Department of Electronic & Electrical Engineering
University College London

A framework for the dynamic management of Peer-to-Peer overlays.

Theofrastos Koulouris

Submitted in partial fulfilment of the requirements for the degree of Doctor of Philosophy of University College London

STATEMENT OF ORIGINALITY

I, Theofrastos Koulouris, confirm that the work presented in this thesis is my own. Where information has been derived from other sources, I confirm that this has been indicated in the thesis.

Theofrastos Koulouris

ABSTRACT

Peer-to-Peer (P2P) applications have been associated with inefficient operation, interference with other network services and large operational costs for network providers. This thesis presents a framework which can help ISPs address these issues by means of intelligent management of peer behaviour. The proposed approach involves limited control of P2P overlays without interfering with the fundamental characteristics of peer autonomy and decentralised operation.

At the core of the management framework lays the Active Virtual Peer (AVP). Essentially intelligent peers operated by the network providers, the AVPs interact with the overlay from within, minimising redundant or inefficient traffic, enhancing overlay stability and facilitating the efficient and balanced use of available peer and network resources. They offer an "insider's" view of the overlay and permit the management of P2P functions in a compatible and non-intrusive manner. AVPs can support multiple P2P protocols and coordinate to perform functions collectively.

To account for the multi-faceted nature of P2P applications and allow the incorporation of modern techniques and protocols as they appear, the framework is based on a modular architecture. Core modules for overlay control and transit traffic minimisation are presented. Towards the latter, a number of suitable P2P content caching strategies are proposed.

Using a purpose-built P2P network simulator and small-scale experiments, it is demonstrated that the introduction of AVPs inside the network can significantly reduce inter-AS traffic, minimise costly multi-hop flows, increase overlay stability and load-balancing and offer improved peer transfer performance.

ACKNOWLEDGEMENTS

I would like to thank the following people:

My supervisor, Dr. Miguel Rio, for his direction, support, patience and kindness. Thank you for being a mentor and a friend.

My examiners, Prof. Jon Crowcroft and Dr. Ioannis Andreopoulos for their constructive criticism of this work and invaluable guidance in producing the best possible written thesis I could.

My colleagues and friends Drs. Eleni Mykoniati and David Griffin for their support and insightful comments during the last stages of my writing.

My family for being there and putting up with me in spite of the many times I tried their patience.

Finally, I would like to dedicate this thesis to the memory of Prof. Chris Todd who supervised me during the first half of this journey. Sorry for taking so long.

T. Koulouris

TABLE OF CONTENTS

STATEMENT OF ORIGINALITY **1**

ABSTRACT **2**

ACKNOWLEDGEMENTS **3**

TABLE OF CONTENTS **4**

INDEX OF FIGURES **7**

INDEX OF TABLES **9**

LIST OF ABBREVIATIONS **10**

1. INTRODUCTION **12**

1.1 Motivation and contributions 12

1.2 Structure of the thesis 14

1.3 A note on terminology 16

2. PEER-TO-PEER NETWORKS **17**

2.1 Introduction and historical context 17

2.2 What is Peer-to-Peer networking? 19

2.3 Why P2P? The user's perspective 22

2.4 P2P application domains 26

2.4.1 Content distribution 26

2.4.2 Communication and collaboration 27

2.4.3 Internet service support 27

2.4.4 Database systems 28

2.4.5 Distributed computation 28

2.5 Lessons from the deployment of contemporary P2P services 28

2.6 P2P from the network provider's point of view 32

2.7 Conclusions 35

3. PROTOCOLS, TOPOLOGIES AND PEER BEHAVIOUR **37**

3.1 Introduction 37

3.2 Topological categorisation of P2P networks 38

3.2.1 Centralised server P2P networks 39

3.2.2 Unstructured decentralised P2P networks 42

3.2.3 Structured decentralised P2P networks 47
3.2.3.1 Deterministic topologies 47
3.2.3.2 Content-addressable networks 49
3.2.4 Hierarchical P2P networks 55
3.3 Peer behaviour characteristics 59
3.3.1 Free-riding 60
3.3.2 Peer availability and churn 62
3.3.3 Content-related characteristics 65
3.4 Summary and conclusions 67

4. THE ACTIVE VIRTUAL PEER 69
4.1 Introduction 69
4.2 Objectives and requirements of P2P overlay management 70
4.3 The Active Virtual Peer concept 71
4.4 Implementation of the AVP with ALAN 76
4.5 Overview of AOC design 79
4.5.1 Router module 80
4.5.2 Connection Manager module 81
4.5.3 AOC administrative interface 81
4.6 AOC functions 81
4.6.1 Access control 81
4.6.2 Routing control and load balancing 84
4.6.3 Topology control 87
4.7 AVP policy model 88
4.8 The Network Optimisation Component 95
4.9 AOC prototype implementation 98
4.10 Summary 101

5. THE VIRTUAL CONTENT CACHE 102
5.1 Introduction 102
5.2 VCC design 104
5.2.1 Overview of the design 104
5.2.2 Cache strategies and replacement policies for the VCC 106
5.2.3 Chunk versus full file caching 112
5.2.4 Distributed VCC caching 114
5.3 Implementation of the VCC prototype 118
5.4 Legal considerations of caching P2P traffic 119
5.5 Summary 120

6. THE AVP SIMULATOR 121
6.1 Introduction 121
6.2 Why develop yet another simulator? 122
6.3 AVP Simulator design 123

6.3.1 General principles 123
6.3.2 Methodology 124
6.3.3 Random number generation 127
6.3.4 Topology generation and BRITE 128
6.3.5 Basic AVPsim data structures 128
6.3.6 Presentation of results 131
6.4 Simulation model 133
6.4.1 Connection model 133
6.4.2 Ordinary peer model 135
6.4.3 AVP model 136
6.5 Summary 137

7. EVALUATION OF THE AVP 138
7.1 Introduction 138
7.2 Simulation setup 138
7.3 Evaluation of AOC routing/topology control and application performance improvement 141
7.4 Evaluation of VCC caching 148
7.4.1 Ideal cache performance 148
7.4.2 Cache replacement strategy performance 151
7.4.3 Multiple-VCC deployments 163
7.4.3.1 Autonomous VCC operation 163
7.4.3.2 Cooperative VCC operation 167
7.5 Evaluation of the AVP as a system 170
7.5.1 Effect of AVP placement on network utilisation 170
7.5.2 Economics of AVP deployment 174
7.6 Proxylet testing and trials 181
7.6.1 Controlled Domain creation and signalling restriction 181
7.6.2 VCC operation 183
7.6.3 Routing control 184
7.6.4 Protocol message tunnelling 186
7.7 Summary 187

8. CONCLUSIONS AND FUTURE WORK 188
8.1 Summary and contributions 188
8.2 Related work 191
8.3 Future directions 197

REFERENCES 200

APPENDICES 214
Appendix A: Advanced Chord routing scheme 214
Appendix B: AVP Policy XML Schema 216
Appendix C: AVP Policy example 220
Appendix D: Basic Gnut commands 223

INDEX OF FIGURES

Figure 1: The Client/Server and P2P models. 20
Figure 2: P2P application layer overlay. 21
Figure 3: The centralised P2P topology 40
Figure 4: Random graph versus scale-free networks. 44
Figure 5: HyperCup topology. 48
Figure 6: HyperCup message forwarding. 49
Figure 7: A Chord ring. 51
Figure 8: Simple Chord routing. 51
Figure 9: A super-peer network 56
Figure 10: The AVP component architecture. 73
Figure 11: An example of an AVP deployment. 74
Figure 12: The funnelWeb EEP control interface. 77
Figure 13: AOC proxylet running on an EEP. 78
Figure 14: Relationship between network plane and application overlay. 79
Figure 15: Structure and information flow inside the router module. 80
Figure 16: Access Control on Gnutella by the AOC. 84
Figure 17: Operation of the “Probabilistic Routing” module. 86
Figure 18: Initial overlay condition. 88
Figure 19: Overlay after AOC control. 88
Figure 20: The IETF policy model. 89
Figure 21: AVP policy structure. 92
Figure 22: Condition element structure. 93
Figure 23: Structure of the “Actions” element. 95
Figure 24: AOC protocol message structure. 99
Figure 25: Using the AOC prototype. 100
Figure 26: P2P traffic flows without and with content caching employed. 103
Figure 27: Application-level caching by the VCC. 104
Figure 28: VCC component diagram. 118
Figure 29: Basic AVPsim data structures and their relationship to the network. 130
Figure 30: Looking-up overlay connection details. 130
Figure 31: The peer list and its association to other data structures. 131
Figure 32: Basic ISP router topology model. 141
Figure 33: CDF of number of IP hops between source and destination peers under normal operation and when an AVP is present. 144
Figure 34: CDF of connection round-trip delay between source and destination peers under normal operation and when an AVP is present. 146
Figure 35: CDF of transfer completion time under normal operation and when an AVP is present. 147

Figure 36: Transit traffic volume with and without a VCC present. 150
Figure 37: Object hit ratio performance per cache capacity. 153
Figure 38: Cache replacement operations per policy and cache capacity. 155
Figure 39: Byte hit ratio performance per cache capacity. 156
Figure 40: Object hit ratios per cache capacity (2nd workload). 160
Figure 41: Byte hit ratios per cache capacity (2nd workload). 161
Figure 42: Different degrees of P2P traffic localisation. 171
Figure 43: Effect of AVP placement on localisation of P2P transfers. 173
Figure 44: Effect of AVP placement on peer transfer performance. 174
Figure 45: Costs over a 24-hour period for α=0, β=10. 178
Figure 46: Costs over a 24-hour period for α=2.5, β=10. 178
Figure 47: VCC operation test set-up. 184
Figure 48: Routing control test set-up. 185
Figure 49: Topology control test set-up. 187
Figure 50: Simple Chord routing scheme. 214
Figure 51: Advanced Chord routing. 215

INDEX OF TABLES

Table 1: Comparison of three popular super-peer networks. 58
Table 2: Types of connections supported by the AOC prototype. 99
Table 3: Basic AVPsim events. 126
Table 4: Key simulation parameters of the featured runs 140
Table 5: Traffic and connection characteristics of no AVP/single AOC deployments. 143
Table 6: Degree of correlation of connection characteristics data between runs. 143
Table 7: Peer traffic and connection characteristics in “no AVP”, “single AOC” and “AOC+VCC” deployments. 150
Table 8: Replacement policy object hit ratios per cache capacity. 154
Table 9: Degree of correlation of OHR simulation results between runs. 154
Table 10: Replacement policy byte hit ratios per cache capacity. 157
Table 11: Degree of correlation of BHR simulation results between runs. 157
Table 12: Object hit ratios per cache capacity (2nd workload). 160
Table 13: Degree of correlation of 2nd workload OHR simulation results between runs. 161
Table 14: Byte hit ratios per cache capacity (2nd workload). 162
Table 15: Degree of correlation of 2nd workload BHR simulation results between runs. 162
Table 16: Variation between different VCC deployment configurations. 164
Table 17: Variation between different cooperating VCC configurations. 168
Table 18: Variation of overlay metrics at different levels of replication. 170
Table 19: Degree of correlation of completion time data between runs. 174
Table 20: Reduction in traffic to ‘Gnut2’ with different threshold values. 185

LIST OF ABBREVIATIONS

ADSL	Asymmetric Digital Subscriber Line
ALAN	Application Level Active Networking
AN	Active Network
AOC	Application Optimisation Component
API	Application Programming Interface
AS	Autonomous System
ATM	Asynchronous Transfer Mode
AVP	Active Virtual Peer
AVPsim	Active Virtual Peer Simulator
BGP	Border Gateway Protocol
BHR	Byte Hit Ratio
CD	Controlled Domain
CDF	Cumulative Distribution Function
CDN	Content Delivery Network
CPU	Central Processing Unit
CSV	Comma-Separated Values
DNS	Domain Name Service
EEP	Execution Environment for Proxylets
FTP	File Transfer Protocol
IETF	Internet Engineering Task Force
IM	Instant Messaging
IP	Internet Protocol
IPv4	Internet Protocol version 4
IPv6	Internet Protocol version 6

IT	Information Technology
MAN	Metropolitan Area Network
MPLS	Multi-Protocol Label Switching
NOC	Network Optimisation Component
OHR	Object Hit Ratio
OSPF	Open Shortest Path First
P2P	Peer-to-Peer
QoE	Quality of Experience
QoS	Quality of Service
PoP	Point of Presence
RFC	Request For Comment
RTP	Real Time Protocol
RTT	Round Trip Time
SLA	Service Level Agreement
SSH	Secure SHell
TCP	Transmission Control Protocol
TTL	Time To Live
UDP	User Datagram Protocol
UML	Unified Modelling Language
URL	Uniform Resource Locator
VCC	Virtual Content Cache
VoIP	Voice over Internet Protocol
WWW	World Wide Web
XML	eXtensible Markup Language

1. INTRODUCTION

1.1 Motivation and contributions

In recent years, Peer-to-Peer (P2P) applications have achieved widespread popularity and extensive deployment. The ability to access a vast pool of resources at no direct extra cost and with surprising ease of use, has made them so attractive that P2P-generated traffic has become a major component of overall Internet traffic. What distinguishes P2P from "traditional" client/server-based Internet services like the World Wide Web (WWW) are the symmetric roles of all participants: P2P services gain their value from the networked cooperation of equals. This is in contrast with the client/server architecture where asymmetric roles are typical.

Current P2P services can be identified by three main characteristics: sharing of pooled and exchangeable resources found at the edges of the Internet, all nodes having similar roles and all nodes being highly autonomous. The lack of separate roles between peers and non-reliance to supporting infrastructure has allowed P2P applications to boast admirable ease of deployment and fault-tolerance. As a consequence, however, the majority of P2P systems operate by creating overlays on top of the application layer, where peers form their application-level virtual topologies. These overlay topologies rarely match the underlying network infrastructure leading to inefficient operation and large operational costs for the network and Internet Service Providers (ISPs).

While considerable work by the research and developer communities is devoted to developing the next generation of P2P systems, most of it concentrates on improving the scalability, performance and functionality of specific systems or problem areas in a protocol-centric manner. Part of that can be attributed to the diversity of user requirements and application domains, meaning that what might work well for one service may be unsuitable for another. Moreover, the diversity of goals and scope of the various stakeholders (end-users, P2P application developers, ISPs) and the lack of established communication channels between the latter two in particular, result in leaving the issue of efficient use of network resources lower in the agenda. As a result,

ISPs pressed to minimise costs quickly, take less elegant approaches by implementing bandwidth caps, rate-limiting or blocking of P2P traffic. Such approaches are drastic and in general do not differentiate between the various P2P services regardless of how each one operates or affects the network. This creates an environment where the wider adoption and evolution of P2P services is stifled, widens the gap between users, application and network providers and unnecessarily harms innovation and advancement of the Internet ecosystem in general.

Therefore, the evolution and promotion of P2P networking does not only rely on developing better protocols and techniques but also on ensuring that the resulting services can co-exist and seamlessly integrate within existing service structures. Part of that approach requires thinking beyond the application layer and adopting a holistic view of the network which includes the ISPs' perspective. This is the main driving force behind the work presented in this thesis.

Specifically, what is proposed in this thesis is that ISPs should be able to influence specific aspects of P2P service behaviour in order to accomplish a more considerate and balanced use of network resources, while allowing such services to operate in an unimpeded and efficient manner. In that direction, this thesis contributes a framework of tools and mechanisms designed to reduce the impact P2P services have on the network infrastructure in a way that is both transparent and compatible to them. The framework leverages the unique position and detailed knowledge of their networks ISPs have to provide guidance to peers and help them refrain from making suboptimal decisions which have a substantial cost in network and peer resources. Crucially, peer operation is not fundamentally restricted. Instead, correct peer behaviour and cooperation is promoted and encouraged by offering performance gains.

In summary, the work presented in this thesis makes the following contributions:

- The areas where management of peer behaviour can be effective while being transparent to peers and compatible to the way P2P services operate are identified.
- Realistic scenarios of how these high-level management goals can be translated into implementable controls for existing protocols are formulated.
- A novel framework architecture that implements the proposed management functions is designed.
- A policy model for the automated configuration and management of the framework elements is defined.

- Caching strategies which are suitable and effective for P2P workloads and take advantage of the capabilities provided by the framework are developed and evaluated.
- A prototype of the framework is developed for proving the concept and evaluating framework capabilities through experiment[1].
- A novel simulation software for P2P networks is developed, to examine large-scale P2P service behaviour and evaluate the operation of the proposed framework in a variety of settings.

1.2 Structure of the thesis

The rest of the thesis is organised as follows:

Chapter 2 ("Peer-to-Peer Networks") provides a broad introduction to P2P networking and sets the scene for the work presented in this thesis. The historical context which led to the development of P2P networking in its current form is briefly presented first. P2P networking is then defined and the differences between P2P and client/server architectures are discussed. A review of application domains where P2P services have appeared follows. Next, general characteristics current P2P services have demonstrated are briefly discussed. Finally, the effect of P2P service deployment from an ISP's point of view is investigated.

Chapter 3 ("Protocols, Topologies and Peer Behaviour") builds upon the discussion of general P2P service characteristics presented in the previous chapter and looks closer into the behaviour of different classes of P2P networks and its effects on service operation, performance and impact on the Internet infrastructure. Large part of this analysis focuses on the role fundamental types of P2P overlay topologies play on that behaviour. For each class, the way representative protocols operate is reviewed, discussing their strengths and shortcomings. Complementing this macroscopic view is the investigation of the effects of individual peer behaviour. This examination gives the

[1] The foundations of the AVP concept along with an early prototype implementation of the AOC component were developed in collaboration with Hermann De Meer (University College London), Kurt Tutschku and Robert Henjes (University of Wurzburg, Germany).

necessary insight on the pervasive nature of P2P services, the way different functions are implemented and the technical challenges associated, providing the link to the following chapters where the proposed framework for managing P2P services is presented.

Chapter 4 ("The Active Virtual Peer") introduces the proposed management framework for P2P overlays. The chapter begins with a discussion of areas where P2P service control can be beneficially applied. Then, the AVP concept and framework design are presented. Core modules are described, along with scenarios of how the framework can be utilised to manage and improve P2P service operation. A detailed definition of the AVP policy model follows. The chapter is concluded with a brief discussion of the implementation details of the prototype component.

Chapter 5 ("The Virtual Content Cache") presents the framework component responsible with caching of P2P traffic. The value of P2P caching is investigated and the design principles behind the caching component, including a deployment scenario, are presented. A discussion of cache replacement in general, along with a deeper examination of proposed strategies, suitable for P2P workloads follows. Finally, a brief description of the prototype implementation is presented.

Chapter 6 ("The AVP Simulator") presents the simulation software developed for the evaluation of this work. A review of available third-party simulators along with the reasons that led to the development of a purpose-built simulator is presented. Then, the design of the simulator is examined, discussing various aspects and important technical decisions. Finally, important details of the simulation models employed are discussed.

Chapter 7 ("Evaluation of the AVP") is devoted to investigating the effects of AVP framework deployment on peer and network performance. First, the basic settings and assumptions of the simulation setup are defined. Then, the effect of peer traffic localisation is investigated. An examination of single-VCC deployment performance follows. This includes direct comparison of different caching strategy performance using two distinct workload scenarios. Having established the necessary background to caching strategy effects, multiple AVP deployments are evaluated next. The evaluation continues with an investigation of the effects of AVP placement on performance. This is followed by a brief examination of the economics of AVP deployment. Finally, the chapter is completed with the discussion of component testing of the AVP prototype.

Chapter 8 ("Conclusions and Future Work") concludes the thesis. A summary of key points and the contributions made is presented. This is followed by an examination

of work related to the AVP. Finally, possible directions towards which the research presented in this thesis may be extended in the future are explored.

The bibliography section ("References") and Appendices complete the thesis.

1.3 A note on terminology

Throughout this thesis, the terms *network provider* and *Internet Service Provider* (ISP) are used interchangeably to describe the single administrative entity which controls an Autonomous System (AS). Furthermore, the terms *inter-AS*, *inter-ISP*, *inter-domain* and *transit* traffic are all used to describe network traffic which traverses the boundaries of a single AS to reach other parts of the global Internet through another ISP(s). Unless stated otherwise, the terms imply a billed service, provided by a higher-tier ISP.

While not strictly synonymous, in the context of this thesis the terms *P2P service* and *P2P application* are also used interchangeably as a means to avoid tedious repetition in the text. Where distinction between these terms is meaningful, it is clearly noted.

2. PEER-TO-PEER NETWORKS

2.1 Introduction and historical context

Over the past few years, we have witnessed the rapid emergence of a seemingly new paradigm in computer networking, called "peer-to-peer" (P2P) networking. P2P networking is rather a paradigm and not a technology because it promotes a different approach to the way a computer network can be utilised, without necessarily dictating specific changes to the existing network infrastructure [Oram, 2001]. P2P networking offers a model of developing network services designed to enable better use of available resources while addressing some of the limitations faced by existing approaches.

The predominant network service architecture employed today is the "client/server" model [Clay, 1998]. In that, a designated computer, the *server*, provides a specific service to any number of other computers that connect to it, called the *clients*. The server is assumed to be always on, always connected to the Internet, and assigned a permanent IP (Internet Protocol) address. Furthermore, in order to ensure it provides a high-quality service, exhibiting high availability and fault-tolerance, the server usually requires high-end hardware and runs specialised software. On the other hand, the users of a service, the clients, are generally assumed to have limited capabilities in comparison. Depending on the nature of the transaction, the clients may send data necessary for the completion of the service to the server or carry out some tasks locally but, generally, it is assumed that all major service intelligence lies with the server [Edelstein, 1994]. As a consequence, the client/server model is thought of as a centralised service architecture (the server being the central service component) where the clients and the server have highly asymmetric roles. A result of the state-of-the-art during the early years of the Internet and of computing in general, when computational power was scarce and users had to rely on centralised infrastructures for their computing needs, the client/server model became the de facto way of building Internet applications. However, at the time the main users of the Internet were universities and research centres with the appropriate knowledge base, resources and actual utility for it,

so it was not uncommon for client terminals to also be always connected and have permanent IP addresses and DNS (Domain Name Service) entries assigned like the servers.

The advent of the World Wide Web (WWW) [Berners-Lee, 2004] however, made the separation between client and server computers more evident. The graphical web provided an application of such value and ease of use, that people outside the "early adopter" communities started connecting to the Internet to use it. These people did not own the resources and connectivity of the large institutions and in order to run a web browser connected their PCs (Personal Computers) to the Internet through a modem by means of an ISP (Internet Service Provider). This created a second class of connectivity because, in contrast to the permanent presence model described previously, these computers would enter and exit the network frequently and unpredictably. People would connect to the Internet for long enough to access some web pages and then disconnect to free their telephone lines. Furthermore, because the 32-bit IPv4 address space was not sufficiently large to accommodate this new crowd with individual static IP addresses, ISP's started allocating IP addresses dynamically. This solution meant that different IP addresses were allocated to a computer every time a new session started, and that prevented them from obtaining permanent DNS entries.

The lack of permanent connectivity and DNS registration prohibits the majority of network-enabled computers from providing services on their own. Furthermore, the fundamental assumption of asymmetric capabilities between servers and clients of the client/server model impedes clients from playing a larger role in the provision of a service. These days, when the processing power found at the desktop is very substantial and broadband Internet connectivity is the norm rather than the exception [Ferguson, 2007], the existing service model is, for many, highly inefficient. A modern PC, if used for web browsing purposes for example, remains idle most of the time with its processing power "wasted" between keystrokes and the communication lines lying idle while the content is viewed on screen. At the same time, computing has experienced a paradigm shift. People do not use computers just for discrete, self-contained computational tasks anymore. Today's world is all about collaboration, information exchange, flexibility. This is where the P2P philosophy comes into place: P2P computing comprises a class of distributed network applications that take advantage of the computer resources available at the edge: storage, processor cycles, and information. Resources that the traditional client/server model cannot easily tap into.

2.2 What is Peer-to-Peer networking?

The term peer-to-peer characterises a family of network applications where all the entities interacting together are considered to be equal partners[2]. That is, each entity has the same set of service capabilities and responsibilities as any other. There is no distinction between client and server machines; every participating computer can play the role of a client and a server[3] at the same time. A P2P network service is, thus, provided cooperatively by the peer population. This gives P2P applications some interesting properties: Firstly, P2P systems are highly decentralised by nature. As no node has necessarily more authority or exclusively facilitates parts of a service, there is no need for central points of control. Therefore, although it is possible to have some form of centralisation or hierarchy to meet certain application design requirements, P2P services are generally decentralised. Furthermore, peers are highly autonomous. They do not explicitly rely on supporting infrastructure in order to function and may enter or leave the network at will. Thus, although the longer a peer stays online the longer it can assist in the provision of a service, the overall service is designed so that it does not take permanent availability for granted. As a consequence, peers have the ability to self-organise into transient networks which adapt to failures while maintaining an acceptable level of service. Finally, by forcing a symmetric relationship between computers, P2P systems manage to pool the resources committed by every participant and not only these of designated central servers. Every computer that joins a P2P network makes some of its resources available to the rest. Figure 1 illustrates this fundamental architectural difference between the client/server and P2P models.

In essence, the core elements of P2P applications can be described by four words: Presence, Identity and Edge resources (PIE) [Shirky, 2001]. These elements not only characterise the P2P model, but also show how it contrasts with the client/server model. *Presence* signifies the ability to tell when a resource is online. Determining the

---

[2] A number of different definitions of P2P computing exist in literature (e.g. [Shirky, 2000; Oram, 2001]), with some focusing on the architectural aspects of P2P while others concentrating on a more resource-centric view. This section attempts a consolidation of both these views, providing a broader – and thus longer – definition.

[3] As it will be shortly discussed (but not stated outright for clarity), a peer can play the role of a client, a server *and* a router at the same time.

presence of a resource is necessary for P2P networks since the permanent availability of resources is not guaranteed. Once the online presence of a user or a resource is established however, any number of highly personalised services can be offered. Presence is vital for the creation of user-centric systems, with instant messaging an immediate example.

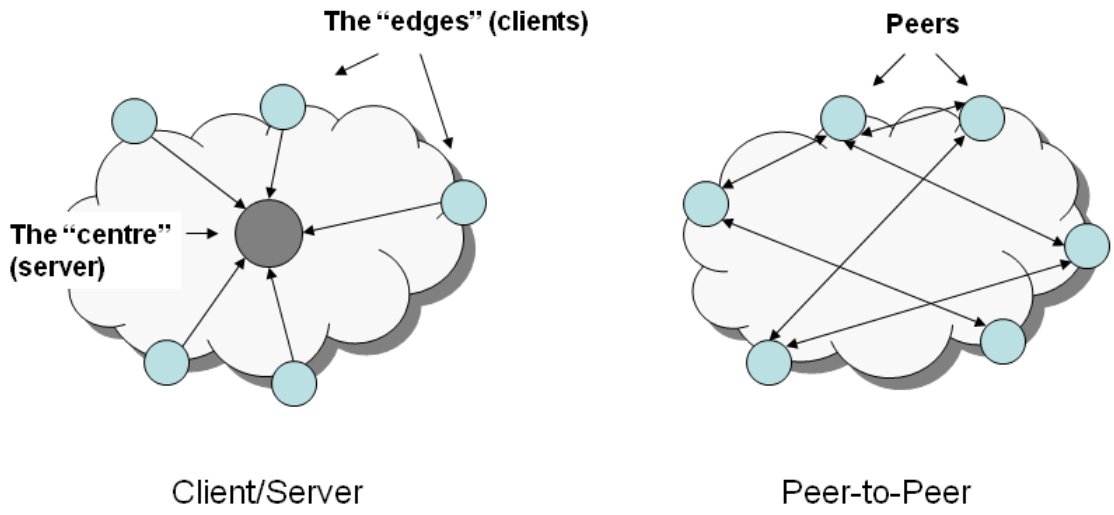

Figure 1: The Client/Server and P2P models.

