol is a proactive link-state routing protocol [34]. In a similar fashion to the previous two protocols, OLSR may be extended to provide multicast routing support [24]. Unlike the previous two protocols, however, OLSR's table-driven nature implies that routes to all destinations are maintained by each node in the network, independently of whether or not they are used. To reduce the burden imposed by control traffic, OLSR makes use of selected nodes for relaying. Termed Multi-Point Relays (**MPR**s), the relaying nodes minimise the control information by reducing redundant retransmissions. The use of MPRs, however, also implies that the network architecture is not entirely flat.

Each node in an Ad Hoc network making use of OLSR periodically broadcasts a **HELLO** packet to its neighbours. The message is not relayed to further nodes. Within this HELLO packet, the sender lists the IP addresses of all the one-hop nodes with which it has a bidirectional link. Additionally, it lists the IP addresses of the one-hop nodes from which it received a HELLO packet, but has not yet validated whether the link is bidirectional [34].

As a result of this procedure, a node finding its own IP address within a received HELLO packet may consider the link with the sender as bidirectional. With the reception of HELLO packets, a node also learns of the nodes which are two hops away. In summary, it is important to note that through the reception of HELLO packets, a node learns about:

- All of the neighbours and the corresponding links it has with them. This information is stored in tables called **Link Set** and **Neighbour Set**.
- All of the two-hop nodes with which its neighbours have a link. This information is stored in a table called **2-hop Neighbour Set**.

Subsequently, a node may perform selection of its MPR set based on this information. The set is selected out of all neighbouring nodes, such that the node may reach all nodes that are two hops away, using only bidirectional links [24]. The **MPR Set** for a given node is

therefore a subset of all the neighbours of the node. The smaller the MPR set, the less control information overhead is incurred. Figure 4.4 illustrates an example of an MPR set for the central node in the figure. As may be observed, the union of the four MPRs selected covers all two-hop nodes. For the example given, the figure illustrates that the MPR set in this case contains only half of the central node's neighbours, thereby leading to a reduction in control information.

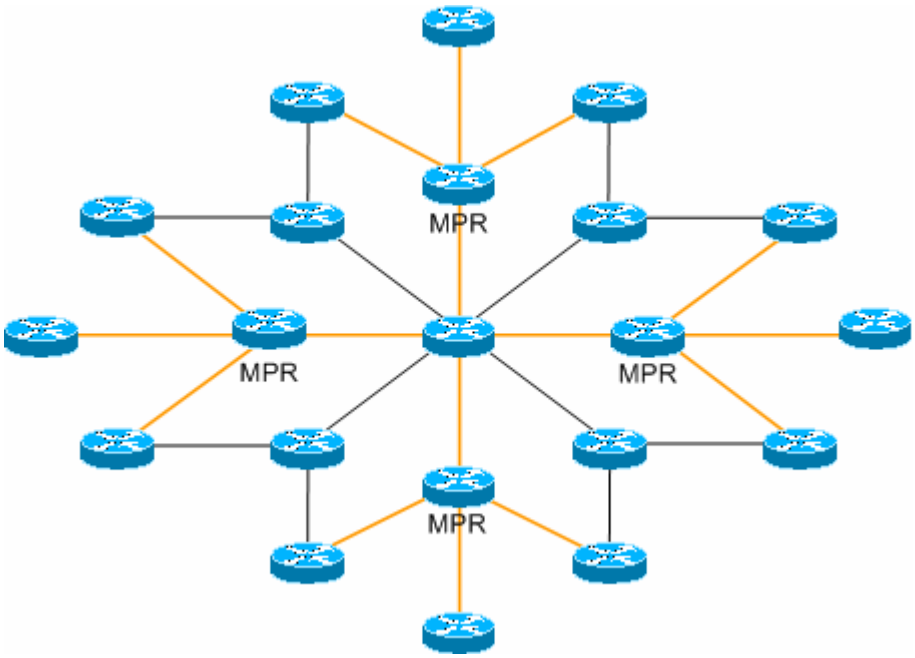

Figure 4.4: Example of an MPR set. Adapted from [34].

Nodes selected as MPRs are indicated within HELLO packets. This permits a node to keep track of all nodes that have selected it as one of their own MPRs [24]. This set of nodes is stored within an **MPR Selector Set** table at each node. MPRs are recalculated when a change in bidirectional links between the node in question and its neighbours, or between its neighbours and the two-hop nodes, is detected. Subsequent HELLO messages also reflect this change so that neighbouring nodes may update their own MPR selector set.

In order to build a **Routing Table**, each node periodically broadcasts control packets called Topology Control (**TC**) packets. In a similar manner to other link-state protocols, TC packets are broadcast over the entire network, with the exception that only the transmitter's MPRs relay these messages beyond the first hop [34]. A reduction in control traffic is thus

achieved. A TC packet declares the transmitter's MPR selector set (i.e. the list of nodes that have selected it as one of their MPRs). Transmission of TC packets may also be triggered by a change in the MPR selector set. It is important to note that TC packets, as well as HELLO packets, contain a **Sequence Number**, so that messages may be discerned.

The reception of TC packets allows a node to build a **Topology Set**, containing the MPR set for every node in the network. In conjunction with the link set, neighbour set and 2-hop neighbour set, this information is then used to build a routing table using a shortest path algorithm. The use of such an algorithm ensures that the path with the minimal number of hops is always chosen. Routing table updates occur when any of the sets change.

4.3.4. Topology Dissemination Based on Reverse-Path Forwarding

TBRPF is a proactive, link-state routing protocol [25]. As a result, it shares a great degree of similarity with OLSR, however, a number of significant differences do exist, mainly in the type of link-state information exchanged and how this information is exchanged.

The TBRPF routing protocol consists of two main modules: *Neighbour discovery* and *routing* [25]. The neighbour discovery module allows a given node to detect the presence of bidirectional links between it and its neighbours. It follows that discovery of link breakages in any direction (including both) is also inherent.

Neighbour discovery is performed with the use of **HELLO** packets, much like OLSR. The key difference, however, is that in TBRPF, the HELLO packets are differential. In other words, they only report changes in the status of links. As a result, HELLO packets are considerably smaller on average, and may therefore be sent more frequently. The information gathered from HELLO packets is stored in a **Neighbour Table**. Thus for every link a node has with its neighbours, an entry exists in the Neighbour table, which states whether the link is unidirectional or bidirectional. At least one HELLO packet must be sent by a node periodically. A **Sequence Number** is also present within every HELLO packet, so that messages originating from the same source may be discerned.

Routing in TBRPF is carried out with the use of a source tree, **T**, which provides shortest paths to all reachable nodes [25]. This source tree is updated using information obtained

from a node's **Topology Table**. The information held within this Topology table is in turn acquired from neighbouring nodes, which report part of their own source tree periodically. Additionally, differential updates may also be used more frequently. It is important to note that, in **Topology Updates**, nodes only provide part of their own source tree, and not their full source tree. The partial source tree is known as the **Reported Subtree**, **RT**.

In contrast to other link-state routing protocols, Topology updates are not forwarded beyond the first hop. In other words, their propagation is restricted to neighbouring nodes. The information received in the Topology updates may, however, cause a change in a node's reported subtree and hence, the change would be reflected in subsequent Topology updates sent by the node. To an extent, the operation of this protocol is therefore closer to that of distance-vector protocols than to traditional link-state protocols.

The complexity in TBRPF lies in how the RT is computed. In order to do this, a node must first compute a **Reported Node Set**, **RN**. At a given node, for example node I, the following operations are performed to calculate RN:

- It includes Node I (i.e. itself) in RN.
- For every neighbouring node, Node I computes the shortest paths (up to two hops) from each neighbour to every other neighbour, using only itself (i.e. Node I) or other neighbours as intermediate nodes [25].
- It includes a neighbouring node in RN, say Node J, if and only if it determines that any other one of its neighbours may select Node I (i.e. itself) to be its next hop (i.e. the next hop of the neighbouring node) on its shortest path (i.e. the shortest path for the neighbouring node) to Node J [25]. Node I is in a position to determine this from the computations performed as a result of the previous operation.
- Having determined which neighbouring nodes are to be included in RN, Node I can then determine the other non-neighbouring nodes that are to be included in RN. Node I includes a given non-neighbouring node, say Node U, in RN if and only if the next hop on Node I's shortest path to Node U is in RN [25].

Having computed RN, Node I can then compute RT. This computation is considerably easier, as RT consists of links present in Node I's source tree T, which also

originate/terminate at a node listed in RN. As a result of this procedure, RT contains all bidirectional[9] links that Node I has with its neighbours, as Node I initially lists itself in RN. Additionally, however, links that originate/terminate at other nodes are also listed if the said nodes are in RN. This fact also stands out against the operation of traditional link-state protocols, in which updates only contain information about local links.

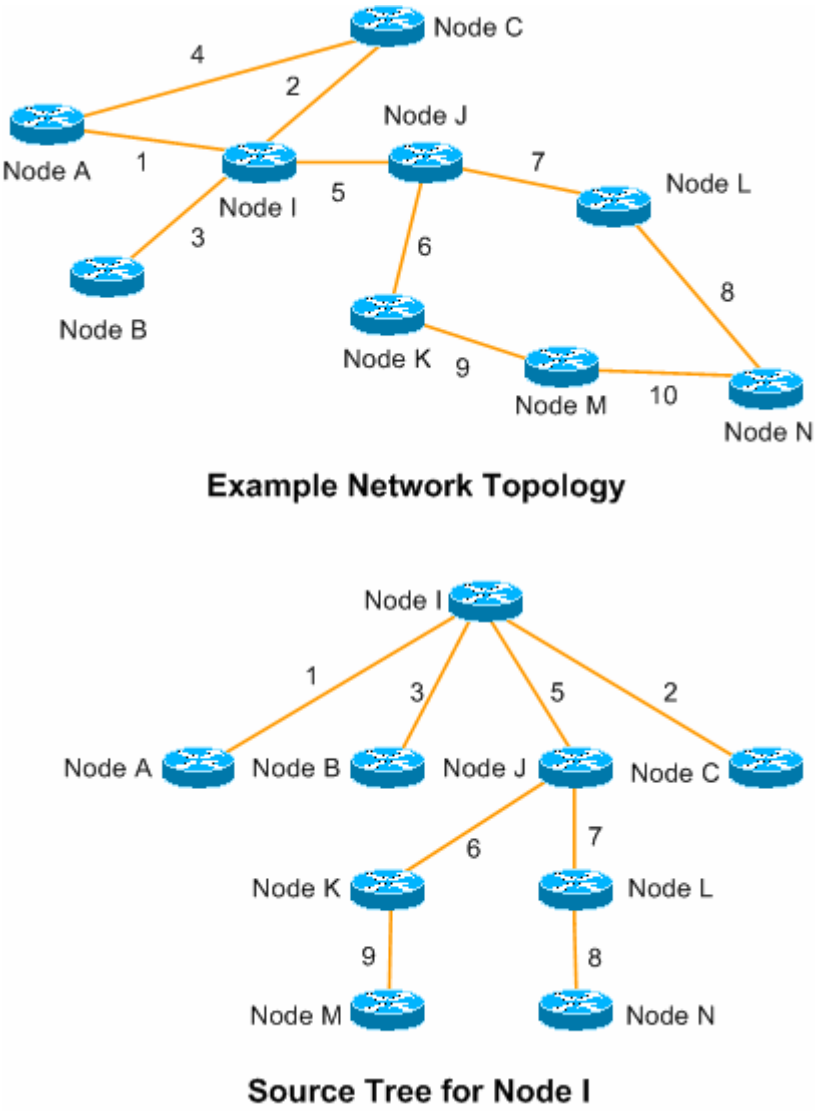

Figure 4.5: Example of routing in an Ad Hoc network using TBRPF.

Figure 4.5 presents an example of how the TBRPF protocol operates. The example presents a fictitious network topology and the corresponding source tree for Node I in the network.

[9] In TBRPF, only bidirectional links are made use of for routing.

For simplicity, only bidirectional links have been illustrated and the links have been numbered for reference. A letter has also been assigned to each node within the Ad Hoc network for the same reason.

As mentioned previously, Node I first compiles RN. Following the previous operations, it first includes itself in RN. Subsequently, it computed the shortest paths for all neighbouring nodes as described, and determines which neighbouring nodes should be included in RN. Performing this operation yields the following results:

- Node A should be included as neighbouring Nodes B and J may use Node I for their shortest path to Node A.
- Node C should be included as neighbouring Nodes B and J may use Node I for their shortest path to Node C.
- Node B should be included as neighbouring Nodes A, C and J may use Node I for their shortest path to Node B.
- Node J should be included as neighbouring Nodes A, B and C may use Node I for their shortest path to Node J.

Having decided which neighbouring nodes are to be included in RN, Node I then has to determine which non-neighbouring nodes are to be included in RN. Using the previously stated condition, Node I in this case has to include nodes K, L, M and N in RN, as the next hop on Node I's shortest path to these nodes is Node J, which is in RN.

Having compiled RN, Node I may then compile the reported subtree RT. As stated previously, links included in RT should be in Node I's source tree and also originate/terminate at one of the nodes in RN. Links 1, 2, 3 and 5 should therefore be included since Node I is in RN. Similarly, Links 6 and 7 are included due to Node J. Finally, Links 8 and 9 are included due to Nodes K, L, M and N. Conversely, Links 4 and 10 have been excluded from RT, as these are not part of the source tree. For the example presented here, Node I's reported subtree RT is in effect Node I's entire source tree, as all of the links in the source tree have been included in RT. Clearly, this is an extreme case and on average one would expect RT to be significantly smaller than a node's entire source tree. For example, node B's RT would only consist of Link 3 in the example of Figure 4.5.

4.4. Comparison of the IETF MANET Routing Protocols

Table 4.1 summarises the attributes of each of the four routing protocols reviewed in the previous section. Although a greater degree of similarity is present between reactive protocols and between proactive protocols, all four protocols are essentially different.

	AODV	DSR	OLSR	TBRPF
Proactive/Reactive.	Reactive.	Reactive.	Proactive.	Proactive.
Link-state/Distance-vector.	Distance-vector.	Not applicable.	Link-state.	Link-state.
Unidirectional link support.	Explicitly avoided.	Yes.	Explicitly avoided.	Explicitly avoided.
Support for alternative metrics.	Possible but not specified.	Unspecified.	Yes.	Yes.
Security techniques.	None.	None.	None.	None.
Support for multicast.	Yes, but not inherent.	Yes, but not inherent.	Yes, but not inherent.	Yes, but not inherent.
Route caching.	Yes.	Yes.	Not applicable.	Not applicable.
Support for multiple routes.	No.	Yes.	Unspecified.	Yes.
Network architecture.	Flat.	Flat.	The use of MPRs may be regarded as hierarchical. Otherwise flat.	Mainly flat, but can be combined with hierarchical routing techniques [25].
Intended environment.	Generic.	Ad Hoc networks with up to two hundred nodes and works well even in environments with high mobility rates [23].	Particularly suited towards large and dense networks [24].	Suited for large and dense networks [25]. Simulations show that support for up to 250 nodes can be achieved [25].

Table 4.1: Attributes of the IETF MANET routing protocols.

AODV may be regarded as a generic Ad Hoc routing protocol in that it is associated with low resource usage, making it suitable for most Ad Hoc networks. These properties are achieved, in part, since AODV is a reactive distance-vector routing protocol. This, however, also implies that AODV is not particularly suited for applications which require certain QoS parameters to be guaranteed. Reactive protocols generally incur a significant delay for route setup. Additionally, support for alternative metrics is not inherent. Finally, AODV excludes the use of multiple paths for resiliency and load balancing.

The source routing concept used in DSR implies that, although the protocol is a reactive one, control overhead may still be significant as data packets exchanged between nodes are required to have the entire route which they should follow, as part of the protocol encapsulation. On the other hand, it is the only protocol which can make use of unidirectional links when forwarding packets and therefore has the upper hand when compared to the other protocols in terms of resource utilisation.

Since OLSR is a proactive link-state routing protocol, its bandwidth and resource usage are expected to be higher than in reactive routing protocols. As a consequence, one may say that the lower the average node mobility rate, the better the protocol would perform, since less control traffic would be triggered. The inherent advantage of using a proactive protocol is that of having routes available for immediate use with a node's routing table, thereby incurring no route discovery delay. The use of MPRs helps in decreasing the amount of control traffic. Load balancing may, however, become an issue as a result, since routing tables are built from routes that use a chain of MPRs to arrive at a destination; potentially not capitalising on other bidirectional links available.

TBRPF, like OLSR, may also be associated with a relatively high degree of resource usage. Bandwidth usage in TBPRF, however, may be considerably less, especially when one takes into consideration the fact that all control messages are not forwarded beyond neighbouring nodes in TBRPF. The trade-off in this case is with computational power required to calculate the subtree at each node.

Finally, all four protocols currently do not specify any explicit security mechanisms. Their current specification, however, allows for generic security techniques, such as those reviewed in Chapter 6, to be used in conjunction. Similarly, all four protocols can be extended to support multicast routing, although the operation of multicast routing is not specified. A multicast extension for AODV is reviewed in Chapter 7, as are other multicast routing protocols that may be used in conjunction with the IETF MANET routing protocols.

5. Content Discovery and Dissemination in P2P Networks

5.1. Introduction

Content discovery and dissemination are possibly the two most important aspects of content sharing P2P networks. The aspect of content discovery deals with techniques that can be used by a peer to search for and locate desired content at other peers. Content dissemination deals with the download of this content from one peer to another, once it has been located by the latter.

This chapter provides a comparison of routing in Ad Hoc networks and content discovery in P2P networks, following a review of content discovery in P2P networks. Subsequently, a review of content dissemination in P2P networks is provided, and the applicability of content discovery techniques used in P2P networks, to Ad Hoc networks, is analysed.

5.2. Content Discovery in P2P Networks

Due to the popularity that content sharing P2P applications such as Gnutella and Kazaa have gained over recent years, this aspect of P2P networks has received considerable attention. As a result, several techniques have been proposed for content discovery in P2P networks to date [35].

This review has been restricted to a selected subset of these techniques. The first four sub-sections review content discovery techniques that have been used in popular P2P applications, whilst the last two sub-sections review two techniques that have been proposed in literature.

5.2.1. Content Discovery in DCP2P Networks

In server-mediated P2P networks, a DCP2P server may be made use of to mediate content discovery. As mentioned in Section 2.5, a DCP2P server maintains a database of content and the corresponding peers from which the content may be obtained.

Index Number	Content	IP Address Lists	TCP Port Number Lists
1	Music1.mp3	144.82.214.58; 102.12.101.1	5060; 7023
2	Video1.mpg	201.10.156.2	7002
3	Picture1.jpg	144.82.214.156; 210.105.203.5	6023; 3012
4	Program1.exe	180.102.210.23; 156.101.87.45	15002; 11032
5	Music2.mp3	120.13.98.152	9321

Table 5.1: Example of a DCP2P database.

Like UCP2P, the specifications of such a database are not disclosed. A fictitious example is therefore provided in Table 5.1 for illustrative purposes. As may be observed, the database typically contains an entry for every item of content that may be located at one of the peers in the network. Each entry is also indexed with an index number for searching and sorting purposes. For every entry, a list of IP address and corresponding TCP port numbers is kept, these being the connection identifiers for the peer/s at which the given content is located.

In DCP2P, content requests generated by peers are directed to the DCP2P server, which in turn would search for the required content in the database. For example, given that a peer desires Picture1.jpg, it would issue a content request to the DCP2P server. The server would in turn search through its database and find the entry corresponding to this content. Subsequently, it would return a reply to the peer that requested the content, containing the connection identifiers 144.82.214.156:6023 and 210.105.203.5:3012. This peer may then establish a direct connection with the peers identified by these connection identifiers so as to download the desired content.

Much like a UCP2P server, a DCP2P server introduces a single point of failure in the network and is also associated with a degree of informational and computational complexity. Additionally, a company maintaining such a server is subject to legal attack due to the nature of the (potentially illegal) content identified in the database. A DCP2P server was in fact used by Napster, before this P2P network was shut-down.

5.2.2. The Flooding and Rumour Mongering Techniques

The techniques described in Sub-sections 3.3.3 and 3.3.4 for peer discovery may also be used for content discovery. For brevity, this sub-section has therefore been restricted to outlining how these techniques are applied to content discovery.

Having established connectivity with a number of other peers, a Gnutella peer may then make use of a **Query** message to search the network for content [30]. Query messages are flooded through the Gnutella network in exactly the same manner that Ping messages are flooded, with one exception. Since a Gnutella peer would have established connectivity with other peers with the use of Ping messages, the Query message is then sent to all peers with which the requesting peer is connected. As a result, given an average node degree of four, a Query message will, at maximum[10], reach four times the amount of Gnutella peers then a Ping message, for the same TTL value.

Besides forwarding to other peers, a Gnutella node receiving a Query message will also check whether it has the required content locally. If this is the case, the peer will reply to the request with a **Query Hit** message. This message supplies the necessary connection identifiers so that the content may then be downloaded by the requesting peer [30].

As noted in Sub-section 3.3.3, the flooding technique's main drawback is scalability. This is especially true in the context of content discovery since Query messages typically account for over half of the control traffic generated by Gnutella messages [30]. Having said this, the flooding technique is attributed with one nominal advantage over other content discovery techniques. Its content discovery technique is deterministic in nature [36]. In other words, given that:

- The required content can be found on at least one peer in the network.
- The network is not fragmented.
- The value of the TTL field can be larger than the maximum path-length[11].

Then, a given search request will always be satisfied.

[10] In practice the amount of peers reached is likely to be less, since peers receiving the same message more than once will not forward the copies received after the first.
[11] The path-length is the number of hops between two given peers.

In contrast to this, the rumour mongering technique may be thought of as probabilistic in nature [31]. In other words, the selection of the parameters *B* and *F* determines the reach of a content request. One may appreciate that, if *B* is not equal to the node degree, then no guarantee can be given as to whether the content request will reach all peers in the network. In practice, this disadvantage is outweighed be the benefit derived from an increase in the scalability of a probabilistic content discovery technique.

Intuitively, the rumour mongering technique may also be applied to content discovery. The operation of the technique, as described in Sub-section 3.3.4, does not change, with the exception that the requesting peer may now send the Query message to *B* peers out of all peers with which it has a connection, as opposed to the situation in which a Ping message may initially only be sent to the stable peer due to lack of connections with other peers.

5.2.3. Hierarchical Content Discovery – Supernodes in FastTrack

The flooding and rumour mongering techniques both employ a flat architecture; one in which content requests propagate between equal peers. This is in sharp contrast to the use of a DCP2P server, where a clear cut difference exists between the DCP2P server and the peers. This difference in effect results in a strict hierarchy, with the peers generating content requests and the DCP2P server servicing the requests.

A compromise between these two different architectures is achieved in the FastTrack P2P network. The architecture employed in this network is worth noting for two reasons:

1) It tries to combine the advantages of a flat architecture with those of a strict hierarchical one for content discovery [4].
2) The FastTrack network is the network used by the Kazaa P2P client, this being a very popular P2P file sharing application.

In the FastTrack P2P network, the compromise is achieved by creating a distinction between two types of peers. Specifically, peers in the FastTrack network may be elected as *supernodes*, upon logon with the FastTrack portal[12]. The selection of these supernodes is performed by the portal itself, which in turn bases its decision on how much bandwidth and

[12] The FastTrack portal is the ‘stable peer’ in the FastTrack network.

processing power is available at the peer in question [4]. The selection is done in such a way so that only peers with a relatively high degree of these resources are chosen.

A node selected as a supernode is subsequently provided by the portal, with a list of other supernodes to which it connects. As a result of this procedure, a network is established between supernodes. Supernode discovery is therefore mediated by the portal in this manner. Other non-supernode peers that logon to the network establish a connection with one of these supernodes; the connection details for which are provided by the portal. Non-supernode peers do not require any further connections with other peers and hence the issues involved in performing peer discovery as described in Chapter 3, are avoided.

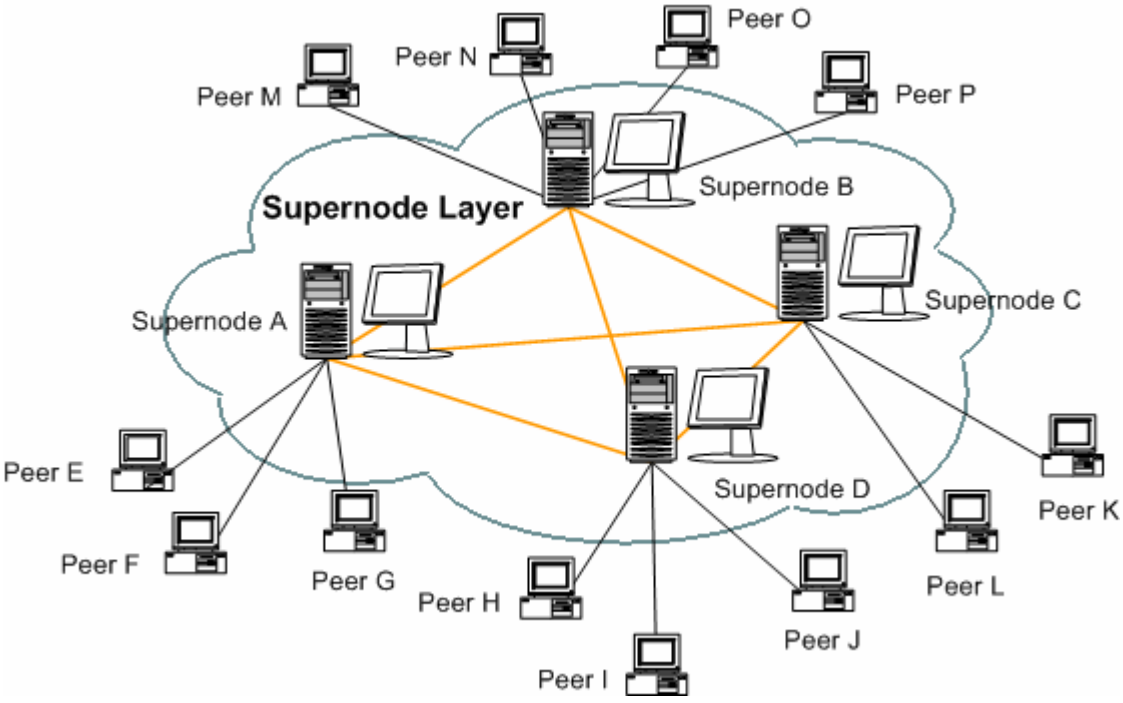

Figure 5.1: FastTrack P2P network architecture.