The second element of a P2P application is *Identity*. P2P networks must be able to uniquely identify the resources that are available each time. The DNS system in use today was designed and is only suitable for machines that are permanently connected to the Internet [Shirky, 2000]. Users who do not own static IP addresses cannot be identified by DNS servers, and even if workaround solutions such as Dynamic DNS [Vixie, 1997] are employed, under IPv4 [Postel, 1981b] there are not enough IP addresses to satisfy everyone. P2P systems address this issue by employing their own, DNS-independent naming schemes. P2P applications like ICQ [ICQ], Groove [Groove] and Skype [Skype] bypass the DNS system and use their own directories of protocol-specific addresses that map to IP addresses in real time. By doing so, they devolve connection management to the individual nodes and abandon the machine-centric view dictated by the DNS. Therefore, no matter which IP address a user has been assigned during a session, he can still be identified appropriately by his peers and make full use of the advantages of a permanent identification scheme.

The final core element of a P2P application is that of the *Edge resources*. P2P networks enable the use of the resources available at the "edges" of the Internet: processing power, storage, content, human presence. This is in contrast with today's

client/server services where the usable resources are these concentrated in the servers: the "central" areas of the Internet. P2P services organise a variable-sized pool of distributed resources owned by the participating peers and allows them to use it collectively.

The triptych of decentralisation, autonomy and symmetric roles that is so characteristic of P2P services, and the inevitable departure from the original end-to-end model of the Internet due to the use of Network Address Translation (NAT), firewalls and various other "middleboxes" (e.g. proxies, etc) leads P2P services to create service-specific logical topologies that operate on the application layer. These logical topologies are called overlays. The use of overlays gives P2P service developers significant flexibility in providing new types of services that overcome the loss of the end-to-end symmetric relation and are not heavily reliant on the structure of the physical network infrastructure. A result of this flexibility is that overlay topologies can differ significantly from the underlying infrastructure, as illustrated in Figure 2, below. For that reason, peers are expected to play the role of routers, enabling the routing of messages at the application layer.

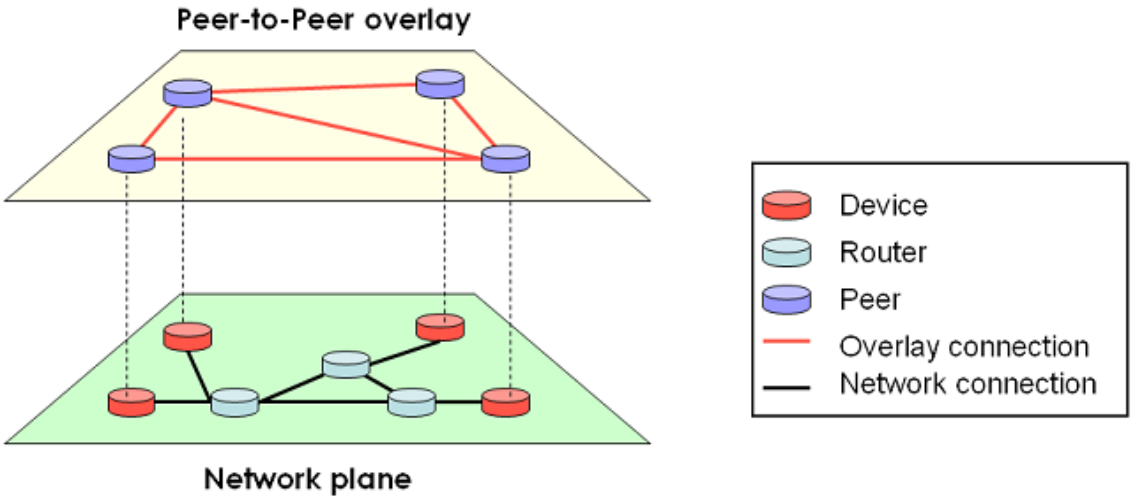

Figure 2: P2P application layer overlay.

It is common practice to describe as "peer-to-peer" any relationship in which multiple, autonomous, hosts interact as equals. An autonomous host is useful in its own right, even in the absence of others. The peering relationship creates the possibility of

additional functions made available to the peers collectively as a consequence of their collaborations with others. Known as the "network effect", the value and extent of these added powers increases dramatically as the number and variety of the peers grows[4] [Economides, 1996]. The more peers join the network; the more resources are committed to be collectively utilised.

The importance of the network effect becomes even more apparent if it is realised that peers do not have to be similar. P2P networking can take advantage of physical disparity to offer novel services. Indeed, the availability of a broad spectrum of wired and wireless communication technologies and the abundance of bandwidth and processing power available not only to typical computer platforms but various other devices makes peering between disparate entities both possible and desirable. While peering between computers may remain central, other devices that contain processors, memory and have network connectivity can engage into P2P communication. The possibilities for novel and ubiquitous services arising from this realisation are, indeed, numerous.

2.3 Why P2P? The user's perspective

The almost universal adoption of the client/server model as the de facto architecture for Internet services can be largely attributed to its inherent suitability for centrally-managed services and simple and lightweight clients. These characteristics helped develop applications of great importance, such as the World Wide Web and FTP (File Transfer Protocol), in an era when client-side complexity was a large limitation. Nevertheless, as the Internet becomes increasingly important in communication, business, entertainment and social life, the client/server model is beginning to show its limitations.

The most evident is scalability. Scalability is the capability of a system to keep functioning efficiently as it grows in orders of magnitude [Hill, 1990]. In other words, scalability is a sign of how a network will react as more and more people use it and for

[4] A number of empirical laws (Metcalfe's law [Shapiro, 1999], Reed's law [Reed, 1999], or more recently a refutation of the former by Briscoe et al [Briscoe, 2006; Simeonov, 2006]) describe the rate of growth.

increasing numbers of tasks. As server resources are limited, it is clear that the client/server model cannot scale infinitely. A server can only serve a limited number of clients as each service request consumes a portion of the server's resources. Consequently, on larger service request loads than the server can handle, the latter can become unresponsive causing service unavailability. Indeed, the lack of scalability of the client/server model can easily be witnessed when dealing with "flash crowds". The term "flash crowd" is used to describe large numbers of clients trying to access a popular service within short timescales. A large enough flash crowd can overwhelm a server with simultaneous requests and render it unavailable to any further clients, severely disrupting the service[5]. The issues stemming from the limited scalability of the client/server model become further pronounced due to the architecture's poor load-balancing (explicit techniques such as server clusters and load balancers need to be employed to alleviate server load) and redundancy (the server poses a single point of failure or attack). As Internet applications become more resource-hungry and the Internet grows in size due to ubiquitous networking, the move to IP-based telephony and the expected integration of Internet television platforms and mobile networks, the scalability of the client/server model becomes a major limitation. Current approaches that attempt to tackle the problem by constantly adding more resources to address the demand, present only short-term solutions. Resource over-provision is an inherently inefficient and uneconomical way to deal with the scalability problem.

In contrast, the client side remains largely under-utilised. The client/server model does not provide a clear way for client resources to be tapped upon as part of offering novel network services. This is particularly inefficient today, when even low-range desktop computers have clock frequencies measured in multiples of the Gigahertz as well as Gigabytes of memory. As Moore's law [Moore, 1965], is still surprisingly accurate, one thing becomes evident: The computational resources found in a typical home or desktop environment are too rich to be ignored. Unfortunately, the inability of the DNS system to support this class of networked computers, the extensive use of

[5] Sometimes called the "Slashdot effect" in the context of WWW, from the effect the popular technology news site has on directing "en masse" its large audience on other newsworthy web sites in short timescales.

Network Address Translation (NAT) to mitigate the problems arising from the scarcity of the IPv4 address space, the existence of firewalls, proxies and various other "middleboxes", have caused a departure from the end-to-end principle that was a fundamental value of the Internet and preclude most of these computers from providing services and being used to their full extent.

This has a direct effect on the availability of information. Being a centralised architecture, the client/server model has essentially put obstacles in the free flow of information by concentrating it in central points, owned by distinct entities. This centralisation makes the management of a service easier, but at the same time makes possible the control of information according to the server owner's criteria (commercial, cultural, political etc). Naturally, this creates an asymmetric relationship where few can publish information and the rest can only access it. For most, the web experience so far has been a passive, television-like consumption of the information available there rather than the full-blown, bidirectional interaction one would expect made possible by the "Inter-network". The "explosion" of blogging[6] in the recent years, which essentially removed just a few technical barriers from the process of creating a personal web page, is an indication that people are as much interested in publishing their own ideas and content if given the opportunity, as they are in accessing original material from others. The information available at the edges has value. In that light, the client/server model has been restrictive. If (contrary to [Odlyzko, 2001a]) "content is king", services that wish to leverage public's creativity or knowledge need to enable a more symmetric relationship with it, something which under the client/server model is inherently challenging to achieve.

P2P networking has been proposed as a way to overcome the aforementioned limitations of the client/server model. It is decentralised by nature, thus better dealing with scalability issues. As the network grows, there are no single points of stress: The burden of providing the service is distributed among the peers. Secondly, the P2P architecture is very flexible, in the sense that P2P services are designed to be as agnostic as possible of the underlying infrastructure. As such, current P2P services have shown

[6] The neologism "blogging" refers to the creation of web-based online diaries and journals, called weblogs or, simply, blogs. Apart from text, blogs may feature rich multimedia content.

that they can be deployed "into the wild" with very little or no infrastructural support. Moreover, as will be discussed more extensively in the next chapter, P2P services boast high resilience. Peers are expected to be volatile and the failure of any peer does not significantly impact the overall service. This volatility also accommodates the behaviour of intelligent mobile devices such as smartphones and PDAs (Personal Data Assistants) whose timescales of online availability are much different of that of typical computers.

P2P architectures leverage their ability to function, scale, and self-organise in the presence of a highly transient population of nodes, network, and computer failures, without the need of a central server and the overhead of its administration, to offer new, more effective approaches on addressing existing needs[7]. Administration, maintenance, responsibility for the operation, and even the notion of "ownership" of P2P systems are also distributed among the users, instead of being handled by a single company, institution or person [Androutsellis-Theotokis, 2004]. Furthermore, P2P architectures have the potential to accelerate communication processes and reduce collaboration costs through the ad-hoc administration of working groups [Schoder, 2003]. The result is more flexible, user-centric and feature-rich network applications. So, while the P2P approach is not a panacea to every challenge faced in providing better Internet services - and indeed, regardless of the architecture's said limitations a lot of network applications are naturally better implemented in a client/server fashion - for many application domains a P2P approach is advantageous. Indeed, the author argues that the early success of P2P services like Instant Messaging, played an important role in the realisation of the potential of social networks and the recent attempts to capitalise on that potential with a new breed of web-based applications and communities (for example Wikipedia [Wikipedia], YouTube [YouTube] or flickr [flickr]), collectively put under the "web 2.0" banner [O'Reilly, 2005].

To summarise, from the user's point of view P2P is particularly appealing because it offers:

---

[7] For example, both the file-sharing and instant messaging application domains existed long before the emergence of relevant P2P services, with FTP and the UNIX "talk" being the most famous client/server-based applications respectively.

- Easy access to a wide range of, otherwise unavailable, resources.
- Better scalability and exploitation of the "network effect".
- Direct involvement in the service provision - more immediate and natural exchange of information.

2.4 P2P application domains

To this day, P2P architectures have been employed or proposed for a variety of different applications. Some present novel services made possible by the employment of the P2P concepts while others constitute "remakes" of existing network services, originally built using the client/server architecture, which leverage the P2P model's strengths.

2.4.1 Content distribution

P2P content distribution (under which *file-sharing* is included) enables the pooling and retrieval of content available at the edges of the Internet. This way, not only the pool of available information is increased many fold, but information that would not be easily accessible under the client/server model (e.g. the "hidden web" [Bergman, 2001; Raghavan, 2001], information that does not have commercial interest for a company to promote, etc) becomes readily available. The peer-to-peer relationship is more intuitive and enables users to make available their own content to the world without many of the obstacles presented by client/server approaches. Furthermore, P2P file-sharing enables content distribution on a larger scale making every user a potential distributor. Instead of having a single server bear the burden of distributing a file to each client separately, which becomes very costly if the file proves popular, peers undertake this effort collectively, at virtually no cost to the publisher and with much better load-balancing and redundancy. Apart from the obvious effects this has for the dissemination of information, new business models can be created pushing further the proliferation of the Internet as a content distribution medium. The embrace of P2P in the form of using BitTorrent [BitTorrent] for the legal distribution of movies and large files (such as the Linux operating system image files), or the recent BBC iPlayer [BBC iPlayer] offering by the BBC show that companies and organisations are keen to offer new ways of delivering content while minimising their requirements and investment for service infrastructure.

In general, P2P content distribution systems cover a broad spectrum, from light-weight file-sharing applications, to more sophisticated systems that create a distributed storage medium for indexing, publishing, organising, searching, updating, and retrieving data with various degrees of fault-tolerance or security. Some concentrate on performance (for instance BitTorrent and Kazaa [Kazaa]) while others enforce anonymity or resistance to censorship (Freenet [Clarke, 2000]). Other examples include Gnutella [Gnutella], Oceanstore [Kubiatowicz, 2000], PAST [Druschel, 2001], Chord [Stoica, 2001], and eDonkey [eDonkey].

2.4.2 Communication and collaboration

P2P Instant Messaging (IM) or "collaborative" systems provide simple, fast and effective one-to-one, one-to-many and many-to-many communication and collaboration between peer devices. Voice-over-IP (VoIP) and real-time video capabilities alongside text have made P2P IM services particularly attractive even for segments of the population with no significant technical background, making a lot of the traditional telcos reassess their long term strategies and business plans. Users are in the millions[8] [Mark, 2004], while the convergence of IM protocols (AIM [AIM], MSN [MSN] and Yahoo messengers [yahoo]) [Naraine, 2004] hints to the potential the vast social networks formed through such services have for the creation of value-added services. Other examples of IM and collaborative P2P applications include Skype [Skype], ICQ [ICQ], Groove [Groove] and Jabber [Jabber].

2.4.3 Internet service support

A variety of Internet supporting services based on P2P infrastructures has been proposed. These include new types of services as well as "remakes" of existing Internet services that were originally based on the client/server model. Examples include Internet indirection infrastructures [Stoica, 2002], DNS systems [Ramasubramanian, 2004; Park, 2004], P2P multicast systems [Castro, 2002a; Vanrenesse, 2003; Bhargava, 2004], P2P-based publish/subscribe systems [Rowstron, 2001b; Pietzuch, 2003; Chirita,

[8] Skype [Skype], the popular P2P VoIP service reached the 100 million user mark on April 2006 [Reardon, 2006].

2004] and security applications, offering protection against denial of service or virus attacks [Keromytis, 2002; Janakiraman, 2003; Vlachos, 2004].

2.4.4 Database systems

Database systems benefit from good scalability and fault-tolerance properties and, naturally, creating distributed database systems based on P2P infrastructures has attracted a lot of attention. The transient nature of peers introduces challenges to maintaining data consistency, making this area particularly active. PIER [Huebsch, 2003] is a distributed and scalable query engine built on top of a P2P overlay network topology that allows relational queries to run across large numbers of computers. The Piazza system [Halevy, 2003] provides an infrastructure for building Semantic Web [Berners-Lee, 2001] applications based on P2P. Finally, Edutella [Nejdl, 2003] builds on the W3C RDF (Resource Description Framework) metadata standard, to provide a metadata infrastructure and querying capability for P2P applications.

2.4.5 Distributed computation

This category includes systems that enable the processing of computationally-intensive tasks by pooling the processing resources (CPU cycles and memory) made available by the participating peers. Tasks are broken down into smaller work units, manageable by typical computers, and distributed to peers where they are carried out. Results are then reported back and combined with these of other peers. This category of P2P services bears very close similarities with Grid computing [Foster, 2002]. It can be argued that Grid systems focus on infrastructure whereas P2P distributed computation systems concentrate on fault-tolerance. [Foster, 2003] discusses the matter in detail. Examples of such systems include projects such as Seti@home [Sullivan, 1997], genome@home [Larson, 2003], evolution@home [evolution@home] and others.

2.5 Lessons from the deployment of contemporary P2P services

The enthusiasm and, perhaps, hype created by the profound popularity of early P2P applications such as Napster and Kazaa [Kazaa] and the ability to share files “for free” has relatively subsided and a more elaborate observation on the high-level behaviour of P2P applications and their overall impact on the network can be made.

Spearheaded by file-sharing applications, P2P services are steadily gaining in popularity. Despite the fact that a few years ago such applications did not exist, more

than 50% of all the traffic measured on the Internet backbones nowadays is generated by P2P-based applications [netflow; Sprint], while others claim that for residential customers P2P traffic amounts for up to 80% of the overall [Goldman, 2004; Ipoque, 2007]. These figures are not only a measure of their popularity, which is reported to be steadily increasing [Cho, 2006], but also of their operational behaviour.

The current Internet evolved from the expectation of carrying traffic generated almost predominantly by client/server applications and is, thus, engineered with these traffic characteristics in mind. Traffic was assumed to be highly asymmetric, match time-of-day patterns and be characterised by the popular "mice" and "elephants" analogy[9]. The influence of the client/server model's asymmetry in roles and volume of uplink/downlink information exchange cannot be seen anywhere more clearly than in current access technologies like ADSL (Asymmetric Digital Subscriber Line) which allocate disproportionately more capacity for the downlink compared to the uplink. By making the exchange of large volumes of data easy and popular, P2P file-sharing applications invalidated many of these assumptions. The number of "elephants" increased dramatically in a network, not engineered with these traffic characteristics in mind. High-volume traffic flows can now appear on short timescales at almost any location at the network's edge, where bandwidth is most scarce. The long-range dependence (LRD) and degree of traffic self-similarity properties of aggregate traffic taken for granted for years [Leland, 1994; Floyd, 2001] are also reported to decrease with the predominance of P2P traffic [Azzuna, 2004]. This smoothing-out of traffic has implications for buffer dimensioning, bandwidth provisioning and congestion control. In addition, unlike client/server applications, P2P applications not only seek to utilise uplink capacity to its fullest, but also may operate unattended for days[10]. These

[9] The "mice" and "elephants" analogy describes the macroscopic behaviour of TCP (Transmission Control Protocol) [Postel, 1981a] traffic, as was observed for many years. In that, the large majority (around 80%) of Internet traffic was carried by a few long-lasting connections, called "elephants", while the rest was caused by a large number of very short-lived connections, dubbed "mice" [Guo, 2001].

[10] Current P2P file-sharing applications, for instance, do not require user presence after the request for content has been formulated. A user can create a list of downloads and leave the application running in the background.

dynamics clash with practices such as link oversubscription[11] or other traffic engineering techniques network providers apply based on statistical assumptions drawn from a client/server-oriented perspective. P2P applications, with their symmetry, long connection lifetimes and traffic patterns, do not fit extremely well in the current Internet and, due to their aggressive use of multiple parallel TCP connections, can disrupt the operation of other network applications by over-consuming scarce network resources.

As mentioned earlier, nodes in a P2P service have the same roles, must be highly autonomous and, in most circumstances, completely decentralised. In order for a P2P application to satisfy these requirements, and provide mechanisms to address the technical implications arising from the presence, identity and edge resources triptych outlined earlier, it usually contains its own protocol set for communication. The majority of P2P applications create overlays on top of the application layer where peers form their application-specific virtual topologies. Inside an overlay peers, amongst other tasks, need to discover their neighbouring nodes, locate resources, find out topology information, advertise their capabilities and make routing decisions, and they need to do these in a decentralised, infrastructure-independent manner. These tasks are achieved by message exchange, either in broadcast or unicast mode depending on the purpose of the message. Discovery and advertisement messages are in most cases broadcasted[12], in order to reach as many recipients as possible, while point-to-point communication is used for direct interaction. Furthermore, because of the autonomous nature of peers, a P2P overlay is a highly dynamic environment. Connections may be formed or destroyed at will. Peers may join or leave without notice. This imposes additional problems as information needs to be refreshed often enough to ensure peers have a realistic picture of the conditions inside the overlay.

As a result, considerable bandwidth is consumed by the current generation of P2P applications for signalling purposes [Azzuna, 2004]. At an extreme, according to

---

[11] Link oversubscription is the practice of allocating the available link bandwidth to more than one customer based on the fact that statistically, no subscriber will use all of the available bandwidth at any given time. This allows the provider to increase its revenue per link. A typical oversubscription ratio for residential ADSL connections is 50:1.

[12] This is a common case with unstructured decentralised P2P topologies (examined in the next chapter), as well as with many structured.

measurements done in [Ripeanu, 2002], signalling traffic in the early Gnutella network made at points up to 55% of the overall traffic during the measurement period. In such cases, the excellent - in theory - scalability properties of P2P services no longer hold. Good scalability is especially critical in the P2P domain because, due to the network effect, the service increases in value with the addition of more peers and there is a strong incentive for new peers to join. Newer P2P services acknowledge these issues by employing more complex protocols and leveraging the characteristics of overlay topologies, as will be discussed in detail in the next chapter. Nevertheless, it becomes clear that autonomy and decentralisation come at a cost.

Another realisation that came with the large-scale deployment of P2P services is that even under the P2P paradigm, "not all peers are the same" [Saroiu, 2002]. Peer devices have very diverse processing power, memory, storage or connectivity capabilities to mention just a few. Furthermore, there are differences in peer uptime and participation. In many systems, peers act selfishly avoiding contributing resources for the "greater good" of the service (the so called "free-riding") [Chu, 2002; Figueiredo, 2004]. For certain types of services, opting for a "lowest common denominator" approach to peer equality prohibits the exploitation of all available resources to their full potential and also may cause large service inefficiencies. In the P2P file-sharing domain, for instance, treating dial-up connected peers the same as their broadband-connected counterparts leads to link saturation and crippling of service for the former and overlay instability for everyone.

Finally, by creating their application-specific overlays, P2P services form topologies which can deviate significantly from the underlying physical Internet infrastructure. This results in large discrepancies between the peers' physical location and their place in the P2P overlay. It is not uncommon for peers that are physically located in different countries to be first-hop neighbours inside a P2P overlay [Klemm, 2004; Cho, 2006]. As a consequence, P2P systems may suffer from non-optimal links, large packet round-trip times and trigger large and unnecessary costs for the network providers. This last point is particularly important for the network providers and will be discussed further in the following section.

To conclude, the current generation of P2P applications exhibits the following high-level behaviour:

- P2P overlays are extremely dynamic environments due to the continuous arrival and departure of peers at fast and unpredictable rates.

- The unpredictability of peer behaviour, in conjunction with the lack of any supporting infrastructure (which is one of the main characteristics of any P2P system), necessitates the generation of significant volumes of signalling traffic.
- P2P systems can interfere with other network services because of their different traffic characteristics, capabilities and operation timescales, and the overall design of the current Internet.
- P2P systems that fail to identify the different capabilities of participating peers and do not take into account the topological characteristics of the underlying physical Internet infrastructure, can become significantly inefficient, both for their users and for the network as a whole.

2.6 P2P from the network provider's point of view

It can be argued that applications such as Skype and BitTorrent have achieved widespread success and entered the public's consciousness as indispensable conveniences, making P2P the main driver in the adoption of broadband connectivity and the persistent demand for faster products from ISPs [Mennecke, 2005].

Despite that, most ISPs see P2P networking as a threat to their business rather than an ally. By enabling access and sharing of resources found at the edges of the network, P2P services disrupt the business models and planning ISPs conducted for years based on the asymmetric nature of client/server-based application traffic. Based on the assumption that such traffic will dominate their networks, ISPs built their infrastructure around technologies that accommodate asymmetry (such as ADSL) and developed business models that take advantage of it, providing attractive flat rate pricing schemes based on link oversubscription and time-of-day utilisation statistical models.

With peers collectively providing a P2P service at the edges of the network however, the ISPs see their infrastructure at constant stress. As discussed earlier, a file-sharing P2P application for instance will not only seek to utilise an ISP customer's connection to its full capacity in both directions, but may operate for days unattended. It has been reported that P2P services roughly double the total traffic and peak load on the ISP access links [Karagiannis, 2005]. Such utilisation has adverse effects on the ISP network causing undesirable latency or responsiveness for time-sensitive applications (Voice-over-IP, streaming, online gaming, etc), low responsiveness for web browsing during peak hours and increased packet loss. These in turn result in increased customer

dissatisfaction due to low quality of perceived service, complaints, customer churn and inability to support additional subscriber growth rates without further degrading the offered quality of service.

More importantly however, P2P applications incur ISPs a more immediate type of cost. For an ISP not all traffic costs the same: While traffic that is confined within an ISP's network carries relatively little cost, traffic that needs to traverse its boundaries and be passed to other ISPs and AS's (Autonomous Systems) is much more expensive [Sandvine, 2002] and subject to often inflexible service level agreements (SLAs). For that reason, ISPs try to minimise this second class of "transit" traffic as much as possible, traditionally by peering agreements and web caching (e.g. [Norton, 2003]). Critically, each network undergoes extensive traffic engineering to ensure network resources are optimally utilised not only in terms of technical (e.g. resilience, load, etc.) but also economic (e.g. existing peering or transit agreements, business relationships etc.) objectives. P2P services create and operate within application-level virtual overlays whose structure is determined by the different P2P protocols they use. These overlay topologies rarely take into account and reflect the topology of the underlying network infrastructure, meaning that a direct connection between two neighbouring peers inside a P2P network may in actuality traverse many different routers, AS's and even continents. As a result, P2P applications create as part of their operation large numbers of inter-domain connections and costly transit traffic. These factors lead ISPs to view P2P services from a negative perspective and compel them to act accordingly.

Responses in the direction of blocking or applying traffic shaping techniques on P2P application generated traffic, however, are short-term measures that fail to successfully address the broader issue. Firstly, most modern P2P applications use dynamic port addressing instead of static ports, making the efficient blocking or rate-limiting of specific P2P application traffic non-trivial [Karagiannis, 2004; Sandvine, 2003]. The encryption and obfuscation of P2P traffic as witnessed by recent BitTorrent clients make such a task even more complex. Moreover, such measures from ISPs may cause an "arms race" where P2P application developers will continuously implement features that make their applications more elusive to such techniques in order to avoid low performance, at the expense of the ISPs who will have to respond accordingly with increasingly intrusive measures. Such an arms race may prove a costly endeavour for the ISPs in its own right, as application-level (deep) packet inspection or traffic pattern identification involves considerable overheads that need to be compensated with additional expenditure on the infrastructure side. Even so, traffic shaping can only

provide temporary relief since it does not do anything to help improve the overall efficiency of the P2P overlay network's use of the network resources in place. At the same time, it leaves the problem of identifying unnecessary inter-ISP traffic unresolved. Finally, blocking or capping P2P traffic will probably be interpreted negatively by the P2P service-using customer base, if certain P2P applications have degraded performance or cannot function, with clear implications for customer satisfaction and churn.

Acknowledging the appeal of P2P applications and the fact that they have become a driver for subscribers to migrate towards faster broadband services, a lot of providers adopt a different approach and choose to promote themselves as "P2P-friendly". In that direction, they focus on regularly upgrading their infrastructure by adding higher capacity links and equipment to maintain sufficient headroom for sensitive network applications to gracefully coexist with their resource-hungry P2P counterparts. This is not, on its own, a sustainable solution either. Apart from being very costly and often technically complex[13], it alleviates the problem only temporarily. As more customers join, often with the motivation to use their high-speed connections, advertised as "unlimited", to their fullest, there will be a point where bulk P2P transfers will again stress the network dictating a new round of upgrades. And again, in a market used in a competitive flat-rate pricing scheme, it may be very difficult to pass the costs of sustaining the infrastructure to the customers. Finally, as many ISPs do not own their own infrastructure but rather resell wholesale products this approach is not always straightforward. It thus becomes clear that ISPs cannot apply "traditional" responses to the challenges brought by P2P services on their networks and businesses. A different approach is needed to tackle the unique nature of P2P services.

The pressing challenge is therefore to provide attractive P2P services, without however compromising other network services and sacrificing user experience in them. Management of P2P services can be a powerful tool to reach this goal. The central idea of this research is that ISPs should be able to manage certain characteristics of P2P

[13] Complexity arises not only because solving the optimisation problem - identifying how the finite resources can be allocated in the best possible way given a unique set of constraints (e.g. topology, demand, business ecosystem, etc) is hard, but also because the possibility of physical installation complications, incompatibility with existing infrastructure or downtime needs to be eliminated.

services that affect their infrastructure in a way that allows them to lessen that impact while allowing such services to operate in an unobtrusive and effective manner. Such an approach will help ISPs view P2P computing as an asset to their business while maintaining an environment where such services can flourish. Understanding P2P services and identifying the aspects of their behaviour where management can be applied beneficially is crucial. Too much intervention or control will render such services underperforming or near-inoperable and will reward these that take measures to evade such attempts. Correctly applied, on the other hand, P2P management can help create value for both the customer and the provider.

2.7 Conclusions

P2P applications have emerged as a powerful alternative to "traditional" client/server architectures. From a user perspective, they present a very attractive class of applications, due to the appealing "free" and unmediated use of networked resources found at the edges of the Internet. Nevertheless, it is clear that the deployment of P2P systems in a larger scale and of a broader scope - especially in the corporate environment where current P2P uptake is limited - depends strongly on providing more efficient and manageable P2P applications that address the issues discussed in the previous sections. This is especially critical from the network providers' point of view. While P2P services, and especially file-sharing applications, have proved to be a driver for subscribers to continuously migrate towards faster broadband services, network providers view P2P as a technology that creates value for end users and content providers at their expense. ISPs, thus, need to manage aspects of P2P behaviour in order to provide a high quality network service to all their subscribers (P2P users or not) and incorporate P2P into a viable business model. After all, the strong demand for P2P services in today's competitive market indicates that sooner or later ISPs will have to address customers' needs or lose them[14]. The challenge is to do it in a way that is sustainable both from a customer quality of service and a business perspective.

[14] In the United Kingdom, for example, operate approximately 200 ISPs, while 99% of the population has a choice of more than 10 [Conti, 2007].

This chapter provided an introduction to P2P networking. The differences between P2P and client/server architectures were presented, along with a number of application areas and a discussion of general characteristics current P2P services have demonstrated. In the latter, common traits observed from the large-scale deployment of P2P services in general were presented, without delving into specific protocols and functions. The following chapter examines representative P2P protocols in detail, discusses how overlay topologies affect performance at the peer and service level and looks into individual peer behaviour and its effects.

3. PROTOCOLS, TOPOLOGIES AND PEER BEHAVIOUR

3.1 Introduction

As stated in the previous chapter, P2P services create virtual overlays at the application layer, with their own application-driven logical topology and routing mechanisms. Inside these overlays, peers may form networks whose topology is much different from that of the underlying physical infrastructure. These peer topologies, as will be shortly discussed, can be highly structured (where peers are interconnected in a specific way), completely unstructured, or somewhere in between. The existence of different topology classes is due to the fact that each class exhibits different strengths and weaknesses which limit its suitability to particular problem areas only. Indeed, in many cases a trade-off is evident; any advantages brought by the topology in carrying out certain functions come at the expense of inefficiency in other areas. The choice of the "right" topology is, therefore, a result of careful assessment of the P2P service's design goals and requirements and in turn will have a large influence on the way the service will be implemented and operate.

Building on the introduction to P2P network characteristics given in the previous chapter, this chapter delves deeper into the behaviour of different classes of P2P networks and the effect they have on the Internet ecosystem. Large part of this analysis focuses on the examination of the role the fundamental types of P2P overlay topologies play on that behaviour. For each class, the way representative protocols operate is reviewed, discussing their strengths and shortcomings. Then, the focus is shifted to the behaviour of individual peers and how this affects the system as a whole. This examination gives the necessary insight on the kinds of challenges involved in creating and operating P2P services, and provides the link to the following chapters where the proposed framework for managing P2P services is presented.

3.2 Topological categorisation of P2P networks

Networks like the Internet have gained such complexity that many of their topological properties cannot be determined with precision [Willinger, 1998]. Graph theory [Bollobas, 2002] is central to the study of such networks and the following terms and notation, ubiquitous in the relevant literature, are used throughout this chapter:

- **Definition 1:** A network can be represented as a directed or undirected graph $G = (V,E)$, where V is the set of vertices (nodes) and E is the set of edges (i,j) describing the connections between nodes $i,j \in V$.
- **Definition 2:** The term *node degree* signifies the number of edges (connections) a node possesses.
- **Definition 3:** The *characteristic path length* of a network is the number of edges in the shortest path (in hops) between two vertices, averaged over all pairs of vertices.
- **Definition 4:** The *network diameter* is the longest hop distance in the network.
- **Definition 5:** The *clustering coefficient* denotes the probability that two neighbours of a node are themselves neighbours.

Because each network has different goals to reach and faces different constraints, formalising a set of properties is necessary to the evaluation of network topologies. According to Minar [Minar, 2001], network topologies can be evaluated using seven fundamental properties:

- **Manageability:** Complex systems such as large networks require management, such as updating, repairing and logging. Manageability is a measure of how easy it is to manage a network.
- **Information coherence:** Indicates how authoritative is information found inside the system. Criteria of information coherence include non-repudiation, audit-ability and consistency.
- **Extensibility:** Specifies how easy it is to add new resources (new capabilities) to the network.
- **Fault tolerance:** Specifies how tolerant the network is to failures.
- **Security:** Identifies how well protected is the network from adversaries.

- **Scalability:** Specifies how large the network can grow, without degrading performance.
- **Resistance to politics:** Specifies how hard it is for an authority to shut down the system.

Until recently, three basic categories of P2P network topologies had been identified [Cohen, 2002]: Centralised, decentralised and unstructured, and, finally, decentralised and structured. A fourth category, hierarchical P2P topologies, needs to be added. The latter, as will be shortly discussed, combines elements from more than one of the other topology types to create two-tiered or multi-tiered networks, with the aim of combining the strengths of each tier's topology. Each category will now be reviewed in detail.

3.2.1 Centralised server P2P networks

An approach followed by the first version of Napster[15], Skype [Skype] and most Instant Messaging systems, this model utilises a central server which indexes content or other resources made available by the participating peers. Peers connect to the central server for search queries, on which the server replies with the IP addresses of available peers that possess that particular resource[16]. Peers then establish a direct link between them to share that resource or communicate. The resulting topology is also called a "hybrid" P2P topology because the central server is essentially mediating in a client/server manner, but the two parties then establish a P2P communication.

Although paradoxical, centralisation readily addresses some issues inherent in all P2P networks. The most significant is peer and resource location. The use of a centralised index guarantees that if a resource exists inside the network, it will almost certainly be located during a search - something that is not always possible in decentralised P2P networks, as will be soon discussed. Furthermore, the performance of

[15] Now Napster operates as a (client/server) pay-per-download music service, similar to Apple iTunes and other.

[16] In the case of IM systems, the server usually holds the individual contact lists ("buddy lists") and relevant authentication information and responds with the current IP addresses of contacts.

such approach is also unrivalled by fully decentralised schemes [Yang, 2001]. Additionally, a central server makes it easier to incorporate mechanisms for access control, authentication and trust management whose architecture so far relies heavily on centralisation, and to monitor the service. Finally, centralised server P2P networks are more capable of dealing with the "legacy client" problem[17] and generally can have a tighter control of the overall service due to the existence of a non-distributed service component – the indexing server.

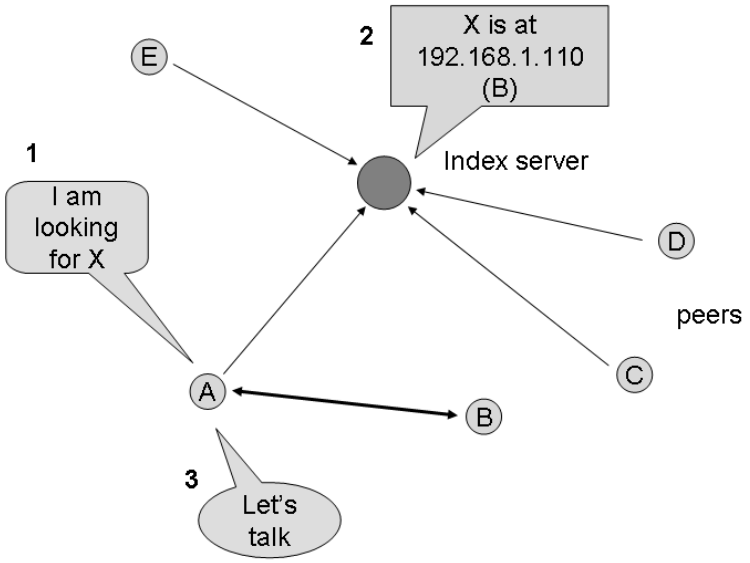

Figure 3: The centralised P2P topology

This model, however, has some considerable limitations. A hybrid P2P network essentially signifies a centralised topology with all the downsides this involves at the server side, such as introducing a single point of failure, a single point of stress, bandwidth bottlenecks etc. Since a large part of the overall service relies on a centralised component, unavailability of the index server due to failure or excessive load renders the service unusable. Denial-of-service attacks have the same effect: Loss of service, no matter how large the peer population is. Similarly, for the network as a whole to scale, the organisation that operates the server has to constantly provide more

[17] This term is used to describe older versions of P2P application software that are incompatible or support different functions from the latest version.

resources (e.g. more powerful indexing servers, higher capacity links) as more users join the network, at a significant financial cost.

A fundamental characteristic of the centralised P2P topology is that the ownership of such a network is clear, in the sense that the service is managed by the entity owning the central indexing servers. This has certain effects, with most prominent the susceptibility to censorship or legal action if users engage in illegal activity (such as the sharing of copyrighted material), even if the central servers do not contain (store) such material. Reliance on a central server means that a legal attack can have the same effect as a denial-of-service attack. This makes hybrid P2P topologies unsuitable for services with a high requirement for anonymity, resistance to censorship, or simply resilience.

Additionally, despite the existence of a centralised server, a P2P network remains an extremely dynamic environment where peers join and leave at will. As any central server cannot update its database sooner than a set period of time for system efficiency and scalability reasons, some of the entries will be inaccurate. Depending on the service, however, these inconsistencies may not have a significant impact on its quality as perceived by the user.

A slight variation of the centralised P2P model is employed by the BitTorrent file-distribution protocol [Legout, 2005; Izal, 2004]. In that, a central server is running the centralised component of the BitTorrent service called the “tracker”. Trackers are discovered using special metadata files called torrent files, stored and retrieved from an ordinary web server, using a standard web browser. The web server and tracker can co-exist on the same computer although this has implications for the overall scalability of the system. The tracker maintains an index of the IP addresses of all the peers currently downloading a particular file and provides BitTorrent peers with the IP addresses of a number of other already-connected peers, with which they engage in a P2P communication. The basic difference of BitTorrent with the hybrid model described above is the existence of the separate web server, facilitating out-of-band resource discovery in the form of providing torrent files since the protocol does not has such functionality built in. Apart from this difference, the tracker plays the role of the central server, giving BitTorrent networks the same advantages and disadvantages of every system based on a hybrid topology.

The centralised P2P model attempts to address issues inherent in any P2P service, such as searching, resource and peer discovery and identity management, by using a central server providing these services in a centralised manner. This way

essentially, the challenge of implementing these functions satisfactorily in a distributed fashion is sidestepped for the simplicity and peace of mind the tried-and-tested techniques developed for client/server services offer. The strength of the P2P concept however lies on the fact that it does things differently from the client/server way. By introducing centralisation, the hybrid model loses significant flexibility and becomes limited to specific application areas and needs. Scalability and reliability issues arise, while the end of Napster and the closing-down of numerous torrent sites provide the best example of how easy it is to shut down networks like these without attacking a single peer. For these reasons, a lot of developers shy away from the hybrid P2P topology and instead focus on its fully decentralised counterparts, examined next.

3.2.2 Unstructured decentralised P2P networks

Unstructured decentralised P2P networks are networks whose topology, as their name implies, has no structure or any form of centralisation. In these networks, services are fully distributed and peers have to collaborate with each other to perform basic functions such as resource location and peer discovery. For that reason they are often called "pure" P2P networks. Lack of structure means that peers can connect to each other any way they see fit, forming topologies that evolve dynamically in a random manner. Because of that, the examination of unstructured decentralised network topologies borrows a lot of concepts from the study of other complex networks.

Traditionally, networks of complex topology have been modelled using the random graph theory of P. Erdös and A. Rényi [Erdös, 1959; Bollobas, 2001]. Erdös and Rényi suggested that complex networks should be modelled as nodes connected with randomly placed links. More specifically, if N nodes are assumed and every pair of them is connected with probability p, a graph with approximately $pN(N-1)/2$ edges distributed randomly will be created. This results in network models where, despite the random placement of the links, most nodes have the same amount of links. The nodes follow a bell-shaped Poisson distribution which makes it rare to find nodes with more or fewer links than the average. Such networks are also called "exponential", because the probability that a node is connected to k other nodes decreases exponentially for large k (i.e. the degree distribution is quickly decaying).

On the other extreme, networks with clear design principles were traditionally assumed to be completely regular. The work of D. Watts and S. Strogatz in [Watts, 1998] showed that there exist many biological, technological or social networks that lie

somewhere between these two extremes: Networks which are not completely random, nor completely regular. In order to examine this type of networks they proposed models where regular networks are "rewired" to introduce increasing amounts of disorder. These systems can be highly clustered, like regular lattices, yet have small characteristic path lengths, like random graphs. In other words, they are networks with a high representation of "cliques" where most pairs of nodes are connected with at least one short path. Networks of this type were called "*small world*" networks, to acknowledge the work of S. Milgram in the 1960's, when he quantified the phenomenon of the "six degrees of separation"[18] [Milgram, 1967]. Kleinberg [Kleinberg, 2000], based on Milgram's experiments, demonstrated that in small world networks nodes are able to route messages to unknown targets.

On their seminal paper, Faloutsos et al [Faloutsos, 1999] argued that the Internet topology obeys power laws. Power laws are expressions of the form $y \propto x^{\alpha}$, where α is a constant and x and y are the measures of interest. The authors showed that an Internet router's degree distribution $P(k)$ follows the power law $P(k) \sim k^{-\gamma}$ with $\gamma = 2.48$. Barabási and Albert showed that the World Wide Web page distribution also follows a power law with $\gamma = 2.1$ [Barabási, 1999]. The difference from random networks is that while random networks are characterised by their bell-shaped distributions, power laws are continuously decreasing functions, as shown in Figure 4 below. The essence of that difference is that while in random networks nodes enjoy a "democratic" distribution of links, power laws describe systems in which a few hubs dominate. Hubs are vertices that possess a very large number of edges compared to the average vertex. Both the random network and the Watts-Strogatz small world network models do not have great, big hubs [Evans, 2004]. Power laws, on the other hand, which describe networks whose degree distribution has a much longer tail, are suitable for this type of networks. Such networks, described through power laws, are called *scale-free*

[18] According to the phenomenon of the "six degrees of separation", any two people in the world are very likely to be connected through six immediate acquaintances. This observation is a result of a series of experiments carried out in the 1960's by S. Milgram, and shows that vast social networks can have small characteristic path lengths.

networks. Scale-free networks maintain the small world property of high clustering and short characteristic path length.

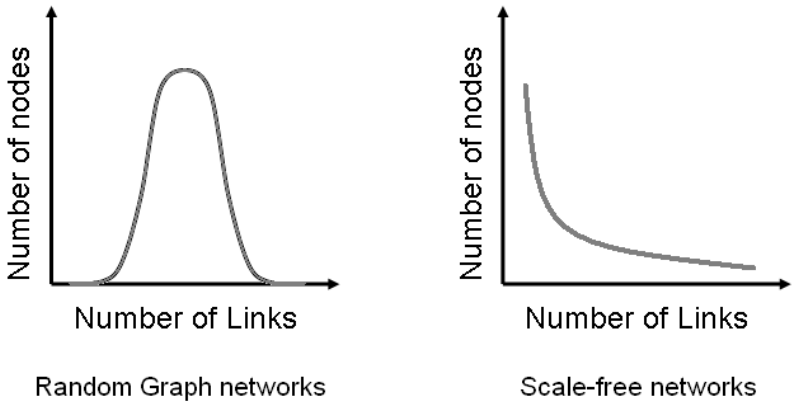

Figure 4: Random graph versus scale-free networks.

The existence of hubs in scale-free networks is mainly accredited to two mechanisms: *growth* and *preferential attachment* [Barabási, 1999]. In the Erdös-Rényi model, it is assumed that the full inventory of nodes exists from the beginning; before placing any links. This is highly unrealistic, as most networks do not maintain a constant number of nodes over time. The World Wide Web, for instance, grows by having more and more pages added over time. Similarly, the Internet grows with the addition of more routers and hosts.

A product of growth is preferential attachment. As new nodes appear, they can connect to any already connected node. However, they tend to connect to nodes that already have a lot of established connections than to nodes maintaining just a few. This can be easily explained: By connecting to already well-connected nodes a new node can take advantage of the improved reachability of the network offered by these nodes. This leads to these popular nodes acquiring more and more links as the network grows[19] and becoming hubs. Interestingly, the mechanism of preferential attachment tends to be linear [Barabási, 2003].

The existence of hubs indicates that such networks display self-organising properties [Evans, 2004]. Self-organisation is considered here as the process in which

[19] A phenomenon quoted as "the rich get richer" [Barabási, 2003].

the internal organisation of a system, normally an open system, increases automatically without being guided or managed by an outside source.

The fully decentralized, unstructured overlay network formed by Gnutella[20] was found to have the small world properties of small diameter and high clustering, as well as a node degree distribution which followed a power law with exponent $\gamma=1.4$ [Jovanovic, 2001]. Therefore, there is strong support that Gnutella and similar unstructured decentralised P2P applications form networks with scale-free properties.

This may seem surprising at first. Because no control is imposed on the way nodes join and leave the network by the Gnutella protocol, the network evolves topologically in an uncontrolled and unpredictable manner that does not outright indicate the formation of a scale-free network. Hub formation can thus be mainly attributed to individual peer behaviour and in particular to the existence of a small set of long-lasting peers. Measurements in [Saroiu, 2002] showed that although as much as 60% of all peers stayed connected for an hour or less, there was a small percentage of peers that stayed connected for longer (360 minutes or more). These are peers which enjoy good connectivity, as it is the low-capacity peers that usually join the network for short periods of time. By being available for longer and serving more links, they end up being preferred by other peers and become hubs inside the overlay.

The presence of hubs has certain effects on the fault-tolerance of Gnutella and similar P2P networks. Scale-free networks are extremely robust to accidental failures or random attacks [Albert, 2000]. While the random removal of a number of already-connected nodes will result in a severe fragmentation of a random network into tiny, non-communicating "islands", it will not significantly disrupt the topology of a scale-free network. This is due to its inhomogeneous nature: The random removal of nodes will inevitably focus on the ones maintaining few connections, as they are a lot more than the hubs. Consequently, the removal of little-connected nodes will not affect the network topology significantly because most connections go through the hubs. Indeed,

[20] The discussion in this section applies to the early, completely unstructured Gnutella network (protocol version 0.4). Later versions of the protocol (such as version 0.6) added super-peer support, and will be discussed later in this chapter.

Albert et al report that as much as 2.5% of Internet nodes can be removed without affecting its diameter [Albert, 2000].

It has to be noted that, at the same time, the presence of hubs reveals a serious weakness. Despite the fact that it provides resilience to failures or attacks on random nodes, it makes the network extremely susceptible to planned attacks on the hubs. If a hub fails or is attacked by an adversary who possesses suitable topology information, all nodes connected to it lose connectivity. This leads to extreme network fragmentation. Indeed according to the aforementioned study, if as little as 5% of the most-connected nodes are removed, the network diameter doubles.

Gnutella has demonstrated that unstructured decentralised P2P systems can be deployed "in the wild" without any form of support or intervention from additional infrastructure in order to operate, and will show considerable resilience. This important capacity is mainly accredited to the simplicity of the protocol which enables the peer network to attain self-organisation characteristics and form a scale-free topology. However, the early Gnutella network suffered from serious drawbacks in scalability and performance [Ritter, 2000; Portmann, 2001], almost crippling the service in the summer of 2000 and effectively hampering the deployment of such systems in a larger scale. At the core of the problem was the simple but inefficient communication scheme employed due to the lack of topology information.

In Gnutella, searching and collection of topology information is accomplished by broadcasting messages on all available overlay links. This results in excessive overhead traffic due to messages being delivered several times to the same peers (since they may be reachable via multiple paths) while at the same time they reach peers that are not capable of contributing anything to the resolution of the query issued (in case of searching). In fact, the presence of hubs accentuates this issue as due to their prominent role they inadvertedly make the generation of duplicate messages over multiple paths unavoidable. A hop-count horizon is used to limit the effect of uncontrolled flooding, by which a message is dropped after travelling a specified number of hops. Nevertheless, this feature is not effective enough as the traffic generated inside this fixed horizon is still high. Measurements in [Ripeanu, 2002] and [De Meer, 2003] found that the signalling traffic (PING and PONG messages) inside the Gnutella network was excessively and unjustifiably high, especially when compared with user-triggered traffic (QUERY messages). Because there is no knowledge of the network properties, peers cannot know metrics such as the diameter of the network and make decisions

accordingly. Therefore, even a conservatively set TTL (Time To Live) hop count value may exceed the network diameter.

Equally important is the lack of search guarantees offered by Gnutella. Since searching by broadcast merely reaches a random set of peers in the network due to the random coupling of peers, it does not provide any guarantees that the results of the search process are conclusive. Content that is actually available in the network may not be visible to all peers because the reachable horizon is restricted by the hop count. At the same time, the traffic caused by the search request may consume excessive resources in other parts of the network with no results.

More efficient search techniques such as a distributed version of iterative deepening search or directed BFS (Breadth First Traversal) have been proposed to rectify this situation [Yang, 2002; Yang, 2003], by adapting the TTL value of search messages to iteratively reach larger parts of the network according to the results returned, or by broadcasting on selected routes. Others like [Cholvi, 2004], [Fessant, 2004] and [Sripanidkulchai, 2003] propose the formation of communities of peers that share common interests where searches can be concentrated.

3.2.3 Structured decentralised P2P networks

3.2.3.1 Deterministic topologies

As seen so far, nodes in an unstructured decentralised network can only have a limited view of the network: They have a set of neighbours which determines their scope (for instance the Gnutella “horizon”) but cannot practically know the global topology. Operations on unstructured P2P networks become ineffective primarily due to this fact. Peers do not know where in the network a specific resource might be located; hence their search technique is usually restricted to simple broadcasting. But as discussed earlier, broadcasting is a very inefficient and ineffective message passing technique that introduces scalability constraints and cannot offer search guarantees.

Deterministic topologies address this issue - not by giving any node a global view of the network, which would imply centralisation - but by maintaining a deterministic topology of the network, which is known to all nodes. This way, nodes can have an idea of what the network beyond their scope looks like. This globally available information can then be used to reach locally optimal decisions while routing and broadcasting search messages. The information on the topology is packaged in a protocol that is used to police peers joining and leaving the network: Instead of allowing

peers to join and depart without any restrictions, peers are made to connect to specific peers already in the P2P network upon joining – in a way that maintains the topology in the desired state at virtually every moment in time.

Such approaches include the HyperCup protocol [Schlosser, 2001; Schlosser, 2002], the work of Saffre et al [Saffre, 2003], or aim at constructing random regular graphs [Law, 2002]. The HyperCup protocol borrows ideas from the area of multiprocessor computing and attempts to tackle the problem of scalability by organising peers into a hypercube-based graph structure. A complete hypercube graph consists of $N = b^{Lmax+1}$ nodes, where b is the base of the graph, i.e. the number of nodes in each dimension, and $(L_{max}+1)$ is the number of dimensions. In Figure 5, below, a hypercube with $L_{max} = 2$ and $b = 2$ (i.e. a 3-dimensional cube with 2 nodes in each dimension) is presented. The links between the nodes in the figure are numbered, and by definition a node B is called the i-th neighbour of node A if node B is A's neighbour in dimension i. For instance, in Figure 5, node 6 is the 2-neighbour of node 3. Edge labels start at 0 and each node can only have one i-th neighbour.

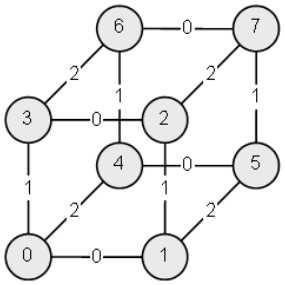

Figure 5: HyperCup topology.

The HyperCup protocol employs two basic techniques (creating/destroying dimensions and using temporary or permanent neighbours) to maintain a topology close to the ideal hypercube (Figure 5), depending on the rate of peers joining or leaving and their positions[21]. Using this structured topology, the HyperCup protocol achieves the following goals: The network is completely symmetric: Each node in the network is thought to the same capabilities and duties, and no node incorporates a more prominent

[21] It is beyond the scope of this section to discuss the HyperCup protocol mechanisms in depth. Interested readers are kindly directed to the referenced bibliography.

position than the others. That ensures that (signalling) load balancing is optimal and no hotspots are created. Correspondingly, any node is allowed to accept and integrate new nodes in the network. Finally, HyperCup achieves a $O(\log_b N)$ complexity in terms of messages sent when a node joins or leaves the network.

Optimal broadcast can be achieved on hypercube topologies (i.e. broadcasted messages reach each peer exactly once), reducing the traffic load on the network compared to unstructured systems. HyperCup guarantees that exactly $N-1$ messages are required to reach every single node in the network, and that the last nodes are reached after $\log_b N$ forwarding steps. Additionally, the characteristic path length L is $L \approx \log_b N$ [Schlosser, 2001]. The broadcasting scheme is achieved as follows: A node invoking a broadcast sends the message to all its neighbours, tagging it with the edge label of the link on which it was sent. Nodes receiving the message forward it only on links tagged with higher edge labels. This is illustrated in Figure 6, below. Peer 0 broadcasts a message to all its neighbours, namely peers 1, 3 and 4. Upon receiving the message, Peer 1 examines the tag and forwards it only to links with labels higher than 0, i.e. to peers 2 and 5. Similarly, Peer 3 only forwards it to 6, since it received it from a level-1 link. Likewise, Peer 2 forwards it to Peer 7, it's only level-2 link.

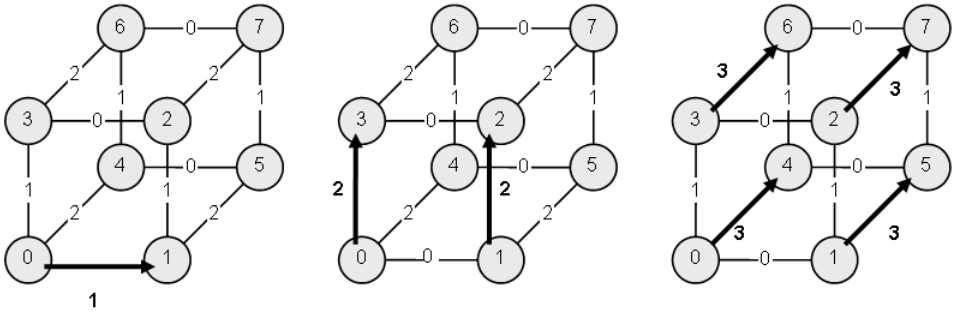

Figure 6: HyperCup message forwarding.

3.2.3.2 Content-addressable networks

If meta-information on a peer's resources is available, this information can be used to organise the network in order to be able to route queries more specifically based on their desired information and carry out more efficient searching.

A very popular technique towards that goal comes in the shape of the so-called distributed hash tables (DHTs): Content descriptors such as file names are hashed, and peers in a P2P network are assigned to cover a particular area of the hash space. Thus, the location of any hashable content in the network can always be deterministically established by and reached from any peer. In essence, DHT-based systems tightly

control both the data placement and overlay topology. Chord [Stoica, 2001], CAN [Ratnasamy, 2001], Kademlia [Maymounkov, 2002] and Pastry [Rowstron, 2001a] are all based on the above concept. A brief analysis of how such systems operate follows, using Chord, one of the most-researched proposals, as its case study[22].