Figure 5.1 provides an illustration of the network architecture formed in FastTrack, as a result of the procedure described above. As may be observed, a relatively high bandwidth network is formed between supernodes, whilst all other peers using the network are attached to one supernode. The FastTrack network has a two-level hierarchy. The top layer is known as the supernode layer [4], whilst all non-supernode peers form the lower layer.

As an illustration of how content discovery is performed, consider that Peer E has just connected to Supernode A. Upon connection, Peer E uploads information about all the content it hosts to Supernode A. As a result of this procedure, all supernodes are aware of the content they themselves host, and the content that their attached peers host. After this procedure, Peer E may then generate content requests. When such a request is generated, the request is sent to Supernode A, which in turn searches for the required content in its own database [4]. This database will effectively contain information about all the content hosted at Supernode A, and Peers E, F and G. If the required content is found, the connection identifiers of the peer hosting the content, are sent back to Peer E.

Additionally, Supernode A also floods the supernode layer with this content request. Supernodes B, C and D in this case would receive the request and query their own respective databases. If the required content is found within the database of any of these supernodes, information regarding the servent hosting the content is sent back to Supernode A. The latter in turn relays this information back to Peer E.

There are two advantages in the use of supernodes, both resulting in increased scalability:

1) The supernode layer is used for content discovery; this being associated with a high degree of resources. Transfer of the actual content is performed between the peer that requested the content and the peer/s that host/s it.
2) The use of a supernode layer for content discovery implies that the broadcast of content requests is restricted to a small part of the whole network [4].

5.2.4. Content Discovery in Freenet and Small World Networks

Freenet is another P2P file sharing application designed to run as overlay network over the Internet. Having said this, it has a number of interesting differences when compared to the other content sharing P2P applications described. One of the main goals behind Freenet is to provide a fully decentralised, scalable content sharing P2P network [37].

Content within Freenet is identified using hash keys, which are derived using the Secure Hash Algorithm-1 (SHA-1). This one-way hash algorithm is applied to the content itself or alternatively, to a description of the content, from which a 160-bit key identifier results.

Content discovery is performed with the use of these keys. When a peer issues a content request, the request specifies the desired key as opposed to a textual description of the content. This effectively permits peers within Freenet to perform *routing* according to the value of the required key. Much like Gnutella's operation, **Key Requests** in Freenet have a **hops-to-live**, upon expiry of which, the request is not forwarded. Additionally, a unique **Identifier** is also present with key requests so as to achieve freedom from loops [37].

A peer receiving a request for a given key will first check whether the required key is present within its local **File Store**. If the key is contained within the file store, the desired key (i.e. content) is returned, together with the peer's TCP/IP connection identifiers. If, on the other hand, the content is not located within the peer's file store, the peer consults its own **Routing Table** in order to find an entry for the desired key. If an entry is found, the Key Request is forwarded to the corresponding peer indicated within the routing table. If no entry for the key exists within the routing table, a key entry that is as similar[13] as possible to the requested key is used instead [37]. Using this mechanism, a peer node should eventually be reached, which holds the desired key within its file store.

Given that the desired key is found, the key and connection identifiers are sent back along the reverse route followed by the Key Request. As a result, all intermediate peers traversed on this route may cache the key (i.e. content) in their file store and, additionally create an entry for the key and corresponding connection identifiers in their routing table.

Content discovery in Freenet is nominally[14] deterministic, as a peer may use all of its routing table entries, in a sequential manner, in order to forward a Key Request. This may be necessary when the target peer is not available or when routing loops have to be avoided. This sequential operation is performed in order of key similarity, starting with the closest match (between the requested key and the routing table entries) first. Having exhausted all of its routing table entries, a peer may report a **Failure** back to the peer from which it received the Key Request, which in turn would also perform this sequential operation. This operation is known as *backtracking* [37].

[13] Key similarity is determined according to lexicographic distance [37].
[14] A very large hops-to-live value would be required, making the technique probabilistic in practice.

The technique as described, has two important advantages [37]:

1) Peers tend to collect content for which the keys are similar. This comes about as a result of receiving key requests that are similar and thereby a copy of the actual key (i.e. content) when the key is sent back along the reverse path.
2) Peers gain knowledge about sourcing content for which keys are similar, due to the routing table entry added when the key and connection identifiers are sent back.

Due to these two advantages, routing in Freenet improves as routing table entries are added and clustering of similar keys at a given peer occurs [37]. A further advantage of this technique is that Key Request replication does not occur at any point. In other words, only one copy of a given Key Request exists at any point in time on the Freenet network. This clearly improves the technique's scalability when compared to the flooding technique.

For peer discovery, Freenet uses a technique similar to the one used in Gnutella. A peer initially bootstraps to the network with the use of a stable peer. The new peer then sends an **Announcement** message to the stable peer, which in turn forwards this message to another peer randomly selected from its routing table [2]. This random forwarding proceeds from peer to peer until the hops-to-live expires. Each peer receiving this message will in turn add an entry for the new peer in its routing table and reply to the new peer with its own connection identifiers. The new peer in turn adds the peers learned into its routing table.

Content discovery as performed in Freenet is based on the concept of a small world network. In a small world network, most peers are connected to a small number of nearby peers, however, a small number of peers are connected to many other peers [2]. When analysed using graph theory, a small world network has an interesting property in that it has a *clustering coefficient*[15] that is comparable to a highly regular network, whilst still achieving a relatively low *characteristic path-length*[16] [2]. The latter is a property typically associated with random networks, such as the overlay network formed in Gnutella.

[15] The clustering coefficient is a measure of connectivity between 'nearby' peers [2].
[16] The characteristic path-length is the average number of hops between two peers [2].

Since the Freenet network exhibits a low characteristic path-length, the median number of hops taken for a Key Request can be very low. However, the worst case number of hops taken can be very high [2], due to incorrect routing decisions than can be taken. In other words, although the characteristic path-length is low, this does not imply that the route with the lowest path-length will be chosen as peers have no absolute knowledge of the shortest path, when performing routing. In contrast, the flooding technique performs relatively well even in its worst case [2]. The trade-off here is therefore one of delay versus scalability.

5.2.5. Content-Addressable Network

The concept of a Content-Addressable Network (CAN) was proposed in [38] to describe the use of a large-scale Distributed Hash Table (DHT) as a data structure that maps keys onto values. A corresponding design for one such CAN is also proposed in [38] and the design itself is also known as CAN, due to the concept on which it is based.

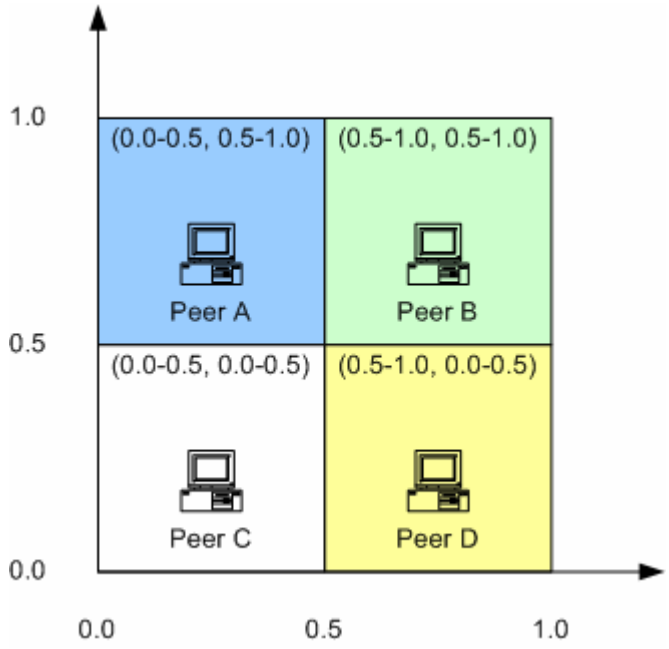

Figure 5.2: Example of a 2-dimensional coordinate space in CAN. Adapted from [38].

The design is centred on the use of a virtual *D-dimensional coordinate space*. An example of a 2-dimensional coordinate space is given in Figure 5.2. The coordinate space is divided amongst all of the peers that form part of the P2P network, such that each peer is allocated

its own individual virtual space [38]. In this example, the coordinates for each peer's virtual space are indicated in brackets.

Content within CAN is identified with the use of keys, much like Freenet. The key and the corresponding content is referred to as a **(key, value) pair**. Such (key, value) pairs are distributed amongst peers in the network, however, this is done in a deterministic manner. Specifically, a hash algorithm in this case is used to map keys onto a point *P* on the coordinate space. Subsequently the (key, value) pair for the key in question is assigned to the peer whose virtual space includes the point *P* [38].

Any peer wishing to retrieve a given key may then apply the same hash algorithm so as to determine the point *P* for the key. Having done this, a request for the key is routed to point *P* (and hence to the peer responsible for point *P*) by passing this request to the nearby peer that is closest to Point *P*. Nearby peers in CAN are peers that are allocated virtual spaces adjacent to the virtual space of a given peer[17]. For example, in Figure 5.2, Peers A and B are nearby peers, whilst Peers A and D are not. Each intermediate peer subsequently performs the same routing decision, as a result of which, the request finally reaches the peer holding the desired (key, value) pair.

In view of this, it is critical for each peer in CAN to know who its nearby peers are. A peer, therefore, maintains a **Routing Table** containing the TCP/IP connection identifiers and the corresponding virtual space of every nearby peer. This information is provided to a peer upon joining a CAN network. It subsequently learns about changes in nearby peers with the reception of **Update** messages.

As with all other techniques described so far, a peer wishing to join the network has to bootstrap to a stable peer in order to join a CAN network. Having done this, it sends a **JOIN** message to a randomly selected point *P* [38]. Once this message reaches the peer associated with the virtual space on which point P lies, this peer splits its virtual space and assigns half of it to the new peer. The original peer then provides the new peer with the required routing information and updates its own routing table. Both peers then update their nearby peers with update messages. It is important to note that (key, value) pairs belonging

[17] The virtual space must be adjacent in at least one dimension.

to the virtual space of the new peer also have to be redistributed from the old peer to the new peer [38]. Similarly, this process has to be reversed when a peer wishes to leave the network. A hand over procedure is therefore also specified.

The biggest advantage of CAN is that it is capable of guaranteeing deterministic discovery of a key in $O(D \times N^{1/D})$ hops [35]; where N is the number of peers and D is the dimension of the coordinate space. The corresponding routing table size would be $O(D)$ [35]. Deterministic content discovery is possible even if a nearby peer fails, since routing to the next best peer would be performed. As a result, even though the algorithm is deterministic in nature, the technique used in CAN is highly scalable, provided a suitable value for D is chosen to balance informational and communicational complexity.

A notable disadvantage, however, stems from the way peers join and leave a CAN network, due to the redistribution of (key, value) pairs that is required. In an actual implementation, this may imply long periods for setup and hand over, especially if the content associated with a given key is significantly large is size.

5.2.6. Chord

The Chord content discovery technique is proposed in [35]. Chord is based on the distributed computation of a hash function (i.e. DHT) that maps keys onto peers that are responsible for them. Following from the definition of a CAN, it follows that Chord is a content discovery technique that may be used for the implementation of a CAN.

The hash algorithm used in Chord is SHA-1; this being applied to both content and peer IP addresses. As a result, each peer is identified with an m-bit key the way content is[18]. The technique used in Chord then orders peer keys in a logical modulo 2^m ring. Having done this, Chord assigns content keys to peers, such that the value of the content key assigned is smaller or equal to the peer key to which it is assigned, but greater than the peer key for the *predecessor* peer on the ring [35].

[18] The terms ‘content key’ and ‘peer key’ will be used throughout this sub-section so as to distinguish between keys used to identify content or peers, respectively.

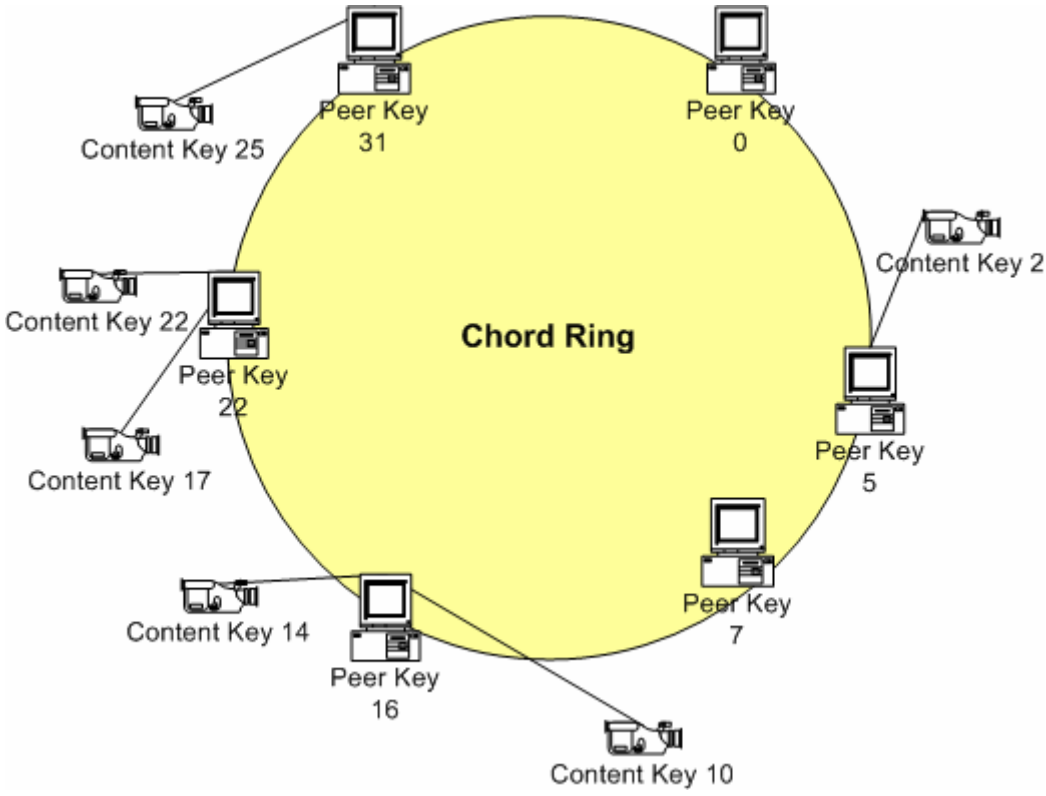

Figure 5.3: Example of a modulo 2^m ring in Chord for $m = 5$. Adapted from [35].

The example presented in Figure 5.3 illustrates the operation described in the preceding paragraph for a value of $m = 5$[19]. As may be observed, peers are ordered according to their peer key value. As an example, Peer Key 7 succeeds Peer Key 5, but precedes Peer Key 16. Content assignment is then carried out according to the condition specified in the previous paragraph. For example, in this case Content Key 14 has been assigned to Peer Key 16, since 14 is smaller or equal to 16, but greater than 7.

When assigned a given content key, a peer would store the content associated with this key. This is therefore similar to the (key, value) concept used in CAN. Clearly, redistribution of content keys and corresponding content is required whenever peers join or leave the network. In the example of Figure 5.3, if a peer with a peer key value of 10 (i.e. Peer Key 10) were to join, then Content Key 10 would have to be transferred from Peer Key 16 to Peer Key 10. The converse procedure would be performed it Peer Key 10 were to leave.

[19] In practice, a much larger value for *m* would be used.

Content discovery in Chord is performed with the use of a routing table, known as a **Finger Table**. The Finger table for each peer contains up to m entries, where m is the number of bits in a key. For a given entry i (where $1 \leq i \leq m$), the entry contains the peer key and TCP/IP connection identifiers of the first peer that succeeds the peer in question by at least 2^{i-1} (modulo 2^m) [35]. In other words, given a peer whose peer key is x, an entry at position i would store the peer key and connection identifiers of the first peer in the Chord network whose peer key is at least modulo 2^m of $(x + 2^{i-1})$. Consider an example for Peer Key 7 in Figure 5.3: In this case m has a value of *5* and x has a value of 7. The Finger table would therefore contain the following:

- For $i = 1$, modulo 2^m of $(x + 2^{i-1}) = 8$. Peer Key 16's details are therefore held.
- For $i = 2$, modulo 2^m of $(x + 2^{i-1}) = 9$. Peer Key 16's details are therefore held.
- For $i = 3$, modulo 2^m of $(x + 2^{i-1}) = 11$. Peer Key 16's details are therefore held.
- For $i = 4$, modulo 2^m of $(x + 2^{i-1}) = 15$. Peer Key 16's details are therefore held.
- For $i = 5$, modulo 2^m of $(x + 2^{i-1}) = 23$. Peer Key 32's details are therefore held.

It is important to note that the first entry in a Finger table is always the immediate *successor* of the peer. In this case, the entry for $i = 1$ is Peer Key 16, which is the immediate successor of Peer 7 in Figure 5.3.

With this Finger table, a peer may then r*esolve* any key (i.e. content or peer). A key query is forwarded according to its value. If its value falls between the peer key value of the current peer[20] and that of its immediate successor, the process is complete. The key is resolved into the peer key and the connection identifiers of the current peer's successor. Otherwise, the key query is forwarded to the peer listed in the Finger table, whose peer key immediately precedes the key to be resolved [35].

Following the example of Figure 5.3, if a peer with a peer key value of 7 (i.e. Peer Key 7) wanted to resolve Content Key 10, the process is complete since this resolves into the connection identifiers of Peer Key 16. If on the other hand, it wanted to resolve Content Key 17, it would forward the key query to Peer Key 16, since Peer Key 16 is the entry in its

[20] The current peer is the peer at which the key query is currently located.

Finger table which immediately precedes Content Key 17. Finally, if Peer Key 7 wanted to resolve Content Key 2, it would forward its key query to Peer Key 32 since this is the largest key that Peer Key 7 has in its Finger table for a peer that precedes Content Key 2 (modulo 2^m). In other words, all key queries for which the keys fall beyond the range of peer keys in the Finger table, are forwarded to the last peer listed in the Finger table.

Peers on the Chord ring receiving a key query forward it using the same routing decisions just described. When the value of the key to be resolved lies between the current peer's peer key and the peer key of the current peer's immediate successor, the current peer resolves the key query by returning the peer key[21] and the TCP/IP connection identifiers of the successor peer, to the peer that issued the key query. Subsequently, if the key resolved was in fact a content key, the peer that issued the key query may contact the peer that holds the desired content (i.e. the successor peer in this paragraph) directly.

The successful operation of a Chord network is clearly dependent on each peer maintaining a Finger table. The table is initiated when a peer joins a Chord network and maintained periodically, using a **Fix_fingers** procedure. A peer that wants to join a Chord ring will first send a **Join** request to a stable peer. The stable peer in turn supplies the new peer with the latter's immediate successor[22]. The new peer then informs its successor about its existence.

Each peer in the network periodically runs a **Stabilise** function. When this is done, a peer asks its successor for the latter's predecessor *p* and decides whether *p* should be its successor instead. This would be the case if *p* recently joined the system [35]. In other words, this procedure allows a peer to update who its successor in the Chord network is. If *p* is the new successor of a peer *n*, then *n* would also inform *p* about its existence and hence, *p* would learn about its predecessor. As a result of this procedure, a given peer is always aware of its successor and predecessor.

Having gone through the procedure described in the above paragraph, a new peer may then execute the Fix_fingers procedure. This procedure relies upon the operation of resolving keys in Chord, so as to resolve peer keys, which are to be inserted in the Finger Table, into

[21] This is necessary since the value of the key in the key query and the resolved key are usually not identical.
[22] The stable peer obtains this through a key query.

corresponding TCP/IP connection identifiers. One should also note that initially, all peer key queries for peer keys that have a value greater that the successor's peer key are forwarded to the successor, as no other entries exist in the Finger table.

Like CAN, Chord is a deterministic content discovery technique that achieves a high degree of scalability. A peer in a Chord network in fact may resolve a key in O(log *N*) hops, where *N* is the number of peers on the network [35]. Informational complexity is also relatively good, as a Finger table has O(log *N*) entries[23]. As a result, the properties of the technique are very similar to those of CAN. A slight advantage of Chord is that communicational scalability is only dependant on *N*, whereas in CAN this is also dependent on *D*, the value of which may be hard to adjust dynamically. Both techniques should also have the advantage of load balancing, due to the random distribution with which keys should be generated. On the other hand, they both have a common drawback in having to perform redistribution of content keys and corresponding content when peers join or leave.

5.3. Comparison of Ad Hoc Routing and P2P Content Discovery

A basis for comparison is present between routing in Ad Hoc networks and content discovery in P2P networks. A number of similarities may in fact be identified, so much so that content discovery in P2P networks is at times referred to as routing in P2P networks.

Fundamentally, both aspects deal with the same issue. Routing in Ad Hoc networks deals with route discovery. Content discovery in P2P networks deals with content discovery.

The use of broadcast packets for routing in Ad Hoc networks, and the use of flooding in P2P networks, is similar. AODV and DSR in fact both make use of broadcast packets (i.e. route requests) to discover a route to a given destination. Essentially, this is the same procedure used when flooding is performed in Gnutella. The only difference is that, whilst a network layer broadcast is used for the Ad Hoc routing protocols, the flooding technique in Gnutella has to perform replication of unicast Query messages at the application layer.

[23] Note that this is not equal to *m* as duplicate entries for a given peer can be removed.

With the use of broadcast messages, a third similarity stems from the use of a TTL field to limit the range of the broadcast messages and hence, increase scalability. The value of the TTL field is also a trade-off for routing as well as content discovery. When used in Ad Hoc routing, the TTL field effectively limits the size of an Ad Hoc network. In the flooding technique used in Gnutella or the technique used in Freenet, the TTL field dictates if the content request is deterministic or probabilistic, and determines the content request's reach.

A fourth similarity may also be identified in the way RREP messages propagate in AODV and the return of keys (i.e. content) in Freenet. In both cases, the message travels back along the reverse path of the original request. In AODV, this allows nodes to set up a forward route table entry. In Freenet, peers may cache the key in their file store and create an entry for the key and corresponding connection identifiers in their routing table. If bidirectional links only are used, this similarity is also applicable in DSR.

On the other hand, the primary difference is that, unlike routing in Ad Hoc networks, proactive content discovery techniques cannot be used in P2P networks, without significantly burdening the network with control traffic [4]. In other words, a solution whereby each peer within a P2P network actively maintains a list of all the content that can be sourced and from where this content may be sourced, would not scale well. Content discovery therefore has to be reactive in nature.

Secondly, the support for unidirectional links in DSR demonstrates another important dissimilarity. It demonstrates that unidirectional links are an important issue for routing in Ad Hoc networks, whilst this issue is non-existent in P2P networks. As a result, whilst most of the complexity of content discovery in P2P networks lies in how the content request reaches a peer that holds the content, routing in DSR also has to ensure that a route along which routing information may be sent back to the destination node is present. The use of route discovery in the reverse direction may be necessary in this case.

In conclusion, routing in Ad Hoc networks, especially reactive routing, bears a high degree of similarity with content discovery in P2P networks. On the other hand, some fundamental differences are also present, implying that the extent to which research in the two fields can converge is limited.

5.4. Content Dissemination in P2P Networks

Once the required content is found, the content itself has to be transferred from the peer hosting the content to the peer requesting it. Since a P2P network is an application layer overlay network, the issues associated with the transmission and routing of the data are not for consideration, as these issues are generally handled by the underlying layers.

Having said this, it is clear, from the reviews given in Section 5.2, that a number of different techniques may in fact be adopted for content dissemination. The issues that have to be considered when selecting a content dissemination technique are the following:

- Is the content relayed back through the peers that participated in content discovery, or is the content transferred directly between the host peer and requesting peer?
- Does the requesting peer act as a client and the host peer act as a server for content transfer, or vice versa?
- If multiple copies of the same content may be discovered, what benefit does this provide? Is segmented download possible?

With regards to the first issue, the majority of techniques reviewed employ a direct transfer between requesting peer and host peer. A notable exception to this, however, is the technique used in Freenet, where both key (i.e. content) and connection identifiers traverse the reverse path used for content discovery. The benefit derived in this technique is that replication of the key at the peers along the reverse path is possible and routing table entries may be added, thereby increasing content availability as well as decreasing the time taken for content discovery. The disadvantage of this technique, however, is that more bandwidth has to be expended by the network, since the optimal network layer route from host peer to requesting peer is not necessarily the same as the route involving all intermediate peers.

The delay may also be significant in this case, for two reasons:

1) The resulting network layer route going though all intermediate peers is likely to involve a larger amount of hops than the optimal network layer route between host peer and requesting peer, the delay incurred is likely to be greater.

2) Since all peers now participate in the transfer of content, the effective bit rate is limited by the bit rate available at each peer. As an example, consider a case in which five peers are involved in transferring the content, of which four peers have a broadband Internet connection at disposition, whilst the remaining peer has a dial-up Internet connection. The latter peer will effectively be the rate determining peer.

With regards to the second issue, the choice is highly dependent on how content discovery is performed. If multiple copies of the same content may be discovered, then the requesting peer has to act as a client to the host peers, as a selection of which copy/copies is/are to be downloaded is necessary. Conversely, if content discovery may yield only one result, then the host peer may initiate transfer of the content, with the requesting peer acting as a server.