Chord, like all DHT-based systems, provides a distributed lookup protocol, that given a key, maps the key onto a node. Chord assigns keys to nodes using consistent hashing, which has some desirable properties. Firstly, the hash function balances the load (all nodes receive roughly the same number of keys) with high probability. Furthermore, it is also highly probable that when the N^{th} node joins or leaves the network, only a $O(1/N)$ fraction of the keys are moved to a different location.

Consistent hashing in Chord is facilitated using the SHA-1 hash function [Eastlake, 2001]. Each node and key is assigned an m-bit identifier. A node's identifier is obtained by hashing its IP address (called the "node ID") while a key identifier is produced by hashing the key (called the "key ID"). The value of m is large enough to make the probability of two different values hashing to the same identifier negligible. Identifiers are ordered on an identifier circle, called a "Chord ring", of size $modulo(2m)$. Therefore, under Chord there exist $N=2m$ nodes in a 1-dimensional circle. Nodes are mapped on the Chord ring according to their node ID. A key k is assigned to the first node whose identifier is equal to or follows the identifier of k in the identifier space. The latter is called the "successor node" of key k.

Figure 7 illustrates a Chord ring with $m = 6$. Six nodes ($N8$ to $N56$) and four keys ($K24$ to $K54$) exist on the ring. The first key, $K24$, is mapped to node 30, as the successor of identifier 24 is node 30. Similarly, $K28$ is also mapped to node 30. $K37$ is mapped to node 37 since their identifiers are equal. Finally, $K54$ is mapped to node 56, as the successor of identifier 54 is node 56.

There are two basic ways a node can locate a key in Chord. The first is a simple implementation that requires minimum per-node state information but is slow and inefficient. In that, each node only needs to know how to contact its current successor on the Chord ring. Queries for a given identifier are passed around the ring via these

[22] Brief discussions of CAN and Pastry can be found on Appendix C.

successor pointers, until two nodes that are on both sides of the identifier in question are encountered. The second of that pair is the node for which the query is aimed. An example of this technique is illustrated in Figure 8, where node 8 searches for key 54.

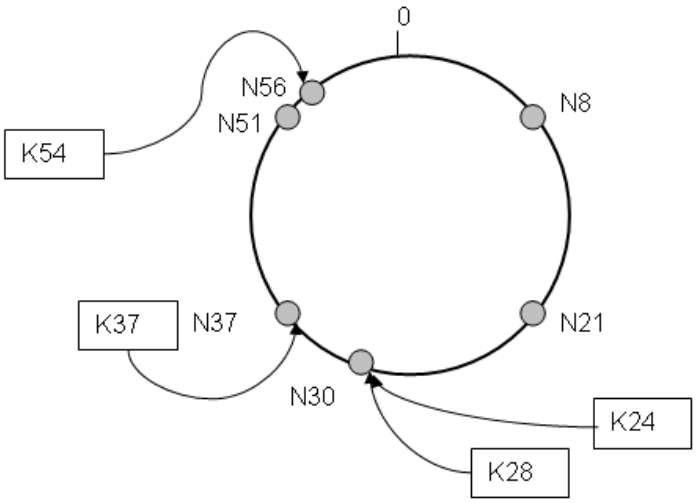

Figure 7: A Chord ring.

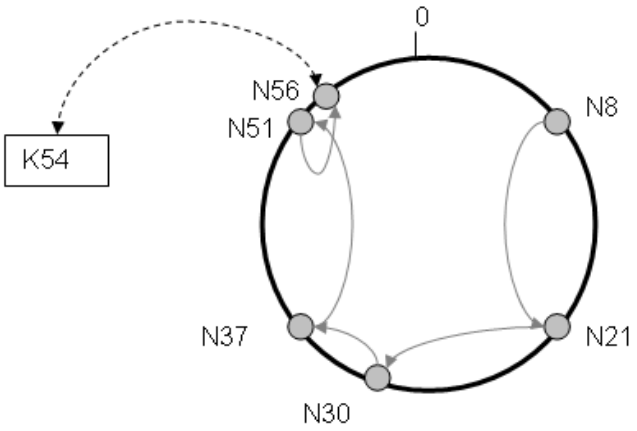

Figure 8: Simple Chord routing.

A scalable key location method also exists, which is a lot more efficient, but requires nodes to maintain additional routing information apart from information on their successor. Examining it goes beyond the scope of this overview, but a brief discussion is presented in Appendix A.

When peers enter or leave the network, Chord attempts to maintain the system in the desired state by rearranging the hash mapping. When a node `n` joins the network, certain keys previously assigned to `n`'s successor are mapped to `n`. Similarly, when a node `n` leaves the network, all its assigned keys are reassigned to its successor. Because

of the dynamic nature of the environment, the Chord protocol includes a stabilisation function that each peer runs periodically to update its successor pointers and finger table. Additionally, each peer attempts to maintain a list of a few of its next immediate successors so that it can still operate if its first successor fails.

When in steady state, for efficient routing in an `N`-node system a node needs to maintain information about `O(logN)` other nodes and resolves all lookups via `O(logN)` messages to other nodes. Its designers claim that Chord can still operate with less information, but with degraded performance.

The modus operandi of most other DHT-based systems is very similar to that of Chord. More specifically, all DHTs base their operation on lookup protocols that can map a key to the peer providing a resource. They dictate which connections peers should form with the rest of the network and which part of the keyspace they are responsible for. Where they mainly differ is in the routing schemes employed and the topological geometries created. For example, Chord and Pastry arrange peers in one-dimensional rings, while CAN uses a virtual d-dimensional Cartesian coordinate space on a d-torus. Other geometries include trees [Plaxton, 1997] and butterflies [Fiat, 2002; Malkhi, 2002]. The significance of geometry lies in the different degrees of flexibility afforded by different geometries [Gummadi, 2003a]. The term flexibility here denotes the algorithmic freedom left after the basic routing geometry is chosen. This freedom is exercised in the selection of neighbours (how the neighbours of a node are picked) and routes (how the next hop when routing a message is picked). This flexibility is important because the ease of selection of routes and neighbours signifies how well the system can respond to an extremely transient environment.

The size of the routing table (or state) versus the number of hops a message needs to travel in the worst case (or equivalently the network diameter) constitutes a fundamental trade-off in DHT-based P2P system design. Smaller search cost can be achieved by maintaining a larger routing table. At an extreme, maintaining state information for every peer in the network would offer the highest possible efficiency (a search cost of `O(1)`). Such an approach, however, would be impractical as the rate of peer arrivals and departures in the network would make the maintenance of the routing table very costly. On the other hand, a minimal routing table would lead to a very large network diameter and resulting search latency. As a result, most existing DHT-based systems (for example Chord, Pastry and Tapestry [Zhao, 2001]) opt for `O(logN)` efficiency and state [Xu, 2003].

CAN is a notable difference to DHTs which settle for $O(logN)$ state and efficiency. In CAN each node maintains $O(d)$ state for d dimensions, and the lookup cost is $O(dN^{1/d})$. Thus, in contrast to Chord or Pastry, the state maintained by a CAN node does not depend on the network size N. If $d=logN$, CAN lookup times and storage needs match Chord's. However, CAN is not designed to vary d as N (and thus $logN$) varies, so this match will only occur for the correct value of N corresponding to the fixed d. That makes CAN more suitable than Chord for dynamic environments, because only a few nodes have to be informed when a node arrives or leaves. However, the trade-off is still present. The smaller routing table makes searches longer, because in CAN the lookup cost increases faster than $logN$.

DHT-based solutions enjoy performance and scalability characteristics that current unstructured decentralized systems cannot reach. Furthermore, they offer search guarantees, since if a key exists in the network, the node responsible for that key can always be found by any other node. Finally, the use of consistent hashing provides an inherent level of load balancing. Nonetheless, few of them have escaped the sphere of proof-of-concept and were deployed as part of real-world P2P services, since some practical issues have not yet been properly addressed. The first is the transient nature of the peer population inside the overlay. Peers come and go at fast and unpredictable rates. Measurements in [Saroiu, 2002] indicate a median uptime for a node of 60 minutes. For a network of 100,000 nodes that implies a churn rate of over 1600 nodes arriving and leaving per minute [Chawathe, 2003]. This rate causes little problem for Gnutella and other unstructured decentralised P2P systems, as long as a peer does not become disconnected by losing all its neighbours simultaneously. Even so, it can re-join the network by repeating the bootstrap procedure at little loss. For DHTs however, these rates impose significant burden. In order to preserve the correctness and efficiency of their structure and routing schemes, most DHTs have recovery and stabilisation algorithms which they run to account for nodes entering and leaving the overlay. Recovery algorithms take time to operate and create overhead. Most DHTs require $O(logN)$ repair operations after each node failure. If the churn rate is too high, the overhead caused by these repair operations can become substantial and cripple the system. Under bandwidth-limited conditions, a positive feedback cycle can occur which overloads the network, causing lookups to have high latency or to return inconsistent results [Rhea, 2004]. The same happens for the stabilisation algorithms. On the other hand, if these repair or stabilisation routines are not run, information consistency is

risked. Information loss is more critical in DHT-based systems than in unstructured networks. In Chord, the ring can become partitioned to smaller sections, whereas in CAN zones can disappear. In the same context, denial of service attacks can be more easily mounted on DHT-based systems by having rogue nodes just cause "failures" fast enough or report incorrect information. Sit and Morris discuss a number of such attacks for DHTs in [Sit, 2002].

Another limitation of current DHT-based P2P protocols is the requirement for exact identifiers when performing queries. DHTs require the exact name of an object or resource, so that they can translate it to a key and perform a lookup. This can be an issue, especially in file-sharing P2P applications. The exact name of a file may not be available in advance, especially since there are no naming conventions or global standards in place. Even when such information is available, however, such as system appears inflexible and unattractive for the majority of users, who are accustomed to using keyword search techniques both in the web and existing unstructured P2P applications. Keyword search is much more powerful, natural and easy to use. Supporting keyword search in DHTs, in contrast with unstructured P2P networks, is a hard challenge. Some approaches have been proposed, such as in [Harren, 2002] and [Shi, 2004], but they add considerable complexity to DHT designs and seem resource-expensive to maintain in the face of the extremely dynamic movement of nodes, and thus content.

In addition, the use of consistent hashing does not entirely solve the problem of optimum load balancing in a P2P network. While schemes based on consistent hashing offer a degree of load balancing, they do not prevent the emergence of "hot spots" and peer load imbalance in general. The random assignment of peer and resource identifiers as implemented in most existing DHTs, does not tackle the non-uniform distribution of objects in the identifier space and high degree of heterogeneity in peer loads and capacities. As a result, a peer may become responsible for a larger part of the key space than average, consequently being assigned a greater number of items and tasked with handling more messages. Additionally, a peer's load may vary greatly over time due to continuous insertions and deletions of objects and arrivals and departures of other peers. Identifying this limitation, alternative algorithms have been proposed, such as in [Byers, 2003] and [Godfrey, 2004], designed for P2P systems. While a step towards the right direction, these approaches introduce overheads themselves and may interfere with the performance or stability of current P2P protocols not designed to accommodate them. As such, further research in the area can definitely be beneficial.

As noted in [Lua, 2005], the mismatch between the underlying network path and the DHT-based overlay path between two peers can result in high lookup latency and could adversely affect the performance of applications running over DHTs. In other words, the deterministic short overlay path of `O(logN)` of a DHT does not necessarily guarantee that high network delay and unnecessary long-distance network traffic will be avoided.

Apart from [Garces-Erice, 2003] and [Ganesan, 2004b], none of the DHT-based systems examined seem to acknowledge or take advantage of the fact that peers vary in capabilities. For that reason, their designs are totally symmetric. However, peers have different capabilities and users stay connected for different periods of time. The level of contribution is also diverse with some peers sharing few resources while others displaying more altruistic behaviour towards the peer community. Overlooking these facts leads to "lowest common denominator" designs which lose some efficiency. As super-peer implementations (discussed shortly) have demonstrated, a level of capacity-awareness can go a long way towards increasing the stability of the overlay while better bandwidth utilisation is achieved.

Finally, lately there is a lot of attention paid to locality and topology-awareness for DHTs in order to improve performance. Almost all proposed systems have either these features directly built in their design (like Pastry and [Garces-Erice, 2003]) or suggest augmentations of their original design to include topology-awareness features with significant performance improvements when compared to their plain versions (for example Vivaldi for Chord [Dabek, 2004]). Other research in this area includes [Zhao, 2002], [Ganesan, 2004a] and [Lua, 2004]. Work in this area indicates that topology-awareness can offer noticeable improvements in routing performance in general. However, such capabilities can enhance any P2P system, not just DHTs. Hierarchical networks, discussed next, are an obvious choice.

3.2.4 Hierarchical P2P networks

Hierarchical P2P networks are essentially multi-tiered topologies with each level of the hierarchy representing a tier. Peers in a tier have additional responsibilities from

peers lower in the hierarchy. In their simplest form, hierarchical P2P networks are two-tiered topologies where the top-tier peers, usually called super-peers[23], have a superset of the functionality of ordinary peers (e.g. [Yang, 2003]). This typically involves assisting in searching, aggregating signalling traffic and keeping the network in an optimal state. For that reason, super-peers are generally peers that enjoy good connection characteristics (high-bandwidth connectivity, large uptimes and high availability) and adequate hardware capacity to support their additional functions. Under this scheme, the rest of the peer population (i.e. peers that are not super-peers) are usually called "leaf nodes".

Conceptually speaking, super-peer networks occupy the middle ground between centralised and entirely symmetric P2P networks: Super-peers play a role similar to that of indexing servers in the centralised P2P topology, handling search requests and other operations for their leaf nodes. Nomination for the role of the super-peer is usually performed in real time, with the P2P protocol containing all the necessary intelligence to dynamically "promote" a capable peer to super-peer status. In systems like Kazaa [Kazaa] and the newer implementations of Gnutella [GDF] (protocol version 0.6 added super-peer support), leaf nodes are only allowed to maintain connections for signalling to super-peers and not between them. A simple illustration of such a super-peer network can be found in Figure 9.

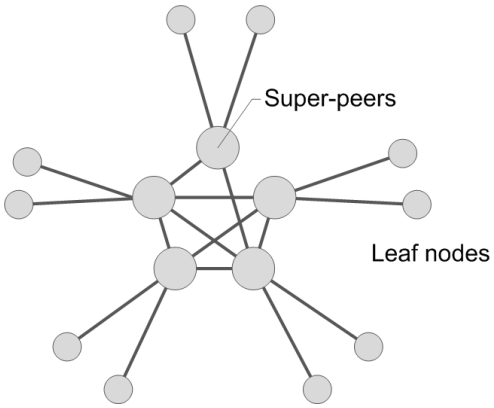

Figure 9: A super-peer network

[23] They can also be encountered as "supernodes" or "ultra-peers" in the relevant literature.

Super-peers index content that is present on the leaf nodes which maintain connections with them. A leaf node searching for some content, sends its query only to the super-peer(s) it is connected to. The super-peer performs a local search of its index to determine whether another of its leaf nodes holds that resource. If not, the query is broadcasted amongst the super-peers, which, if successful, forward it to the appropriate leaf nodes. A successful search is completed with a direct connection between two leaf nodes.

Hierarchical networks allow the formation of different topologies at each level of the hierarchy, creating systems that can potentially combine the strengths of each one. For instance, a tier may form a DHT-based topology taking advantage of search guarantees and lookup efficiency while another may be unstructured for higher fault-tolerance. Hierarchies can also be built to accommodate interest-based or locality-based communities of peers. The use of hierarchy allows for such clustering without sacrificing other desirable network properties like short characteristic path length or fault-tolerance. Furthermore, there is clearer scope for identifying peers' different capacities and exploiting them to a greater extent (for instance as in [Srivatsa, 2004]).

By reducing the number of nodes responsible for message handling and routing, signalling traffic is significantly reduced, and the network becomes more scalable compared to fully-decentralised unstructured P2P networks. Moreover, the search performance of such a network is closer to that of a centralised P2P system while defying the downsides of an entirely centralised system. More specifically, while a search takes $O(N)$ steps in a completely symmetric system of N nodes like early Gnutella, it takes $O(N/M)$ steps in a super-peer network like Kazaa, where M is the average number of peers connected to a super-peer. Thus, while the complexity of search cost is still linear, it has a less steep slope allowing for better scalability and performance. The search cost can be more drastically minimised when the top-level hierarchy uses a DHT.

For unstructured topologies, a super-peer hierarchy brings additional stability to the network. Because the nodes elected to become super-peers tend to stay connected for longer compared to regular peers, the overlay attains a dense core of highly available nodes which supports the bulk of highly volatile peers [Stutzbach, 2005]. By having more stable points of connection for regular peers, the overall stability of the network is improved and each peer departure has a smaller impact on the network. Furthermore,

redundant connections to other super-peers can minimise the effect of super-peer departure or failure.

The biggest issue for hierarchical P2P networks is that by definition they lose symmetry: Some peers in the overlay have additional duties. This raises the questions of how super-peer selection is best carried out, what is the optimal ratio of leaf nodes to super-peers for a particular scenario, what the additional duties mean for the load, performance and operation of the super-peers and what effect does this lack of symmetry has for the fault-tolerance and other properties of the service as a whole. The flexibility in employing and combining different topology models means that answers to such questions are specific to a particular implementation and often can only be provided after the service is deployed. Indeed, several early hierarchical P2P protocols relied on heuristic values for triggering super-peer promotion and maintaining a leaf node per super-peer ratio that were adjusted manually as the network evolved, instead of employing a purpose-built mathematical model (for example as described in [Kleis, 2005]).

Table 1, provides a comparison of three popular super-peer networks[24]. It becomes clear that each network builds its super-peer hierarchy differently to better address size and service requirements.

	Gnutella (ver. 0.6)	**FastTrack (Kazaa)**	**Overnet (eDonkey)**
Super-peers	10,000 - 100,000	5,000 - 40,000	20 - 40
Leaf nodes	≈ 1 million	≈ 2 millions	≈ 1.5-2 millions
Average leaf-to-SP connection duration	≈ 93 minutes	≈ 34 minutes	> 24 hours
Super-peer promotion mechanism	Protocol election	Protocol election	Voluntary installation of server software

Table 1: Comparison of three popular super-peer networks.

[24] The information contained in the table was compiled from [Liang, 2004], [Loo, 2004] and statistics obtained from www.slyck.com, http://ocbmaurice.no-ip.org/index.html and the eMule client (collected on 1/9/2005).

Hierarchical P2P networks have been created as a response to the poor scalability and performance of their early unstructured counterparts. At the moment, the majority of P2P services deployed in the "real world" are based on super-peer topologies, displaying that this approach presents a desirable trade-off between performance, resilience and complexity. However, current super-peer topologies demonstrate only part of the flexibility offered by deploying multi-tiered overlays. The ability to combine different topologies in each tier allows for customised solutions that better fit the diverse requirements modern P2P services may have. Systems can be built that are capability-aware, locality-aware or interest-based while maintaining the necessary flexibility, characteristic of P2P networking.

3.3 Peer behaviour characteristics

So far, this chapter focused on protocol-specific aspects of P2P services and especially how the adopted overlay topologies affect functions like searching and routing. Equally important is the examination of individual peer behaviour; in particular because it is dependent on a large number of parameters external to the protocol. The user is perhaps the greatest. The user influences peer behaviour to a very large extent, especially regarding peer uptime, query workload and resource contribution. Peer uptime signifies availability and participation in the provision of the service to other peers. Small uptime results in higher churn rate and overlay volatility which affect the overall quality of service. Query workload describes the number, rate, type and other characteristics of queries made. Finally, the contributed resources encompass everything from the number, volume, heterogeneity and popularity of shared resources to the level of cooperation in making these resources available to other peers (i.e. whether the user shows altruistic or selfish behaviour). Quantifying and measuring these characteristics is crucial to understanding the macroscopic behaviour of P2P systems, not only so that better P2P services can be built but also to provide the necessary direction on how the infrastructure needs to evolve towards more graceful coexistence of P2P and other network services. Specifically for this thesis, an examination of such characteristics is necessary for the realistic simulation of peer behaviour discussed in the following chapters.

3.3.1 Free-riding

The level of participation of peers in sharing resources is fundamental to the value one derives from a P2P service and, ultimately, its success. When peers contribute little, the overall wealth of pooled resources diminishes and so does the perceived value of the service. In this case, due to the network effect, users may choose to migrate to a similar service with a higher degree of participation, which in turn will decrease the original service's value further. Nevertheless, behaviour where users choose to share disproportionately little to what they consume is common in P2P services, and particularly file-sharing services. Called "free-riding" in reference to similar behaviour observed in society and linked to the "tragedy of the commons" [Hardin, 1968], it can be attributed to several factors. In the context of P2P, the most prominent is the scarcity of individual peer resources (e.g. disk space, bandwidth, etc) which leads self-interested users to limit their contribution to the community for personal benefit. For instance, due to the asymmetry of residential connectivity and the nature of TCP, limiting the upstream capacity allocated to a P2P application allows less congestion and delays for ACK packets to be transmitted and maximises downlink performance.

Free-riding has been the subject of several studies. In 2000, Adar and Huberman [Adar, 2000] reported that 66% of Gnutella peers were free-riders. Three years later, a similar study on Gnutella found the percentage of free-riders to be 25% [Saroiu, 2003]. More recently, measurements in [Stutzbach, 2007] reported the percentage of free-riders to be closer to 13% of the population. This measurement regime however classified as free-riders those peers which did not provide their sharing list to the crawler[25], thus excluding "easy-riders" (peers who contribute until they fully download a file, after which they stop sharing it [Figueiredo, 2004]) or peers which did not satisfy such a request due to privacy concerns[26]. Consequently, it is probable that the actual free-rider population is higher than reported, especially if "easy-riding" behaviour is considered

[25] A crawler is an application speaking the P2P protocol of interest, used to collect real-time data about a P2P system by connecting successively ("crawling") to all available peers.

[26] A number of applications offer this option as a countermeasure to copyright enforcement agencies using this method to collect information on unauthorised sharing of copyrighted content in P2P networks.

equally abusive to the system. Finally, Fessant et al [Fessant, 2004] observed that 68% of peers were free-riding in eDonkey.

Systems like Gnutella and eDonkey offer no incentives or policing for sharing, making free-riding easy to accomplish. Newer systems take explicit steps to address this issue by incorporating incentive schemes, or algorithms that regulate individual performance according to the level of contribution at the protocol level. Such schemes utilise economic or game-theoretical concepts to present incentives to the user to share more, or directly limit the performance, or even participation, of peers who consume more resources than they contribute. The BitTorrent system developed an interesting amalgamation of both concepts. At the protocol level, BitTorrent incorporates a "tit for tat" algorithm which regulates individual download performance according to the peer's upload performance [Izal, 2004]. This ensures that peers who upload at slow speeds or do not upload at all will not be able to achieve high download performance from other peers. In addition, separately to the protocol, many trackers incorporate mechanisms to monitor and record a peer's upload-to-download ratio and penalise or altogether deny service to those peers who do not maintain an acceptable level of contribution to the community. This mechanism operates on a different timescale to the "tit for tat" algorithm as it judges a peer according to its overall contribution to the community over many torrents and not that of individual sessions. Moreover, this policing is done by the tracker and not the protocol, allowing an extra layer of resistance against modified clients (for instance as described in [Liogkas, 2006]). The combination of these two mechanisms ensures acceptable peer contribution at different degrees of granularity.

In any case, free-riding is a reality that must be taken into account both when designing P2P services and when simulating P2P protocols and peer behaviour. Users will seek to maximise the value they derive from a service and may engage in selfish behaviour if the gains from such behaviour justify the costs. Free-riding is most prominent in systems where fair participation is not actively pursued or rewarded. The lack of mechanisms to ensure fair participation makes free-riding readily achievable at no obvious disadvantage (cost) to the user. P2P systems which incorporate such mechanisms essentially remove any gains arising from free-riding by either rewarding participation or by making it costly to engage in selfish behaviour.

3.3.2 Peer availability and churn

Peer availability describes the capacity of peers to be present and accessible so that they can carry out their part of service provision. Peer availability is affected by short term (e.g. packet loss) or longer term (e.g. software/hardware crash) transient effects or even one-off events such as permanent departure from the service [Bhagwan, 2003]. Churn, on the other hand, describes the continuous variation of the group of participating peers due to arrivals and departures. Availability can be characterised by measuring a peer's session duration (uptime) against its downtime. A session describes the time a peer stays connected to the overlay between its arrival and departure. It is expected that a peer will have many sessions over its lifetime. As a result, churn can also be quantified by measuring arrival and departure events.

Churn and availability are important system characteristics due to the effect they have on the performance, stability and overall quality of service. High overlay dynamics lead to increased search cost and necessitate more signalling traffic while diminishing search performance and resource stability. DHT-based systems are particularly vulnerable since they need to maintain a structured topology, but unstructured systems suffer from the effects of high churn rate as well. Coping with churn and ensuring graceful operation in the face of frequent peer unavailability is central to the design of robust P2P systems. Towards that, it is valuable to examine these quantities in existing P2P networks.

A number of studies have offered measurements of churn and availability in deployed P2P networks. Saroiu et al. [Saroiu, 2002] used a crawler to measure peer availability in Gnutella and Napster and compared it to the actual host availability at the IP level. Their findings suggested that 80% of the peers in Gnutella were available for less than 45% of the host machine's uptime while in Napster that percentage was closer to 83%. Furthermore, the median of session duration was for both Gnutella and Napster approximately 60 minutes. Chu et al. [Chu, 2002] did a similar study on Gnutella and reported that peer availability is influenced by the time of day in their geographical location. Their session duration measurements agreed with [Saroiu, 2002]. In [Sen, 2002] Sen and Wang used offline analysis of flow-level data collected passively from multiple routers across a large tier-1 ISP backbone. They reported that 60% of peers in the FastTrack network (Kazaa) stay connected for 10 minutes or less while 20% of the overlay connections last for a minute or less. They also confirmed the correlation between time of day and peer availability. Qiao and Bustamante [Qiao, 2006] reported

that in the Gnutella network 50% of all peers have a session length smaller than 4,300 seconds, and 80% have session lengths smaller than 13,400 seconds (less than 4 hours). Only 2.5% of the session lengths measured were longer than one day. Similarly, for the Kademlia-based Overnet network (eDonkey) the median session length was around 8,100 seconds, 80% of the peers' sessions lasted less than 29,700 seconds, and only 2.7% of all session lengths lasted more than a day. In [Bhagwan, 2003] the authors made similar measurements which also demonstrated time-of-day effects in peer availability. They reported that on average a peer joins and leaves the network 6.4 times a day. They also approached peer availability by taking into account both short-term (daily) and long-term (permanent) arrivals and departures of peers. Their short-term results were compatible with these discussed above. Regarding long-term availability they suggested that over 20% of peers arrive for the first time or leave permanently every day.

Due to the decentralised nature of P2P networks, measuring churn is not a precise process. The use of network crawlers, necessary due to the lack of a global vantage point, precludes the generation of an accurate snapshot of the global overlay; as by the time even the fastest crawler completes its crawl, a number of peers have left or joined the network. Also, otherwise available peers may be unreachable to the crawler because they are behind NATs or refuse crawler connections. More crucially, the measurement strategy followed may introduce bias. Stutzbach and Rejaie [Stutzbach, 2006], identify peer sample selection as applied by [Saroiu, 2002] and [Chu, 2002] and length of measurement window (i.e. how sessions longer than the measurement period are accounted for) as two areas where bias can be introduced.

Consequently, the findings presented in the aforementioned studies should not be considered highly precise, but rather indicative of the scale of dynamics in a constantly changing and evolving system. Nevertheless, certain observations can be drawn from where these studies agree, especially where different measurement regimes were used. Firstly, median peer session times for all the networks studied are in the order of tens of minutes, increasing slightly in newer studies. This increase can be attributed to the greater penetration of broadband connectivity and the employment of incentive schemes by newer systems as discussed in the previous section. Median availability is around 30% with less than 3% of peers staying available for more than a day. In all studies a number of peers have relatively long session times. A large part of the population, however, joins the network for a few minutes only, attributing to the low median figures. In general, these data suggest high overall churn rate for all networks.

Statistical characterisation of peer session length and peer inter-arrival times is particularly important for P2P network simulation. The appropriate distributions to match the real-life peer-arrival and peer session durations patterns for use in simulation have been the subject of much debate. A number of studies have assumed both distributions to be exponential [Liben-Nowell, 2002; Rhea, 2004], as traditionally done when modelling independent events occurring at a constant average rate. Stutzbach and Rejaie [Stutzbach, 2006] found the use of Weibull or log-normal distributions more fitting to the set of data they collected. The majority of studies, however, consider the session length distribution as Pareto (heavy-tailed) [Bustamante, 2003; Leonard, 2005; Wang, 2007]. The results presented in [Gummadi, 2003b] and [Sen, 2002] mentioned earlier, also point towards a Pareto distribution. Given these results, modelling session length using a Pareto distribution and peer arrival using an exponential distribution (as in [Qiao, 2004]) appears to be the most appropriate route.

The observation that there is strong correlation between peer availability and the local time of day suggests that peer availability is affected by the daily schedule of users. The period of lowest availability in a time-zone is observed between early morning and early evening – typically during normal working hours [Klemm, 2004]. Depending on the size and scope of a network this characteristic can have significant consequences. At the time of lowest availability, small networks with strong peer locality may experience a discernible reduction of the available pool of resources and inadequately replicated resources (such as files) may become temporarily unavailable. Distributed storage systems or databases, which are vulnerable to such a scenario, may thus need to take this issue into account and explicitly replicate objects to geographically-disparate peers to ensure availability at all times. In general, however, networks where peers extend to different time-zones or enjoy adequate participation at non-peak times are not susceptible to such effects.

In conclusion, high churn rates are evident in all current P2P networks, regardless of protocol. Peers with very short session times have the largest effect on overlay dynamics. While P2P services are built on the premise that peers will have short and unpredictable lifecycles, high levels of churn are harmful as they reduce performance and affect their stability. The proliferation of broadband connectivity alone cannot be expected to lessen the rate of churn drastically. In fact, faster connectivity may lead to an increase in churn as tasks can be completed in shorter sessions, allowing self-interested users to leave the overlay sooner. As such, mechanisms to manage churn and minimise its impact on basic service functions need to be at the core of P2P service

development and not be added as an afterthought. This is particularly important for many DHT-based systems which although tuned to perform well in ideal conditions, suffer when deployed in actual networks.

3.3.3 Content-related characteristics

The exchange of content comprises one of the most popular application domains where P2P networking is applied. The unique impact on the network caused by the use of content-sharing P2P applications by a sizeable part of the public[27] makes the examination of workload characteristics necessary for the comprehensive understanding of how such services behave. The analysis of volume, types and popularity of shared resources as well as how these attributes vary with time stimulated by user behaviour, can provide invaluable insights on how to build more efficient file-sharing infrastructures, while being critical for their realistic simulation.

In P2P networks, content is generally made available in the form of distinct objects (files) shared by each peer. As such, P2P-accessible content is immutable. Because of that immutability, a peer needs to obtain a particular file only once. In contrast, web-based content is highly dynamic and may be updated, modified or personalised. As a result, it is natural for web pages (for instance of news sites, etc) to be fetched multiple times per client. As pointed out by Gummadi et al [Gummadi, 2003b], unlike the Web whose workload is driven by document change, the primary forces in P2P file-sharing networks are the creation of new objects and the addition of new users. The P2P workloads are also typically larger than these of the web. A download of a large file in a P2P network may take multiple sessions to complete. Finally, the longer download times make user cancellation of P2P downloads before they are completed more common compared to web content.

These differences between web and P2P workloads mean that concepts extensively researched for the former cannot be unreservedly applied to P2P file-sharing networks. The distribution of file popularity is one such area. While it is common

[27] The three most popular P2P file-sharing networks (eDonkey, Gnutella and FastTrack) claimed an approximate total of 7.5 million users in 2005 (source: www.slyck.com statistics 01/09/2005).

practice to use a Zipf distribution[28] to represent object popularity on the web (e.g. [Breslau, 1999]), a number of studies deem it unsuitable for P2P workloads. Gummadi et al [Gummadi, 2003b] examined a 200-day trace of Kazaa traffic and observed that the file popularity distribution was not accurately represented by a Zipf distribution, especially for the most popular files. Instead, the graphical representation of the distribution curve produced using their collected data demonstrated a considerably flattened "head". The consequence of that observation is that the most popular objects in a P2P network are significantly less popular than a Zipf distribution would predict. In other words, a Zipf distribution would overestimate the popularity of the most popular files. That difference was attributed to the "fetch-at-most-once" behaviour of peers and large object sizes in contrast with the web. The large size and resulting long time to download, in particular, motivates users to decide whether they are really interested in a file before downloading it. The same observations on the flattened head of the distribution were made by Klemm et al [Klemm, 2004] and Saleh and Hefeeda [Saleh, 2006]. The former combined two Zipf distributions with different parameters to best fit the two parts of the curve (main body and tail). The latter proposed a Mandelbrot-Zipf distribution. Chu et al [Chu, 2002] similarly noted that Zipf did not provide an accurate representation of their data on Gnutella. They found that a log-quadratic distribution fitted better. However, they did not justify why that occurred or whether log-quadratic distributions should be appropriate for characterising file popularity in P2P networks.

These studies show that in P2P networks a few files are highly popular but there exists a long tail of unpopular files. This observation is valid both when examining popularity by measuring instances of files (replication) and numbers of queries [Fessant, 2004]. File replication per peer is reported to be highly skewed but does not follow a power law [Stutzbach, 2007]. Specifically, according to the aforementioned study most peers share a moderate amount of files (or bytes) while a few peers contribute an enormous amount. The median value for shared files is around 70 files while 0.01% of

[28] Zipf's law (named after linguist George K. Zipf) states that the frequency of the *i*th-most popular object is proportional to $1/i^s$, where *s* is the "Zipf coefficient" characterising the distribution. In the context of object popularity a Zipf distribution means that a small number of objects are extremely popular, but there is a long tail of unpopular ones.

peers share more than 7,500 files. The median value of shared space is around 650 MB while 0.1% of peers contribute more than 85 GB.

In the same study the authors reported that 60% of all files in Gnutella are a few megabytes in size (between 1 and 10 MB). Audio and video files collectively constituted more than 73% of all files in the system, occupying more than 93% of the aggregate capacity (bytes). The audio files accounted for 67% of files and 40% of bytes while video files constituted around 6% of files but 52.5% of bytes (due to size). In comparison, an analysis of Gnutella in 2002 [Chu, 2002] reported that audio files constituted 67.2% of files and 79.2% of bytes and video files amounted to 2.1% of files and 19.1% of bytes. It can be assumed that file-sharing networks move towards larger workloads.

It has to be noted that advances in storage technology and broadband connectivity penetration as well as the appearance of more advanced file-sharing systems will influence user behaviour and thus both workload and traffic volumes. The aforementioned reports therefore are relevant for the time frame and protocol they examined. Nevertheless, they provide valuable information on workload characteristics as well as user trends.

3.4 Summary and conclusions

Having examined the basic types of P2P systems, it becomes apparent that two predominant architectural routes exist: systems that form unstructured and systems that form structured overlay topologies. Their differences are fundamental: In the former, no control whatsoever is imposed on the way peers are interconnected, ultimately resulting in a topology of no discernible structure. Structured systems, on the other hand, have their peers form deterministic topologies, where each peer is attached to a particular point in the overlay. This way, the location of any peer or resource can be determined in a bounded number of steps.

Unstructured systems are characterised by their simplicity and inherent self-organisation features, but at the same time cannot scale efficiently to accommodate large numbers of peers. Structured systems in comparison enjoy superior scalability and performance and offer search guarantees. However, they cannot handle the dynamic nature of peers as well and in many applications require more complex mechanisms for translating and mapping a user query to a resource.

A lot of work is devoted to addressing the shortcomings of each side. Better search techniques, for instance, improve the search capabilities of unstructured networks. Similarly, better geometries make DHT-based systems more resilient to node failures. Such efforts, however, remain largely protocol-centric and rarely consider how the resulting service will affect the network ecosystem. The manageability of P2P services is equally important no matter which approach is taken to construct the overlay. The next chapter presents the proposed framework for the management and control of P2P overlays, aiming to address the broader issues arising from their use of network resources, while helping them operate at their maximum potential.

4. THE ACTIVE VIRTUAL PEER

4.1 Introduction

As discussed in the previous chapters, a key issue with current P2P architectures is that while they enjoy desirable self-organising properties and can be deployed with very little effort and support, they consume resources inefficiently and unfairly, starving other network services from bandwidth, conflicting with applied network engineering functions and traffic cost optimisations and even suffer from degraded performance themselves. The near-collapse of the early Gnutella network because of poor scaling [Ritter, 2000], helped identify two important issues: Firstly, that peers have different capabilities and should be treated as such [Saroiu, 2002], and secondly, that the introduction of structure and limited control inside a P2P network can be valuable [Lv, 2002a].

In order to exploit heterogeneity into unstructured, decentralised P2P services, the introduction of hierarchy has been proposed. Super-peers and distributed mediation servers, as in later versions of Gnutella [Yang, 2003], Kazaa [Kazaa] and eDonkey2000 [eDonkey] are based on such a concept. Others suggest the introduction of structure in the overlay, in order to improve performance and scalability by having a deterministic view of the network. DHT-based topologies represent the latter concept.

These approaches, however, comprise only partial solutions to a more complex control problem. In particular, variability in service demand or load patterns can only be dealt with in a limited way. The demand for services may form hotspots which may shift within an overlay from one location to another over time. Furthermore, these protocol-specific designs rarely adopt a holistic view of the network which would take into account co-existence with other network services and the needs of network providers. In particular for the latter, the obliviousness of most popular P2P applications to the underlying network infrastructure cannot be dealt with by structure or hierarchy alone. The diversity and distributed nature of P2P protocols is in itself an obstacle to addressing these issues satisfactorily or in a consistent manner. P2P applications

therefore require a more flexible and dynamic method of control and management. In particular, different control methods should be in place when and where needed, should be usable in combination with each other as flexibly as possible, and should be extensible in an evolutionary manner. In essence, the goal is to introduce and implement control and structure into P2P services on demand. But first, the nature and amount of control has to be identified.

4.2 Objectives and requirements of P2P overlay management

Acknowledging the nature and unique characteristics of P2P applications, it is important to examine how adaptive and un-supervised control mechanisms can be implemented without diminishing the virtues of the P2P model or introducing further complexity and overhead to the network. It is vital that control does not interfere with the autonomy of peers, impose obstacles to carrying out normal protocol functions or affect their performance negatively. Too much control may limit P2P application usability, causing user dissatisfaction and prompting developers to adopt evasive tactics. In order to be effective therefore, any control mechanisms must first and foremost understand and be compatible to the way P2P applications operate.

The work presented in this thesis identifies as the primary causes behind the inefficient and unfair utilisation of network resources by P2P applications the separation and mismatch between P2P overlay and the network layer, the short and unpredictable lifecycles of peer relations and the inability to distinguish peers in terms of their individual capabilities, behaviour and motivation. Starting from there, four areas where control may be beneficially applied have been identified.

The first is *access control*. Participants of P2P overlays can typically connect to any peer present and are granted access to all resources offered by other peers. Such unregulated access leads to inefficient operation and mismatching of overlay and underlay. Thus, the resource provider (network provider or content provider) needs to regulate the admission to the overlay. For instance, through access control the resources certain peers may access may be specified or sets of peers may be prevented from accessing them altogether. Additionally, access control can be the basis for building more complex management functions.

The second area is *resource management*. The resources of individual peers are both limited and valuable and need to be treated with care. For instance, low-bandwidth or lightweight peers (e.g. mobile devices) should not be overloaded with requests and

exploited equally compared with their better-connected counterparts. This includes network resources which are limited and need to be shared fairly. Resource management has the task of protecting both peer-owned and network resources from inefficient uses.

A third area of importance is traffic load. Overlay *load control* is concerned with the characteristics of traffic flows inside the overlay and their impact on both overlay and underlay. Its goal is to balance the load of overlay connections and map it optimally onto the underlying network infrastructure in order to maintain sufficient throughput for peers while also protecting other network services and uses.

Finally, the fourth area of beneficial control is adaptive *topology control*. Overlay connections may be established or destroyed arbitrarily by peers since they can join or leave the network at any time. Topology control may enforce redundant connections and create alternative overlay paths, thus increasing the reliability of the service. In addition, topology control may influence the structure of the overlay to increase efficiency of broadcast traffic and, in combination with other forms of control, alleviate temporary hotspots of demand.

The identified areas support the aim of having adaptive and application-suited, management strategies for P2P services. The outlined control objectives might at first appear to violate the populist concept of unlimited access to free resources in P2P services, but if properly used, control mechanisms governed by these objectives can increase the stability of P2P services based on overlays. The rest of this chapter describes how the proposed management architecture applies these forms of control in a way that guarantees application autonomy and unrestrained operation.

4.3 The Active Virtual Peer concept

The support infrastructure presented in this thesis comes in the form of a framework which facilitates a flexible and adaptive mode of P2P service and overlay management. Its main element is the Active Virtual Peer (AVP). As its name implies, an AVP is a virtual entity which interacts with other peers inside a P2P overlay. To these peers, an AVP is viewed as another peer, able to communicate with them via the P2P protocol in use. Its purpose is two-fold: the AVP offers the means for management of the P2P overlay, while enhancing the P2P service experience of its users. By inserting an AVP inside the P2P network an ISP can manage aspects of the service's behaviour in a manner that is transparent to the rest of the peers while having an

"insider's" view of it at the application level. At the same time, the AVP will attempt to make the P2P service operation more efficient by leveraging its broader view of the network, unique knowledge of the network infrastructure available only to the ISP, high-speed connectivity and high availability[29] to serve other peers in its community. Being part of the P2P network, the AVP can enjoy the advantages of flexibility and self-organisation inherent to P2P networking. Additionally, an AVP can perform functions not expected by an ordinary peer. An AVP will use the P2P protocol messages it collects to deduce real-time information about the state of the overlay and its peers and may create or modify such messages in order to affect their behaviour.

An AVP consists of various distributed and coordinated components that facilitate different forms of management. By combining these components based on network conditions or administrative policies, AVPs of different scope or functionality can be created. The AVP architecture is P2P-based itself. It is envisaged that ISPs will deploy a number of AVPs throughout their networks, according to their needs. These AVPs will self-organise in a group and exchange information about peer behaviour in their vicinity that may not be visible from a single vantage point. Furthermore, in true P2P fashion, a group of AVPs will collaborate with each other to enlarge the scope and effectiveness of their operation and carry out tasks collectively.

The AVP functionality is separated into three basic components. These AVP components are depicted in Figure 10.

The main component of the AVP architecture is the *Application Optimisation Component* (AOC). The AOC contains the core functionality of the AVP and as such comprises the minimum configuration an AVP may have. Its task is to monitor, control and optimise the state of the P2P overlay at the application level. Towards that, the AOC captures, examines and modifies P2P protocol packets, creates and destroys connections with other peers and applies application-specific routing in conjunction with any access control policies in place. The routing performed by the AOC may be in response of inferred peer state or link state measurements, thus changing the load imposed on peers by protocol functions and overlay link characteristics like throughput

[29] It is assumed that AVPs, being part of the management infrastructure of a provider, will be run on adequately provisioned hardware and have high-speed connectivity.

or delay. The AOC allows for active overlay topology control, which is accomplished in two ways: The Active Virtual Peer may initiate, accept or terminate overlay connections based on access restrictions or topology features. Topology characteristics such as the number of available alternative overlay paths or characteristic path length may be enforced on the overlay structure. Furthermore, the AOC component makes use of the ALAN (Application-Level Active Networking) control mechanisms, examined shortly, for implementing its self-organisation features. The AOC can instantiate modules implementing AOC or other AVP functions whenever and wherever[30] needed. These features enable the AOC to adapt the virtual overlay structure to varying demand, traffic patterns and connectivity requirements by launching new overlay connections and new virtual peers. These self-organisation features offer flexibility to the AVP to respond to the ever-changing conditions inside a P2P overlay.

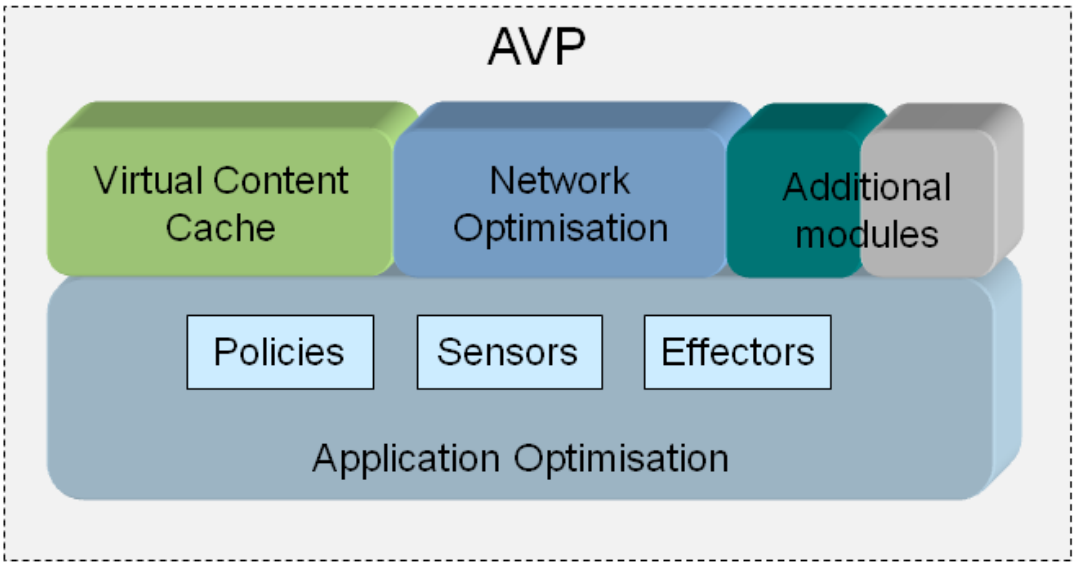

Figure 10: The AVP component architecture.

Traffic management by the AVP is assisted by another component, called the *Virtual Content Cache* (VCC). The VCC provides P2P content caching capabilities at the application-level. By maintaining often-requested content in close proximity, and in

[30] The ability of AVP components to be instantiated on different parts of the overlay depends on the numbers and locations of the execution environments an ISP may provide. In that context "wherever" means "in any of the available locations".

particular inside an ISP's domain, large economies in resources and performance gains can be achieved. The design and implementation of the VCC are discussed in depth in the next chapter.

Finally, the capabilities of the AVP architecture can be augmented with the addition of the *Network Optimisation Component* (NOC). The role of this component is to provide the AVP with dynamic traffic engineering capabilities which will map the P2P overlay traffic onto the actual network layer much more effectively than current systems do on their own. A discussion of the NOC is presented later in the chapter.

It is envisaged that the AVP architecture presented here will be extended with other components that can run on top of the AOC and complement the architecture with additional functionality. Such components may, for example, contain service-specific functionality to accommodate IM or P2P video streaming services. The AVP components themselves are developed in a modular way, so that multiple P2P protocols can be incorporated with little or no changes to the rest of the software.

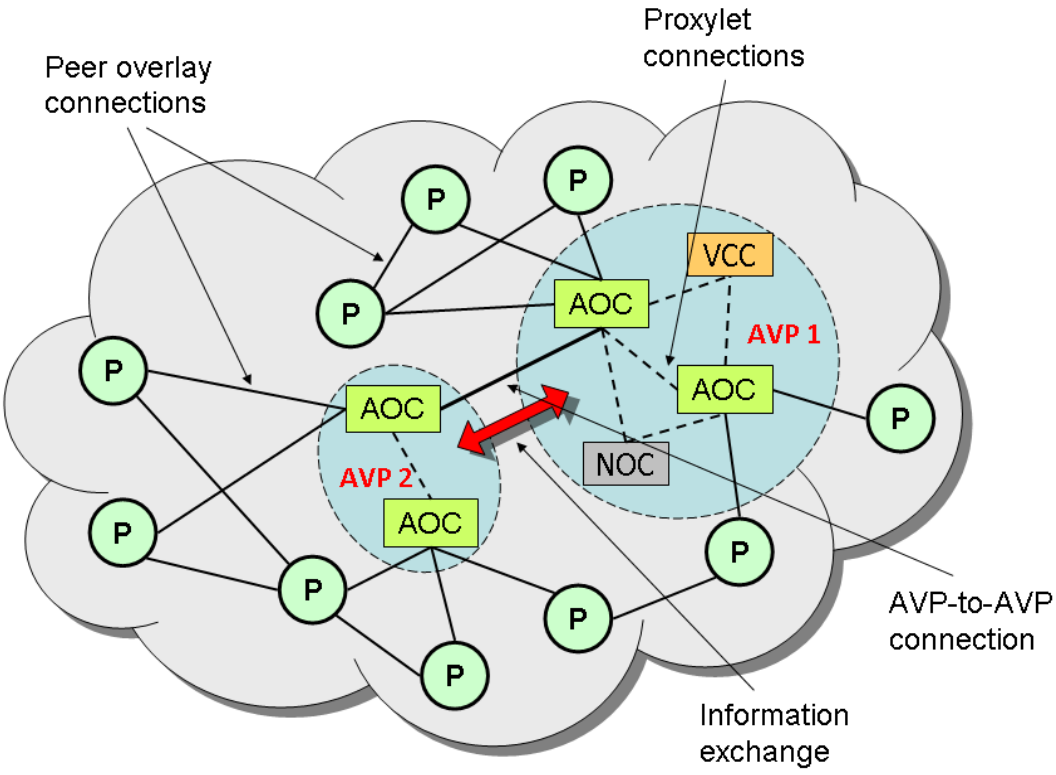

Figure 11: An example of an AVP deployment.

Figure 11 depicts a scenario where two AVPs, AVP 1 and AVP 2, are located within a single administrative domain (e.g. within an AS). AVP 1 consists of two AOC, one VCC and one NOC component, while AVP 2 comprises of two AOC components.

Multiple ordinary peers, denoted by "P", maintain connections to them and to each other. The two AVPs maintain overlay connections with each other.

The AVP architecture was designed to meet a number of objectives deemed vital for the applicability and success of the concept. These are:

- Transparency to P2P applications: Any solution requiring the active cooperation of P2P applications involves their modification, restricting its applicability to only those applications whose developers agree to incorporate the necessary features. Establishing communication channels and convincing developers is a major task, especially if the latter have concerns about performance degradation or feature control of their applications. The AVP design, therefore, was dictated by the need to operate without the active knowledge and cooperation of peers.
- Compatibility to P2P philosophy: Effective management should originate from understanding how P2P networks operate, why they carry out certain functions the way they do and from respecting the desire of their users/ISP customers to run such applications without obstacles. Specifically, the management framework must avoid imposing control measures that affect the connectivity or degrade the search and download performance of peers. Moreover the system must be compatible to P2P timescales of operation, churn, and decentralised communication.
- Improvement of P2P application performance: The capability to interact with P2P applications and control aspects of their behaviour gives an ISP the opportunity to leverage its role and unique knowledge of the network to improve application performance, especially since the difficulty of applications to infer such information on their own leads to inefficient decisions. Furthermore, it strengthens compatibility with the P2P concept and serves as proof to both customers and developers that the ISP does not apply management to restrict P2P usage but rather to streamline utilisation and protect network resources. This creates a win-win situation where both the ISP and the end-users benefit and ensures the acceptance of the management framework by the latter.
- Scalability: P2P usage is already very substantial and appears to be steadily increasing [Cho, 2006]. Therefore the system must be able to scale with increasing numbers of peers without performance degradation that may affect its effectiveness.

- Manageability: Given the dynamic nature and complexity of P2P overlays, the system should not require constant supervision by a human operator, should be largely autonomous and easily manageable.
- Extensibility: The design of the AVP should allow the system to be extended easily, both in terms of deployment size and in terms of capabilities.

4.4 Implementation of the AVP with ALAN

The proposed implementation of the AVP is based on the Application Level Active Networking (ALAN) concept [Fry, 1999; Ghosh, 2000; Ghosh, 2001][31]. ALAN enables the dynamic deployment of active services in the network, but at the application rather than the router level (as in [Tenenhouse, 1996]). The aim of such an approach is to realise the advantages of active networking (the rapid creation of intelligent, personalised services), without facing the disadvantages of router-level implementation (implementation complexities, deployment costs, security issues, disinclination by the providers to change something that "works" etc). The ALAN infrastructure offers rapid deployment of network services and their on-demand provision to specified users or communities. ALAN is based on an overlay technique: Active nodes, which operate on the application level, are strategically placed within the network. These nodes, called Execution Environments for Proxylets (EEPs), enable the dynamic loading and execution of active code elements, called "proxylets", from designated proxylet repositories. The resulting services may interfere with data transport and control. ALAN provides mechanisms for dynamic EEP discovery, application specific routing, and service creation by deploying a web of proxylets across the physical infrastructure. The "Self Organizing Application-level Routing" (SOAR) protocol [Ghosh, 2000], which is a key component of ALAN, enables clustering and grouping of proxylets. This way, ALAN facilitates the creation of an application-specific connectivity mesh and the

[31] ALAN comprises work carried out as part of the ALPINE (Application Level Programmable Inter-Network Environment) project. ALPINE was funded by BT Labs and was a collaboration between University of Technology-Sydney, University College London, Imperial College London, University of Lancaster and University of Sussex.

dynamic forming of topology regions. Finally, ALAN provides the basic administrative mechanisms necessary for managing such an architecture.

ALAN allows new services to be deployed in "the wild" and their effectiveness tested without compromising any existing network architectures. Moreover, infrastructures which interact with the application layer such as the AVP concept are better suited to be provided on the application layer itself, rather than involving lower layers, as claimed by the end-to-end argument [Saltzer, 1984; Reed, 1998]. Finally, the use of ALAN allows for AVP components to self-organise and be deployed when and where needed.

An ALAN implementation of the EEP environment along with a control and a monitoring interface are available under the name "funnelWeb". Proxylets for the funnelWeb implementation are written in the Java programming language [Arnold, 1996] to take advantage of the language's portability and security model properties. Proxylets are packaged in the form of single Java archive (JAR) files and can be stored in normal web-servers. These web servers need no modification to serve as proxylet repositories, and as a result proxylets can be easily referenced via URLs (Uniform Resource Locators). Proxylets are loaded by reference, meaning that EEPs do not need to store copies. This makes the management of different versions of proxylets easy, as only the original copy of each proxylet needs to be controlled. The control interface available with funnelWeb (depicted in Figure 12) can be used to load, run, stop and pass parameters to a proxylet. The monitor interface provides status information for the proxylets that run on a particular EEP.

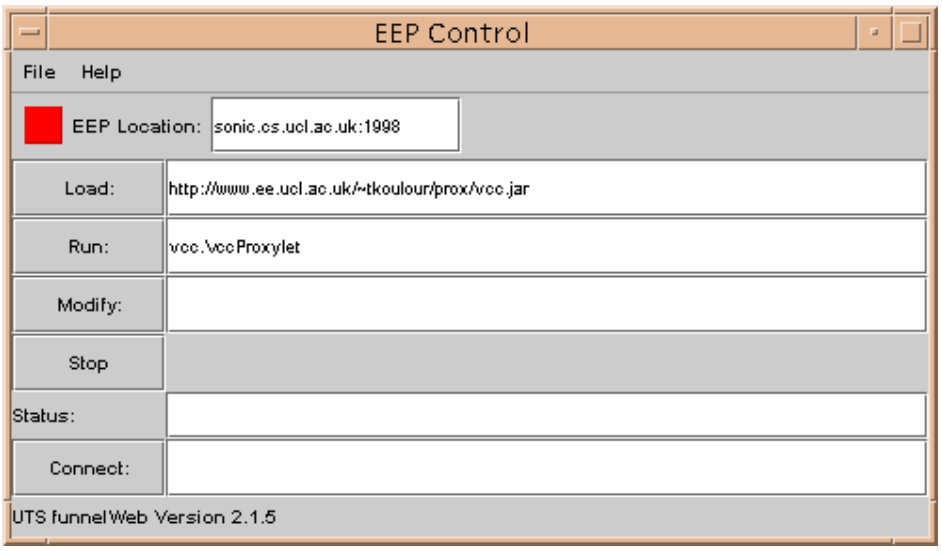

Figure 12: The funnelWeb EEP control interface.