With regards to the third issue, segmented download is possible if multiple copies of the same content may be discovered. Segmented download is a technique whereby the required file (i.e. content) is split into segments. Subsequently, a segment is transferred from a host peer to the requesting peer, such that all the different segments are downloaded from the different host peers that host a copy of the required content. When all the segments have been downloaded, the file is reassembled from these segments at the requesting peer. The advantages with which this technique is associated are twofold:

1) A requesting peer with a high bandwidth connection may still take advantage of this connection, even if the host peers have low bandwidth connections.
2) A degree of load balancing is attained, as the transfer is split amongst several of the hosting peers.

Amongst others, this technique has been successfully used in the FastTrack P2P network. A successful implementation of this technique has, however, to ensure that sufficient computational resources are available at the requesting peer in order to manage the multiple TCP connections. An even more important constraint is that the P2P application at the requesting peer has to make sure that the different copies available at the different host peers are truly identical. For this reason, the P2P application typically has to correlate several properties such as file size and metadata, before segmentation is performed.

5.5. Applicability of Content Discovery Techniques to Ad Hoc Networks

Research into content discovery for Ad Hoc networks is still within its infancy. The network layer aspects, such as routing, still dominate research interest in Ad Hoc networks as many issues still have to be overcome. Regardless of this fact, the development of suitable content discovery techniques will undoubtedly gain importance in the future, if Ad Hoc networks are to support applications that are similar to those supported by current fixed networks. In view of this, the first sub-section provides a review of a content discovery technique that has been proposed for use in Ad Hoc networks. The second sub-section then concludes this chapter with an analysis of the extent to which the other content discovery techniques presented in Section 5.2 may also be applied to Ad Hoc networks.

5.5.1. Nom

Nom is a content discovery technique that has been proposed in [6] for use in Ad Hoc networks. Interestingly, Nom is based on the flooding technique as used in Gnutella. The authors provide the following two reasons for this choice:

1) There are several open source implementations to the Gnutella protocol, simplifying access to it.
2) The Gnutella network has wide reach across the Internet.

The reason behind the relevance of the second point is that, according to the authors, a content discovery protocol for Ad Hoc networks should be interoperable with an equivalent protocol used over fixed networks. The objective in this case is that of allowing two-way content discovery (i.e. from Ad Hoc networks to fixed networks and vice versa).

An Ad Hoc node running Nom can perform the following operations [6]:

- Send and receive Nom messages. Nom messages which have already been received (determined with the use of a Message ID) are ignored.
- Retrieve a list of neighbouring nodes.
- If a **Query-Initiate** message is generated by the application running on the Ad Hoc node, build a Query message and send it to neighbouring nodes.

- If a Query message is received, check whether the content requested in the message is available locally. If this is the case, then generate a Query Hit message and send it to neighbouring nodes. If not, then forward the Query message to the neighbouring nodes, provided that the TTL has not expired.
- If a Query Hit message is received, check whether the corresponding request was sent by this node. If this is the case, return the result to the application. If not, forward the message to the neighbouring nodes.

In comparison to Gnutella, one can immediately identify a great degree of similarity in operation. This is necessary in order to achieve the required interoperability. For example, the message format used is identical to content discovery messages used in Gnutella. The use of a Message ID and a TTL field in every message is also similar.

Two important adaptations to the flooding technique may, however, also be identified from Nom's operation. The first adaptation concerns the use of neighbouring nodes in Nom. Since, neighbouring nodes are discovered by underlying protocols in Ad Hoc networks as described in Section 3.2, discovery of peers is unnecessary in Nom. The use of Ping and Pong messages provides the equivalent mechanism in Gnutella, as described in Sub-section 3.3.3. An assumption being made in this adaptation is that at least one neighbouring node is also running Nom. This assumption may be problematic in a practical Ad Hoc network where Nom doesn't necessarily form part of the basic service set offered by each node.

The second adaptation is in how Query Hit messages are treated. From the description given, one can note that no provision is made to ensure that this type of message traverses the reverse path taken by the original Query. This adaptation is necessary for two reasons:

1) Since an Ad Hoc network is highly dynamic, the network is continually reconfiguring itself. Within the time elapsed between the transmission of a Query message and the reception of a corresponding Query Hit message, the network may have adopted a different topology, making it impossible to follow the reverse path.
2) The support for unidirectional links in routing protocols like DSR implies that messages might need to be routed on a different reverse path if the message is to ultimately reach the requesting node.

5.5.2. Applicability of Other Techniques

The rumour mongering technique, like the flooding technique may also be adapted for use in an Ad Hoc network, essentially because its operation is based on the Gnutella protocol. In practice, however, it is unlikely that any benefit will be derived from doing so, since the technique essentially trades-off network reach with scalability. The low node count in Ad Hoc networks, however, implies that reducing network reach may significantly impact the probability of discovering the required content.

The hierarchical content discovery technique used in FastTrack, whilst achieving good scalability, introduces the concept of supernode; one which is not easily adapted to an Ad Hoc network. Since all nodes in an Ad Hoc network generally make use of the same bandwidth, nodes with a higher bit rate connection cannot be identified. Secondly, within the supernode layer, all supernodes can communicate directly with each other at these higher bit rates. Even if nodes with a higher bit rate connection are identified in an Ad Hoc network, these nodes wouldn't necessarily fall within radio range of each other. The FastTrack architecture may, however, prove useful in situations where gateways are provided to Ad Hoc networks. A gateway in this case could be designated as a supernode and would have a fixed connection with the Internet. In this way, a technique such as that used in FastTrack can be extended to encompass Ad Hoc networks.

Content discovery as performed in Freenet is also unsuitable for Ad Hoc networks, since the advantages associated with this technique are not applicable to an Ad Hoc network. Specifically, peers in Freenet are said to acquire a degree of specialisation in sourcing similar keys, as a result of route table entries added and content caching when a key traverses the reverse path. This reverse path is not necessarily present in Ad Hoc networks.

On the other hand, the properties of CAN and Chord make them good candidates for use in Ad Hoc networks. Their low informational and communicational complexities are two such desirable properties. Having said this, an adaptation is definitely necessary in order to minimise or avoid the transfer of keys when peers join or leave the network. This is necessary due to the low bandwidth available to Ad Hoc nodes and due to the dynamic nature of Ad Hoc networks that is likely to make such transfers frequently necessary.

6. Identity, Security and Anonymity

6.1. Introduction

The aspects of identity, security, and anonymity may be summarised as follows. Identity is concerned with the identification of any entity (e.g. nodes, peers, users, and content) within the network, in a way that distinguishes it from all other entities. Security encompasses a number of goals, the most important generally being the preservation of data confidentiality. Finally, anonymity deals with the non-disclosure of user identity in network applications. Intuitively, there is a large degree of interdependence amongst these aspects, thus favouring the consolidated approach taken in this comparative study.

The first seven sections in this chapter deal with the three aspects of identity, security, and anonymity in Ad Hoc and P2P networks, with the exception that Section 6.4 provides a general overview of the security techniques used in communications. The chapter is subsequently concluded with a comparison of these three aspects in the two network types. The comparison also includes an analysis of the applicability of techniques, used in one network type, to the other.

6.2. Identity in Ad Hoc Networks

As noted in Section 2.8, current research and development in the field of Ad Hoc networking is relevant at the lower three layers of the OSI model. As a result, the identification techniques used in Ad Hoc networks are intended for the identification of nodes and packets exchanged between these nodes, at the network and data-link layers.

The first sub-section within this section provides a review of the identification techniques used at these layers, whilst the second sub-section reviews the use of Mobile IP for wireless networks.

6.2.1. Node and Message Identification in Ad Hoc Networks

Like fixed networks, nodes in an Ad Hoc network are assigned a globally unique Medium Access Control (MAC) address for use at the data-link layer. For example, in Bluetooth each device is assigned a unique 48-bit address, known as the Bluetooth device address [19]. These addresses are allocated by the IEEE 802 committee and belong to the same set of addresses used in other IEEE 802 standards. Similarly, the IEEE 802.11 standards for WLANs also use the 48-bit addressing scheme managed by the IEEE 802 committee.

At the network layer an IP address is also used so as to allow interconnection between different data-link layer technologies. Each node in an Ad Hoc network would have an IP address and hence routing may be performed with the use of IP addresses.

The last identification technique worth noting is the use of sequence numbers to identify packets exchanged between nodes in an Ad Hoc network. The routing protocols described in Section 4.3, for example, all make use of such sequence numbers to some extent, in order to identify packets. The uniqueness of a given packet's identity is derived from combining the sequence number with the source IP address.

6.2.2. Mobile IP

In fixed networks routing is performed on a per subnet basis. This implies that a change in IP address is required if a node were to be relocated from one subnet to another. Nodes, such as a server, which require a permanent network layer identity, are typically assigned a static IP address. The use of static IP addressing is clearly not an issue in this case, as movement of such nodes between different subnets does not occur in a fixed network. Conversely, routing in Ad Hoc networks is typically performed on a per IP address basis, thereby eliminating the need for any change in IP address as nodes move through the network. In isolation, both solutions therefore provide a way how a node can be associated with a permanent network layer identity.

A problem arises when Ad Hoc networks are interconnected with fixed networks. The problem occurs when an Ad Hoc node traverses from one interconnection point to another. From a fixed network perspective, these interconnection points are likely to fall under different subnets. Thus, if an Ad Hoc node were to retain the same IP address when

traversing these subnets, routing from a fixed network to the Ad Hoc node would not be possible [39]. On the other hand, if an Ad Hoc node were required to change its IP address at each interconnection point, the node's network layer identity would not be preserved.

To solve this problem, the IETF has standardised mobile IP. Mobile IP allows a mobile node to retain its IP address, whilst making routing between fixed and Ad Hoc IP networks possible. The standard itself is not restricted to Ad Hoc networks alone and may be applied to other wireless networks, such as a cellular network.

A mobile node making use of mobile IP has to be associated with a *home network*. In this case, a mobile node is assigned an IP address on its home network, which is referred to as the *home address*. This address is not assigned to any other nodes, even if the mobile node is not present within the home network. The home address provides the mobile node with a permanent identity, as all other nodes may exchange IP packets with the mobile node by using this address as the destination address, irrespective of the mobile node's location.

When the mobile node is located at a *foreign network* (i.e. a different interconnection point), IP packets addressed to the home address still arrive at the home network. These packets therefore have to be re-routed from the home network to the foreign network where the mobile node is currently located. This is accomplished by using a *home agent* in the home network. When a mobile node is located on a foreign network, the mobile IP specification allows for two possible scenarios:

1) The mobile node registers with a *foreign agent*, whilst still retaining its own home address as the active IP address.
2) The mobile node does not register with a foreign agent, in which case it has to be assigned an IP address that is valid on the foreign network. This can be performed by using DHCP on the foreign network.

For the first scenario, upon registration with the foreign agent, the mobile node informs its home agent about the foreign agent with which it is registered. An IP tunnel is then established between the home agent and the foreign agent. IP packets received at the home network for the mobile node are then tunnelled by the home agent to the foreign agent. In

other words, the IP packets are encapsulated within other IP packets, the destination address for the latter being the IP address of the foreign agent. The foreign agent's IP address thus serves as a *care-of-address*. Upon receiving such packets, the foreign agent removes the encapsulation and forwards the contents (i.e. the original IP packet) to the mobile node.

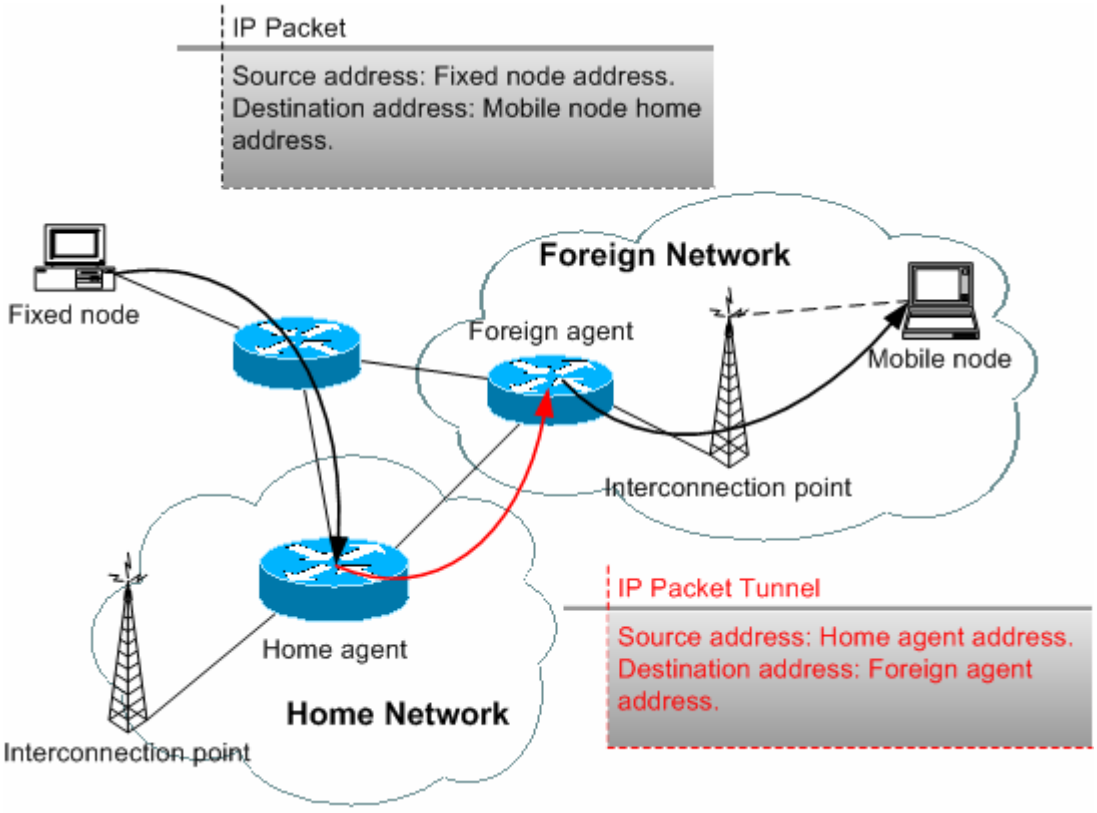

Figure 6.1: Example of re-routing in Mobile IP with the use of a foreign agent.

Figure 6.1 provides an example of the re-routing procedure performed in Mobile IP, when a foreign agent is used. The arrows in bold indicate the path taken by IP packets from a fixed node on an IP network to a mobile node located in a foreign network. The arrow in red indicates that an IP tunnel is used between home agent and foreign agent.

For the second scenario, the mobile node itself serves as the endpoint for the IP tunnel. The mobile node informs its home agent about its new address and an IP tunnel is then established between the home agent and the mobile node. The address assigned to the

mobile node via DHCP now serves as the destination address used for the tunnel and hence is known as a *collocated care-of-address*.

Home and foreign agents advertise their services by periodically sending out **Agent_Advertisement** packets [39]. In turn, a mobile node may send out **Agent_Solicitation** packets to find agents. Additionally, a mobile node is also required to inform its home agent about its current care-of-address on a periodic basis. In this way, the home agent can keep a mapping between the mobile node's current care-of-address and its home address. This mechanism allows the home agent to act as a proxy server for the mobile node [39]. Network layer identity portability is thus achieved with Mobile IP.

6.3. Identity in P2P Networks

Since P2P networking deals with the application oriented layers of the OSI model, the use of an IP address at the network layer is assumed. Moreover, this IP address is used in conjunction with transport layer TCP port numbers to identify the peer within the P2P network. Additionally, a key may be used to identify a peer; this being generated by applying a hash algorithm to the IP address[24]. Sequence numbers for the identification of application layer protocol messages are also used in P2P networks.

Depending on the application at hand, user and content identification may also be required in a P2P network. The techniques used for the identification of users and content are described within the following two sub-sections, respectively.

6.3.1. User Identification in P2P Networks

The identity provided by the TCP/IP connection identifiers alone does not suffice for permanent user identification due to the transient connectivity inherent to peers in a P2P network. As a result, the connection identifiers may change between successive connections of a given peer (e.g. due to the use of DHCP).

Instant messaging P2P applications in particular require the user to have such a permanent identity, due to the following reasons:

[24] As noted in Sub-section 5.2.6, a peer key is used in the Chord content discovery technique.

- Instant messaging P2P applications are generally mediated with a UCP2P server. Thus, an entry for a user in the database of this server is only created once (upon registration) if a permanent identity is used. The entry only requires updating to reflect the current status of the user and the connection identifiers for the user.
- A contact list containing the permanent identities of peer users can be associated with the permanent identity of a user. This cannot be done if users do not have a permanent identity. The contact list provides additional information to the user, such as the status and profile of other users.
- A user's contact list may be stored at the UCP2P server with the user's permanent identity. Thus, as long as the user knows his/her permanent identity, he/she may use the instant messaging application on different computers, without having to rebuild the contact list each time a new computer is used. Amongst others, this facility is provided in MSN Messenger.

In instant messaging applications, this identity typically takes on the form of a user ID and a password. For example, in MSN Messenger the permanent identity is provided by a .NET Passport, which consists of an email address and a password. In ICQ, the user ID is a multi-digit number with which a password is also associated.

6.3.2. Content Identification in P2P Networks

In content sharing P2P networks, content may be identified in the following ways:

- A key generated using a hash algorithm.
- Content attributes such as file name and file size.
- Metadata that describes the actual content.

The use of keys for content identification is made use of in Freenet, as well as in CAN and Chord. As described in Section 5.2, the use of keys for content identification facilitates content discovery. A notable disadvantage of using keys, however, is that a peer is required to know the exact key value for the content it wishes to discover. This is considerably hard to achieve, since the key value has to be determined by the peer from some user provided input (such as a textual description of the desired content).

More often, a mixture of metadata and content attributes are used for content identification. For example, content is identified by content attributes over the FastTrack P2P network. Additionally, however, metadata such as ID3 tags in Motion Pictures Expert Group Audio Layer 3 (MP3) music files, is also used when this is provided.

6.4. Security in Communications

Within any communications system, the security techniques employed have to ensure that the following objectives are met [40], [41]:

- *Identification*: To be able to identify the transmitter and the receiver and, moreover, to be able to verify these identities.
- *Non-repudiation*: To ensure that the transmitter cannot deny having sent a message and that the receiver cannot deny having received it.
- *Confidentiality*: To ensure that messages exchanged between the transmitter and intended receiver/s cannot be eavesdropped.
- *Integrity*: To ensure that messages exchanged between the transmitter and intended receiver/s cannot be modified by anyone during transfer.
- *Availability*: To ensure the survivability of a system's resources/services despite attack.

The objectives are met by employing security techniques that collectively perform the following four functions:

1) *Authentication* of transmitter and receiver ensures that identification and non-repudiation are achieved.
2) *Encryption* ensures that the confidentiality of messages is preserved.
3) *Integrity checks* ensure that the integrity of messages is preserved.
4) *Authorisation* aids in ensuring the availability of a system, since access rights dictate what may be performed on the system. Other system specific techniques may, however, be required to ensure that DoS attacks are unsuccessful.

6.4.1. Authentication and Encryption Techniques

Authentication and encryption are both performed with the use of ciphers. There are two main categories under which ciphers may be classified: *Symmetric* and *asymmetric*.

Symmetric ciphers are based on the use of one shared *secret key* that is known to the transmitter and receiver. There are two main types of symmetric ciphers: *Stream ciphers* and *block ciphers*. Stream ciphers use a key to generate a *keystream* which is mixed in some reversible way[25] with the *plaintext* so as to produce the *cipher text*. An example of a popular stream cipher is Ron's Code-4 (RC-4) [42]. Block ciphers operate on blocks of plaintext, with a given block being encrypted using a key to produce a block of cipher text [43]. An example of a popular block cipher is the Data Encryption Standard (DES).

Most symmetric ciphers support a number of key sizes (in bits). For a given symmetric cipher, the key size used is a measure of the cipher's strength. Encryption and decryption can be done very fast using a symmetric cipher and hence, symmetric ciphers are typically preferred over other ciphers to ensure confidentiality. Key management, however, poses a problem in symmetric ciphers, since both transmitter and receiver need to agree on a secret key before cipher text can be exchanged between them.

Asymmetric ciphers form the basis of Public Key Cryptography (PKC). Two keys are used in asymmetric ciphers: A *public key* and a *private key*. The public key is known to everyone, whilst the private is only known by the one to whom it belongs. This pair of keys is used as follows:

- For confidentiality and for implicit authentication of the receiver, the transmitter encrypts using the receiver's public key, whilst the receiver decrypts using its own private key.
- For authentication of the transmitter, the transmitter encrypts using its private key, whilst the receiver decrypts using the transmitter's public key. When used in this way the private key provides the transmitter with its *digital signature*, whilst the receiver verifies this digital signature using the public key.

[25] An XOR operation of the plaintext with the keystream is often used.

To attain confidentiality as well as authentication of the transmitter and the receiver, both key pairs may be used. The encryption and decryption processes performed in this case are illustrated in Figure 6.2. To obtain the cipher text, the transmitter first encrypts the message with its private key and then with the receiver's public key. In turn, the receiver first applies its private key and then the transmitter's public key, in order to obtain the plain text.

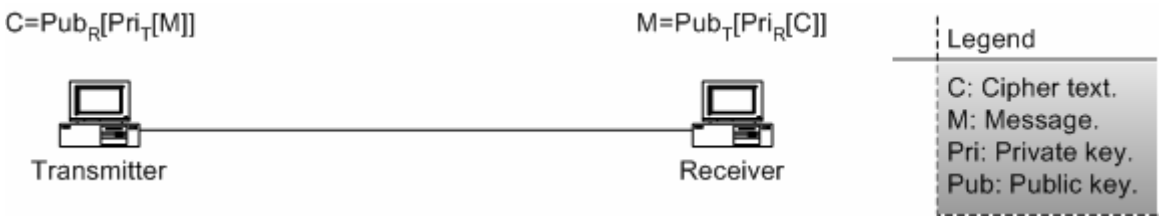

Figure 6.2: Authentication and confidentiality in asymmetric ciphers. Adapted from [43].

A popular example of an asymmetric cipher is the Rivest Shamir Adelman (RSA) cipher. The main advantage of using asymmetric ciphers is that the objectives of identification, non-repudiation and confidentiality are all met. Key management is also simpler in this case, as the transmitter and receiver do not need to share a key.

A key pair is generally issued by a Certifying Authority (CA), this being a *trusted* third party to the transmitter and receiver. The CA issues a *digital certificate*, binding an entity to its public key. Digital certificates are signed with the CA's private key and published (on a server), thereby making them accessible to everyone. The corresponding private key is only provided to the entity to which it belongs. If each entity were allowed to issue its own digital certificate, impersonation becomes possible, as an entity may then issue a digital certificate in another entity's identity. The CA thus plays an important role in ensuring that digital signatures may be trusted. The use of a CA in the network types dealt with in this dissertation is, however, problematic due to their highly decentralised nature.

A second drawback of asymmetric ciphers is that the encryption and decryption processes are computationally expensive; ultimately limiting the speed with which these processes may be performed. Hybrid systems are used to overcome this drawback. In hybrid systems, PKC is used for authentication and exchange of a secret shared key. This shared key is then used in conjunction with a symmetric cipher to preserve the confidentiality of messages.

6.4.2. Integrity Check Techniques

Integrity check techniques may also be classified under two main categories: *Message digests* and *Message Authentication Codes* (MACs). Both categories are reviewed here.

Message digests are generated using hash algorithms, such as those used in some of the content discovery techniques described in Section 5.2. Hash algorithms are one-way algorithms that are applied to a message in order to obtain a fixed length key. A good hash algorithm ensures that probability of two different messages generating the same key is negligibly small and that two similar messages will generate very different keys. A popular application of hash algorithms is for the storage of user passwords on a computer[26]. Two popular hash algorithms are Message Digest-5 (MD-5) and SHA-1 [42].

In communications, the hash algorithm is applied to the message, thereby generating a message digest (key). The message digest is sent along with the message, after encryption of both has been performed. If the message has been tampered with, the receiver should be able to detect this as the recomputed message digest should differ significantly from the one received with the message. The procedure is thus very similar to the use of a Cyclic Redundancy Check (CRC) to detect transmission errors. The difference is that a hash algorithm provides better security since the value of the message digest computed is unpredictable.

MACs are a particular form of message digests. A MAC is also generated with the use of a hash algorithm; however, a secret key known as a MAC secret is combined with the message in order to generate the MAC. The receiver thus requires the MAC secret in order to validate the MAC [42]. An attacker would therefore also require access to the MAC secret if the MAC is to be recomputed so that modification of a message would go by unnoticed.

[26] Keys, as opposed to plain text passwords, are stored.

6.5. Security in Ad Hoc Networks

The security techniques described in the previous section are all made use of in Ad Hoc networks. As an example of the application of some of these techniques, the first sub-section in this section reviews Wired Equivalent Privacy (WEP). Subsequently, the second sub-section reviews a technique proposed for the use of PKC in Ad Hoc networks.

6.5.1. IEEE 802.11 Wired Equivalent Privacy

WEP is the security protocol specified by the IEEE 802.11 working group for WLANs in order to preserve confidentiality and integrity. To attain these objectives, WEP makes use of RC-4 for encryption and a CRC for integrity check. WEP operates as follows [9]:

- A 32-bit CRC is applied to the frame body.
- The result is appended to the frame body and is known as an Integrity Check Value (ICV) trailer.
- A 40-bit WEP key is then combined with a 24-bit Initialisation Vector (IV) to obtain a 64-bit RC-4 key.
- This 64-bit RC-4 key is used to generate a keystream.
- The keystream is then XORed with the frame body and ICV to perform encryption.

At the receiver, decryption is performed by performing an XOR of the keystream with the encrypted frame body and ICV. The CRC for the frame body is then recomputed and compared to the ICV, thereby checking the frame body's integrity.