The AVP components are implemented by single or multiple interconnected proxylets. This allows the implementation of the AVP architecture in separate components. For instance, a proxylet may execute the AOC functions whereas an additional proxylet may form the Virtual Control Cache or the Network Optimisation Component. This approach facilitates flexibility and efficiency in the constantly changing conditions of a P2P overlay. Different configurations of AVPs (i.e. AVPs with or without VCC or NOC components) can be deployed in parts of the network that experience different characteristics, or even in the same part of the network at different times of the day when conditions have changed. In addition, it is possible that different proxylets exist which implement the same functions differently. This gives further choice over the functionality of the AVP. For the prototype implementation using funnelWeb, AVP proxylets are written in Java. An AOC proxylet running on top of an EEP is conceptually illustrated in Figure 13.

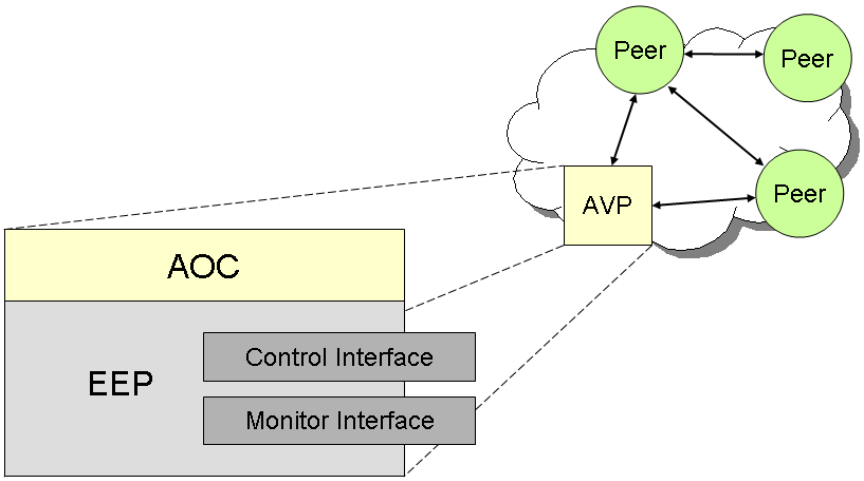

Figure 13: AOC proxylet running on an EEP.

Figure 14 illustrates the relationship between the network layer and the P2P overlay after the introduction of an AVP consisting of two proxylets. Although the proxylets execute on two different hosts, from within the application overlay, they are perceived as a single AVP.

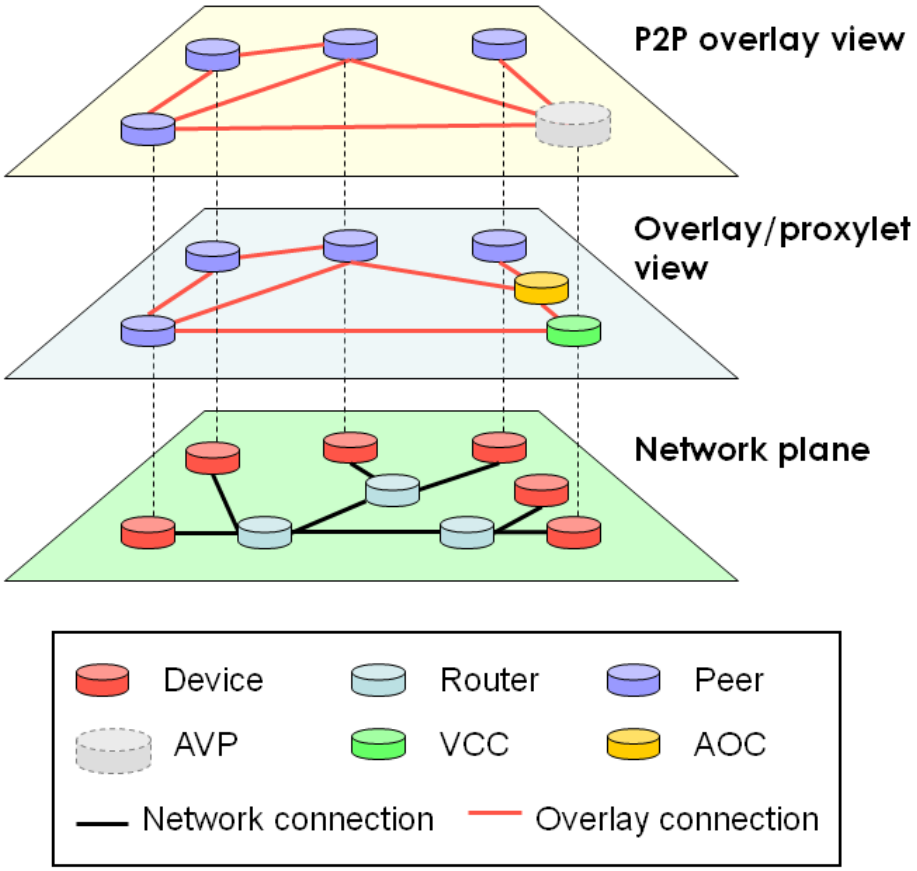

Figure 14: Relationship between network plane and application overlay.

4.5 Overview of AOC design

The AOC component contains P2P protocol-specific sensors and effectors to monitor and interact with the P2P overlay respectively. As each P2P protocol differs, sensors and effectors are designed in a modular way so that the ability to understand new protocols can be accommodated as they appear. The sensors continuously collect information about the state and activity of peers an AOC is connected to, such as searches, shared resources, active connections or other information the protocol may provide. Information gathered from captured P2P protocol messages is supplemented with that collected from light-weight network probing (e.g. [Ng, 2003]). This information is combined with similar information collected by AOCs that reside in other parts of the overlay. The exchange and correlation of peer and overlay state information allows for coordinated control of the overlay. Especially for unstructured networks, a group of AVPs is more suitable to evaluate the conditions inside a part of a P2P overlay from multiple vantage points than a single monitoring entity and this information is distributed in order to achieve better results. Control is applied by AOC effectors, namely the “Router module” and the “Connection Manager module”. The Router

module governs the relaying of messages at the application level according to local or federated constraints, for instance access control or performance optimisation policies. The Connection Manager module handles overlay connectivity and enforces control through its selective manipulation.

4.5.1 Router module

The router module is itself internally organised in a modular, tree-structured way, wherein each supported P2P protocol has its own independent routing element. This way it is possible to keep protocol-specific functions encapsulated and separated from other functions. In addition, this allows for protocol-specific optimisations to be offered where available. For instance, the Gnutella protocol element has a probabilistic routing mechanism incorporated, which will be discussed shortly. Finally, an AOC protocol element supports routing between AOCs.

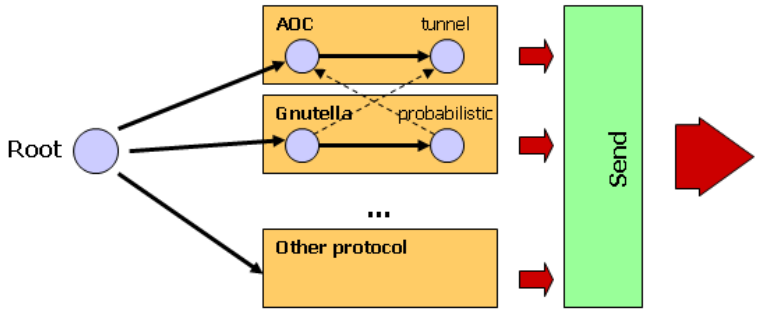

Figure 15: Structure and information flow inside the router module.

As illustrated in Figure 15, at the entry point of the router module lays the “root” routing element. Every received packet is passed to the root element, where all active connections are added as possible routes. From there, packets are passed to all protocol-specific elements attached to it, which process them according to their specific capabilities. For instance, a Gnutella router element can perform Gnutella protocol routing while an AOC router module routes AOC protocol messages. Every element decides independently whether it can handle a packet or not. Different elements are grouped according to their purpose (e.g. the Gnutella element has a probabilistic routing capability, discussed shortly). At the end of each path the send element is found. The send element passes the processed packets to the connection manager for transmission to another peer or AOC proxylet.

4.5.2 Connection Manager module

The connection manager module is responsible for the handling of all active connections (incoming and outgoing) and connection listeners maintained by an AOC. The module consists of two main components: the message processing queue and the connection list. As its name implies, the message processing queue queues and processes all incoming packets as they arrive. Packets may be passed to an appropriate P2P protocol handler for further processing, if an operation is required on the message contents, before forwarded to the router module described earlier. After the desired forwarding path(s) is established, the packets return to the connection manager for transmission.

The connection list holds information about all connections handled at a given moment (e.g. connection type, protocol, destination, etc) to enable stateful management of overlay connections. Real-time connection statistics and error logs, which are collected by the relevant sensors, are also stored in the connection list.

4.5.3 AOC administrative interface

The AOC contains an administrative interface which facilitates the control and configuration of the proxylet by an administrator via a remote console over the network. The administrator can issue commands to execute AOC functions manually or modify configuration parameters. Additionally, the administrative console can provide the operator with run-time module information, connection information or display AVP routing tables.

Configuration and management of AOC functions can also be carried out through the use of AVP policies. AVP policies are discussed in detail in Section 4.7.

4.6 AOC functions

4.6.1 Access control

One of the core capabilities of the AVP is access control. The AOC component can create areas of control inside a P2P overlay, where all communications between the controlled domain and the global overlay are inspected and appropriately managed by the AOC. The goal is to control who can access the peers and their resources inside the domain of interest. An AOC proxylet imposes access control by blocking and/or modifying P2P protocol packets communicated between the controlled domain and the

rest of the network. The desired result is for peers inside the controlled domain to be able to reach directly only each other and become practically invisible to any peer outside that domain that is not granted access. At the same time, the AOC proxylet becomes the mediator between the controlled domain and the global network.

The effectiveness of such a scheme depends on the P2P protocol of interest and the amount of autonomy the AVP operator wishes for the P2P service to retain. More specifically, how effectively an AVP deployment manages to separate the controlled domain from the rest of the overlay depends on how overlay operations such as peer join and searching are carried out by a particular protocol and whether the AVP operator wishes to apply a mild manipulation of these operations (for instance with the AOC capturing and manipulating only the messages it can receive naturally) or greater control. In the latter case, an AOC can become a sort of proxy for peers belonging to an ISP network, whose every communication with peers from other ISP networks passes through the AOC.

Full access control gives AOCs an almost complete picture of local peer behaviour since all requests and transactions with other peers pass through the former. This is desirable because it makes other forms of management more effective. For instance, if VCC caching (discussed in the next chapter) is applied, it makes sense to set up fully controlled domains in order to maximise transit traffic savings. Given the peers' inherent autonomy, AVP access control may seem challenging, especially since no changes to the P2P protocol are required. However, it has been demonstrated that the initial neighbours of a peer, which are the outcome of the bootstrapping process, have a very large influence on its future search and download performance [Karbhari, 2004]. In other words, by ensuring that the proposed neighbour list sent to a peer during bootstrapping contains a number of AVPs (the more the better) and local peers only, a large step has been made towards ensuring that this peer will remain locally connected for the entirety of its session. This can be achieved with a trivial DNS redirection to a locally managed bootstrapping server (e.g. for Gnutella a GWebCache server [GWebCache]), as casually performed by Content Delivery Networks (CDNs). This approach can be strengthened further by ensuring unmediated P2P connections to the

rest of the network are rate-limited, although this is not necessary. Interestingly, even after bootstrapping, exchanged peer messages and "word of mouth" will steer peers towards connecting to AVPs and other local peers. Furthermore, the high availability and performance offered by the AVPs compared to ordinary peers is progressively "memorised"[32] by local peers who will opt to connect to AVPs instead of random peers at their own accord. Lastly, an additional advantage of managing a local bootstrapping server besides access control is that it can be used to perform some initial load balancing of the local overlay.

The clustering of local peers as a result of controlled domain creation may raise concerns that overlay stability and resilience is compromised. The AVP framework addresses this issue by having each AOC maintain numerous and purposefully varied connections to peers in different ASes to ensure that effective local peer scope remains robust even at high churn conditions. Inside the domain peers are redundantly connected to more than one AOC, minimising the risk of becoming disconnected in the case of AOC failure.

Using Gnutella as an example, a scenario of access control will now be presented. A typical Gnutella packet contains the following information [GDF]:

- Source and destination connection attributes (IP address, port number, GUID[33], Message ID[34])
- TTL (Time To Live) and hop values
- Payload type (Ping, Pong, Query, QueryHit, Push)

Other protocols use similar attributes for their operation. Gnutella uses "Ping" messages for neighbour discovery. In Figure 16a, Peers 1 to 5 reside inside the global Gnutella overlay. Peer 2 sends out a Gnutella "Ping" message in order to discover other peers. Under the Gnutella protocol, a Ping has to be forwarded by the receiving peer (in

[32] Peers maintain "known host" lists of peers with whom they had successful transactions in the past for performance and resilience reasons.

[33] A GUID (Globally Unique Identifier) is a 16-byte long string uniquely identifying a peer inside the Gnutella network.

[34] A Message ID is a 16-byte long string identifying a particular message inside the Gnutella network.

this case Peer 5), to any peer directly connected to it. Moreover, every peer that receives a Ping message has to respond with a "Pong" message. Thus, Peer 2 eventually receives "Pongs" from Peers 1, 3, 4, and 5. In the access-controlled scenario, illustrated in Figure 16b, an AVP manages a controlled domain (CD), where Peers 1 and 2 reside. The AOC proxylet examines all communication to and from the CD. When Peer 2 sends out a "Ping" to discover other peers, the AOC proxylet intercepts the "Ping" message and forwards it unmodified to Peer 1 which also is part of the CD. In addition, the AOC proxylet modifies that "Ping" so it seems like it was initiated by itself, i.e. it changes the source connection information (IP address and GUID) of the message. Then, the AOC proxylet relays the modified message to the outside world, on selected routes if deemed necessary. Peer 2 receives "Pongs" by Peer 1 and the AOC proxylet and concludes that only these two peers comprise its neighbourhood. The AOC proxylet captures all messages originating from the global Gnutella network, modifies them if necessary, and forwards them inside the controlled domain. This way, the AOC proxylet makes sure connections to the CD are not created from the outside and gathers information about the global Gnutella network that can be indirectly utilised by the peers (e.g. searches) inside the CD.

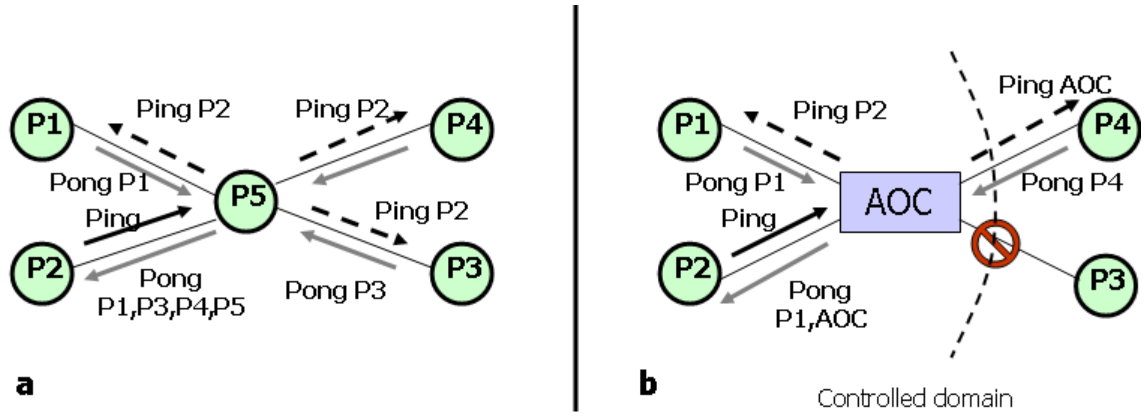

Figure 16: Access Control on Gnutella by the AOC.

4.6.2 Routing control and load balancing

The AOC routing module represents the core mechanism for application of control. As seen earlier, appropriate routing of messages can influence peer behaviour to a large extent. As a result, most other controls utilise the routing module as part of their operation.

A particularly valuable form of routing control, which also overlaps with the goals of topology control, is control of local peer relations. Specifically, apart from

influencing the way local peers interconnect with each other for overlay membership and signalling functions as per access control, the AOC can also bias to a certain extent the selection of download sources by local peers and thus resulting transfer connections without diminishing typical transfer performance. When a local peer searches for an object and adequate sources exist both globally and inside the ISP's network, the AOC can promote the local sources by suppressing most (or all, depending on the amount of appropriate local replicas) non-local responses while forwarding local ones. The intended effect is the formation of local rather than inter-AS peering relations. Crucially, by leveraging the ISP's privileged knowledge of the underlying network infrastructure and their ability to obtain accurate measurements much more effectively than peers on their own, AOCs can direct peers towards optimal paths and appropriate neighbours even within the ISP's network. Therefore, routing control can be used as a tool to help the typically topology-agnostic peers avoid making costly and inefficient peering decisions.

This traffic localisation does not only help in reducing costly transit traffic for the ISP, but also takes advantage of the typically lower latency and less congestion between peers of the same domain, which may manifest on improved download performance. Within the domain, source balancing can achieve load distribution between multiple similar sources. When no local sources exist, AOCs can still influence peer selection and avoid inefficient overlay peering decisions by minimising the number of ASes traversed (i.e. promote "close" foreign peers from the list of query replies), selecting peers from ASes with which the ISP has a favourable agreement (i.e. peering agreement or lower transit traffic rates) or, if such information is available (for instance through a NOC module), preferring peers on non-congested paths.

The modular architecture of the routing module allows for protocol-specific optimisations or improvements to be incorporated for particular P2P protocols, which build upon AVP's performance-enhancing role. The "Probabilistic Routing" module is such an enhancement for unstructured decentralised networks which employ message flooding. The probabilistic routing module identifies broadcasted protocol packets (e.g. query messages, neighbour discovery messages, etc) and instead of forwarding them to all available neighbours as expected, it does only to a subset of them in order to minimise overheads and load from message flooding. This module builds on the idea of "random walks" described in [Lv, 2002b] and [Chawathe, 2003]. As such it is not a novel contribution on its own but rather a service, incorporated to demonstrate the modularity of the architecture. The overlay connections over which a message is to be

forwarded are selected based on a random value assigned to the message compared to a given threshold per connection. If the random value for a message is larger than the configured threshold, the packet is forwarded, otherwise it is suppressed. This module bases its operation on the fact that unstructured decentralised systems largely approximate the behaviour of scale-free networks where most of the messages pass through a few hub nodes. As long as messages pass through these hubs, there is a very large probability they will reach their destination without the need to be broadcasted on every available connection. Connection thresholds are adjustable as a means to apply basic overlay-based load control. If upon probing an AOC detects increased delay on the physical path between an AOC and its neighbour, it can increase the threshold of the corresponding overlay connection to prevent the further degradation of the path and/or overloading of the peer.

It must be noted that probabilistic routing is applied only to messages that are broadcasted and contain request information, such as Pings and Query messages in Gnutella. Messages that are sent in response to another message, and thus contain valuable information for the peers, are not manipulated.

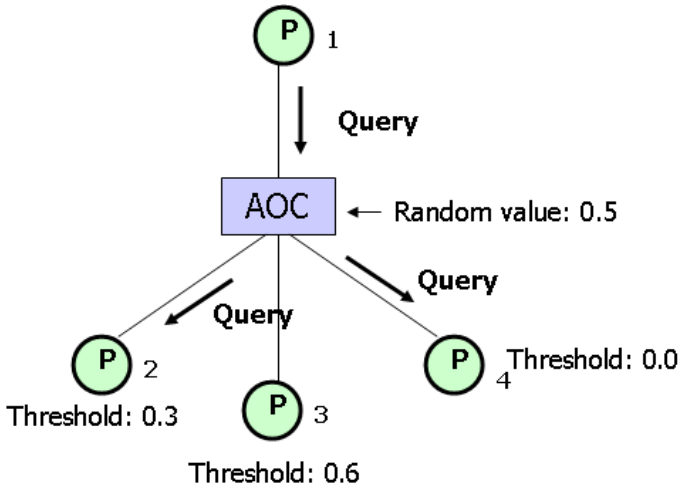

Figure 17: Operation of the "Probabilistic Routing" module.

An example of probabilistic routing is depicted in Figure 17. In this example four Peers (1, 2, 3, 4) are directly connected to an AOC proxylet. The proxylet has assigned different threshold values for the links to each of these peers. Peer 1 sends a broadcast message to the AOC. The AOC determines a random value of 0.5 for this Query (values are uniformly distributed in the interval [0, 1]). Since the random value is smaller than the threshold value on the link to Peer 3, the AOC component does not

forward the packet along this connection whereas it forwards it as normal on the links to Peers 2 and 4.

4.6.3 Topology control

Topology control as enabled by the AVP aims to enforce optimal P2P relations inside the overlay, based on information collected from overlay measurements, available NOCs or the peers themselves. The AOC component of an AVP performs topology control by selectively setting up or terminating connections to other AVPs and ordinary peers, creating alternative paths where needed and restricting reachability to others. These mechanisms are complemented by neighbour selection biasing, discussed in the previous section, an application of topology control based on routing control mechanisms. By shaping the way peers are connected and communicate inside the overlay, AVPs can improve overlay stability and the performance of peer functions.

Stability is attained by ensuring that peers which act as hubs within the overlay are well-connected and can be reached via alternative overlay paths, critical paths are not formed through high-churn peers and in general, the effect of peer departure is constrained. Search performance is improved by bolstering peers' ability to reach other peers in the global overlay and their search scope, as discussed earlier. Furthermore, paths may specifically be created between AVPs to reduce the characteristic path length of the overlay.

The following scenario, as depicted in Figure 18 and Figure 19, examines how the AVP can apply beneficial topology control on a part of the overlay. Figure 18 shows a number of peers organised in three controlled domains, managed by an equal number of AOCs. Heavy activity between peers of CDs 1 and 2 is all mapped on the path between routers A and B and thus may compete with non-P2P traffic, causing congestion. The three AOCs exchange overlay routes in order to re-route part of the traffic between CDs 1 and 2 over an alternative path. As illustrated in Figure 19, AOC 3 opens connections to AOC 2 and a peer of CD 2, creating an overlay path that can reach CD 2 over another physical link. By tunnelling traffic between AOCs 1 and 3, P2P link load between CDs 1 and 2 is balanced.

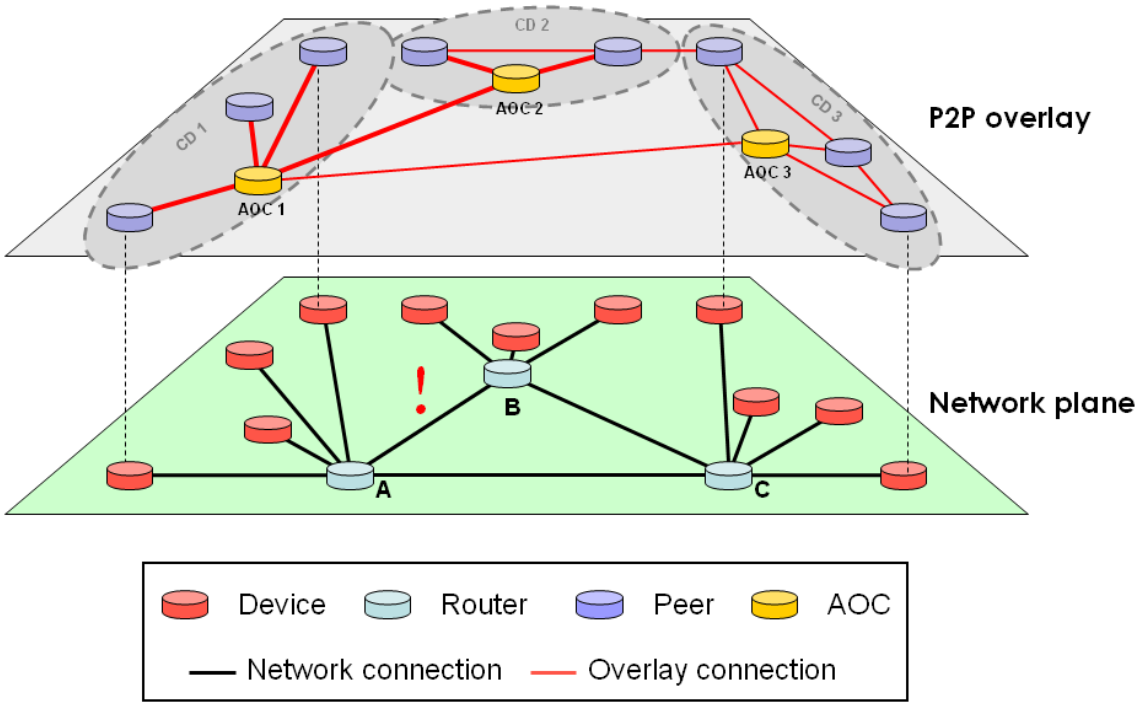

Figure 18: Initial overlay condition.

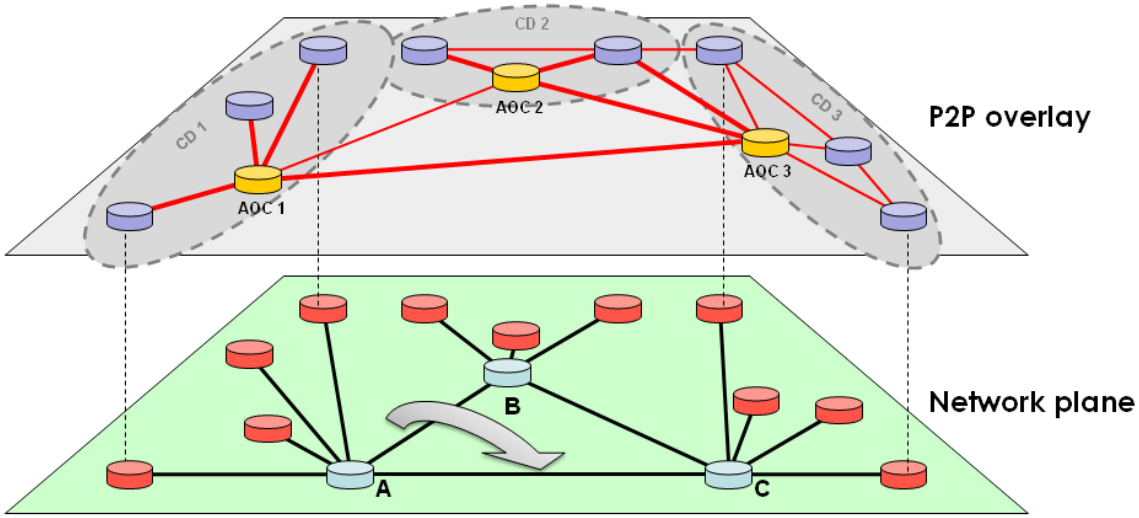

Figure 19: Overlay after AOC control.

4.7 AVP policy model

The AVP administrative interface allows the manual configuration and control of the AOC. An AVP operator may connect remotely to this interface through a console and configure an individual AVP or execute available commands. While such an interface is useful, it does not provide a practical or scalable solution for large AVP deployment management. Furthermore, AVPs are envisaged to function with little real-time support from the network provider, ideally being "fire-and-forget" entities that self-organise and self-coordinate with minimal attendance from an administrator.

Nevertheless, in order to function, AVPs need configuration data and information necessary for coordinating their efforts. To address this issue, ideas from the area of policy-based network management are employed. Policy-based network management enables the configuration and management of a large number of network elements and resources through the use of abstract rules (policies), allowing the network to be managed as an entity and not as a sum of many and disparate components [Hewlett-Packard, 1999].

Policy-based network management (PBNM) was proposed to make the accomplishment of several IT tasks easier:

- Reduce the time, cost, and problems associated with individual device configuration and enforcing operational coherency.
- Meet users' varied needs and expectations regarding network application performance and quality of service.
- Optimise the use of network resources and slow down the cycle of network over-provisioning to improve performance.

Policies contain one or more rules. Rules specify a set of conditions that, when evaluated true, result in an action being taken. Put simply, a rule makes the logical statement: `IF (conditions) THEN action` [Moore, 2001]. This is conceptually illustrated in Figure 20:

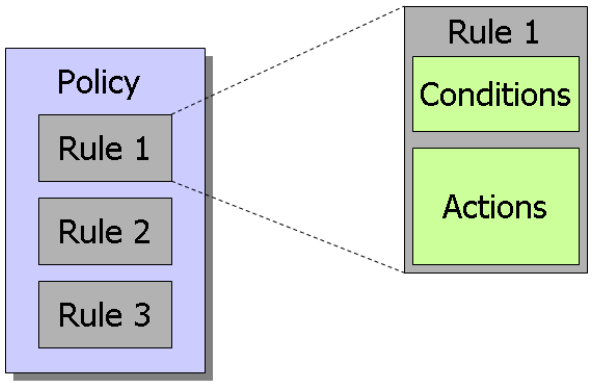

Figure 20: The IETF policy model.

In the context of this research, AVPs regulate their behaviour and determine their scope through policies that they receive either from the network operator or other AVPs. AVP policies are a simplified adaptation of the work carried out in the policy-based management area, suited to the needs and design criteria of the AVP. The provision of a full-scale policy-based management supporting infrastructure (for

example as discussed in [Waters, 1999]) was deemed incompatible to the notion of AVPs being peers themselves, as it would reduce their autonomy and make their deployment more complex. Furthermore, supporting such an infrastructure would introduce overheads and increase the memory footprint of the AVP proxylets. Consequently, only the necessary features to support AVP functions were considered. For that reason, throughout the text the term "policy" refers to AVP-related policies unless explicitly stated otherwise.

AVP policies contain configuration data, constraints to which capabilities an AVP should be allowed to use and instructions on what an AVP should do in relation to a particular event. Policies are normally assigned to an AVP upon activation, but additional policies may be sent and activated during operation.

Two categories of AVP policies exist: *System-wide policies* that contain rules and information that every AVP in a deployment should be aware of, like the range of IP addresses or the list of AVP IDs owned by the network operator. This will prohibit, for example, AVPs from interfering with peers run by customers of other ISPs. The second category is *individual configuration policies*, which apply to individual AVPs or groups of AVPs that manage specific parts of the overlay. These may, for instance, contain a list of IP address subnets inside the operator's network that need to be grouped in a controlled domain (CD). Through configuration policies, the AVPs can identify which peers to apply a form of control to, which to block, or whether they are allowed or not to use certain capabilities, such as activating a VCC component or other AOC components. A configuration policy will also specify the conditions that need to be met before such activation.

To avoid cases where two or more policies dictate conflicting actions, large-scale PBNM infrastructures (for example as discussed in [Waters, 1999]) implement "conflict resolvers" or "conflict checkers" which examine policies and either indicate or additionally take action to resolve any conflicts discovered. To avoid complexity, such functionality was not explicitly developed and incorporated into the AVP infrastructure. A policy priority attribute is included, however, in the AVP policy model which classifies a policy's priority as "critical", "normal" or "best-effort" according to its significance. This may assist in categorising policies according to significance and identifying conflicts manually. The consequence of this scheme is that it becomes the responsibility of the AVP operator to classify policies appropriately and avoid cases of conflict between policies of the same scope and significance, without relying on automated checking by the AVPs. Nonetheless, given the clear definition of the AVP

policy model and use of XML, discussed shortly, it is possible that existing tools may be modified to support AVP policies and facilitate automated conflict identification.

AVP policies are represented using XML [BRAY, 2004], due to its platform-independence and extensibility. The wide use of XML ensures a high degree of familiarity by IT professionals as well as an abundance of editors, parsers and other software to assist in the creation and management of AVP policies. An XML-Schema is provided to ensure the proper definition and validation of policies. The complete schema is included in Appendix B. An example of an AVP policy is presented in Appendix C.

At the highest level, an AVP policy has the following structure (Figure 21):

- Policy Identifier (`polId`): a unique identifier of the policy (e.g. policy_1).
- Policy Group Identifier (`polGroupId`): Interrelated policies may be grouped together for easier management or to create policy bundles that collectively carry out a complex operation. Policies of the same group share the same group identifier.
- Policy Type (`polType`): Indicates the type of policy – individual or system-wide.
- Policy Description (`polDescr`): A string containing a brief description or comments on the policy's purpose.
- Policy Priority (`polPriority`): Indicates the priority of the policy. A policy can be classified as critical, normal or best-effort.
- Validity Period: The validity period of a policy indicates the time period during which the policy should be in effect. This way the automatic activation or deactivation of policies based on system time is made possible. The validity period can also be set to indicate permanent scope or expiration date only.
- Conditions: The set of policy conditions.
- Actions: The prescribed actions that must be taken if the policy conditions are met.

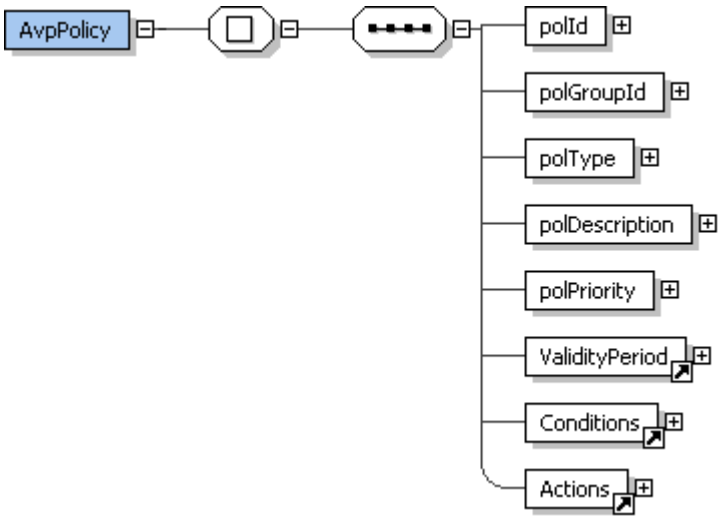

Figure 21: AVP policy structure.

In order to enable the formulation of conditions with sufficient flexibility and clarity, they are broken down into three parts: condition objects, condition requirements and evaluation parameters. The condition objects (`CondObject`) specify which parameters of the system need to be monitored or evaluated as part of checking the validity of the rule. The condition requirements (`CondReq`) contain the constraints applied on these condition objects. The evaluation parameters (`EvalParams`) specify the relationship between the requirements and objects and any other parameters necessary for the evaluation of a condition. Figure 22 illustrates the structure of the "Conditions" element.

A condition object may be quantifiable or not. Quantifiable objects such as, for instance, the number of lost packets or queries made can be represented by the *variable* complex type. The *event* complex type of the schema is used to represent non-measurable or non-quantifiable objects that occur at unpredictable times. The appearance of a new peer in the overlay or the transmission of a P2P protocol message are such cases and can be represented as events.

The `Variable` complex type contains the following elements:

- Variable identifier (`varId`): an identifier of the variable in question.
- Variable type (`varType`), e.g. "numberOfQueries".
- The syntax (data type) of the variable (`varSyntax`), e.g. "unsigned integer".

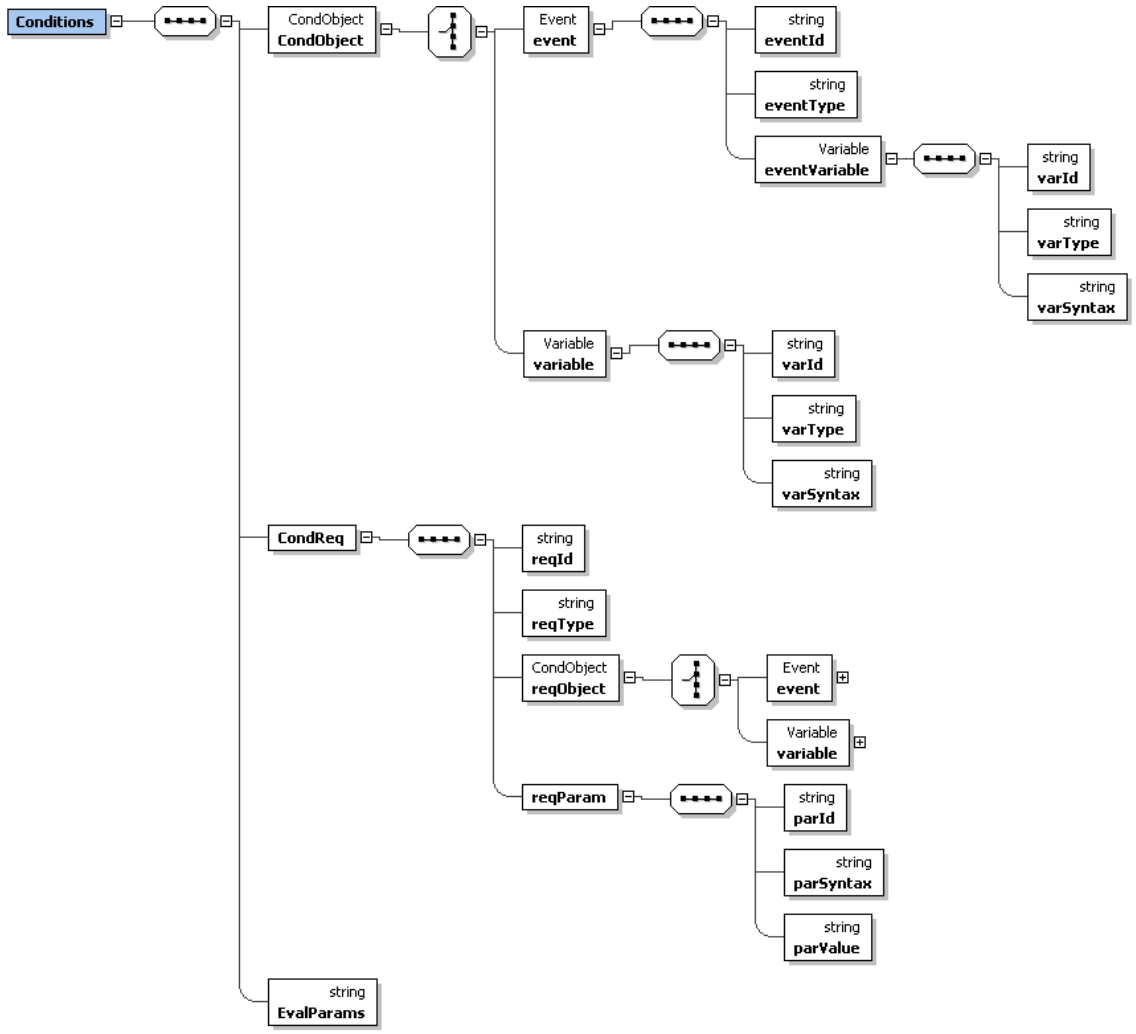

Figure 22: Condition element structure.

Objects represented using the `variable` complex type have values that can vary with time. The values of these variables can be monitored and updated by AVPs depending on the policy. An example of a variable object in an AVP policy could be written as:

```
<Variable>
    <varId>noq1</varId>
    <varType>numberOfQueries</varType>
    <varSyntax>unsignedInt</varSyntax>
</Variable>
```

The `Event` complex type contains the following elements:

- Event identifier (`eventId`): an identifier of a particular event.
- Event type (`eventType`), specifying the type of event (e.g. “NEW_PEER_ARRIVAL”).

- Event data variable (`eventVariable`), used for reference to any event-specific information that may be available.

Although, event variables reuse the structure of the `Variable` complex type (i.e. have an identifier, type and syntax), they are used for characteristics that are not expected to change over time. An example of an event variable may be the IP address of a peer that just entered the overlay (the event in this case is the peer arrival). During the duration of that peer's session this address is not expected to change.

The condition requirements contain the requirements or constraints that need to be applied on the values of the condition objects so that the evaluation can be carried out. A complete condition may have more than one condition requirements applied on a number of objects. The condition requirement (`CondReq`) complex type contains the following elements:

- Requirement identifier (`reqId`): an identifier of a particular requirement.
- Requirement type (`reqType`). Three operators are provided:
- Greater than (`GT`)
- Less than (`LT`)
- Equal (`EQ`)
- Requirement object (`reqObject`): the condition object (event or variable) that this requirement corresponds to.
- Requirement parameters (`reqParam`): the values or thresholds with which the objects will be compared. These have a similar structure to the variable type (i.e. parameter identifier and syntax) and hold the actual comparison values.

For instance, reusing the aforementioned "number of queries" example, an accompanying requirement specifying that the number of queries should be less than ten could be:

```
<CondReq>
    <reqId>Req1</reqId>
    <reqType>LT</reqType>
    <reqObject>noq1</reqObject>
    <reqParam>
        <parId>lessThanTen</parId>
        <parSyntax>unsignedInt</parSyntax>
        <parValue>10</parValue>
    </reqParam>
```

</CondReq>

The evaluation parameters (EvalParams) bind together the condition objects and requirements by specifying the logical operators (AND, OR, NOT) that need to be applied to carry out the evaluation of the rule. For example, an evaluation parameter could be of the form: (Req1 OR Req2). The evaluation parameters are represented as strings.

The Actions complex type contains the elements necessary for the specification of the desired course of action, should the defined conditions be met (Figure 23). These are:

- Action identifier (actionId): an identifier of a particular action.
- Action type (actionType): the action that needs to be implemented (e.g. "drop_connection"). This corresponds to a function call (or sequence of) that will carry out the actual desired action.
- Action parameter (actionParam): a parameter that may need to be provided in order for a specific action to be successfully carried out. This parameter is specific to the particular action and has the same structure as the requirement parameters (i.e. identifier, syntax and value). Not all actions may require a parameter, in which case this field is not used.

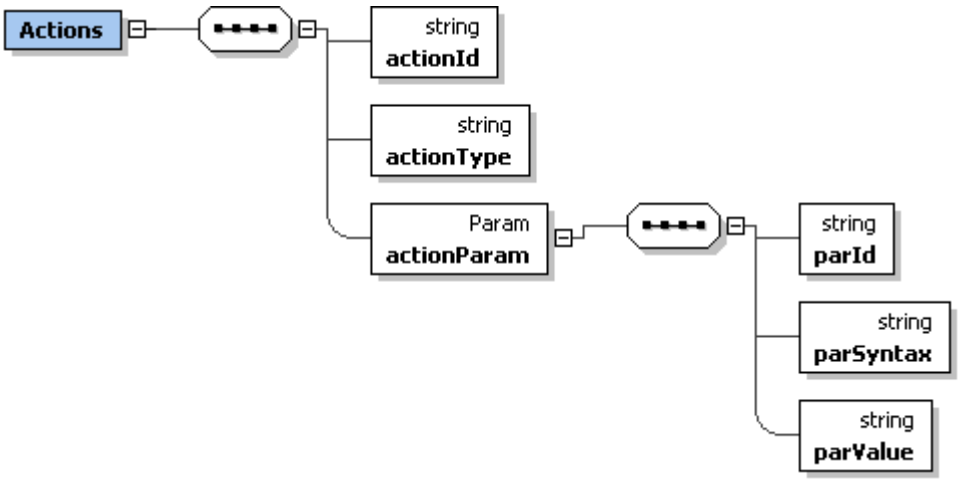

Figure 23: Structure of the "Actions" element.

4.8 The Network Optimisation Component

By enabling peers to create their own, physical infrastructure-independent overlay topologies and make their own routing decisions inside them, P2P services

cause a mismatch between these two topologies and routing layers. While this mismatch can often prove advantageous for the service's resilience or performance, it is also responsible for two major issues: The first is the steep increase of transit traffic volumes associated with P2P services, caused by the creation of overlay peerings spanning multiple ASes despite the availability of local sources, as discussed earlier. The second is the interference or even conflict between the separate routing layers (i.e. the underlay and the overlay) operating independently, which may lead to unnecessarily congested paths, race conditions and undermining the intended effects of any traffic engineering applied. For instance, the overlay route selected based on some active measurement (typically delay) and mapped on certain network links may cause one of them to become congested. Moreover, as noted in [Keralapura, 2004], co-existing routing layers can experience race conditions and become synchronised, leading to route and traffic oscillations and cascading reactions. In general, the discrepancy between the overlay topology and the physical network topology may not only make P2P services less efficient but, more importantly, may also affect the performance of other network services as well.

The AVP architecture attempts to address the aforementioned heavy utilisation of costly and scarce inter-AS links via the topology control capabilities of the AOC and via P2P content caching as enabled by the VCC, which is presented in the next chapter. At the same time, in order to address the more general problem of overlay/underlay mismatch, the Network Optimisation Component (NOC) was proposed as a part of the AVP architecture.

The NOC is envisaged to provide the AVP with dynamic traffic engineering capabilities which will map the P2P overlay traffic onto the actual network layer much more effectively than current systems do on their own. By matching the physical and overlay topologies in locations of importance or ensuring that IP traffic engineering is not invalidated by overlay routing mechanisms, a P2P service may realise performance gains while the proper state of the network is maintained. Towards that, the NOC can be used to provide the AVP deployment with information on the state of the network that can be used to make informed overlay control decisions.

The NOC provides the interface that the AVPs operating at the application layer can use to learn current information about the provider's network. At the very least, the NOC can be built as a database where ISP-specific information as well as network measurements and statistics from various sources can be collected, combined and provided from a single place in a consistent format. For instance, the NOC may be

queried for various kinds of ISP-specific information on local peers, such as the PoP (Point-of-Presence) serving the subscriber whose assigned IP address matches the one of a peer in question, as well as his/her subscription's connection type and capacity, contention ratio, imposed bandwidth caps and other relevant information. Such information can be vital in inferring a peer's true characteristics and capabilities with a certainty not afforded by probes or other methods, or by relying on what information P2P protocols report on their own. Furthermore, the NOC may extract and process information from interior and exterior gateway protocols which will provide AVPs with valid and current insights on how overlay routes and connections translate on the actual network infrastructure. For example, BGP (Border Gateway Protocol) [Rekhter, 2006] readily supplies information on the number of ASes separating two peers as part of its Network Layer Reachability Information (NLRI) updates. Such information can be used to help AOCs promote peerings between peers that are closer together or over suitable paths.

Being part of infrastructure owned and operated by an ISP, the NOC can have access to network-related information that a user application (e.g. a peer client) cannot due to its lack of a suitable network vantage point or for confidentiality/security reasons. In other words, the NOC can take advantage of the ISP's privileged position and knowledge of the network to gather valuable information that regular peers cannot collect on their own.

Due to the size and scope of the AVP architecture and the coverage of so many diverse research areas as part of designing and building the AOC and VCC components of it, it was not possible for the NOC to be developed further beyond this initial concept as part of PhD research. It is envisaged that the addition of a functional NOC can bring a manyfold improvement to the ability of an AOC to make optimal overlay topology control decisions and limit the impact overlay formation and routing have on the underlying network topology. Since however an NOC prototype was not built, all further discussion and evaluation of the AVP takes care not to assume the availability of such capabilities in either the prototype or the simulation model.

It is a testament to the foresight and validity of the AVP architecture, first presented in 2003 [Koulouris, 2003], that towards the completion of this thesis independent work on the topic proposed that ISPs provide P2P applications with guidance on suitable neighbours and paths in the form of an "oracle" service while protecting ISP topology confidentiality [Aggarwal, 2007]. The similarity between the NOC and an oracle along with other related research is discussed in Chapter 8.

4.9 AOC prototype implementation

For the prototype implementation of the AVP, an unstructured decentralised P2P file-sharing system was considered the most appropriate platform for AVP capabilities to be tested upon. P2P file-sharing applications are known to create large numbers of overlay connections with a lot of temporal and spatial connection variability, generate large amounts of traffic and exhibit high frequency of peer arrivals and departures. These characteristics were considered important in evaluating certain elements of the AVP such as access control or the VCC component (discussed in the next chapter). Consequently, the Gnutella protocol was chosen for the development and testing of the AVP prototype as it is well-researched, open source and actively used by large numbers of users.

As discussed earlier, AVP components are written in the Java programming language, in the form of proxylets. The AOC prototype proxylet contains five basic sub-components:

- AOC router module
- AOC connection manager module
- Administrative interface module
- Policy module
- Gnutella protocol module (packet interfaces, protocol-specific functions)

One of the main features of the AOC router module is the ability to handle multiple protocols simultaneously. For the prototype version of the AOC proxylet, two different protocols have been implemented: the Gnutella version 0.6 protocol and the AOC inter-communication protocol, called the *AOC protocol*. The prototype version of the AOC has thus the ability to route Gnutella packets and AOC-to-AOC packets as well as route between Gnutella and AVP networks. Table 2 lists all types of connections implemented by the AOC prototype.

The AOC protocol, used for AOC-to-AOC communication, facilitates the exchange of information vital for the communication and self-organisation of the AVPs. The protocol was designed to be independent of existing P2P protocols and does not rely on any specific P2P protocol mechanisms, although it borrows ideas from unstructured P2P protocols such as Gnutella and eDonkey. A major feature of the AOC

protocol is the tunnelling of other protocol messages (for instance Gnutella messages) between AOC proxylets for topology control purposes.

An AOC protocol packet contains the following fields, illustrated in Figure 24:

- A 5-byte header specifying payload (message) type and packet size.
- If the message is a route advertisement message (to exchange route and overlay topology information between AOCs), then only the source connection attributes (AOC identifier, IP address, Port number) and data payload are provided.
- If it is a tunnelling message then it contains:
- Source connection attributes (AOC identifier, IP address, Port number)
- Destination connection attributes (AOC ID, IP address, Port number, alternative route)
- Payload
- Priority

Connection type	Protocol
AOC-to-AOC	AOC protocol (TCP)
Gnutella-to-AOC	Gnutella Protocol v0.6 (TCP)
AOC-to-Gnutella	Gnutella Protocol v0.6 (TCP)
Administrative interface	Telnet (TCP)
EEP-to-AOC	ALAN proxylet interface

Table 2: Types of connections supported by the AOC prototype.

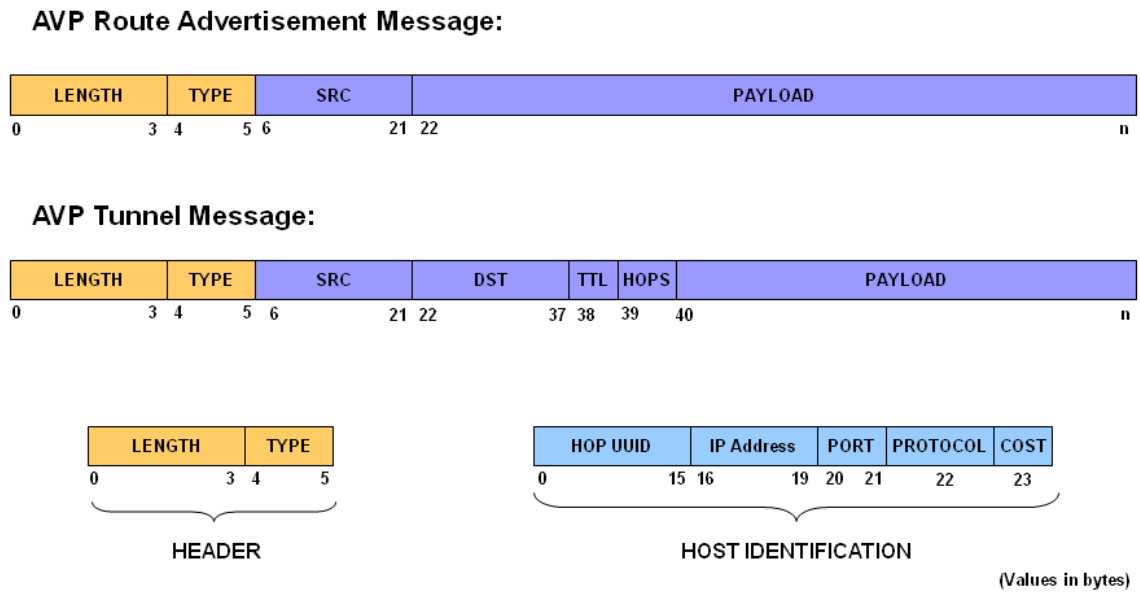

Figure 24: AOC protocol message structure.

When P2P protocol messages are relayed, they are encapsulated in AOC protocol messages with a header pre-pended. The entire header information occupies 40 bytes. The header allocates four bytes for packet size and two bytes for packet type identification information. Additionally, the header contains two 16-byte fields for source and destination information, as well as two one-byte sections for the "time-to-live" and the hop count fields. The overhead introduced by the encapsulation is not trivial, but permits the distinct handling of messages without holding state information.

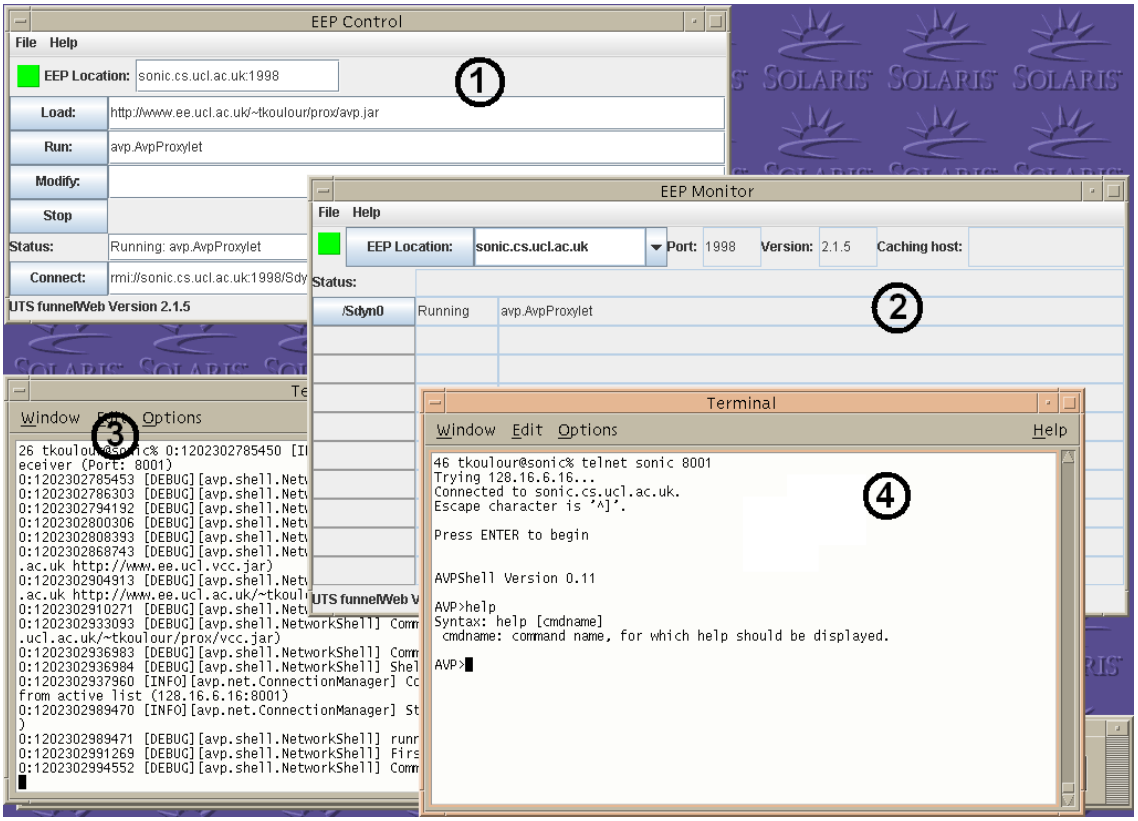

Figure 25: Using the AOC prototype.