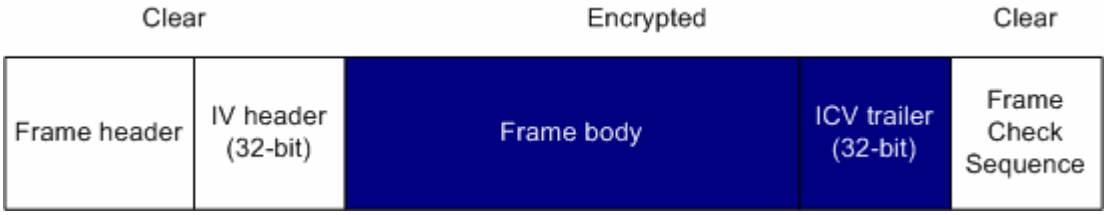

Figure 6.3: IEEE 802.11 frame format. Adapted from [9].

Figure 6.3 illustrates the IEEE 802.11 frame format. As may be observed, the frame body and the ICV trailer are both encrypted. The figure also illustrates that an IV header is used, part of which contains the IV. This is included since each frame may be encrypted with a

different IV value, chosen at random, ultimately resulting in a different keystream for each frame. WEP also allows up to four default WEP keys to be used amongst nodes. For a given frame, the WEP key used is indicated in the last byte of the IV header.

As its name implies, WEP was only intended to provide marginal security. Nonetheless, it has been heavily criticised for its shortcomings, especially since it was completely broken in 2001 [9]. Amongst the shortcomings noted are the following [9], [44]:

- No authentication mechanism is provided. WEP relies on MAC addresses for identification. These can, however, be stolen and impersonation is thus possible.
- The 40-bit WEP key is too small to provide adequate security. This limitation is a result of US export laws that prohibited the export of stronger security techniques.
- Since no mechanism for the exchange of shared keys (such as PKC) is provided, key management is problematic, often requiring manual configuration of keys.
- The integrity check is based on the use of a CRC, as opposed to a proper hash algorithm. Since a CRC function is linear, integrity may violated with relative ease since the ICV value may be adjusted without having to decrypt it.
- The 24-bit IV value is small, hence the same value is reused in a short time span.

To this end, the IEEE 802.11 working group is currently working on IEEE 802.11i, this being a new security standard to replace WEP. An interim standard was also co-developed by the working group and the Wireless Fidelity (WiFi) alliance. The standard is called WiFi Protected Access (WPA) and forms part of the upcoming IEEE 802.11i standard.

6.5.2. Public Key Cryptography in Ad Hoc Networks

As noted in Sub-section 6.4.1, the use of a CA in PKC poses a problem in decentralised networks. This problem is particularly significant in Ad Hoc networks since no fixed infrastructure is present. Even so, a technique for using PKC in Ad Hoc networks has been proposed in [40]. The technique does not require the use of a CA whilst still ensuring that digital signatures can be trusted.

The key pair for the CA in this case is replaced by a key pair for a key management service. The public key for this service is known by all the nodes within the network and these trust

digital certificates signed with the corresponding private key [40]. The key management service itself is split amongst *n* nodes which are termed 'servers'. Besides storing the public keys for all other nodes in the network, each server also has its own key pair.

To operate the key management service, a type of cryptography known as *threshold cryptography* is employed. In this case, the private key for the service is split into *n* shares, with each share assigned to a server. Out of the *n* servers, $t + 1$ servers are required to issue a digital certificate, where $n \geq 3t + 1$. The technique thus allows for up to *t* servers to be compromised, whilst still maintaining correct operation of the service.

To issue a valid digital certificate, $t + 1$ servers have to generate a partial signature with their share. These partial signatures are then used by a combiner to compute the private key for the service so that the digital certificate may be signed[27]. Any server may act as the combiner, since no additional information about the service key is provided to the combiner [40]. Since the combiner may be compromised, $t + 1$ servers may be used as combiners, so that at least one server is able to issue the digital certificate. Additionally, since all nodes know the public key for the service, a combiner is thus in a position to check the validity of the computed private key for the service. If the key isn't valid, a different set of servers is asked to generate partial signatures.

The key management service also employs share refreshing to combat mobile adversaries. Mobile adversaries are adversaries which compromise one server after another, thereby moving from server to server. Over time, a mobile adversary may therefore obtain access to more than *t* shares, allowing it to generate valid digital certificates. Share refreshing circumvents this problem essentially by providing servers with a way how to generate new shares from old ones, thereby eliminating the use of the old shares[28]. The new shares are computed in such a way that old shares cannot be combined with new shares to construct the private key of the service [45]. The mobile adversary in this case would therefore have to compromise $t + 1$ servers within this time period in order to generate valid digital certificates.

[27] In threshold cryptography, any $t + 1$ shares may be used to compute the private key for the service.
[28] The servers collaborate with each other to do this.

6.6. Security in P2P Networks – JXTA Security Model

As with Ad Hoc networks, the security techniques described in Section 6.4 may also be applied to P2P networks. Having said this, surprisingly few security measures are implemented in current P2P networks. To our knowledge, none of these security techniques are in fact employed in content sharing P2P networks. Instant messaging P2P networks are only marginally superior in this respect, since most of them even transmit the password associated with the user identity in clear text [46]. MSN Messenger, however, transmits a key generated from the password using MD-5. Jabber also provides support for the Secure Sockets Layer (SSL) protocol; however, the use of SSL is not mandatory.

As part of the JXTA framework for P2P networks, a security model is also provided so that JXTA based P2P networks may adopt security measures. This security model is reviewed here. The security model is based on the following three components [47]:

1) The adoption of Transport Layer Security (TLS), this being an IETF standard based on SSL, for secure transport of information.
2) Exploitation of the JXTA protocols' end-to-end transport layer independence.
3) Use of digital certificates in a way that neither mandates nor excludes CAs.

TLS was adopted in the JXTA security model to perform authentication, encryption and integrity checks. TLS defines two protocols: The *record protocol* and the *handshake protocol*. These protocols allow for a number of ciphers and integrity check techniques to be used, such as those described in Section 6.4.

The record protocol is layered on top of a reliable transport layer protocol, such as TCP. The record protocol performs encryption and integrity checks as follows:

- Message confidentiality is achieved with the use of a symmetric cipher, such as DES or RC-4. The record protocol generates a shared key for each connection.
- Message integrity is assured since each message includes an integrity check. A MAC is used for this integrity check. The MAC is generated using a hash algorithm such as MD-5 or SHA-1.

The handshake protocol performs authentication of both transmitter and receiver using PKC. An asymmetric cipher such as RSA is used in this case. Having performed authentication, the protocol is then used for the exchange of the shared key and the MAC secret. When the record protocol and handshake protocol are used together, TLS therefore provides a hybrid security system.

Much like TCP/IP, the protocols defined by the JXTA P2P framework are independent of the underlying protocols in use. From a security viewpoint, this is advantageous as it assures end-to-end security since, in no circumstance, is protocol conversion required.

Since authentication is performed in TLS, digital certificates are required. Since the security model takes into account the fact that the use of a CA may not always be possible in P2P networks, it provides three possible alternatives for the issuing of digital certificates:

1) A well known CA may be used if this is possible.
2) Peers may generate their own digital certificates.
3) Since the formation of peer groups is possible in JXTA, a peer within such a group may be designated as the CA of the peer group.

Although the last two alternatives dispense with the use of a well known CA, the solution provided is trivial in nature, since the issue of trust is not actively addressed.

6.7. Anonymity in Ad Hoc Networks

Techniques for the preservation of anonymity in Ad Hoc networks are still in inception. This situation may be attributed to the following reasons:

- Anonymity is generally considered an application specific issue and is therefore dealt with at the application layer.
- A large amount of the research carried out in Ad Hoc networking is done for the military. Identification of nodes, as opposed to anonymity thereof, is of paramount importance in military applications.

Having said this, in [48], the application of *pseudonyms* for anonymity in Ad Hoc networks is proposed. Three pseudonym types are described as:

1) *Person pseudonym*: A pseudonym that can be used so that a person can be recognised over a relatively long period of time, but not by his/her real identity. A typical example would be the use of a person's email address as a person pseudonym.
2) *Relationship pseudonym*: A pseudonym that is similar to a person pseudonym, but it is only used for communication with one specific other person.
3) *Transaction pseudonym*: A pseudonym that is only used for a single transaction and is thus, relatively short lived.

Of interest is the fact that these pseudonyms are proposed for use not only at the application layer, but also at the network and data-link layers. As a result, each node needs to be capable of changing IP and MAC addresses on a per transaction basis.

6.8. Anonymity in P2P Networks – Freenet

Depending on the intended application of a P2P network, the preservation of anonymity may be a desirable feature, if not a requirement. In this section, the Freenet content sharing network serves as an example of how anonymity can be achieved in a P2P network.

As described in Sub-section 5.2.4, Key Requests in Freenet propagate from peer to peer according to a routing decision, until the desired content is found or the hops-to-live expires. The corresponding reply follows the reverse path of the request path.

Anonymity of the requesting peer in this case is achieved since a peer in the request path cannot tell whether its predecessor in the request path initiated the Key Request or is merely forwarding it [37]. The Key Request itself contains no information about the requesting peer. The reply can be sent back to the requesting peer since each peer in the request path keeps track of the predecessor peer which sent the Key Request.

Since a reply to a Key Request contains TCP/IP connection identifiers besides the key itself, these identifiers are occasionally replaced by any peer on the reverse path with its own identifiers, so as to preserve the anonymity of the peer that supplied the key. As a result, the TCP/IP connection identifiers contained in the reply do not necessarily belong to the peer that supplied the key; the identified peer could be any peer on the reverse path.

In conclusion, the technique by which anonymity is achieved in Freenet ensures that a peer, at maximum, only knows about three other peers that participated in content discovery: Its predecessor, its successor and the peer identified by the connection identifiers. More importantly, none of there three peers are necessarily the requesting peer or the replying peer. Anonymity is therefore achieved.

6.9. Comparison and Applicability of Identity, Security and Anonymity

As noted in Section 6.3, peers in P2P networks are not the only entities that require identification. Users and content also require identification. In contrast to this, the data-link and network layer identification techniques used in Ad Hoc networks only need to provide an identity to nodes within the network. The complexity lies in providing identity portability whilst still maintaining interoperability with fixed IP networks.

In summary, since the fields of Ad Hoc and P2P networking assume relevance at different layers of the OSI model, very little similarity exists between the techniques used for identification. The only commonality present is the use of sequence numbers to identify messages. The divergence in the two research fields is therefore beneficial.

In terms of applicability, the sub-aspects of user and content identification may become relevant to the aspect of identity in Ad Hoc networks, as application layer issues gain importance in the field. To this end, it is worth noting that user identification as performed in P2P networks cannot be applied to Ad Hoc networks because a UCP2P server is relied upon to ensure uniqueness of, and provide storage for, these identities. Conversely, the content identification techniques used in P2P networks can be applied to Ad Hoc networks since no server-mediation is necessary in this case.

The security techniques employed in Ad Hoc and P2P networks are similar. This is because general security techniques used in communications are applied in both network types, and because these security techniques are largely independent of the OSI layer at which they are implemented. Both network types in fact make use of symmetric ciphering and message digests to attain confidentiality and integrity of messages, respectively. Additionally, the use of PKC for identification and non-repudiation is problematic in both Ad Hoc and P2P networks, since trust in digital certificates may have to be attained without the use of a CA. Thus a high degree of similarity in techniques, as well as in issues, is present, suggesting that the research on security in the two fields would benefit from convergence.

P2P networking is likely to benefit from convergence, as well as from the applicability of security techniques used in Ad Hoc networking since research in this aspect is more advanced in Ad Hoc networks, notably due to the military interest in such networks. For example, the technique described in Sub-section 6.5.2 for the use of PKC in Ad Hoc networks provides a better solution than the technique used in the JXTA security model, as it addresses the issue of trust in the absence of a CA. Moreover, the technique does not rely on any particular characteristic of Ad Hoc networks and should therefore lend itself to application in P2P networks.

As demonstrated by the example from Freenet in the previous section, anonymity in P2P networks is generally dealt with in an application specific manner. Moreover, the example demonstrates that anonymity can be layered on top of identity and hence, that anonymity and identity are not mutually exclusive. This contrasts with the use of pseudonyms at the application, network and data-link layers, as proposed for Ad Hoc networks.

The convergence of research and the applicability of techniques to preserve anonymity in Ad Hoc and P2P networks should also be possible for the same reasons given previously with regards to the aspect of security. In this case, the application of techniques would benefit Ad Hoc networks, since the techniques used in P2P networks are more practical then what has been proposed for Ad Hoc networks, due to the fact that network and data-link layer identities may be retained. The retention of identity at these layers would ultimately facilitate identification, routing and security in Ad Hoc networks.

7. Multicasting

7.1. Introduction

Multicasting may be defined as the transmission of data to a group of receivers identified by a single destination address and hence, is intended for group oriented computing [49]. In contrast, unicasting is the transmission of data to only one receiver, whilst broadcasting is the transmission of data to all possible receivers.

The first section within this chapter provides an overview of multicasting techniques used in networks, whilst the following two sections review multicasting in Ad Hoc and P2P networks, respectively. The chapter is subsequently concluded with a comparison between multicasting in Ad Hoc and P2P networks, which includes an analysis of the applicability of techniques, used in one network type, to the other.

7.2. Overview of Multicasting

The need for multicast techniques evolved as a result of the inefficiency perceived in performing broadcasting, or replication of unicast packets, over switched/routed fixed networks. In these networks, the multicast protocols used presently are implemented at the network layer. Two main algorithms are used for performing multicasting in these protocols, these being [49]:

1) *Source tree-based algorithms*: In which the source periodically *floods* the network, relying on intermediate multicast routers to reach all receivers. The multicast data stream thus looks like a tree stemming from the data source. Subsequently, *prune* messages are used to ensure that subnets, in which no receiver is interested in receiving the multicast data, stop receiving it [50]. This algorithm is thus also known as a *flood-and-prune* algorithm or as a *dense mode* algorithm.

2) *Shared tree-based algorithms*: In which Rendezvous Points (RPs) are used, these being specially designated multicast routers. Multicast routers wishing to obtain a copy of the multicast data stream would send a **Join Request** to the appropriate RP. The RP would then cast the data stream to the given router, which would in turn cast it to the receiver/s that requested it. On the other hand, a source would announce its existence to one RP and forward its data stream to it [50]. In this algorithm, the tree is shared since multiple sources may be handled by one RP. This algorithm is also known as a *sparse mode* algorithm.

In both algorithms, the objective is to reduce the total amount of data traffic. When compared to unicasting, this is achieved since replication of the data stream is performed at multicast routers close to the intended receivers; a given multicast router need only be provided with one copy of the data stream, as opposed to a copy for every receiver. On the other hand, in contrast to broadcasting, multicasting ensures that a copy of the data stream is only delivered to receivers that require it. In broadcasting, all potential receivers receive a copy of the data stream, independently of whether or not they require the data stream.

7.3. Multicasting in Ad Hoc Networks

As for fixed networks, multicasting in Ad Hoc networks has the potential to reduce the amount of data traffic significantly, especially if the inherent broadcast property of the wireless medium is exploited. Nonetheless, the design of multicasting techniques suitable for Ad Hoc networks also has to take into consideration the several additional constraints. In this case, mobility is particularly problematic since the delivery path changes as a result [49]. An ideal solution to this problem would also ensure that control traffic does not increase substantially with mobility, thus making the multicast technique scalable.

In an Ad Hoc network, broadcasting may essentially be considered as the simplest form of multicast. Intuitively, broadcast is easy to perform in a wireless medium and, additionally, has the favourable properties of reach and robustness. In applications where such properties are necessary, the use of broadcasting is considered a viable option. Nevertheless, the use of broadcasting severely limits the scalability of an Ad Hoc network, due to the unnecessary

duplication of data streams and the increased probability of data-link layer frame collisions due to this duplication. As a result, other multicasting algorithms have been proposed for use in Ad Hoc networks. These algorithms may be classified into four main categories [49]:

1) *Tree-based algorithms*: Tree based algorithms are equivalent to the algorithms used in fixed networks.
2) *Mesh-based algorithms*: In mesh-based algorithms, multiple paths between source and receivers are used for resiliency purposes. Mesh-based algorithms are particularly suited to Ad Hoc networks in which the topology changes frequently, since a new path does not have to be found each time a link fails.
3) *Stateless multicasting algorithms*: In stateless multicasting, no delivery tree needs to be actively maintained by the network, since the source explicitly mentions the intended receivers in the packet header. The underlying unicast routing protocol is then used for forwarding. Stateless multicasting is adapted to small groups.
4) *Hybrid algorithms*: Hybrid algorithms combine tree-based algorithms with mesh-based algorithms for their efficiency and resiliency, respectively.

All four categories of multicast routing algorithms have their own relative advantages and disadvantages. It is therefore highly unlikely that one category will prevail over the others in the future. For illustrative purposes, an example from each of the four categories has been briefly reviewed in the following sub-sections. A broader review of the different multicast protocols proposed for Ad Hoc networks can be found in [49].

7.3.1. Multicast Ad Hoc On-demand Distance Vector

Multicast Ad Hoc On-demand Distance Vector (MAODV) is a tree-based multicast routing protocol for Ad Hoc networks that builds on its unicast counterpart. In MAODV, RREQ packets are used by a node either when it wishes to join a multicast group or when it wants to send data to the multicast group but it does not have a route to it [49]. If the RREQ packet is not a **Join Request**, the operation is similar to unicast AODV, since any intermediate node with a fresh enough route may reply with a RREP. On the other hand, a Join RREQ may only be answered by a node that is already part of a multicast group. A node that does not satisfy these respective conditions rebroadcasts the RREQ packet.

In a similar manner to AODV, nodes also use the reception of RREQ and RREP to set up reverse and forward unicast route entries, respectively. Additionally, if the RREQ received is a Join RREQ, a reverse entry is also added to the **Multicast Route Table**. This entry is activated later on if the route is selected to form part of the multicast tree[29].

A member node of the multicast group may only reply to a Join RREP if its recorded sequence number for the multicast group is at least equal to that contained in the Join RREQ [49]. Besides adding a forward unicast route entry, the corresponding RREP for a Join RREQ is used by intermediate nodes to create a forward multicast route table entry.

Since a node may receive multiple RREP in response to its RREQ, it selects the one with the highest group sequence number and shortest hop count. Subsequently, it enables the next hop for this RREP in its own Multicast route table and unicasts a Multicast Activation (**MACT**) to the selected next hop [49]. If the next hop is a member of the multicast group, it simply enables the entry in its multicast routing table. No further propagation of the message is necessary. Conversely, if the node is not a member of the multicast group, it would have received at least one RREP from its neighbours beforehand and kept the best next hop for its route to the multicast group. Hence, after enabling the entry in its Multicast routing table, it forwards the MACT packet on this route. This process continues until a member node is reached.

The first member of a multicast group becomes the *leader* of that group and is responsible for maintaining the group's sequence number and disseminating it to other nodes in the group [51]. This is done with the use of a **Group Hello** packet that is propagated to all members of the group. MAODV also specifies how nodes should react to changes in the tree. A prune mechanism is specified so that nodes may leave the group, as well a mechanism to cope with link breakages. The latter also implies that, unlike AODV, MAODV actively deals with link breakages since these affect the tree structure. MAODV may therefore be thought of as taking a *hard-state* approach to multicast routing.

A more in-depth description of MAODV can be found in [51].

[29] Multicast data is only sent along activated routes, so as to avoid replication.

7.3.2. On-Demand Multicast Routing Protocol

On-Demand Multicast Routing Protocol (ODMRP) is a mesh-based multicast routing protocol for Ad Hoc networks. ODMRP uses a *soft-state* approach to maintain group membership, thereby doing away with the use of an explicit mechanism for a node to leave a group. In ODMRP, group membership and multicast routes are established by source nodes on demand [49].

In ODMRP, given that a source node has packets to send to the multicast group, the source node periodically broadcasts a **Join Request** to the entire network [52]. A node receiving a non-duplicate Join Request would store the source node IP address and **Sequence Number** in its **Message Cache** to detect future duplications, and rebroadcast the Join Request. Additionally a reverse route to the source node is added/updated in the **Routing Table**, with the Join Request's previous hop IP address serving as the next hop indicated in the routing table. One should note that, at each node, the node lists its own IP address as the previous hop IP address within the **Join Request** (replacing the previous node).

When a **Join Request** reaches a multicast receiver, the latter sends a **Join Reply** to its neighbours. The Join Reply is computed after selection of the multicast routes since the multicast group may have multiple sources. Each source IP address and corresponding next hop IP address of a multicast group are contained in the Join Reply packet [52]. A node receiving a Join Reply would check whether its own IP address appears within the entries held in the Join Reply. If it does, this implies that the particular node is on the path to a source node and thus, part of the *forwarding group*. The node sets its own Forwarding Group FLAG (**FG_FLAG**) and broadcasts its own Join Reply. The next hop IP address is obtained from the routing table. The procedure is repeated until the source nodes are reached.

As may be noted, a mesh is formed that consists of sources, intermediate nodes, and multicast receivers. Collectively, these constitute a forwarding group so that the exchange of multicast data between sources and multicast receivers is possible.

A more in-depth description of ODMRP can be found in [52].

7.3.3. Differential Destination Multicast

The Differential Destination Multicast (DDM) protocol is a stateless multicast protocol suited for small multicast groups. In DDM, the source has exclusive control over group membership. To join a multicast group, a receiver needs to unicast a **JOIN** packet to the source for that session [49]. The source updates its Member List (**ML**) and replies with an **ACK**. A **LEAVE** packet is also defined so that a member can leave the multicast group.

The source controls group membership by encoding the receiver addresses in multicast data packets using a special DDM header [49]. As a result, each multicast packet contains a variable length list of destination addresses, known as the **Destination List**, thereby providing a way how packets can be self routed to each destination by means of the underlying unicast routing protocol.

DDM supports two modes of operation: *Stateless* and *soft-state* [53]. In the stateless mode, a node performing forwarding of multicast packets need not maintain any forwarding state. It simply consults its unicast routing table, in conjunction with the Destination list, to determine the next hop/s to which a multicast packet should be forwarded.

In soft-state mode, a multicast packet would contain in-band information. This information is used by each node to determine the receivers to which the previous packet was destined. Hence, in soft-sate mode, the Destination list only needs to be present in the first multicast data packet. Any subsequent changes in the destination list may then be indicated in the multicast data packet in a differential manner; hence the protocol's name.

A more in-depth description of DDM can be found in [53].

7.3.4. Ad Hoc Multicast Routing Protocol

The Ad Hoc Multicast Routing Protocol (AMRoute) creates a bidirectional shared tree using only group sources and receivers as tree nodes for data distribution [49]. The protocol has two main components, these being: *Mesh creation* and *tree setup*.

For a mesh to be created in AMRoute, a node has to be designated as a *logical core*. Each node begins by identifying itself as the core of a one-node mesh, consisting of only itself. This core node sends **JOIN_REQ** packets with increasing TTL to discover other members of the same multicast group [54]. Given that a JOIN_REQ packet reaches another member of the same multicast group, but belonging to a different mesh, the member responds back with a **JOIN_ACK**. A bidirectional tunnel is thus established between the core of the original mesh and the member from the second mesh. Both meshes are merged in this way. Additionally, a core resolution algorithm is used to determine which of the cores of the previously disjoint meshes should become the core for the new mesh. This procedure ultimately results in a mesh of multicast group members which have *mesh links* between them[30]. The role of the core can also migrate to other member nodes throughout the mesh's lifetime, so as to cope with network dynamics [49]. A node may leave the group by sending a **JOIN_NAK** packet to its neighbouring nodes in the mesh.

Having formed the mesh, the core is then responsible for the setup and maintenance of a tree to all other members of the multicast group. To do so, the core periodically sends out a **TREE_CREATE** packet on all mesh links. A given group member receiving a non-duplicate TREE_CREATE packet would forward this on all mesh links except for the one on which it was received. Additionally, it marks the incoming and outgoing links as *tree links*. Conversely, a group member receiving a duplicate TREE_CREATE packet would send a **TREE_CREATE_NAK** packet back on the same mesh link from which the packet was received, forcing the node receiving this packet to re-mark the link as a mesh link. Thus, each non-core node considers the link along which a non-duplicate TREE_CREATE packet was received and every other link along which no TREE_CREATE_NAK packet was received to be a tree link for the group [54].

[30] One mesh link between two multicast group members may go through several intermediate Ad Hoc nodes that are not members of the multicast group.

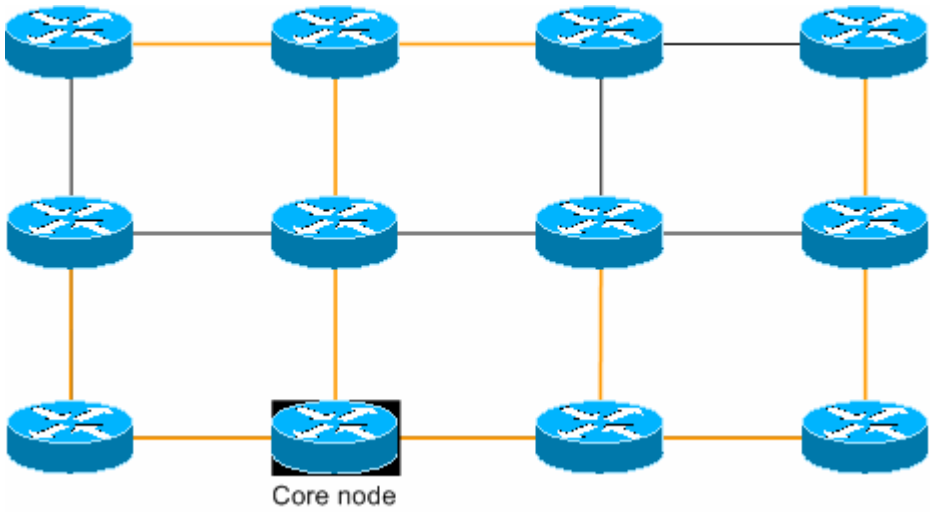

Figure 7.1: Example of tree formation in AMRoute.

Figure 7.1 illustrates one possible tree structure that could be formed over the mesh of multicast Ad Hoc nodes shown. As may be observed, the tree links are in actual fact a subset of the mesh links. This shared tree may then be used for multicast communication amongst members. Maintenance of the mesh itself is also taken care of by the core node periodically; managing mesh mergers and disjoints, as well as node membership changes.

A more in-depth description of AMRoute can be found in [54].

7.4. Application Layer Multicasting in P2P Networks – Overcast

As noted in Section 7.2, multicasting techniques are generally implemented at the network layer. The relevance of this aspect to P2P networks is therefore limited. From an applicability perspective, to our knowledge, none of the current P2P overlay networks make use of IP layer multicasting, for two reasons:

1) TCP is the transport layer protocol of choice in current P2P networks. IP multicast cannot be used in conjunction with TCP, since TCP relies of the use of ACK packets for error, flow and congestion control. The transmission of such packets from a receiver to a source cannot be performed if IP multicasting is used.
2) IP multicast support is not ubiquitous at the Internet's IP layer. Internet based P2P networks therefore cannot rely on the use of IP multicasting.

From a research perspective, multicasting in P2P networks is gaining relevance with the concept of ALM. ALM is the organisation of participants in a point-to-multipoint (multicast) configuration so as to form a delivery tree, in order to increase efficiency [55]. When applied to a P2P network, ALM involves the organisation of peers so as to form an application layer multicast tree. Although not as optimal as network layer multicasting, ALM can still improve efficiency significantly when compared to other techniques, such as application layer broadcasting. Overcast is one such ALM technique, proposed in [56], which has been reviewed here.

Overcast is an ALM technique that provides scalable and reliable single source tree-based multicast, by implementing a simple protocol for building efficient data distribution trees that are adaptive to changing network conditions [56]. A peer that wants to join an Overcast network must first bootstrap onto such a network by contacting a global well-known registry. This registry provides the peer with a list of the various Overcast networks that it may join following which, the peer connects to the chosen network and begins a process of self-organisation so as to place itself in an optimal position within the multicast tree.

Initially, the peer is connected to the source peer itself. Having connected, the peer then performs an iterative procedure in which it tries to locate itself further away from the source peer by replacing its *parent* peer, without sacrificing the amount of bandwidth between it and the source peer [56]. This procedure is known as the *tree protocol*.

At each step of this procedure, the peer compares the bandwidth between it and the parent, to the bandwidth between it and the parent through each of the parent's *children*. One of these children is chosen to replace the parent, provided that the bandwidth to the source peer does not decrease significantly. The procedure stops when no suitable replacement for the parent is found from among the children. Bandwidth is calculated by monitoring the download time of 10 KB of dummy data. The measured bandwidth is considered to be significantly different if it differs by more than 10%.

A peer periodically revaluates its position in the tree by measuring the bandwidth through its *siblings* (i.e. other children of its parent), parent, and grandparent [56]. The peer moves

down or up in the tree according to these measurements. Additionally, the peer keeps an **Ancestor List** which is used for reconnection if the parent peer goes offline.

Overcast also provides a protocol, called the *up/down protocol*, so that the source peer can keep track of the status of other peers in the multicast tree. Each peer within the network maintains a **Table of Information** about all peers lower than itself in the tree, and also logs all changes to the table [56]. The source peer should therefore have information about all peers. This table is updated periodically, by having children contact their respective parent peer to report the following information:

- *Death certificates*: Children of the reporting peer that haven't reported within the last period.
- *Birth certificates*: Peers that have become children of the reporting peer.
- Similar changes reported to the reporting peer by its children.
- Any required additional information, such as a group membership count.

The information provided by this protocol may then be used by the source peer for various application-specific functions. Additionally, it may be used for the placement of a newly connected peer (by the source) in an optimal position in the network; thereby significantly reducing the amount of iterations required in the previously described tree protocol.

Results described in [56] show that efficiency can improve significantly with the use of Overcast. The application of ALM techniques such as Overcast to P2P networks is therefore promising in view of increasing scalability. Nonetheless, several outstanding issues need to be solved before the use of such techniques becomes viable. Dealing with transient connectivity of the source itself is one such issue in P2P networks. As presented in [56], the source is in actual fact an actively maintained server, with all other nodes acting as clients to it. A second issue is that only one source is present in Overcast. An Overcast network thus has to be formed for each source that wishes to distribute content to others. With regards to P2P networks, a shared multicast tree is possibly more appropriate for distribution of content amongst peers.

7.5. Comparison and Applicability of Multicasting/ALM

Although implemented at different layers of the OSI model, multicasting and ALM in Ad Hoc and P2P networks respectively, share a great degree of similarity because the techniques used are independent of the OSI layer at which they are implemented. The following similarities observed suggest that convergence of the two research fields would be beneficial:

- The use of tree-based techniques is common to both networks.
- The creation and management of multicast groups is problematic in both networks, due to their decentralised nature. In Ad Hoc multicast routing protocols, the source/s generally have to create and manage the multicast group. Handover of these tasks has to be catered for due to node mobility. In Overcast, the source node also performs creation and management of the group. Additionally, since a server is used as the source node, the issue of transient connectivity is not addressed.
- The creation of shared multicast trees is also problematic in both networks because the transient connectivity associated with nodes/peers implies that the concept of RPs cannot be easily applied to these network types.
- The discovery of multicast groups provides another commonality in that it is also problematic to both network types. In Ad Hoc multicast routing protocols, discovery generally forms part of the source node's management tasks since this transmits Join Requests periodically. Alternatively, knowledge of the presence of the multicast group is assumed in these protocols. In Overcast, the solution relies on a global well-known registry and hence, the issue of transient connectivity is not addressed in this case either.

From an applicability perspective, the aspect of multicasting has received a greater degree of attention in Ad Hoc networks, then ALM has in P2P networks. Various protocols for performing multicast routing in Ad Hoc networks have been proposed under each of the four categories.

To this end, the application of multicasting algorithms for Ad Hoc networks to ALM in P2P networks is worth considering due to the similar properties that the two network types

share. The following list outlines the benefits that can be derived from the application of Ad Hoc multicast algorithms other than tree-based algorithms, to ALM in P2P networks:

- The use of mesh-based algorithms for multicasting in P2P networks could provide better resiliency than the tree-based technique used in Overcast. Resiliency is important in view of the transient connectivity associated with peers.
- Stateless multicasting could possibly be applied to P2P applications in which relatively small groups need to collaborate amongst each other. One such application could be P2P audio/video conferencing. The benefit derived in this case would be the reduction of control traffic required to maintain the multicast group.
- The application of hybrid techniques to P2P networks may be beneficial to applications that require a compromise between resiliency and efficiency.

8. Network Management and QoS

8.1. Introduction

The aspect of network management can be divided into two sub-aspects, these being: *Monitoring* and *Control*. Network monitoring refers to the collection of information about the usage of network resources. Information that is typically collected includes bandwidth usage across network links and computational load at network nodes. Network control refers to the policing of network resources usage. Network control is exercised for reasons of network performance, resource reservation for QoS, and access control, amongst others.

QoS can be defined as a set of application requirements that need to be met by the network whilst transporting a stream of packets from source to destination [57]. Real-time applications, such as audio and video conferencing, in particular pose strict QoS requirements. These requirements are application specific, and typically quantified in a set of desired service parameters, such as: Bandwidth, delay, delay variation (jitter) and BER. A given application would have a specified value for all of the parameters that affect its performance. The network may therefore be thought of as providing a service to the application; the desired service parameters defining the required quality of this service.

In this chapter, a comparison of these two aspects in Ad Hoc and P2P networks is provided. Both aspects are being dealt with in this chapter since a degree of interdependence amongst these aspects is also present in this case. The first three sections review network management in Ad Hoc and P2P networks. The following two sections review QoS in a similar manner. The last section concludes this chapter with the comparison, which also includes an analysis of the applicability of techniques, used in one network type, to the other.

8.2. Network Management in Ad Hoc Networks – ANMP

Network management in Ad Hoc networks is considerably harder to perform when compared to fixed networks, for the following four constraints [58]:

1) The topology of an Ad Hoc network is dynamic, thus requiring automated reconfiguration on network management entities.
2) Due to resource limitations at each node in an Ad Hoc network, and due to decentralisation, network management has to be performed by the combined efforts of all nodes.
3) Ad Hoc networks can be employed for a diverse set of military or commercial applications. These different applications may have specific management requirements.
4) Protocols designed for performing network management in Ad Hoc networks should also be interoperable with their fixed network counterparts, for interconnection to be possible.

The Ad Hoc Network Management Protocol (ANMP) is a management protocol, proposed in [58], for use in Ad Hoc networks. ANMP was developed as a lightweight protocol that is compatible with the Simple Network Management Protocol (SNMP) used in fixed IP networks, thereby assuring that the protocol itself does not burden the Ad Hoc network significantly, and that interoperability with fixed networks is possible. Notable implementation details which ensure that these constraints are met include [58]:

- The packet format used is identical to the SNMP packet format.
- The User Datagram Protocol (UDP) is the transport layer protocol used.
- By default, lost data is not retransmitted, since periodical updates are performed. Nonetheless, retransmission may be requested by the application using ANMP.

Figure 8.1 illustrates the hierarchy adopted in ANMP. As may be observed, a three-level hierarchy is employed; this being a trade-off between the cost of maintaining a hierarchy and control traffic minimisation. At the lowest level are the managed Ad Hoc nodes, known as *agents*. Agents in proximity to one another are grouped into *clusters*, and are managed

by a *cluster head*. In turn, a *manager* oversees the management of the cluster heads. The manager is frequently more than one hop away from the cluster heads.

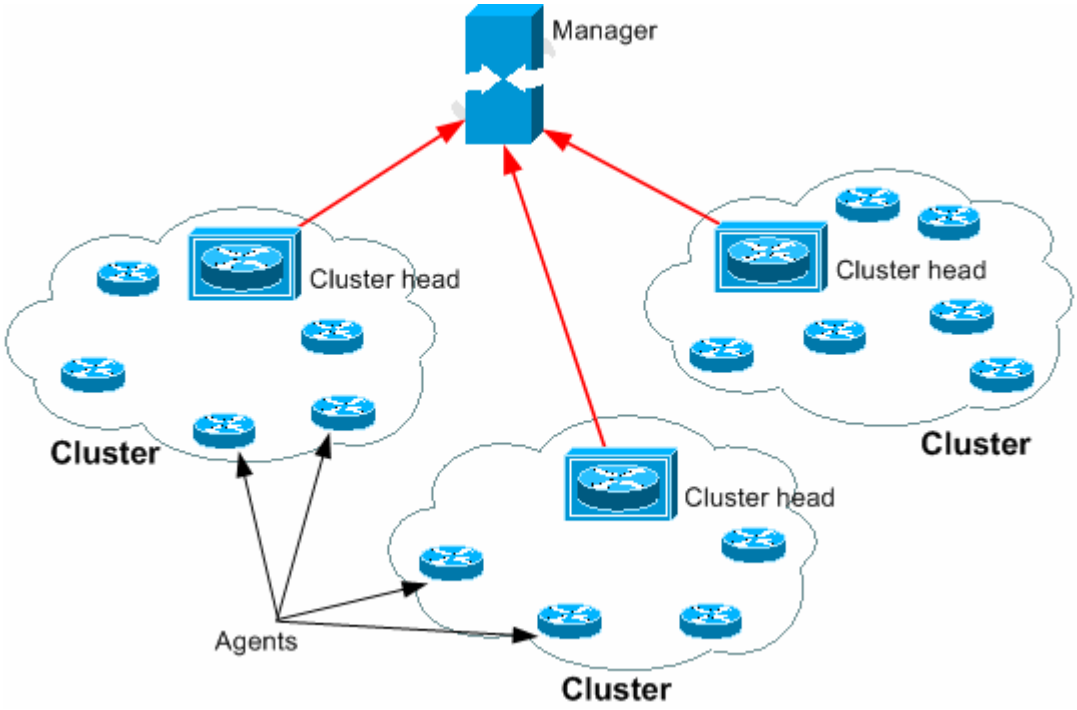

Figure 8.1: Hierarchical architecture in ANMP. Adapted from [58].

Due to mobility of nodes within an Ad Hoc network, the composition of clusters is likely to change over time, as are the nodes serving as cluster heads. In [58], two algorithms are described for forming and maintaining clusters. The first algorithm ensures that the formation of clusters is such that all nodes within a given cluster are two hops apart at most. The second algorithm uses node location information to distribute nodes evenly amongst clusters. Cluster head selection in both algorithms is performed according to **Node ID** and node location within the cluster; other methods can, however, also be used. Additionally, both algorithms are optimised for the following properties [58]:

- The size of clusters is neither too large nor too small. This ensures that the amount of messages exchanged between agents and cluster head, and between cluster heads and manager is minimal.
- Clusters are formed such that node mobility does not result in frequent recomputation of clusters.

- Movement of a node from one cluster to another does not imply immediate incorporation into the new cluster, so that cluster maintenance can be performed periodically as opposed to continuously. In the meantime, the node may be managed directly by the manager if necessary.

In ANMP, Management Information Blocks (MIBs) are also used in a similar manner to SNMP; the difference being that these are extended to include information specific to Ad Hoc networks. Every agent is responsible for periodically entering the required values into the MIB and, subsequently, transferring this to its cluster head. In turn, the cluster head collects all this information from its agents and transfers this information to the manager. In doing so, a cluster head may also summarise the information before transmission so as to minimise the management traffic. Each cluster head also maintains tables with the information gathered from each agent through the MIBs received.

A notable feature of ANMP is that the manager may also exercise control over agents. The protocol's scope is thus not restricted to monitoring. The MIB values for each agent gathered through the cluster heads form the basis for control decisions taken by the manager. For example, if the manager notes that an agent is running low on battery power, the manager may instruct the agent to enter sleep mode. The transmission of such instructions is triggered when a threshold for an associated value is reached [58]. Moreover, the function to be performed for each trigger may also be changed dynamically, thus giving the manager in ANMP the power to completely reconfigure the network nodes.

In conclusion, ANMP overcomes all of the four constraints that have to be dealt with in an Ad Hoc network: It can automatically perform reconfiguration, it is lightweight, it is extensible, and it is interoperable with SNMP. Having said this, the use of one manager as an overall controlling entity may be problematic since it introduces a single point of failure.

8.3. Network Management in P2P Networks – AVP

The lack of a centralised entity that can impose control restrictions in P2P networks can lead to a situation in which the network resources available are not shared equally amongst peers. Since a given peer is at liberty to use resources available as necessary, and since no mechanism is present to force a peer to give up resources for others, a race condition amongst peers may arise for resources. It is therefore hard to guarantee fairness. Secondly, network resources not only have to be shared amongst peers, they also have to be shared with other Internet applications, such as the WWW. The lack of imposed control in a P2P network may therefore affect other Internet users altogether.

To this end, the concept of an Active Virtual Peer (AVP) is proposed in [59]. The proposed architecture is hierarchical in nature and shares a high degree of similarity with the FastTrack P2P architecture reviewed in Sub-section 5.2.3. An AVP in this case is analogous to a supernode in the FastTrack architecture. Ordinary peers maintain a connection to an AVP, such that an AVP may be thought of as being representative of a community of peers [59]. In turn, AVPs maintain connections to each other.

The concept of an AVP is used to implement network monitoring & control functions. As shown in Figure 8.2, these functions are divided across three different vertical planes, with each vertical plane present, to a varying degree, at the three horizontal layers.

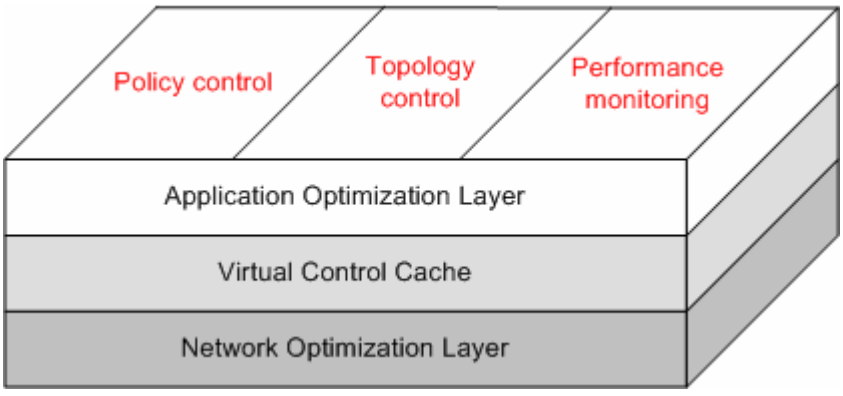

Figure 8.2: AVP Structure. Adapted from [59].

The Application Optimisation Layer (AOL) is responsible for control and optimisation of the P2P relationships at the application layer. At this layer, the vertical planes have the following significance [59]:

- *Policy control* consists of the application of policies for application layer routing. These policies may be influenced by parameters such as *virtual peer state* and *virtual link state*, which are analogous to parameters used in network layer routing. Access restrictions may also be implemented as part of policy control at the AOL.
- *Topology control* at the AOL is performed by enforcing the desired topology according to controlling parameters such as the number of AVP connections or the characteristic path length. Moreover, topology control may be adapted dynamically by using Application Layer Active Networking (ALAN) mechanisms, which allow an AVP to load and execute control code as necessary.
- *Performance monitoring* consists of the monitoring of performance statistics, such as the number of relayed or dropped messages, message inter-arrival times, and response times. It also includes the collection of topological information. This information may then be used to influence policy and topology control dynamically.

The Virtual Control Cache (VCC) layer is responsible for providing content caching at the application layer. Policy control may also be present at this layer, so as to control the use of such a cache or, for example, to implement a distributed cache amongst AVPs.

The Network Optimization Layer (NOL) is responsible for the implementation of dynamic traffic engineering techniques that map the P2P traffic onto the underlying network in an optimised way [59]. The implementation of such techniques is influenced by the control features provided by the underlying protocols in use. Policy control in this case would control the traffic volume allowed and transmitted. Topology control at the NOL is limited to the traffic engineering techniques themselves, whilst performance monitoring may be done for measurements of round trip delay, error rate or throughput.

Four main benefits are noted in [59] for the use of AVPs, these being:

1) On-demand resource aggregation becomes possible at the application layer.

2) Interaction between the application layer and the underlying layers can be controlled as necessary.
3) Caching is provided at the application layer.
4) Self-organisation and dynamic control of the overlay P2P network is made possible.

In conclusion, the concept of an AVP allows a P2P network to maintain a relatively high degree of decentralisation, whilst allowing network management to be performed.

8.4. Free Riding in P2P Networks

Free riding is a network management issue that's particular to content sharing P2P networks. The issue comes about due to the lack of centralised control in such networks. Free riding manifests itself in two forms [60]:

1) Free riding occurs when a number of peers in the P2P network do not share any content with other peers. Peers that do not share content are known as *free riders*. In Gnutella, the amount of peers that share no files has been measured to be 70% [2].
2) Free riding occurs when popular content only resides at a subset of the peers that make up the P2P network. This typically occurs when the said subset of peers are the only peers that place new content on the P2P network.

As a result of free riding, two problems may be witnessed in the P2P network:

1) A relatively small amount of new content is added to the P2P network regularly, since only a subset of the peers adds new content. This ultimately reduces the value of the P2P network.
2) Load balancing is not achieved, since content requests will only be serviced by peers that share content. Moreover, the vast majority of these requests will be serviced by the peers that share new content.

Both problems may ultimately induce users to stop using the P2P network. This is of concern especially when considering that the peers sharing new content are the most likely to leave due to the high service load they sustain, and due to lack of content at other peers.

Various solutions are possible to combat free riding, all with their own relative advantages and disadvantages. A number of these solutions are used in current content sharing P2P networks, whilst others have been proposed in literature.

In DHT content discovery techniques such as CAN and Chord, the second problem resulting from free riding is essentially avoided since peers are allocated content to share in a deterministic manner. The use of a hash algorithm ensures that the keys generated are pseudorandom and thus an equal distribution of keys between peers should be achieved. Additionally, content caching and replication techniques may also be used so that a popular key may be provided by a number of peers. Load balancing is therefore achieved; essentially by creating a distinction between the process of adding new content to the network and the act of sharing stored content. In other P2P networks, this distinction is not present, since adding new content simply involves placing the new content in the same directory as other shared content.

Two disadvantages, however, result from this distinction:

1) The first problem associated with free riding is not circumvented. Ideally, the solution to free riding would also ensure that new content is added to the network by all peers.
2) Peers have no control over what content is stored locally, as this is now decided by the content discovery technique. This second disadvantage may be especially problematic when considering the nature of some of the content shared.

Utility function based schemes may alternatively be used to minimise free riding [60]. The aim behind such schemes is to measure the 'utility' of a peer and subsequently reward/penalise the peer accordingly. A utility function may take into account any/all of the following three factors [60]:

- The amount of files shared by a peer.
- The size of the files being shared.
- The popularity of the files being shared.

A utility function is typically implemented in the P2P application itself and thus requires no centralised control. One such utility function based scheme is implemented in Kazaa; a P2P application that uses the FastTrack P2P network. The utility function used in this case is based on a system of points that reflect the *participation level* of a peer. Points increase when files are uploaded to other peers and decrease when the peer downloads files from other peers. In turn, the amount of points a given peer is allocated is used to control some of the content search functionality that Kazaa offers, and to prioritise between peers attempting to download the same file from another peer [61].

The amount of points a peer is allocated is dependent on the size of the files uploaded/downloaded. Thus, the second factor affects the number of points directly. An increase/decrease in any of the other two factors also indirectly affects the number of points, as the probability of an upload being performed increases/decreases.

The use of a utility function is effective in minimising both problems associated with free riding, since an incentive now exists for each peer to share new content with other peers in the P2P network. Additionally, load balancing should also be achieved since files are replicated at each peer that downloads them.

On the other hand, a notable disadvantage of implementing a utility function is that it cannot discern between peers that have the capability to increase their points significantly and others that cannot. One reason for this difference in capability may be the different Internet connections used by the different peers. In such a situation, the use of a utility function effectively discriminates against peers that only have a low speed (e.g. dial-up) Internet connection at their disposal.

A second problem with the use of utility functions is that of determining the number of points that should be allocated to a new peer, since no usage history exists. Assigning a low number of points may result in the new peer perceiving little value in the P2P network. Assigning a high number of points would encourage churn, since users would reinstall the P2P application in order to obtain these points.

8.5. QoS in Ad Hoc Networks – INSIGNIA QoS Framework

Constraints such as limited bandwidth, limited processing power and node mobility compound the problem of guaranteeing QoS requirements in an Ad Hoc network. As a result, this aspect of Ad Hoc networking is still within its infancy; more so since protocols in other aspects of Ad Hoc networking, such as routing protocols, aren't yet as established as their fixed network counterparts.

One of the more promising approaches to guaranteeing QoS requirements in Ad Hoc networks is the design and implementation of inter-layer QoS frameworks [57]. The reason for this is that conservation of scarce resources becomes possible if OSI layer boundaries are violated. One such framework is proposed in [62] and is called INSIGNIA.

The INSIGNIA framework is based on the following design parameters [57]:

- It is intended for adaptive real-time applications, where a base layer and an enhancement layer are defined. The base layer uses the minimum possible bandwidth, whilst the enhancement layer uses additional bandwidth for better quality. This technique is supported in Motion Pictures Expert Group (MPEG) audio and video.
- It should work in conjunction with a wide variety of routing protocols.
- It makes use of *in-band* signalling (as opposed to *out-of-band* signalling), since this responds faster to changes in a dynamic environment.
- Reservations at each node are made using a *soft-state* reservation (as opposed to a *hard-state* approach), since this is a more flexible approach to reservation management in a dynamic environment.

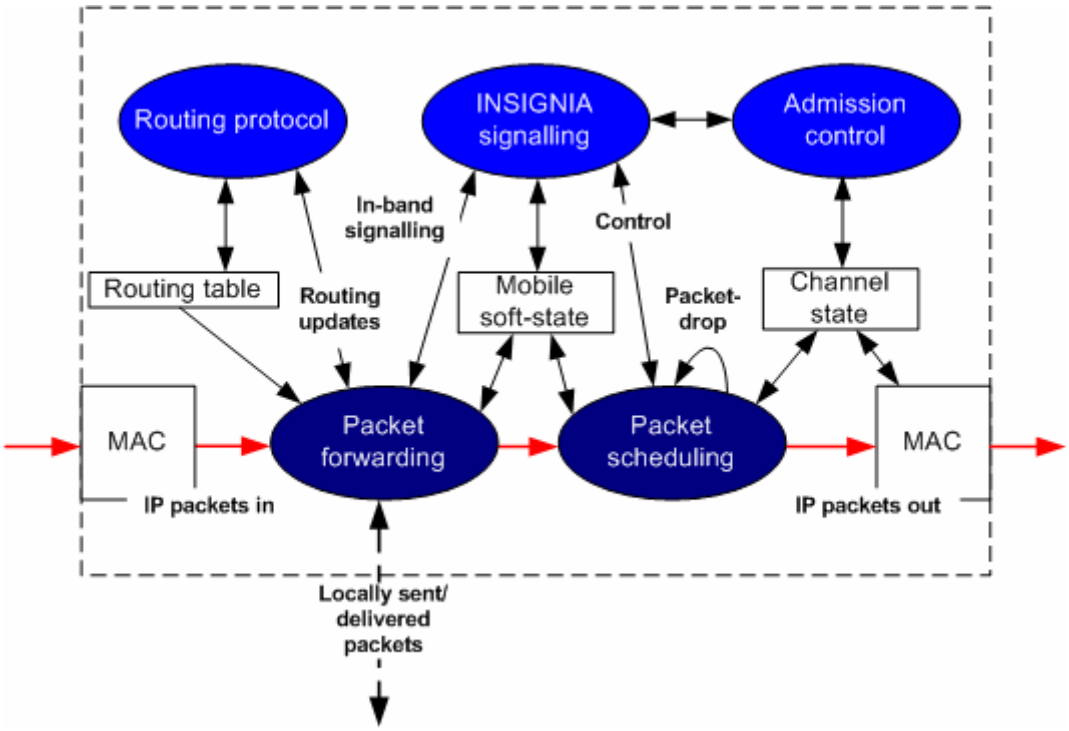

Figure 8.3: INSIGNIA QoS framework. Adapted from [62].

Figure 8.3 provides a system level diagram for the INSIGNIA QoS framework. As may be observed, the QoS framework interacts with both the network and data-link layers. The following is a description of the main modules in the diagram [62]:

- The *MAC* module provides QoS access to the shared wireless medium. The INSIGNIA QoS framework can operate over various data-link layer protocols. Additionally, it can operate over multiple data-link layer protocols at the IP layer.
- *Packet forwarding* classifies incoming packets and forwards them to the appropriate module. If the packet is a signalling packet, this is forwarded to the INSIGNIA signalling module. Similarly, if the packet is a routing update, this is forwarded to the routing module. On the other hand, if a data packet is received, this is either delivered locally or forwarded to the packet scheduling module, depending on the destination IP address.
- The *routing protocol* module provides the necessary routes to other nodes in the Ad Hoc network. As stated previously, the INSIGNIA QoS framework is designed to work with a variety of routing protocols, including the IETF MANET routing

protocols reviewed in Chapter 4. Proactive as well as reactive protocols are therefore supported.

- The *INSIGNIA signalling* module controls the establishment, restoration, adaptation and destruction of QoS paths, between source and destination [57]. Stream restoration algorithms are used to take care of route changes, whilst adaptation algorithms reallocate bandwidth (to the enhancement layers) if changes in the available bandwidth occur. In-band signalling is used, with the signalling information being placed in the **Options Field** of IP packets.
- *Packet scheduling* can be implemented using any of a number of supported scheduling algorithms in the framework. Additionally, location-dependent channel conditions may also be taken into account in packet scheduling.
- *Admission control* is responsible for allocating resources to streams based on base layer and enhancement layer requirements. These resources are held in soft-state; this being refreshed whenever packets pertaining to a given stream are received. Allocation is done by considering the monitored utilisation of resources and the amount requested.

The INSIGNIA QoS framework is an attempt to provide QoS guarantees in Ad Hoc networks. Several other issues remain outstanding, such as providing QoS for non-adaptable real-time applications, as well as the development of algorithms optimised for Ad Hoc networks, which can be plugged into such a framework. The performance of such a framework is therefore hard to predict.

Regardless, the inter-layer approach taken in INSIGNIA, whereby the framework is integrated with other protocols at the network and data-link layers, is a novel approach that might overcome some of the constraints posed by the Ad Hoc networking environment.

8.6. QoS in P2P Networks

The aspect of QoS has received a marginal degree of attention in research on P2P networks. Secondly, to our knowledge, none of the current P2P networks implement any techniques to guarantee QoS. There are two reasons for this state of affairs, these being:

1) The two currently dominant P2P applications, content sharing and instant messaging, require no QoS guarantees to be met. In content sharing P2P applications, delay, delay variation and bandwidth are not bounded in any way. Data loss cannot be sustained, although by using TCP as the underlying transport layer protocol, no data loss occurs. In instant messaging P2P applications, messages exchanged between peers should arrive within a short time span. However, this time span is not strictly bounded, implying that it need not be guaranteed. TCP is also used as the underlying transport layer protocol and hence no loss of messages occurs.
2) Techniques to guarantee QoS are generally implemented at layers below the application layer. Since P2P networks are application layer overlay networks, handling QoS issues is often *incorrectly* considered beyond the scope of P2P network design.

Regardless, QoS in P2P networks may assume importance in future service sharing P2P applications, whereby a peer would be able to utilise a number of services provided by other peers to make up the desired application. An example application could be P2P VoD. In [63], a QoS-aware service aggregation model for P2P networks is proposed. The model is made up of two tiers:

1) *On-demand service composition* is responsible for choosing suitable services to make up the desired application.
2) *Dynamic peer selection* is used to select the specific peers at which the selected service instances are executed.

On-demand service composition is used, by the peer that desires the application, to determine which of the discovered services should be used to make up the application. All

the selected services are then connected into a *service path*. Each service in the path would accept input (from the previous service in the path) with a defined QoS level Q^{in} and generate output with a QoS level Q^{out}. This output then serves as input to the next service in the path. The output quality of given service must therefore match the input quality of the next service in the service path [63]. A quality consistency check algorithm is used to ensure this in on-demand service composition. In summary, on-demand service composition has to ensure that the user's functional and QoS requirements are met.

Having performed this procedure, the peer then initiates dynamic peer selection. This is necessary since, for a given service that is part of the service path, it is conceivable that several instances of that service may be present at different peers in the P2P network, due to replication. Dynamic peer selection is thus employed to select the appropriate instance.

It is important to note that dynamic peer selection is performed in a distributed manner by having each peer select the next peer that should participate in providing the desired application. The selection is done according to locally maintained QoS information about the application layer next-hop candidate peers. The candidate peers are all the next-hop peers that can provide the next required service indicated in the service path. The QoS information used for selection includes parameters such as peer load, network bandwidth and delay. In other words, the selecting peer is responsible for ensuring that the selected peer has enough resources to perform the desired service. An additional parameter called *peer uptime* is also used to take into account the topological variation of a P2P network due to peer arrivals/departures. As can be noted, dynamic peer selection occurs along the reverse path of the application delivery path. More importantly, it ensures that the appropriate peer to perform a given service is selected; one that has sufficient resources and hence, can meet the QoS requirements.

The service aggregation model proposed in [63] for P2P networks presents two new concepts. First of all, it shows how P2P networks may be used for applications beyond content sharing and instant messaging by illustrating how a P2P network may be used for service sharing amongst peers. Secondly, it demonstrates that QoS issues are within the scope of P2P networks and, moreover, how these issues can be dealt with in a P2P network.

8.7. Comparison and Applicability of Network Management and QoS

In both Ad Hoc and P2P networking, the fundamental network management issue to be overcome is that of monitoring nodes/peers and enforcing control policies in a decentralised environment. The absence of a central entity conceivably makes both of these tasks hard. From an architectural perspective, the approach taken in ANMP and AVPs, for Ad Hoc and P2P networks respectively, is similar since a hierarchical structure is used. The use of cluster heads in ANMP is analogous to the use of AVPs.

On the other hand, two important differences may also be noted between the two techniques, which highlight the differences in network management issues between the two network types:

1) In ANMP, a number of design decisions ensure that the protocol is lightweight and interoperable with SNMP. These design decisions are there to meet the constraints of limited resources and interconnection, respectively. There are no equivalent interconnection constraints in P2P networks and, although resources at each peer may also be limited, they are so to a far lesser extent. Both constraints are therefore not of consideration for the design of P2P overlay networks.
2) One of the primary objectives for the use of AVPs in P2P networks is that of ensuring a fair distribution of network resources amongst peers, through the use of policy control. Conversely, in ANMP, the emphasis is placed on network monitoring, even though the protocol does provide support for network control operations. This difference is due to the fact that the techniques are implemented at different layers of the OSI model. At the network layer, monitoring of network nodes assumes more importance, whilst at the application oriented layers, network control is important in order to police the use of bandwidth by applications.

From a network management perspective, there is therefore a limited scope in the convergence of the two research fields of Ad Hoc and P2P networking in terms of network management.

The network management issue of free riding in content sharing P2P networks was given importance in Section 8.4 since equivalent issues may also arise in Ad Hoc networks. For example, Ad Hoc routing protocols generally provide each node with the option of not performing packet forwarding on behalf of other nodes. Although such an option is provided to take into account resource constraints at each node, it may also give rise to a form of free riding in Ad Hoc networks, whereby only a small subset of all nodes performs packet forwarding; for 'egoistic' reasons of resource conservation.

The applicability of techniques to combat free riding in P2P networks to Ad Hoc networks is therefore worth consideration. For example, a utility function at each node in an Ad Hoc network could be implemented, which keeps track of the amount of packets forwarded and assigns points accordingly. In turn, the amount of points could be used to control other routing functionality, such as non-neighbouring node discovery reach.

There is little which is of comparative value in how the aspect of QoS is dealt with in both network types. In Ad Hoc networks, QoS is primarily handled at the network layer. As a result, the QoS techniques employed are by and large concerned with QoS guarantees that have to be met on a per packet basis. The inter-layer approach taken in INSIGNIA deals with the network and data-link layers and therefore also handles QoS requirements for every packet/frame dealt with.

Conversely, the aspect of QoS is P2P networking has to be dealt with at the application layer. Moreover, it only has to be dealt with if the application at hand has QoS requirements. As noted in Section 8.6, current P2P applications have no QoS requirements. Nonetheless, if such QoS requirements are present, they can be dealt with by taking a system's approach. A clear example of this is the QoS-aware service aggregation model reviewed. In this model, the design of the P2P network is done in such a way that services can be offered by peers that can then be used to make up a desired application.

In conclusion, the present research divergence in the fields of Ad Hoc and P2P networking is beneficial with regards to the aspect of QoS, since the way this aspect is dealt with has to take into consideration the OSI layer at which it is dealt with.

9. Proposed Modification to Chord

9.1. Abstract

Within this chapter, we propose a modification to the Chord content discovery technique. As noted in Sub-section 5.2.6, Chord requires the redistribution of content keys and corresponding content, each time a peer joins or leaves the Chord ring. In our proposed modification, Chord maintains its ability to perform content discovery in a deterministic manner, whilst avoiding the need for redistribution of content. To date and to the best of our knowledge, this modification has not been proposed before.

9.2. Problem Statement

The Chord content discovery technique, described in Sub-section 5.2.6, has the notable disadvantage of having to perform redistribution of content keys each time a peer joins or leaves the Chord ring. We consider this attribute of Chord as a disadvantage due to the following three reasons:

1) Redistribution of content keys is in actual fact redistribution of content. Thus, although only an $O(1 / N)$[31] fraction of the content keys need to be moved each time a peer join or leaves [35], the amount of actual data transferred may still be significant. For example, one video file may easily be 700 MB in size, thus making the redistribution of even one content key significant.
2) Transient connectivity of peers is inherent to a P2P network. The more transient the connectivity, the more traffic will be generated due to content key redistribution.
3) In a P2P network, the amount of bandwidth available at each peer varies considerably, depending on the connection in use at each peer. When content key

[31] As in Sub-section 5.2.6, N is the number of peers that make up the Chord ring.

redistribution involves a peer with a low-speed (e.g. dial-up) Internet connection, the problem is compounded since redistribution of content keys will take long.

A naive solution to this problem would be to cache content at each peer so that redistribution is not always required. This solution is problematic for the following reasons:

- There is a limit to how much content may be cached at each peer, which depends on the amount of storage space a peer has at its disposal.
- Peer keys in Chord are generated from a peer's IP address using SHA-1. The peer key may therefore change between subsequent logons of a peer to the P2P network, due to a change in IP address[32]. This typically occurs when DHCP is used for IP address assignment. A change in the peer key implies a change in the range of content keys and hence, in the content that it has to cache.
- The number of peers *N* varies as peers join and leave the network. Moreover, there may be a significant difference in the number of peers at different times of the day and different days of the week, due to user usage patterns. As a result, no strict range of content keys and corresponding content to be cached at each peer may be defined, since this range varies with the amount of peers that make up a Chord ring, due to the inherent load balancing.

9.3. Proposed Modification

In the proposed modification to the Chord content discovery technique[33], the concept of a *content broker* is introduced. As opposed to storing a range of content keys and hence, the corresponding content itself; we propose a modification whereby each peer now locates content at other peers in the P2P network, the corresponding content keys for which fall under the given peer's *responsibility*.

In our modification, a peer is responsible for the same set of content keys that it would have been responsible for in unmodified Chord. The difference with modified Chord is that

[32] Since SHA-1 is used, the peer key is likely to be significantly different if a change in IP address occurs.
[33] Chord will be referred to as unmodified Chord, whilst the modification will be referred to as modified Chord, from here onwards.

responsibility for the set of content keys does not imply actual storage of the corresponding content at the given peer. It implies that the given peer is capable of deterministically locating the corresponding content at other peers in the Chord ring. Intuitively, in modified Chord, only redistribution of this capability needs to be performed when a peer joins or leaves the P2P network, as opposed to redistribution of the content keys and corresponding content in unmodified Chord. Thus each peer is now a content broker since it locates the content for which it is responsible, on behalf of all other peers.

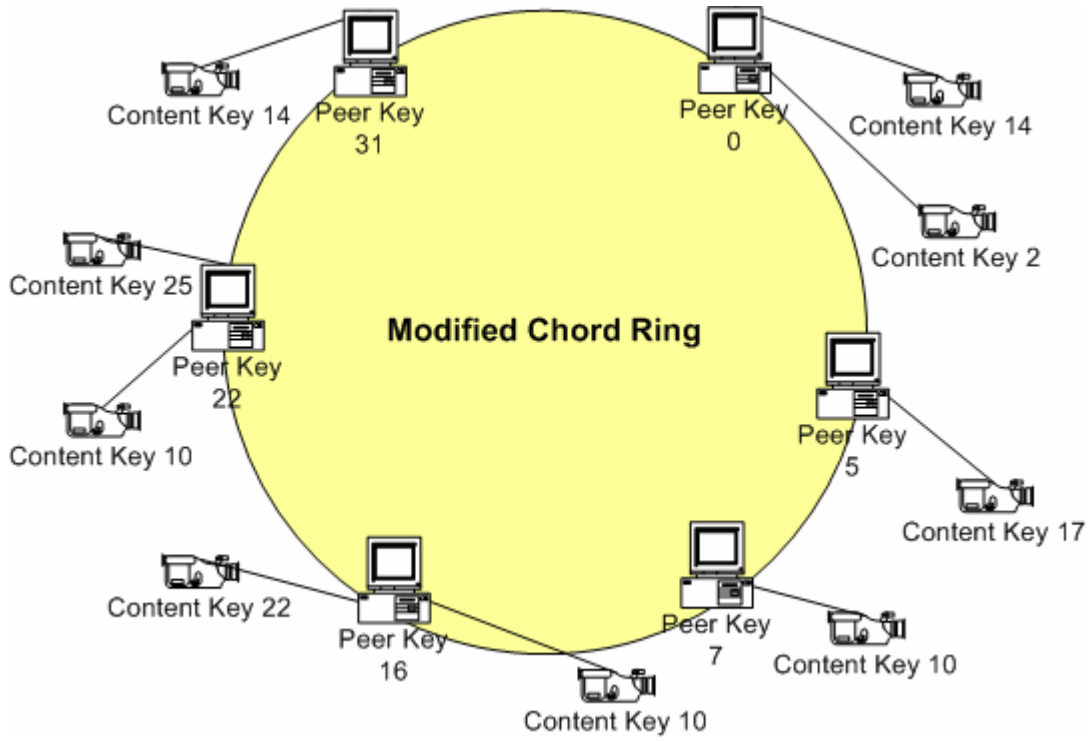

Figure 9.1: Example of a modulo 2^m ring in modified Chord for $m = 5$.

Figure 9.1 provides an example of how content keys may be distributed amongst peers in modified Chord. By comparing this figure to Figure 5.3, one can note that content keys and corresponding content need not be stored at the peer which is responsible for them any longer. Secondly, content keys and corresponding content may be replicated at a number of peers, since content brokers in modified Chord may locate multiple copies.

9.3.1. Content Tables

To locate the content for which it is responsible, a content broker now maintains a **Content Table**. The Content table contains an entry for each unique content key that the content broker is responsible for. Thus, a content table would have a number of entries corresponding to O(1 / *N*) fraction of all the content keys present on the Chord ring. For each content key entry, the Content table contains a list of peer IP addresses and TCP port numbers at which the corresponding content is located. Since a list of addresses is maintained, multiple copies of the same content for a given content key may be located.

Peer Key	IP Address and TCP Port Number
Peer Key 0	144.82.214.156:2001
Peer Key 7	194.158.37.122:4052
Peer Key 16	108.11.123.101:6021
Peer Key 22	214.97.80.21:3587
Peer Key 31	78.144.82.10:9021

Table 9.1: Partial list of TCP/IP connection identifiers for corresponding peer keys, for Figure 9.1.

For illustrative purposes, Table 9.1 provides a fictitious list of TCP/IP connection identifiers for corresponding peer keys, for the example illustrated in Figure 9.1. Table 9.2 provides an example of the Content table for the peer with Peer Key 16 in Figure 9.1. As may be observed, an entry exists in the table for Content Keys 10 and 14, these being the content keys for which the peer with Peer Key 16 is responsible. For each entry, a list of TCP/IP connection identifiers is present, indicating the locations at which copies of the corresponding content are located.

Content Key	Addresses and Port Numbers
Content Key 10	194.158.37.122:4052; 108.11.123.101:6021; 214.97.80.21:3587.
Content Key 14	78.144.82.10:9021; 144.82.214.156:2001.

Table 9.2: Content table for the peer with Peer Key 16, for Figure 9.1.

The presence of multiple locations per content key entry is not fundamental to the operation of modified Chord. Modified Chord can work with a minimum of one IP address and TCP port number per content table entry. As a result, one can fine tune the maximum amount of locations maintained per entry.

Maintaining multiple locations per table entry has the following advantages:

- Multiple copies of the same content may be located. This increases content availability in modified Chord.
- Much like content dissemination in the FastTrack P2P network, algorithms may be used that employ segmented downloading, thereby improving the download speed.

These advantages come at the expense of increased informational complexity, since multiple TCP/IP connection identifiers have to be maintained per table entry, and an increase in the amount of control traffic required to maintain content table entries.

9.3.2. Content Requests

Since a content table is maintained at each content broker, content key requests may be performed in a similar fashion to unmodified Chord, as described in Sub-section 5.2.6. The Finger table is used to forward a key query in exactly the same way as in unmodified Chord. Having resolved the key into the IP address and TCP port number of the content broker, the requesting peer then contacts this content broker directly for the desired content.

Instead of returning the content itself, in modified Chord, the content broker sends a message back to the requesting peer, containing the list of TCP/IP connection identifiers from which the corresponding content may be sourced. This information may be obtained by the content broker by looking up the Content table and finding the entry for the content key, provided that an entry is present[34].

Having received the list of TCP/IP connection identifiers, the requesting peer may then obtain the required content from any one of the peers identified by these addresses and port numbers. Additionally, segmented downloading may be implemented at this stage. As may be noted from this operation, in modified Chord, only two additional application layer hops are required, when compared to unmodified Chord. The first additional hop is incurred when the content broker returns the list of TCP/IP connection identifiers, instead of returning the content itself. The second hop is incurred when the requesting peer requests the desired content from the peer/s identified by the list.

[34] No entry is present if the required content key is not currently located at any peer on the Chord ring.

9.3.3. Content Table Redistribution

In unmodified Chord, redistribution of content keys occurs each time a peer joins or leaves the Chord ring. When a peer joins the Chord ring, it is assigned content keys from its successor, such that the content keys assigned are greater than the peer key of the new peer's predecessor, but smaller or equal to the new peer's peer key. In contrast, when a peer leaves the Chord ring, the content keys are assigned to the peer's successor. In both cases, the redistribution of content keys implies redistribution of the corresponding content.

In modified Chord, redistribution occurs in exactly the same manner, however, what is actually redistributed are the content table entries. Content need not be redistributed. To further simplify this process, a content table should be sorted in numerical order of content keys. In summery, whilst in unmodified Chord O(1 / *N*) fraction of content keys and corresponding content need to be redistributed, in modified Chord O(1 / *N*) fraction of content table entries need to be redistributed.

9.3.4. Content Table Creation and Maintenance

The creation and maintenance of the content table at each content broker is fundamental to the successful operation of modified Chord. To this end, the following sequence of operations is performed when a peer joins the Chord ring:

- In a similar manner to unmodified Chord, a peer that has just joined the Chord ring will first establish its successor and predecessor.
- It will also execute Fix_fingers, so as to build its Finger table.
- The peer will then obtain its content table through its successor.
- For each item of content stored at the peer, the peer will resolve the corresponding content key, using the same mechanism used in unmodified Chord to resolve keys.
- In resolving the content keys, if the newly connected peer determines that it is responsible for the corresponding content itself, it updates its content table as necessary.
- Subsequently, for every content key resolved, a **Content_Location_Advert** message is sent directly to the content broker responsible for it, using the IP address

and TCP port number obtained by means of the content keys resolved[35]. A Content_Location_Advert message contains the following fields:

- The 160-bit (20 bytes) content key that the message represents.
- The 32-bit (4 bytes) IP address of the peer at which the corresponding content resides (i.e. the peer which has just joined).
- The 16-bit (2 bytes) TCP port number being used for the modified Chord application, at this peer.
- A message header that indicates the message type. In this case, this would indicate that the message is a Content_Location_Advert message.

As may be noted from this sequence of operations, a newly connected peer informs the relevant content brokers in the Chord ring about the content which it stores. It is important to note that sequence of operations is not performed periodically, but only upon joining the Chord ring. A similar operation is also performed when the peer leaves the Chord ring. Each content broker would in turn modify its content table according to the following rules:

- Upon receiving a Content_Location_Advert message, the content broker extracts the content key out of the message.
- The content broker then searches for the entry for this content key in the Content table. Search is simplified since the table is sorted in numerical order of content keys. If no entry is found for the content key, a new entry is added.
- Subsequently, the content broker adds the IP address and TCP port number, received in the Content_Location_Advert message, to the entry for the content key in the content table. The list of locations for a given entry in the content table should also be ordered according to when the location was added to the list. The most recently added location should be at the top of the list.

This mechanism provides a way how content key entries may be added or updated in a content table, as new peers join the Chord ring. Equally important is the way in which entries are removed or updated when peers leave the Chord ring. To this end, a peer uses a

[35] In modified Chord, the use of a reliable transport layer protocol, such as TCP, for message delivery is assumed.

similar procedure when leaving the Chord ring to that used when joining. The sequence of operations is as follows:

- For each item of content stored at the peer for which it is not the content broker, the peer will resolve the corresponding content key. This is necessary because the content broker responsible for a given content key may have changed during the time elapsed since the peer joined the Chord ring.
- For every content key resolved, a **Content_Location_Remove** message is sent directly to the content broker responsible for it. The structure of this message is identical to that of a Content_Location_Advert message. The difference lies in the message type indicated in the message header.

In turn, upon receiving a Content_Location_Remove message, a content broker would look up the extracted content key in its content table and remove the IP address and TCP port number, supplied in the message, from the list held for the content key entry in the content table. If the location was the only one in the list, the entire entry is removed from the content table. A mechanism for removing or updating entries in a content table is therefore also provided in modified Chord.

9.3.5. Recovery from Peer Failure

Content table creation and maintenance as described in the previous sub-section works under the assumption that peer failure does not occur. In practice, peer failures may occur. Such failures will affect modified Chord in two ways:

1) The content table of the failed peer is lost, since the failed peer would not have carried out redistribution of this content table. This problem is shared with unmodified Chord, since a failed node in unmodified Chord would not have redistributed content keys and corresponding content.
2) Invalid locations within the location lists of content table entries are inadvertently created, since a given location list may indicate the failed peer as a possible location from where the corresponding content may be obtained. This comes about as a result of the fact that a failed peer would not have sent the required Content_Location_Remove messages.

Depending on the application at hand, the first problem may be actively addressed or not. If the availability of the entire range of content keys is not critical, then the problem need not be addressed. One should note that, in this case, the problem will be gradually solved due to the transient connectivity of peers, since peers joining the Chord ring inform the relevant content brokers about their content, as described in the previous sub-section. The lost content table is therefore rebuilt gradually.

On the other hand, if the availability of the entire range of content keys is critical, a method is proposed in [35] for unmodified Chord, whereby content is replicated at a number of successor peers. Thus if the peer responsible for a given range of content keys fails, its immediate successor may take over the range content keys, since it should have a replica of the corresponding content of the failed peer. The number of replications depends on how critical content availability is and on the probability of simultaneous peer failures. Intuitively, this method may also be applied to modified Chord. Instead of replicating content at a number of successor peers, the content table itself is replicated.

To circumvent the second problem, we propose a solution whereby each content broker periodically verifies the content table entries, by implementing the following procedure:

- Starting with the first content table entry, it goes through the entire table in a sequential manner.
- For each entry, it retrieves the location list for the given content key.
- For a predefined amount of locations, a **Content_Availability_Verification** message is sent to these locations. This message simply contains the content key being verified as present at the queried location.

In turn, any peer receiving such a message would verify whether it in fact holds a copy of the content corresponding to that content key. Provided this is the case, the peer would send back a **Verification_ACK** to the content broker within the same TCP session, thereby informing the content broker that the content is still available.

Alternatively, the reception of a **Verification_NAK** implies that the content is no longer available. One possible reason for the reception of such a message could be that the copy of

the content was deleted by the user of the application. The lack of reception of a message by the content broker is also taken to imply that the content is no longer available, possibly because the peer experienced failure. In both cases, the content broker would remove the given location from the location list of the content table entry being verified. It is important to note the following three properties about this proposed procedure:

1) Since the content broker has access to the IP addresses and TCP port numbers (i.e. locations) listed for each content table entry, only one application layer hop is required for the Content_Availability_Verification message to reach the intended peer. The control traffic generated is thus kept to a minimum.
2) The amount of locations to be verified per content table entry may be adjusted according to requirements. By default, we assume that modified Chord will verify all locations listed, however, this may not be necessary. For example, a policy may be defined in which only the first location listed is in fact verified. This would further reduce the amount of control traffic, at the expense of possibly storing other invalid locations within a given location list. However, if such a policy is combined with a limit on the amount of locations stored within a location list, invalid locations should eventually be removed as new locations are added.
3) Even if all locations in a given location list are periodically verified, the existence of invalid locations is still possible. However, these invalid locations are transient in nature. Their maximum duration is of one *verification period*; from when a peer fails until when the relevant content broker performs verification of content table entries. Depending on the application at hand, the verification period may be set as short as necessary, at the expense of increased control traffic. Additionally, since multiple locations may be maintained, the existence of transient invalid locations is not considered problematic.

9.4. Model Equations

In order to analyse the relative performance of modified Chord, we have derived a number of equations so as to compare it with that of unmodified Chord. The equations model the amount of traffic generated in unmodified Chord due to the redistribution of content and the amount of control traffic generated in modified Chord due to the maintenance and redistribution of content table entries.

In order to derive these equations, a scenario is assumed in which we analyse the cumulative traffic incurred due to a peer joining the Chord ring, spending a defined amount of time in the Chord ring and, subsequently leaving the Chord ring; everything else remaining static. In both cases, the equations model the amount of traffic generated at the application layer.

A number of parameters are used throughout the equations derived. A brief explanation of each parameter therefore follows:

- N: The number of peers that form part of the Chord ring. In calculations, it is assumed that, for large N, $N + 1 \approx N$, as is $\log (N + 1) \approx \log (N)$.
- K: The total amount of content keys being used in the Chord ring.
- C: The total amount of items of content (i.e. files) being stored by the peers in the Chord ring.
- S: The average size, in megabits, of the individual items of content being stored by the peers in the Chord ring.
- U: Is a measure of the average uniqueness of content. The value of U can vary between 0 and 1. For example, a U value of 0.1 implies that, on average, there are 10 copies of an item of content stored across the peers that make up the Chord ring. Note that multiplying C by U provides K.
- P: The verification period, in seconds, used in modified Chord.
- L: The average lifetime, in seconds, of a peer in the Chord ring.

9.4.1. Equation for Unmodified Chord

When a peer joins a Chord ring in unmodified Chord, O(1 / *N*) fraction content keys and corresponding content need to be redistributed from its successor to it. Assuming that, due to the randomness with which content keys are generated, these are equally distributed amongst peers in the Chord ring;

The average amount of content keys redistributed should be: $\frac{K}{N}$.

Substituting for K yields: $\frac{C \times U}{N}$.

Multiplying by the average file size S yields: $\frac{C \times U \times S}{N}$.

Noting that this amount of data needs to be redistributed twice in the lifetime of a peer, we can therefore obtain an expression, in Mbps, which is representative of the average bandwidth that is required by a peer in its lifetime to perform redistribution of content:

$$\frac{2 \times C \times U \times S}{N \times L} \qquad Mbps$$

Equation 9.1: Average bandwidth used by a peer in its lifetime in unmodified Chord.

9.4.2. Equations for Modified Chord

By deduction, content table redistribution in modified Chord is similar to content redistribution in unmodified Chord. Thus, Equation 9.1 can be adapted accordingly. In this case, however, *S* has to be replaced for the average size of a content table entry.

On average, a content table entry is: $160 + \left((32 + 16)\right)/U\right) \qquad bits$.

Where: 160 is the size of an SHA-1 content key in bits.
32 is the size of an IP address in bits.
16 is the size of a TCP port number in bits.

U is the measure of uniqueness. In this case, dividing by U yields the average number of locations entered in the location list.

Equation 9.1 can therefore be rearranged as:

$$\frac{2 \times C \times U \times \left(\left(1.6 \times 10^{-4}\right) + \left(\left(4.8 \times 10^{-5}\right) / U \right) \right)}{N \times L} \quad Mbps$$

Equation 9.2: Average bandwidth used by a peer for table redistribution.

For modified Chord, the control traffic generated as a result of content table maintenance needs to be taken into account. For simplicity, the size of messages is conservatively assumed to be 100 bytes (800 bits). As illustrated in the previous sub-section, most messages are significantly less than this in size. The largest message is in fact likely to be the response to a key query, since this needs to accommodate two keys (40 bytes), an IP address (4 bytes), a port number (2 bytes) and a header to indicate the message type. We therefore presume that allocating 100 bytes per message for these calculations should yield conservative results.

The control traffic generated due to content table maintenance is of two types:

1) The control traffic generated when a peer joins/leaves the Chord ring.
2) The content traffic resulting from verification of content table entries. For these calculations, it is assumed that all locations for a given content table entry are being verified. A peer forming part of the Chord ring generates such traffic since it adds content to the Chord ring when it joins. As a result, all relevant content brokers would have to periodically verify the availability of this content. One should note that the verification of the peer's own content table entries is not considered. When a peer joins the Chord ring, it causes redistribution in responsibility for content verification from its successor to itself, as a result of redistribution of content table entries. Redistribution of responsibility does not result in additional traffic in itself.

On average, a peer stores C / N items of content. When such a peer joins the Chord ring, it will therefore have to perform C / N key queries. As noted in Sub-section 5.2.6, each query traverses O(log N) application layer hops before being resolved. Simulation results obtained in [35] show that the average number of hops is in fact $(1 / 2) \times \log_2 (N)$. Once a given key is resolved, the peer that resolved the key has to transmit this information back to the peer that issued the key query. In this case, upon receiving this information, the newly connected peer would send a Content_Location_Advert message to the relevant content broker. Two additional hops are therefore incurred.

The peer therefore transmits the equivalent of $\frac{C \times (\log_2 (N) + 4)}{2 \times N}$ 1-hop messages.

As may be deduced from the description given in Sub-section 9.3.4, the sequence of operations to be performed when the peer leaves the Chord ring generates the same amount of messages.

Since each message is assumed to be 800 bits in size, a peer would require the following average bandwidth due to the join/leave procedures in its lifetime:

$$\frac{2 \times 8.0 \times 10^{-4} \times C \times (\log_2 (N) + 4)}{2 \times N \times L} = \frac{8.0 \times 10^{-4} \times C \times (\log_2 (N) + 4)}{N \times L} \quad Mbps$$

Equation 9.3: Average bandwidth used by a peer due to the join/leave procedures.

As a result of verification of content table entries, a peer should receive C / N Content_Availability_Verification messages within a verification period P. Within the same period, it should respond with C / N Verification_ACK and/or Verification_NAK messages. Within a period P, a total of $2 \times C / N$ messages should therefore be exchanged between the given peer and the content brokers responsible for the content it stores. By dividing L by P, the amount of times that this exchange of messages occurs, is obtained.

A total of $\frac{L \times 2 \times C}{P \times N}$ messages are therefore exchanged within the peer's lifetime.

In a similar manner to Equation 9.3, multiplying by the message size of 800 bits and dividing by the average lifetime L yields the average bandwidth required for content verification:

$$\frac{L \times 2 \times C \times 8.0 \times 10^{-4}}{P \times L \times N} = \frac{2 \times C \times 8.0 \times 10^{-4}}{P \times N} \quad Mbps$$

Equation 9.4: Average bandwidth required for content verification in a peer's lifetime.

Combining Equation 9.2, Equation 9.3, and Equation 9.4 yields the total average bandwidth used by a peer throughout its lifetime:

$$\left(\frac{C}{N \times L}\right) \times \left(\left(2 \times U \times \left(\left(1.6 \times 10^{-4}\right) + \left(\frac{4.8 \times 10^{-5}}{U}\right)\right)\right) + \left(8.0 \times 10^{-4} \times \left(\log_2(N) + 4 + \left(\frac{2 \times L}{P}\right)\right)\right)\right) \quad Mbps$$

Equation 9.5: Average bandwidth used by a peer in its lifetime in modified Chord.

9.5. Results

Equation 9.1 and Equation 9.5, derived in the previous section, were implemented in a Java program. The source code for this program is available in Appendix A. In order to compare performance, data was collected from a Kazaa client logged onto the FastTrack P2P network. This data is provided in Table 9.3.

Subsequently, the Java program was used to generate data sets for unmodified Chord and modified Chord. In generating each data set, one parameter was varied whilst keeping all other parameters constant. This allowed us to analyse the effect that each parameter has on the average bandwidth used by a peer throughout its lifetime. The following values were chosen as the default values[36] for the parameters in generating the data sets:

- *N*: A value of 3819654 peers was obtained from Table 9.3.
- *C*: A value of 778793167 items of content was obtained from Table 9.3.

[36] A parameter's default value was used when the parameter's effect was not being analysed.

- *S*: A value of 63.838236 megabits was obtained from Table 9.3.
- *U*: Unfortunately, no measured values were available for average uniqueness. A realistic value of 0.1 was therefore chosen.
- *P*: A value of 60 seconds was chosen for the verification period in modified Chord.
- *L*: A value of 1800 seconds (30 minutes) was chosen as the average lifetime of a peer. Measurement results presented in [64] show that 60% of peers stay connected for 10 minutes or less on the FastTrack P2P network. Since a lower value for lifetime favours modified Chord, a value of 30 minutes should yield conservative results.

	Peers [N]	Items of Content [C]	Total Content Size [C × S]	Date	Time
	3873270	814609717	51495627087	04/07/2003	18.13
	3816792	769423057	48905177692	05/07/2003	19.16
	3849000	786124004	50123436575	06/07/2003	18.37
	3866206	769370518	49863951831	07/07/2003	18.12
	3763020	771679051	49349380389	08/07/2003	18.47
	3809160	772302937	49485496493	09/07/2003	18.31
	3760130	768042883	48794401895	10/07/2003	19.12
Average	3819654	778793167	49716781709		
Size [S]			63.838236		

Table 9.3: Data collected from the FastTrack P2P network.

9.5.1. Variation of Uniqueness

Since no measured values for average uniqueness were available, the effect of this parameter was analysed first. *U* was varied from 0.01 to 1 in 0.01 increments. A graph of the resulting data set is shown in Figure 9.2.

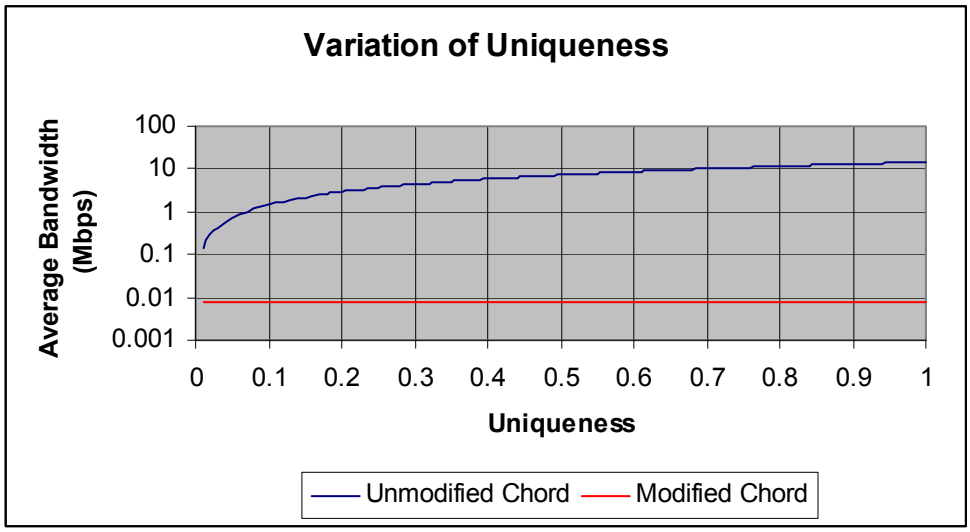

Figure 9.2: Graph of variation of uniqueness.

As may be observed, unmodified Chord shows a linear[37] variation of average bandwidth with respect to *U*. This follows directly from Equation 9.1. Secondly, the graph shows that a variation in *U* has minimal effect on modified Chord. The difference between the two extremes of *U* (1 and 0.01) is in fact of 36 bps. This is due to the fact that a variation in *U* only affects content table redistribution in unmodified Chord, as may be noted from Equation 9.5. Moreover, the graph shows that modified Chord significantly outperforms unmodified Chord throughout the range of *U* values tested. Thus, for example, at the chosen value of 0.1 for *U*, unmodified Chord requires 1.446 Mbps of average bandwidth, whilst unmodified Chord requires 7.795 Kbps.

[37] Note that a logarithmic scale is used for average bandwidth in Figure 9.2.

9.5.2. Variation of Lifetime

The second parameter analysed was average lifetime. L was varied from 300 seconds to 86400 seconds (24 hours), in 300 seconds increments. Figure 9.3 shows that unmodified Chord displays a linear relationship[38] with lifetime, as a consequence of Equation 9.1.

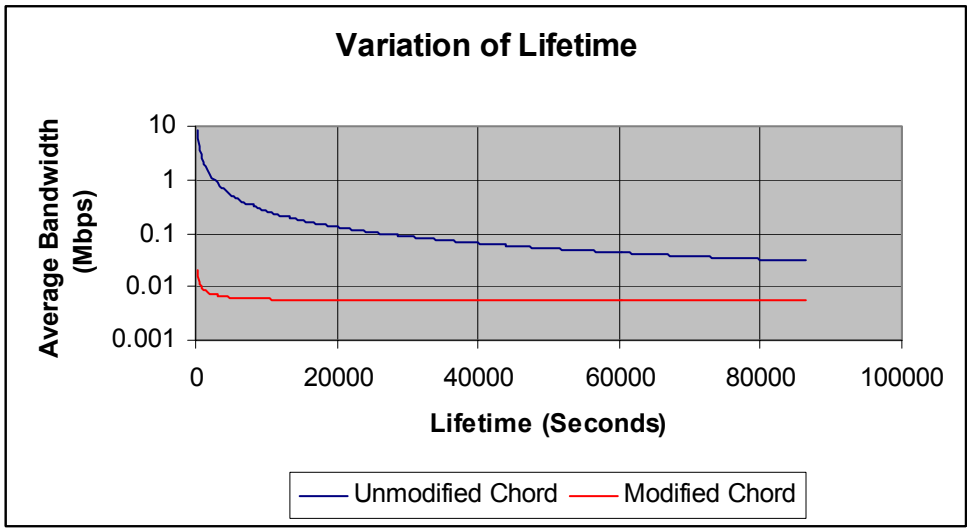

Figure 9.3: Graph of variation of lifetime.

Modified Chord also displays a linear relationship. This is, however, offset by the control traffic component due to content verification, which is independent of lifetime (in terms of average bandwidth), as may be noted from Equation 9.4.

More importantly, the graph shows that at the maximum value of L tested, unmodified Chord requires 30.13 Kbps, whilst modified Chord requires 5.486 Kbps; one order of magnitude in difference. At lower values of L, modified Chord further outperforms unmodified Chord. Modified Chord therefore performs better under transient connectivity.

[38] A logarithmic scale is also used for average bandwidth in Figure 9.3.

9.5.3. Variation of Content Size

The premise under which the modified Chord was designed is that redistribution of content keys and corresponding content in unmodified Chord is likely to result in a significant traffic burden. This traffic burden is directly related to the average size of an item of content. With this in mind, the average content size was varied with the intent of finding the average bandwidth intersection point between unmodified Chord and modified Chord.

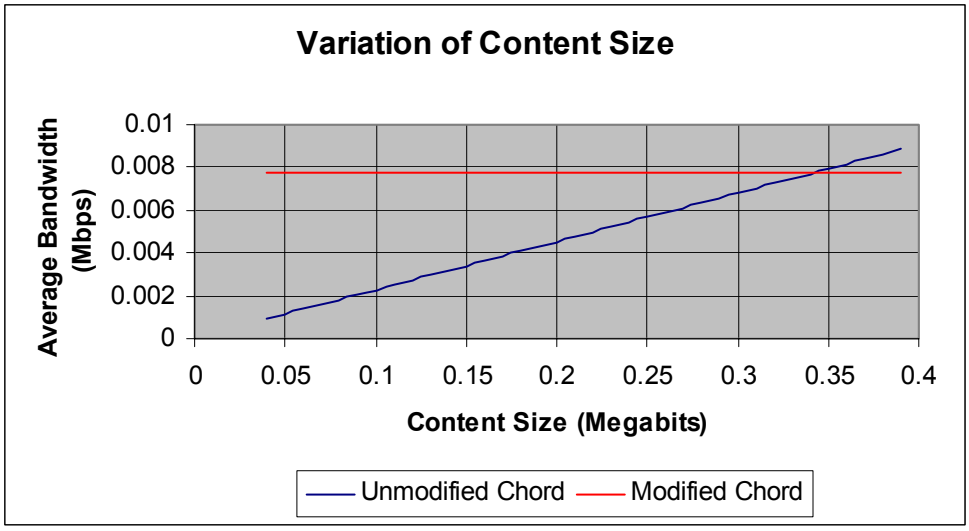

Figure 9.4: Graph of variation of content size.

Figure 9.4 provides the results obtained. The intersection point in this case[39] occurs when the average content size is circa 0.34 megabits (41.5 kilobytes). At higher values of *S*, modified Chord's performance is superior. At lower values, that of unmodified Chord is superior. This result is important since it can be used to discern which type of applications modified Chord would be more suited for and vice versa, according to an estimated average content size.

Equally important, the result shows that, unlike unmodified Chord, the scalability of modified Chord is independent of average content size. Equation 9.5 is in fact independent of *S*. On the other hand, the scalability of unmodified Chord is linear with respect to *S*, as may be noted from Equation 9.1 and Figure 9.4.

[39] That is, with all other parameters set to the default values.

9.5.4. Variation of Verification Period

The default verification period (60 seconds) chosen should suffice for the majority of applications. Having said this, applications in which content availability is time critical, may require a shorter verification period. The verification period was therefore varied from 5 second to 60 seconds, in 5 seconds increments, and the results shown in Figure 9.5 were obtained.

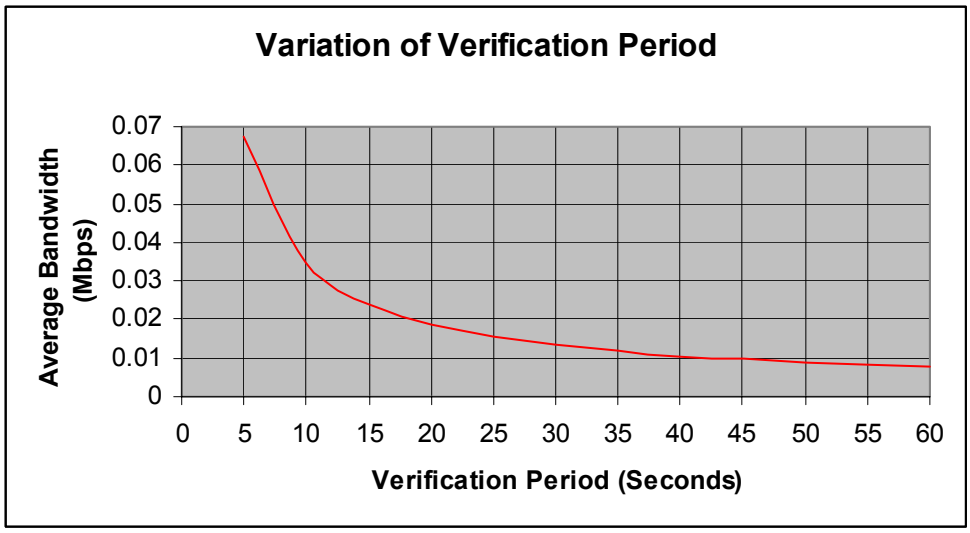

Figure 9.5: Graph of variation of verification period.

As expected from Equation 9.4, the average bandwidth used increases in proportion to 1 / *P*, as *P* gets shorter. Having said this, at a value of *P* equal to 10 seconds, the average bandwidth used is 34.981 Kbps. For the same default values, a peer in unmodified Chord requires 1.446 Mbps as observed in Sub-section 9.5.1.

Nonetheless, since the verification period is of no relevance in unmodified Chord, the results suggest that it may be better suited to applications in which content availability is time critical, so as to avoid transient invalid content table entries.

9.5.5. Variation of Number of Peers Keeping C / N Constant

The fifth parameter analysed was the number of peers that form the Chord ring. In contrast to the other parameters analysed, this parameter was varied in such a way as to keep the ratio of C / N at a constant equal to 204[40]. To do this, the total amount of content C, was varied in the same proportion as N.

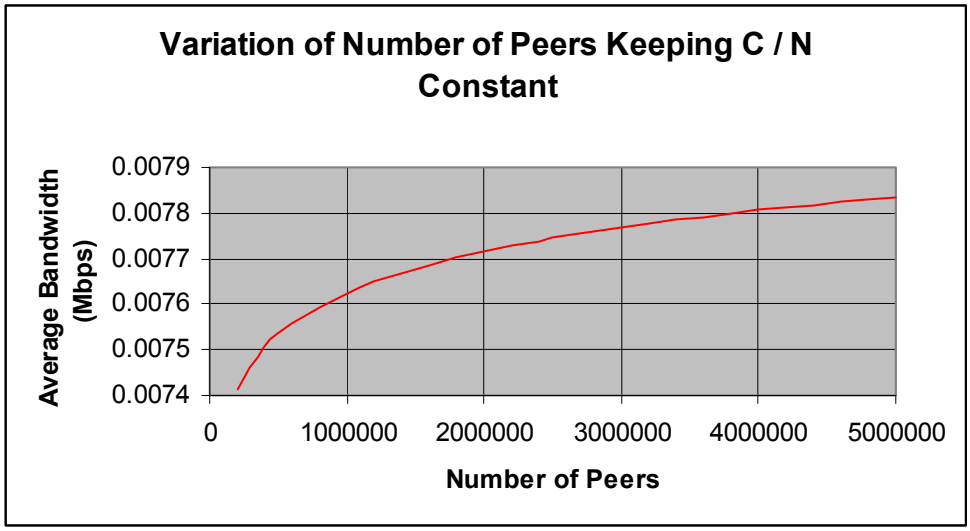

Figure 9.6: Graph of variation of number of peers keeping C / N constant.

As shown in Figure 9.6, the average bandwidth required by a peer in modified Chord increases logarithmically as the number of peers increases. This logarithmic increase is attributed to the $\log_2(N)$ component in Equation 9.3. In keeping C / N constant, the average bandwidth required in unmodified Chord remains at a constant of 1.447 Mbps. Analysis of Equation 9.1 confirms this result.

Even though the performance of unmodified Chord is independent of C / N, we note that the performance of modified Chord is preferable even at high values of N. At a value of N equal to 5 million peers, a peer in modified Chord requires 7.835 Kbps; three orders of magnitude lower than the value for unmodified Chord. In conclusion, the logarithmic scalability of modified Chord therefore ensures that, in absolute terms, it still outperforms unmodified Chord at realistically large values of N.

[40] This is the value of the C / N ratio when both C and N are at their default value.

9.5.6. Variation of the Amount of Content

The last parameter analysed was the total amount of items of content being stored by the peers in the Chord ring. As may be observed from Figure 9.7, unmodified Chord and modified Chord both exhibit linear scalability[41] with *C*. Analysis of Equation 9.1 and of Equation 9.5, respectively confirm these results. The scalability of unmodified Chord and of modified Chord are therefore similar in view of variations in the total amount of content.

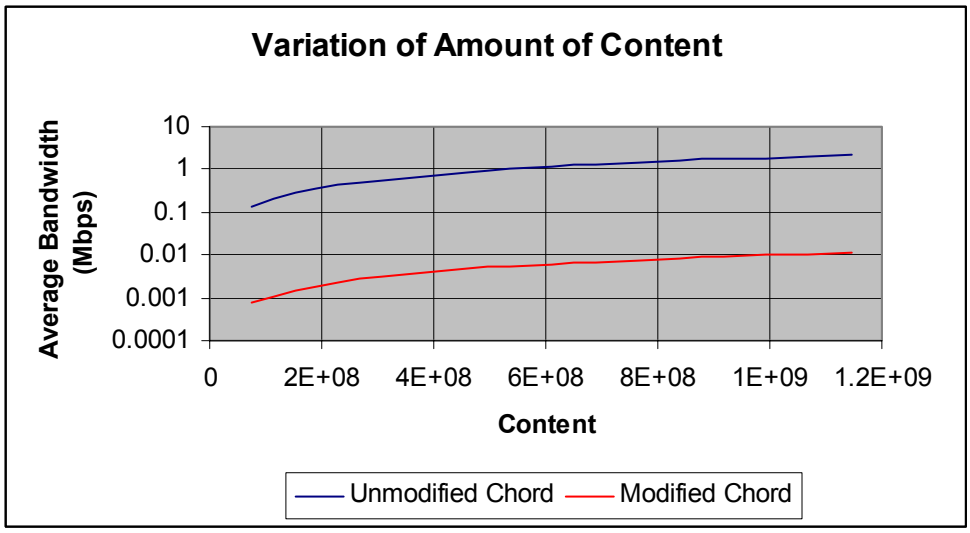

Figure 9.7: Graph of variation of amount of content.

9.6. Conclusions

In view of the work presented on modified Chord in this chapter, the following advantages may be identified when compared to unmodified Chord:

- Modified Chord primarily provides a deterministic content discovery technique, the scalability of which is independent of average content size.
- It allows for the discovery of multiple copies of the same item of content, thereby increasing content availability and allowing for segmented downloading.
- Its design is flexible in nature, since several parameters may be adjusted according to the application at hand. These parameters include the maximum number of

[41] A logarithmic scale is also used for average bandwidth in Figure 9.7.

locations held in a given location list, the number of locations per location list verified, and the verification period.

- In modified Chord, peers have control over what content is stored locally. This may be a significant advantage especially when considering the varied nature of content shared in P2P networks.
- Average content uniqueness has a minimal effect on the scalability of unmodified Chord, as shown in Sub-section 9.5.1.
- At the default values for the parameters used in the performance comparison, modified Chord performs better in view of the transient connectivity inherent to P2P networks, as shown in Sub-section 9.5.2.

Conversely, the following disadvantages may be identified:

- Two additional hops are incurred when content discovery is performed.
- In view of content dissemination, load balancing is not inherent to modified Chord, since content is not necessarily equally distributed amongst peers. To this end, a utility function such as that described in Section 8.4 could be used.
- The existence of invalid locations in content table entries is possible; however, these are transient if verification of content availability is carried out periodically as described in Sub-section 9.3.5.
- Modified Chord introduces an additional logarithmic increase in control traffic with an increase in the number of peers. However, in Sub-section 9.5.5 it was demonstrated that modified Chord's performance is still superior to that of unmodified Chord for a relatively large number of peers.

With these advantages and disadvantages in mind, it is clear that modified Chord is not superior to unmodified Chord in all respects and vice versa. In practice, the application at hand is likely to dictate which content discovery technique is more adapted. For example, modified Chord is more adapted towards content sharing P2P applications, since it scales independently of average content size. On the other hand, unmodified Chord is more adapted to an application such as distributed DNS [35], since the content in this case would consist of a 32-bit IP address, thus outperforming modified Chord at this content size.

9.7. Recommendations for Future Work

A number of areas in which further work on modified Chord could be carried out have been identified. These areas are noted here so that further research may be carried out on modified Chord:

- Simulation: The results presented in the previous section are theoretical in nature, since they are based on a direct implementation of the model equations. A simulation of modified Chord should therefore be performed in order to get a better understanding of its performance.
- Protocol Specification: The design of modified Chord presented within this chapter mainly deals with the specification of the algorithms to be used. Protocol specifications, such as message types, were included only where necessary. A full protocol specification including message types, message formats and header field values is therefore required for an implementation of modified Chord.
- As noted in Sub-section 9.5.1, a peer in modified Chord requires 7.795 Kbps of average bandwidth. In practice, the instantaneous bandwidth required may be significantly more, since events are not evenly spread over a peer's lifetime. Techniques therefore have to be devised to ensure that the control traffic does not exceed the average bandwidth by a significant amount. Such techniques would also ensure that a peer is not overloaded with the processing of control traffic.

10. Conclusions and Recommendations

In this study, the fields of Ad Hoc and P2P networking were compared, and the similarities and differences between the two fields were identified in each comparison made. The key conclusions from these comparisons are summarised in the next section, followed, in the subsequent section, by recommendations for future work.

10.1. Conclusions

From the overviews of Ad Hoc and P2P networking, three fundamental commonalities have been identified between Ad Hoc and P2P networks: Both network types are decentralised in nature; the connectivity of nodes/peers to the network is transient; and resources available at each peer/node may differ from those available at other peers/nodes, thereby leading to heterogeneity of resources.

On the other hand, the fields of Ad Hoc and P2P networking differ in their relevance to the OSI model since Ad Hoc networking generally deals with application-independent network issues, whilst P2P networking generally deals with application-oriented network issues. Secondly, Ad Hoc and P2P networks also differ in network size, since P2P networks are typically orders of magnitude larger than Ad Hoc networks.

With regards to node discovery in Ad Hoc networks and peer discovery in P2P networks, it was concluded that, whilst the need to perform node/peer discovery is common to both network types, there are two differences in the way this is performed, which imply that the current divergence in research on this aspect in the two fields is beneficial. The first difference is that a distinction between neighbouring and non-neighbouring nodes is present in Ad Hoc networking, whilst no equivalent distinction is present in P2P networking. The second difference is that, whilst broadcast packets are used in Ad Hoc networking for

neighbouring node discovery, the use of broadcast messages in P2P networking is limited to LAN applications. When the P2P network spans multiple subnets, a server or a stable peer is required to mediate initial connectivity to the network.

The comparison of the four IETF MANET routing protocols revealed that all four protocols have their own relative strengths and weaknesses. AODV can be regarded as a generic routing protocol for Ad Hoc networks since it has the advantage of low resource usage due to its reactive distance-vector nature. This advantage comes at the expense of QoS, due to the delay incurred for route setup and due to the lack of support for alternative routing metrics. DSR is the only protocol to provide support for unidirectional links and hence, it maximises resource utilisation. Its disadvantage is that control overhead can be significant, since each data packet needs to contain the entire route to the destination. OLSR and TBRPF are both proactive routing protocols and, therefore, share the advantage of having routes readily available when required, at the expense of increased control traffic due to routing table maintenance. To this end, TBRPF achieves proactive routing with a lower amount of control traffic, however, as a result it increases computational complexity.

In comparing routing in Ad Hoc networks to content discovery in P2P networks, we identified that both aspects fundamentally address the same issue. Routing in Ad Hoc networks deals with route discovery; content discovery in P2P networks deals with content discovery. Three other similarities were also identified, these being: The similarity between the use of broadcast packets in Ad Hoc routing and application layer flooding in P2P networks; the use of a TTL field to limit the propagation of messages; and the use of the reverse path for RREPs in AODV and for the return of keys in Freenet. Two differences were also identified, as a result of which, we concluded that the extent to which research in the two respective fields can converge, is limited. The first difference is that proactive content discovery techniques cannot be used in P2P networks since these would not scale well. Secondly, there is no parallel in P2P content discovery, to the presence of unidirectional links in Ad Hoc routing.

When considering the applicability of content discovery techniques used in P2P networks, to Ad Hoc networks, it was noted that the flooding technique and the rumour mongering

technique can both be applied to content discovery in Ad Hoc networks. There is, however, little scope in applying the latter technique due to the low node count in Ad Hoc networks. It was also noted that the technique used in FastTrack cannot be applied since supernodes require a higher bit rate connection than other nodes, and need direct connectivity with all other supernodes. Content discovery as performed in Freenet was also considered as not applicable, since this technique makes use of the reverse path followed by the Key Request, to deliver the key itself. On the other hand, CAN and Chord were identified as suitable techniques for content discovery in Ad Hoc networks, due to their properties; even though these techniques may have to be adapted in order to minimise key redistribution.

In comparing the aspects of identity, security, and anonymity for both network types, we concluded that the current divergence in the two research fields is beneficial with regards to the aspect of identity since Ad Hoc and P2P networking assume relevance at different layers of the OSI model. Conversely, research in the two respective fields on the aspects of security and anonymity would benefit from convergence, since the techniques used so far to attain these two aspects are largely independent of the OSI layer at which they are implemented. Additionally, due to their decentralised nature, both network types pose similar issues to be solved with regards to these aspects.

In terms of applicability, it was noted that content identification as performed in P2P networks may be applied to Ad Hoc networks, thus benefiting the aspect of identity in Ad Hoc networks. With regards to security, P2P networking would benefit from the application of security techniques used in Ad Hoc networks, since research on this aspect of Ad Hoc networking is notably more advanced. Conversely, Ad Hoc networking would benefit from the application of techniques to preserve anonymity in P2P networks, since the techniques used in P2P networks are superior to what has been proposed for Ad Hoc networks.

In comparing multicasting in Ad Hoc networks to ALM in P2P networks, we identified four similarities which suggest that the convergence of research in the two respective fields would be beneficial. These similarities are: The use of tree-based techniques; and the issues posed by decentralisation for the creation and management of multicast groups, the creation of shared multicast trees, and the discovery of multicast groups. From the comparison

carried out, it was also concluded that ALM in P2P networks would gain from the application of multicast algorithms (other than tree-based algorithms) used in Ad Hoc networks, these being: Mesh-based algorithms, stateless algorithms, and hybrid algorithms.

In the comparison of the aspects of network management and QoS in Ad Hoc and P2P networks, it was concluded that there is limited scope for the convergence of the two research fields in terms of network management, and that the current divergence in the two research fields with regards to the aspect of QoS is beneficial. With regard to the aspect of network management, the conclusion was reached after noting that both network types are similar in that network management has to be performed in a decentralised environment. Ad Hoc network management protocols, however, have to be lightweight and interoperable, whilst equivalent techniques in P2P networks have no such constraints. Secondly, in Ad Hoc networking, a greater emphasis is placed on network monitoring, whilst in P2P networking, a greater emphasis is placed on network control. With regard to the aspect of QoS, the conclusion was reached in view of the dissimilarity between the two network types on how the aspect of QoS is dealt with. In Ad Hoc networking, QoS is primarily dealt with at the network layer, on a per packet basis. In P2P networking, where necessary, QoS is dealt with at the application layer, by taking a system's approach.

The network management issue of free riding in content sharing P2P networks was identified as an issue that may also arise in Ad Hoc networks and it was concluded that techniques, such as utility functions, to combat free riding in P2P networks, should be applied to Ad Hoc networks.

In our proposed modification to the Chord content discovery technique, it was noted that the technique has several advantages over the unmodified version, the primary one being that the modification provides a deterministic content discovery technique, the scalability of which is independent of network size. On the other hand, this advantage comes at the expense of losing the inherent load balancing property of the unmodified version. To this end, it was concluded that the decision as to whether the unmodified or modified version of Chord is the better choice depends on the application at hand.

10.2. Recommendations for Future Work

Based on the conclusions drawn from the comparisons performed for each aspect dealt with in this study, further work can be done on each of the aspects in which convergence between research on Ad Hoc and P2P networking is considered beneficial. These aspects are: Routing in Ad Hoc networks and content discovery in P2P networks; security; anonymity; multicasting and ALM; and network management. This purpose behind such a work would be to analyse and streamline all of the research carried out for the given aspect in the two respective fields, with the intent of providing a common basis over which research in the two fields could be converged.

In a similar manner, work could be done on each of the aspects in which the applicability to the other network type, of techniques used in one network type, was identified as being possible. These aspects include: content discovery in P2P networks to Ad Hoc networks; content identification in P2P networks to Ad Hoc networks; security in Ad Hoc networks to P2P networks; anonymity in P2P networks to Ad Hoc networks; multicasting in Ad Hoc networks to ALM in P2P networks; and free riding in P2P networks to Ad Hoc networks. This work would include an analysis of the extent to which the techniques may be applied, as well as the design of any necessary modifications.

Two sub-aspects have not been compared in this study, these being DoS attacks in the aspect of security, and service management in the aspect of network management. DoS attacks have not been dealt with as they are generally system specific. A classification of all possible DoS attacks in Ad Hoc and P2P networks may, however, reveal some similarity between the two fields. Service management has not been dealt with since no basis for comparison of service management in Ad Hoc networks to P2P networks, currently exists. Having said this, as service management in Ad Hoc networks evolves to encompass service discovery, then a basis for comparison with content discovery in P2P networks may exist.

Recommendations for future work on the modification proposed for Chord were also made in Section 9.7. The future work recommended included: A simulation of modified Chord; a detailed protocol specification; and the development of techniques to ensure that the average bandwidth is not exceeded.

Appendix A – Program for Modified Chord

This appendix contains the program listing for the Java program, referred to in Section 9.5, which was used to generate the required data sets for the results presented.

```
import java.io.*;
import java.text.*;

class Chord1 {
    public static void main(String[] arguments) throws IOException {
        double c = 778793167.0; // Items of content (i.e. files).
        double n = 3819654.0; // Number of peers.
        double s = 63.838236; // Average size (Megabits) of one item of content.
        double u = 0.1; // Average uniqueness of Content.
        double l = 1800.0; // Average lifetime of a peer.
        double p = 60.0; // Verification period for modified Chord.
        double unMod; // Average bandwidth (Mbps) required in unmodified Chord.
        double mod; // Average bandwidth (Mbps) required in modified Chord.

        // Opening results file.
        File results = new File("C:/chordres.dat");
        FileOutputStream resFile = new FileOutputStream(results);
        PrintStream pf = new PrintStream(resFile);
        DecimalFormat six = new DecimalFormat("0.000000");

        // Uniqueness loop.
        // for (u = 0.01; u < 1.01; u += 0.01) {
        // Lifetime loop.
        // for (l = 300.0; l < 86700.0; l += 300) {
        // Content size loop.
        // for (s = 0.01; s < 0.4; s += 0.01) {
        // Verification period loop.
        // for (p = 5.0; p < 65.0; p += 5.0) {
        // Number of peers keeping c/n constant loop.
        // for (n = 200000.0, c = 40800000; n < 5200000.0; n += 200000.0, c += 40800000) {
        // Items of content loop.
        for (c = 76393080.0; c < 1222289280.0; c += 76393080.0) {
            // Unmodified Chord equation.
            unMod = ((2 * c * u * s) / (n * l));
            // Modified Chord equation.
            mod = (2 * u * ((1.6 * Math.pow(10, -4)) +
            ((4.8 * Math.pow(10, -5)) / u)));
            mod += ((8.0 * Math.pow(10, -4)) * ((Math.log(n) / Math.log(2))
            + 4 + (2 * (l / p))));
            mod *= (c / (n * l));
            // Storing results in file.
            // The parameter printed depends on which parameter is varied.
            pf.println(six.format(c) + "\t" + six.format(unMod)
            + "\t" + six.format(mod));
        }

        // Closing results file.
        pf.close();
        resFile.close();
        System.out.println("Results Complete!");
    }
}
```

Bibliography

Deitel H. M. and Deitel P. J. *Java – HOW TO PROGRAM*, 4th Edition, New Jersey: Prentice Hall, 2002.

Keshav S. *An Engineering Approach to Computer Networking – ATM Networks, the Internet, and the Telephone Network*, New Jersey: Addison-Wesley, 1997.

Leuf B. *Peer to Peer Collaboration and Sharing over the Internet*, Indiana: Addison-Wesley, 2002.

Oram A. *PEER-TO-PEER – Harnessing the Power of Disruptive Technologies*, California: O'Reilly & Associates, 2001.

Perkins C. E. *Ad Hoc Networking*, New Jersey: Addison-Wesley, 2001.

Stevens W. R. *TCP/IP Illustrated, Volume 1 – The Protocols*, Massachusetts: Addison-Wesley, 1994.

References

[1] R. Ramanathan and J. Redi, "A Brief Overview of Ad Hoc Networks: Challenges and Directions," *IEEE Communications*, no. 50th Anniversary Commemorative Issue, pp. 20-22, May 2002.

[2] A. Oram, *PEER-TO-PEER – Harnessing the Power of Disruptive Technologies*, California: O'Reilly & Associates, 2001.

[3] C. Elliott and B. Heile, "Self-Organising, Self-Healing Wireless Networks," in *Fifth IEEE International Conference on Personal Wireless Communications*, 2000, pp. 355-362.

[4] R. Schollmeier, I. Gruber, and M. Finkenzeller, "Routing in Mobile Ad Hoc and Peer-to-Peer Networks. A Comparison," in *International Workshop on Peer-to-Peer Computing*, 2002, pp. 1-15.

[5] G. Kortuem, "When Peer-to-Peer comes Face-to-Face: Collaborative Peer-to-Peer Computing in Mobile Ad Hoc Networks," in *First IEEE International Conference on Peer-to-Peer Computing*, 2001, pp. 75-91.

[6] D. Doval and D. O'Mahoney, "*Nom*: Resource Location and Discovery for Ad Hoc Mobile Networks," in *Proceedings of the First Annual Mediterranean Ad Hoc Networking Workshop*, 2002, pp. 1-8.

[7] "Dictionary.com," [online] 2003, http://www.dictionary.com (Accessed: 25 August 2003).

[8] "ETSI HIPERLAN/1 Standard," (ETSI Telecom Standards), [online] 2003, http://portal.etsi.org/bran/kta/Hiperlan/hiperlan1.asp (Accessed: 16 August 2003).

[9] M. S. Gast, *802.11 Wireless Networks – The Definitive Guide*, California: O'Reilly & Associates, 2002.

[10] "Overview," (IEEE Standards Association), [online] 2003, http://standards.ieee.org/wireless/overview.html (Accessed: 16 August 2003).

[11] "ETSI HIPERLAN/2 Standard," (ETSI Telecom Standards), [online] 2003, http://portal.etsi.org/bran/kta/hiperlan/hiperlan2.asp (Accessed: 16 August 2003).

[12] R. Schollmeier, "A Definition of *Peer-to-Peer* Networking for the Classification of *Peer-to-Peer* Architectures and Applications," in *First IEEE International Conference on Peer-to-Peer Computing*, 2001, pp. 101-102.

[13] B. Leuf, *Peer to Peer Collaboration and Sharing over the Internet*, Indianapolis: Addison-Wesley, 2002.

[14] D. Barkai, "Technologies for Sharing and Collaborating on the Net," in *First IEEE International Conference on Peer-to-Peer Computing*, 2001, pp. 13-28.

[15] S. Keshav, *An Engineering Approach to Computer Networking - ATM Networks, the Internet, and the Telephone Network*, New Jersey: Addison-Wesley, 1997.

[16] "TCP/IP," (Funet Network), [online] 2003, http://www.funet.fi/index/FUNET/history/internet/en/tcpip.html (Accessed: 17 August 2003).

[17] E. Thelen, "THE SRI VAN AND COMPUTER INTERNETWORKING," (Ed Thelen's Nike Missile Website), [online] 2003, http://ed-thelen.org/comp-hist/CORE-3-1-SRI-TCP-IP.html (Accessed: 17 August 2003).

[18] D. Kristula, "The History of the Internet," (Dave's Site), [online] 2001, http://www.davesite.com/webstation/net-history.shtml (Accessed: 17 August 2003).

[19] H. Wang, "Overview of Bluetooth Technology," *Dept.of Electrical Engineering Penn State University Paper*, pp. 1-42, 2001.

[20] "Bluetooth," [online] 2003, https://www.bluetooth.org (Accessed: 22 August 2003).

[21] Anon., "IEEE 802.11 Technical Tutorial," *Alvarion White Paper*, pp. 1-17.

[22] C. E. Perkins, E. Belding-Royer, and S. Das, *RFC3561 - Ad hoc On-Demand Distance Vector (AODV) Routing*, Status: EXPERIMENTAL, IETF, July 2003.

[23] D. B. Johnson, D. A. Maltz, and Y. C. Hu, *Internet Draft - The Dynamic Source Routing Protocol for Mobile Ad Hoc Networks (DSR)*, draft-ietf-manet-dsr-09.txt, IETF, April 2003.

[24] T. Clausen and P. Jacquet, *Internet Draft - Optimized Link State Routing Protocol*, draft-ietf-manet-olsr-10.txt, IETF, June 2003.

[25] R. Ogier, F. Templin, and M. Lewis, *Internet Draft - Topology Dissemination Based on Reverse Path Forwarding (TBRPF)*, draft-ietf-manet-tbrpf-09.txt, IETF, June 2003.

[26] "IP Routing for Wireless/Mobile Hosts (mobileip)," (The Internet Engineering Task Force), [online] 2003, http://www.ietf.org/html.charters/mobileip-charter.html (Accessed: 22 August 2003).

[27] "Jabber Software Foundation," [online] 2003, http://www.jabber.org/ (Accessed: 22 August 2003).

[28] "Extensible Messaging and Presence Protocol (xmpp)," (The Internet Engineering Task Force), [online] 2003, http://www.ietf.org/html.charters/xmpp-charter.html (Accessed: 22 August 2003).

[29] "Project JXTA," [online] 2003, http://www.jxta.org/ (Accessed: 22 August 2003).

[30] M. Portmann et. al., "The Cost of Peer Discovery and Searching in the Gnutella Peer-to-peer File Sharing Protocol," in *Ninth IEEE International Conference on Networks*, 2001, pp. 263-268.

[31] M. Portmann and A. Seneviratne, "The Cost of Application-level Broadcast in a Fully Decentralized Peer-to-Peer Network," in *Seventh IEEE International Symposium on Computers and Communications*, 2002, pp. 941-946.

[32] M. S. Corson and J. Macker, *RFC2501 - Mobile Ad Hoc Networking (MANET): Routing Protocol Performance Issues and Evaluation Considerations*, Status: INFORMATIONAL, IETF, January 1999.

[33] E. M. Royer and C. K. Toh, "A review of Current Routing Protocols for Ad Hoc Mobile Wireless Networks," *IEEE Personal Communications*, vol. 6, no. 2, pp. 46-55, April 1999.

[34] P. Jacquet et. al., "Optimized Link State Routing Protocol for Ad Hoc Networks," in *IEEE International Multi Topic Conference*, 2001, pp. 62-68.

[35] I. Stoica et. al., "Chord: A Scalable Peer-to-Peer Lookup Protocol for Internet Applications," *IEEE/ACM Transactions on Networking*, vol. 11, no. 1, pp. 17-32, February 2003.

[36] D. A. Menascé, "Scalable P2P Search," *IEEE Internet Computing*, vol. 7, no. 2, pp. 83-87, March 2003.

[37] I. Clarke et. al., "A Distributed Anonymous Information Storage and Retrieval System," in *Proceedings of the ICSI Workshop on Design Issues in Anonymity and Unobservability*, 2000, pp. 1-21.

[38] S. Ratnasamy et. al., "A Scalable Content-Addressable Network," in *Proceedings of the ACM SIGCOMM*, 2001, pp. 161-172.

[39] Y. C. Tseng, C. C. Shen, and W. T. Chen, "Integrating Mobile IP with Ad Hoc Networks," *IEEE Computer*, vol. 36, no. 5, pp. 48-55, May 2003.

[40] L. Zhou and Z. J. Haas, "Securing Ad Hoc Networks," *IEEE Network*, vol. 13, no. 6, pp. 24-30, November 1999.

[41] F. Stajano and R. Anderson, "The Resurrecting Duckling: Security Issues for Ubiquitous Computing," *IEEE Computer*, vol. 35, no. 4, pp. 22-26, April 2002.

[42] M. Mactagarrt, *Introduction to Cryptography*, IBM developerWorks : Security Library, 2001.

[43] V. Buttigieg, *Coding for Communication Systems – Course Notes*, Malta: University of Malta, 1999.

[44] F. Anjum, *Wireless Security – Course Notes*, Pennsylvania: University of Pennsylvania, 2003.

[45] Zhou, L., "Towards Fault Tolerant and Secure On-line Services." Ph.D. Thesis, Cornell University, New York, USA, 2001.

[46] "MSN Messenger Protocol (Practical Implementation)," (Welcome to Venky's World!), [online] 2003, http://www.venkydude.com/articles/msn.htm (Accessed: 21 July 2003).

[47] W. Yeager and J. Williams, "Secure Peer-to-Peer Networking: The JXTA Example," *IEEE IT Professional*, vol. 4, no. 2, pp. 53-57, March 2002.

[48] M. Schmidt, "Subscriptionless Mobile Networking: Anonymity and Privacy Aspects within Personal Areas Networks," in *IEEE Wireless Communications and Networking Conference*, 2002, pp. 869-875.

[49] C. M. Cordeiro, H. Gossain, and D. P. Agrawal, "Multicast over Wireless Mobile Ad Hoc Networks: Present and Future Directions," *IEEE Network*, vol. 17, no. 1, pp. 52-59, January 2003.

[50] Anon., *JUNOS 5.7 Internet Software Configuration Guide: Multicast*, Juniper Networks, 2003.

[51] E. M. Royer and C. E. Perkins, "Multicast Operation of the Ad Hoc On-Demand Distance Vector Routing Protocol," in *Proceedings of IEEE MobiCom*, 1999, pp. 207-218.

[52] S. J. Lee, W. Su, and M. Gerla, *Internet Draft - On-Demand Multicast Routing Protocol (ODMRP) for Ad Hoc Networks*, draft-ietf-manet-odmrp-02.txt, IETF, July 2000.

[53] L. Ji and M. S. Corson, "Differential Destination Multicast - A MANET Multicast Routing Protocol for Small Groups," in *Proceedings of IEEE INFOCOM*, 2001, pp. 1192-1201.

[54] M. Liu et. al., "AMRoute: Adhoc Multicast Routing Protocol," *Technical Report TR 99-1*, pp. 1-16, 1999.

[55] A. Wierzbicki, R. Szezepaniak, and M. Buszka, "Application Layer Multicast For Efficient Peer-to-Peer Applications," in *Proceedings of the Third IEEE Workshop on Internet Applications*, 2003, pp. 1530-1534.

[56] J. Jannotti et. al., "Overcast: Reliable Multicasting with an Overlay Network," in *Proceedings of the Fourth Symposium on Operating System Design and Implementation*, 2000, pp. 1-16.

[57] P. Mohapatra, J. Li, and C. Gui, "QoS in Mobile Ad Hoc Networks," *IEEE Wireless Communications*, vol. 10, no. 3, pp. 44-52, June 2003.

[58] W. Chen, N. Jain, and S. Singh, "ANMP: Ad Hoc Network Management Protocol," *IEEE Journal on Selected Areas in Communications*, vol. 17, no. 8, pp. 1506-1531, August 1999.

[59] H. De Meer and K. Tutschku, "Dynamic Operation of Peer-to-Peer Overlay Networks," in *Fourth Annual International Working Conference on Active Networks*, 2002, pp. 1-3.

[60] L. Ramaswamy and L. Liu, "Free Riding: A New Challenge to Peer-to-Peer File Sharing Systems," in *Proceedings of the 36th Annual Hawaii International Conference on System Sciences*, 2003, pp. 220-229.

[61] "Participation Level," (KaZaA), [online] 2003, http://www.kazaa.com/us/help/glossary/participation_ratio.htm (Accessed: 5 August 2003).

[62] S. B. Lee et. al., "INSIGNIA: An IP-Based Quality of Service Framework for Mobile ad Hoc Networks," *Journal of Parallel and Distributed Computing*, vol. 60, no. 4, pp. 374-406, April 2000.

[63] X. Gu and K. Nahrstedt, "A Scalable QoS-Aware Service Aggregation Model for Peer-to-Peer Computing Grids," in *Eleventh IEEE International Symposium on High Performance Distributed Computing*, 2002, pp. 73-82.

[64] S. Subhabrata and J. Wang, "Analyzing peer-to-peer traffic across large networks," in *Proceedings of the ACM SIGCOMM*, 2002, pp. 1-14.