Figure 25 depicts the control and information interfaces available when operating the AVP prototype. The numbered windows correspond to the EEP control interface (1), EEP Monitor interface (2), console window where real-time proxylet info is displayed for logging/troubleshooting purposes (3) and AOC administrative interface (4) awaiting a command. AOC functions can be executed manually using commands like the following:

- *open [protocol] [IP address] [port number]:* creates a new connection. Example: `open gnutella 123.123.123.123 8000`
- *close [protocol] [IP address]:* closes an existing connection.

- *pgnut add [IP address] [threshold]:* adds a loss-probability threshold to a specified connection (threshold between 0.0 and 1.0). Example: `pgnut add 123.123.123.123 0.5`
- *firewall add [domain identifier] [IP address]:* adds an IP address or address range to the controlled domain referenced by 'identifier'. Example: `firewall add cd1 123.123.123.123`

4.10 Summary

This chapter introduced the core element of the P2P service management framework presented in this thesis: the Active Virtual Peer. The AVP addresses the challenge of introducing controls to the self-organising overlay of autonomous elements a P2P service forms by becoming part of it. An AVP appears as an ordinary peer and applies the intended management functions transparently through its interactions with other peers.

The AVP architecture favours a modular design where components which implement different capabilities can be dynamically combined to form AVPs of different roles. After presenting the architecture and supporting platform, this chapter focused on the AOC, which implements the base functionality of the AVP and serves as its minimal configuration. Its high-level design as well as the way it applies access, routing and topology controls was discussed, before presenting the AVP policy model, created to facilitate better configuration and management of a multiple-AVP deployment. The motivation, value and intended operation of the NOC component was briefly examined next. The chapter was concluded with a discussion of the AOC prototype implementation.

The next chapter continues the discussion of AVP mechanisms with the presentation of the VCC component, which provides content caching capabilities to the AVP.

5. THE VIRTUAL CONTENT CACHE

5.1 Introduction

As discussed earlier, one of the main downsides P2P services have from an ISP's perspective is the amount of traffic they generate that is not confined inside the ISP's network. While a considerable proportion of that traffic is necessary for the correct and effective operation of P2P services, it can be argued that by not taking into account peer proximity or geographic location, P2P services form topologies which lead to an excessive number of not particularly necessary but nevertheless costly inter-AS connections. For instance, it has been reported in [Gummadi, 2003b] that up to 86% of requested P2P objects were downloaded from peers outside the home network despite being locally available. While incorporation of intelligence that can take advantage of locality in the protocol level as suggested in [Zhao, 2002; Castro, 2002b; Castro, 2002c; Lua, 2004] and elsewhere can be a step towards the right direction, the implementation of such functionality remains a decision of the P2P application developer and may be in conflict with other service functional requirements, or viewed as a performance-limiting factor. Furthermore, it may clash with existing traffic management functions present at the lower layers and introduce race conditions between them [Keralapura, 2004]. In concert with the design philosophy of the AVP concept, a more desirable solution would be to minimise transit traffic generated by *a number* of similar P2P applications in a way that does not affect their operation and end-user experience. Content caching is key in such an approach as it allows the reduction of P2P-generated inter-AS traffic without being intrusive to the service.

Caching is routinely employed in the web domain where it describes the storage of copies of popular objects (e.g. web pages, images etc.) in order to minimise bandwidth usage, server load and perceived response lag [Wessels, 2001; Rabinovich, 2001]. In the context of P2P, caching can be highly effective because unlike the web where an increasing proportion of content is dynamic, personalised and thus un-cacheable, content exchanged through file-sharing applications is almost universally

immutable. Therefore, by bringing often-requested content inside its network so that its subscribers can be served from an internal source, an ISP can reduce the amount of inter-domain traffic caused by bulk P2P transfers. The intended effect of P2P content caching is illustrated in Figure 26, below, alongside current normal P2P operation.

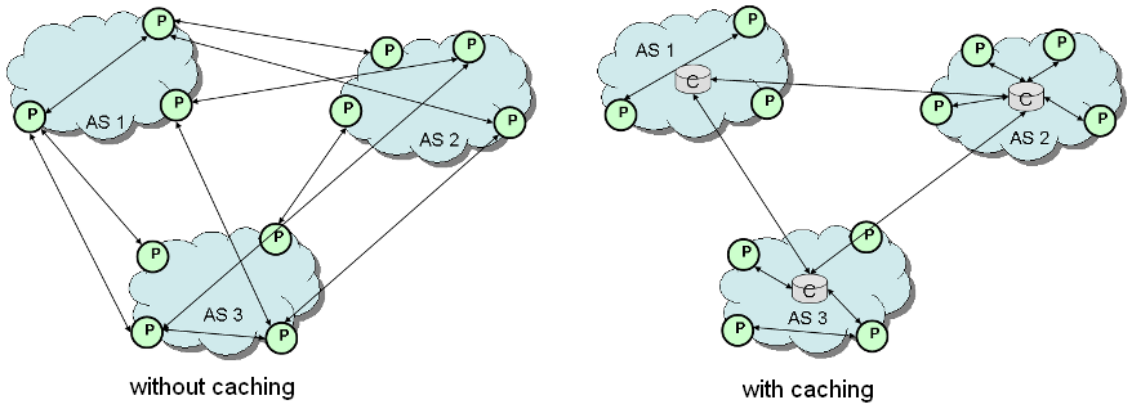

Figure 26: P2P traffic flows without and with content caching employed.

The Virtual Content Cache (VCC) integrates such content caching capabilities for P2P services directly into the AVP management framework. The gains are two-fold: Instead of having each peer fetch the same content from sources outside the ISP's network over expensive transit links, the content is fetched once by the cache and subsequently served from there. Secondly, peers are served from a local source that is expected to enjoy good connectivity and high availability. Depending on the popularity of a particular item and its original location, significant economies can be achieved, while offering a noticeably better quality of service to the ISP's subscribers. Finally, a side-effect of caching, especially if adopted by a number of ISPs, is the reduction of redundant traffic on the Internet backbones as well.

Having presented the AVP architecture and examined the Application Optimisation Component (AOC) in the previous chapter, this chapter focuses on the VCC component of the AVP. The next section presents the design principles of the VCC, including a deployment scenario. A discussion of cache replacement strategies in general, along with a deeper examination of proposed strategies for the VCC follows. The VCC proxylet implementation is presented next. A brief examination of potential legal issues from the employment of P2P caching concludes the chapter.

5.2 VCC design

5.2.1 Overview of the design

The VCC is designed as an AVP framework component which provides P2P content caching capabilities. The modular design of the AVP implies that an AVP may maintain a VCC component or operate without one, depending on the requirements and operational scenarios the AVP operator may have formulated. The capability of an AVP to spawn VCC components is given by the network provider/owner of the AVP either automatically through AVP policies or manually through the administrative console (both presented in Chapter 4). The main aim of this modular architecture is, as discussed earlier, to make the AVP concept as flexible as possible by allowing the use of different AVP configurations as necessary.

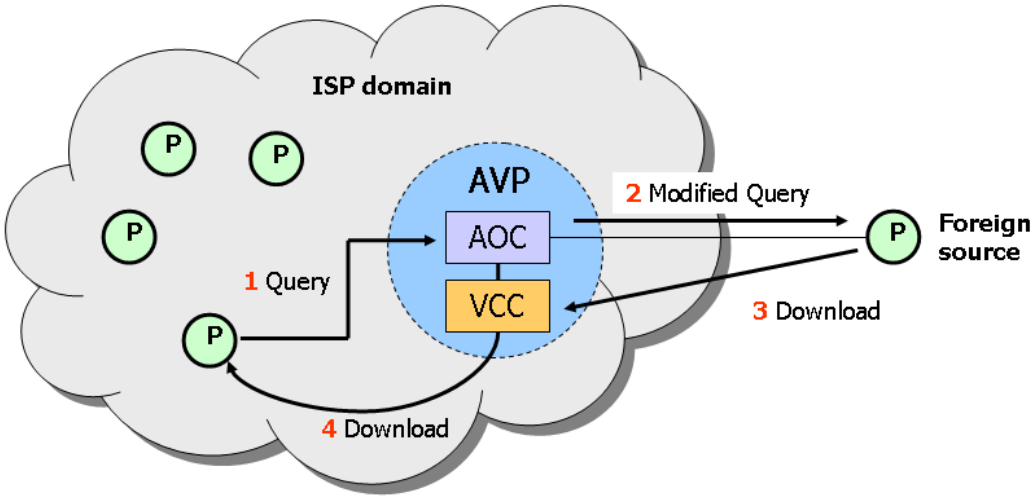

Figure 27: Application-level caching by the VCC.

In Figure 27, a scenario of an AVP containing a VCC is illustrated. The AVP controls a domain of peers by applying routing and access controls as discussed in the previous chapter. The controlled domain may be the entire ISP network or part of it, depending on the particular AVP configuration. The AOC monitors the P2P protocol messages exchanged between peers inside the domain (which the AVP maintains a connection to) and gathers information about the kind of content each peer offers or is looking for. Each time a query for content made by peers inside the domain reaches the AVP, it is inspected but not propagated outside the controlled domain to avoid a possible unmediated reply. The AOC queries the VCC for the content in question and if

positive replies on behalf of the VCC, having the peer download the content from the local cache. If the content is not already cached but is deemed popular, the AOC forwards a modified query message outside the domain, so that "foreign" peers perceive the AVP as the actual initiator of the query. Upon receiving a positive response, the content is downloaded to the VCC and stored for future requests by "local" peers. Replies by local peers which have a copy of the requested content are not blocked but if the content is also cached, the AVP may include the VCC in the list of replies in order to improve user experience by offering another high-quality source. Requests made by peers outside the ISP's domain are not forwarded by default inside the domain, to avoid local responses and the resulting inter-domain connections.

Being a modular design, the functionality and responsibilities of each AVP component are clearly separated. The AOC contains the basic AVP functions, interfacing with the peers, routing, modifying and forwarding messages and generally interacting with the overlay in real time. The AOC decides whether particular content should be cached and crafts the protocol messages necessary to direct eligible peers to download content from the VCC. Generally, the VCC is "invisible" to regular peers, both inside and outside the control domain, with the former downloading content from the VCC only after an AOC's mediation. As such, the VCC is not discoverable using ordinary P2P protocol procedures but only temporarily made visible as a source to eligible peers during a content transfer.

Focused on its content caching responsibilities, the VCC has a more passive role if seen from a P2P perspective. The VCC manages the storage space available for content caching, the data tables containing the file indexes and relevant information (e.g. file sizes, numbers of requests, last-access times etc.) and the functionality necessary to provide a peer with the requested file or part of file, depending on the P2P protocol in use. Operations pertaining to the management of the cache are self-contained and independent of the AOC. For instance, the execution of file replacement operations to release necessary storage space is done automatically by the VCC without intervention by the AOC.

This separation allows for unlimited deployment flexibility, enabling ISPs to choose between having a single large VCC used by numerous AOCs, each AOC having its own VCC or any combination in between. More importantly, configurations can be changed on-the-fly and new capacity can be added easily. Crucially, the AVP architecture allows for a number of VCCs to be deployed on demand throughout the network. New VCC instances may be spawned on available execution environments in

response to changes in network conditions as identified by the AVP deployment in real time. Thus, caching tasks can be distributed to a number of locations, alleviating temporary hotspots, reducing load and increasing available cache capacity as needed. Similarly, when such support is no longer necessary, additional VCCs may be taken off-line.

More importantly, by separating message manipulation/peer redirection from file serving and storage management, AOCs can treat VCCs as "special" peers and follow strategies where VCCs and regular peers are not considered distinct entities with independent roles, but instead are exploited in combination. Specifically, instead of traditional caching according to demand as inferred from peer requests, the VCC can be used to complement natural replication on peers by explicitly replicating medium-popularity content (i.e. letting peers handle the popular content and improve availability of targeted content) or cache selected types of content, if the ISP wants to offer a value-added service. Taking this one step further, the ISP can supplement a cluster of "normal" VCCs with additional specialised VCCs providing a paid-for P2P CDN service.

The downside of this separation is that an AOC needs to query a VCC for information about the cached content instead of having direct memory access, as in a monolithic design. Nevertheless this leads to a more fault-tolerant, decentralised design. In fact, the P2P-based design means that VCCs have practically no reliability requirements and thus can be run on inexpensive hardware. A VCC failure will have no larger impact on the overlay than the departure of any ordinary peer and because by definition the content cached is popular and thus naturally replicated, the loss of the VCC will not affect service availability. Essentially, all the recovery mechanisms are already provided by the P2P service itself.

Finally, depending on the number of P2P protocols supported by the AVP, a number of different P2P applications may be served by the same cached content. Towards that, the VCC can serve entire files or parts of a file to peers to accommodate the way different P2P protocols operate.

5.2.2 Cache strategies and replacement policies for the VCC

The immutability of most cacheable content exchanged in P2P networks as well as the fact that peers will generally obtain that content only once, as discussed in Chapter 3, raise the question whether established web-caching techniques, designed for

content which is less immutable but may be served to the same recipient repeatedly, can be effectively transferred to the P2P domain. Consequently, existing cache replacement policies for the web need to be reassessed for applicability in P2P scenarios and new ones which will take advantage of the unique characteristics of P2P applications may need to be identified. The choice of the appropriate replacement policy, which will retain in the cache the "right" popular documents and replace rarely used ones, will have a direct effect on the performance and value of the VCC and any P2P caching approach for that matter.

Cache replacement policy performance is typically evaluated by object hit-rate (also called "hit-ratio" and used interchangeably hereafter) and byte hit-rate. Object hit-rate denotes the number of cache hits for a cached object as a percentage of the total number of requests for that object. Policies which achieve a high object hit-rate generally favour the caching of small objects as this allows the storage of more objects for the given cache capacity. Byte hit-rate denotes the number of bytes transferred from the cache as a percentage of the total number of bytes resulting from all object requests. Caching of bigger objects tends to increase the byte hit-rate.

For the prototype implementation of the VCC and its subsequent evaluation, the following cache strategies/replacement policies have been considered:

- LRU (Least Recently Used): The file with the oldest last-request timestamp in the cache is removed.
- LFU (Least Frequently Used): The least-frequently requested cached file is removed.
- LRUSS (Least Recently Used of Similar Size): The file with the oldest last-request timestamp in the cache with size similar to that of the file that needs to be stored is evicted.
- LFUSS (Least Frequently Used of Similar Size): The least-requested file in the cache with size similar to that of the file that needs to be stored is evicted.
- LFUTS (Least Frequently Used Threshold Smaller): In this cache strategy only files smaller than a defined threshold value are considered for caching. Eviction is then decided by applying LFU on the group of cached files.
- LFUTL (Least Frequently Used Threshold Larger): Same as LFUTS but with the difference that only files larger than the threshold are considered for caching.
- ILR (Intelligent Least Requested): This policy considers for removal only those cached files whose number of requests is equal or less than this of the file to be

cached. It then applies an improved version of LFU to the candidate list to select the appropriate file for eviction.

- ALR (Averaged Least Requested): This policy calculates the average number of requests attracted by all files currently in the cache and considers for removal only those cached files whose number of requests is equal or less than the average. It then applies an improved version of LFU to the candidate list to select the file to be evicted.

The LRU policy releases capacity by removing the file that has stayed in the cache the longest since attracting a request. This serves as an indication that its relative popularity has dropped and will not offer further significant gains by being retained in the cache. This concept is known as temporal locality of reference and characterises the ability to predict future accesses to objects from past accesses [Podlipnig, 2003]. LRU is a common replacement strategy not only in the web caching domain but also in CPU instruction and virtual memory caching.

The LFU policy operates similarly but considers the number of requests a file attracts instead of its last-access time. In that, the cached file with the lowest number of requests is removed. This policy is also common in the web caching domain [Cao, 1997].

The typical size of P2P content makes the decision whether to retain or evict a file from the cache more critical than in web caching, as the associated costs of transferring and storing that file are much higher. Similarly, an erroneous decision to evict a file only to re-cache it later will have a large impact on traffic savings. For the typically small size (less than 1 MB) of cacheable web objects the current state-of-the-art in storage devices means that capacity is rarely the limiting factor and replacement operations need not occur frequently. As a result, web caching strategies can be optimised to improve factors such as access latency (e.g. Lowest Latency First policy [Cao, 1997]) or may differentiate cached objects according to the day they were cached (e.g. Pitkow/Recker [Pitkow, 1994]) without sacrificing basic object hit-rate performance. P2P policies, on the other hand, primarily need to make optimal decisions regarding transit traffic reduction, indicating that the metrics of importance are byte hit-rate and efficiency of storage space utilisation.

For that reason, apart from the well-known and widely employed LRU and LFU which can also serve as benchmarks, additional policies have been proposed for the

AVP in order to examine whether they can serve as better-suited alternatives for P2P workloads.

LRUSS and LFUSS are identical to LRU and LFU respectively, with the difference that they take into account the amount of space that needs to be released for the upcoming object storage as well. Specifically, these policies create a list of candidates from those cached files whose size is roughly (typically within ±1% of target size in MB) the same to the space required for the soon-to-be cached file. Then, they determine the file to be evicted inside that group based on its last-request time for LRUSS or popularity for LFUSS. As P2P content is typically much larger in size than web content, policies that take file size into account have the potential to be highly effective. For instance, they can avoid cases where a large file, which involved a significant cost to be cached but offers equivalent traffic savings, is erased to make room for a much smaller file based only on relative hit-count. Similarly, to obtain space for a large file, a simple replacement policy may dictate the deletion of a number of smaller files which could collectively offer higher traffic savings than if a single large file was removed. The validity of these assumptions will be put to test in Chapter 7.

ILR takes a proactive view on caching, and especially cache admission, by maintaining information on both cached and non-cached files available in the network, extracted from the query and reply messages intercepted by AOCs. The privileged position of the latter in the overlay allows them to track requests by both local and foreign peers and keep file popularity scores based on them regardless of local peer request activity. This information can then be used to make informed decisions on whether it is worthy to cache a requested file or allow the peer fetch it without mediation. The rationale behind this approach is that it may be more costly to cache files ultimately belonging in the "long tail"[35] at the expense of evicting existing files, than allow peers transfer them on their own. The transit traffic costs due to the unpopular file's transfer will be comparable in both cases (i.e. transfer to cache from external source and then to peer, or unmediated transfer to peer from external source) while the occupied cache space could be used for an equally or more popular, already

[35] Of course, the future popularity of a file is not known at the time of request. The caching of an unpopular file is thus considered a non-optimal caching decision.

cached (and thus transferred) file. While all schemes examined so far treat all requests equally by caching any file and rely on the replacement policy to retroactively judge which of the cached files are the most valuable to retain, ILR will only cache a file if there is an indication that it will offer more savings than retaining an already cached file. With ILR, cache space is treated as the scarcest resource and the strategy intelligence attempts to eliminate non-optimal caching decisions.

What is important is that despite avoiding to cache newly-appeared files before their popularity is proven, the external traffic costs will remain comparable to the other schemes due to the existence of the AOCs. Specifically, while a file transferred from an external source may not be cached by a VCC outright, the AOCs can redirect future requests to the local peer that initially fetched it at no further external traffic cost. Only if there is no reliable local source (e.g. the local peer left the network) a new inter-domain transfer may be allowed. If, in the meantime, the file proves popular enough it is cached, ensuring a highly-available local copy. Thus, with ILR a VCC essentially offloads some of its short-term caching responsibilities to local peers until a file's caching value is determined, with ideally no negative effect on external traffic.

It has to be noted that ILR is not slow to identify really popular files. Because requests by foreign peers that are captured by AOCs are included in monitoring a file's popularity, newly appeared files that attract flash crowd behaviour can be identified and cached early on, even if the ISP's local customer base has not yet started to join the crowd (e.g. due to different time zones, etc).

File eviction is handled by applying a modified version of the LFU policy on the subset of cached files whose number of requests is equal or smaller to that of the file to be cached. The aforementioned modifications are intended to improve the basic LFU algorithm by adding a safeguard against cache pollution and a refinement of evictee selection in the case of similar request counts. Cache pollution occurs when older files which do not hold caching value any longer have already built a high request count due to past popularity and manage to remain cached leading to incorrect evictions. In the modified LFU, if a file does not attract any hits inside a period T (typically 1800 seconds – 30 minutes) it has its hit count halved, similarly in principle to LFU-Aging [Arlitt, 2000]. The second modification dictates that in the case of a tie in selecting the appropriate evictee (i.e. more than one file with the same number of requests), the one with the oldest last-request timestamp is removed.

Given the large number of unique objects present in a P2P file-sharing network, it is not scalable or efficient for the AVPs to maintain permanent information on all of

them. Instead, a moving monitoring window of size T is employed to reduce the amount of information required for employing ILR. The monitoring window operates on the premise that files with actual caching value at a given moment will show current request activity (i.e. temporal locality of requests). Indeed, given the cache capacity scarcity compared to workload size, taking into consideration very old statistics can lead to sub-optimal performance and cache pollution from once-popular files. A limited amount of information is stored on disk (a hash of the file name and contents along with a request counter and update timestamp) allowing the resulting database to remain manageable while holding information on a large number of objects[36].

ALR shares the same selective caching philosophy as ILR but applies slightly more complex criteria to assess caching suitability and identify the appropriate evictees. ALR periodically calculates the average number of hits attracted by all the files currently in the cache and uses that as an indication of the value carried by currently cached objects. Upon a file request, its number of accumulated requests R_f as monitored by the AOCs is compared with that average. If R_f is equal or higher to the average, the file is deemed eligible for caching; otherwise it is assumed that the current group of cached files carries more value than the requested file and the requesting peer is allowed to fetch it from the foreign source without mediation. If the file is eligible, the next step is the identification of the appropriate evictee(s). Any cached files that have attracted a lower or equal number of hits to the current average are inserted into a list where the modified LFU policy described earlier is applied. It is possible that the set of candidates is such that the replacement operation does not release enough space for the new file to be cached. In this case, the requesting peer is again allowed to fetch it on its own without mediation. In conclusion, in ALR a file is cached if two requirements are met: (i) the file has attracted an equal or higher number of requests to the current cache average, and (ii) a suitable evictee or group of evictees is found.

LFUTS and LFUTL enable a simpler form of selective caching compared to ILR and ALR. They both employ a (configurable) threshold value T which determines whether a file may be admitted in the cache or not based on its size. In LFUTS only

[36] Similar indexes are maintained by many centralised or semi-centralised P2P systems (e.g. Napster, eMule) demonstrating that with careful design they can be viable.

files with size smaller than T are cached, while LFUTL does the opposite (i.e. only files larger than T are cached). The LFU policy is applied on the group of cached files to determine the appropriate evictee.

5.2.3 Chunk versus full file caching

It becomes evident that the proposed policies deal with entire files and not individual file parts (chunks). In contrast, recent work ([Wierzbicki, 2004; Saleh, 2006]) proposed the use of file chunk caching. In that, file chunks constitute the smallest caching unit allowing only portions of a file to be cached. Consequently, in chunk caching replacement policies operate on individual chunks and not entire files.

The main strength of chunk caching is its inherently better management of available cache space in the face of user aborts or otherwise uncompleted file requests. Under full file caching, an intercepted request may lead the cache to transfer and store a large file unnecessarily if the requestor aborts the transfer prematurely. Chunk caching, in contrast, is not affected since the external transfer can be stopped as soon as or shortly after it is determined that the requestor left. The result is less external traffic due to aborted requests and ideally maximisation of byte hit rate due to focusing on bytes and not objects.

However, the support of chunk caching involves significantly higher overheads for maintaining popularity and other relevant metadata for each chunk instead of a single set per file. More importantly, functionality to identify and deal with overlapping ranges of a file, cached at different times, needs to be implemented. Given that chunk sizes may vary not only between protocols but also between objects[37], there is also the issue of deciding what the optimal size of the caching unit should be for the vast variety of objects shared. In contrast, the full file caching approach involves much lower overheads, does not involve functionality to deal with overlapping ranges and the required space to be freed in the cache can be known in advance for policies that take file size into account.

[37] For example, files distributed in different BitTorrent torrents may be split in different chunk sizes.

Furthermore, assuming that in normal usage scenarios every part of a file will eventually be requested (since an incomplete file may be unusable) demand for all parts is uniformly distributed. This is confirmed in [Wierzbicki, 2004], where using actual traces of Kazaa traffic the authors note that range requests are short and ask for any portion of the file. In other words, the advantage offered by chunk caching may only be applicable when dealing with unpopular requests that are consistently aborted. In other cases, whether the transfer is done over a number of steps or all at once, the end result will be fetching the same amount of bytes from the foreign source (i.e. the object's size).

The superiority of each approach therefore depends on the amount and similarity of user aborts experienced in a P2P network. If that number is low in relation to successfully completed requests, both approaches will have comparable performance with chunk caching requiring increased complexity. If in contrast aborts are commonplace, that complexity may be acceptable due to the inefficient use of cache space by full file caching.

The author is not aware of any study comparing cache performance in the face of user aborts using traces collected from different P2P networks, to eliminate bias effects and demonstrate consistent abort behaviour. Consequently, for the selection of a caching scheme for the VCC, no assumptions were made on the level of aborts. However, the design and functional requirements of the AVP framework indicated that full file caching is more appropriate. Although the VCC can serve both entire files and file ranges according to the P2P protocol in use, thus essentially supporting both types of caching, cache objects are managed at the file level and policies apply to fully downloaded files. This is to enable support for multiple P2P protocols as different applications may use different chunk sizes or modes of chuck request. In order to operate transparently with different P2P protocols and users, the VCC cannot negotiate with peers the serving of selected file ranges it has already cached instead of the ones originally requested, as many protocols do not support this feature. Therefore, the VCC trades overspecialisation to a single protocol for wider applicability. The effect of inefficient caching decisions due to user aborts is acknowledged and partially dealt with the inclusion of policies like ILR, and ALR which attempt to eliminate incorrect replacement operations, as well as by employing a hybrid admittance approach. In that, a caching operation is halted and the reserved cache space freed if an abort occurs before the requesting peer received at least 50% of the file size in bytes and no other peer has made a request for the same file in the meantime.

5.2.4 Distributed VCC caching

For clarity, all discussion of the VCC and its policies so far implied the use of a single cache. However, the strength of the AVP framework lies with its support for the deployment of multiple VCC components throughout the network. As mentioned earlier, this permits caching tasks to be spread along a number of machines and locations, alleviating temporary hotspots, reducing both peer and cache load and increasing available cache capacity as needed with no downtime. VCC caching therefore offers high scalability while the lack of requirement for specialised hardware makes the provision of additional capacity straightforward. The decentralisation of the caching architecture in the form of a multiple-VCC deployment can be further exploited to support different caching models. Specifically, VCCs may operate as a distributed cluster coordinated by AOCs or they may additionally support cooperative caching.

In the first approach only a single copy of a particular file exists in any cache, ensuring maximum utilisation of available cache capacity. That is, each VCC holds a different set of files from any other. The forwarding of a local request to the right VCC as well as admission decisions are managed by the AOC handling a particular request. An AOC will interrogate each VCC in parallel for the requested file[38] and if it is present in a VCC, it will send a modified reply message to the requestor as discussed in the design overview. If the file is not cached and is to be fetched from a foreign source, the AOC needs to determine in which of the VCCs it will be best stored. The algorithm utilised to evaluate caching suitability takes into account cache space availability, RTT (Round Trip Time) delay between VCC and requesting peer and VCC load, giving a different weight to each factor. If `C` is the capacity availability, `D` the round-trip delay between VCC and peer in milliseconds and `L` the reported VCC load as a percentile value of its total capacity, VCC selection is carried out as described by the following high-level pseudocode:

---

[38] It is assumed that even the largest ISPs will not need to run such a large number of VCCs that this kind of communication will prove non-scalable. Nevertheless, if it proves problematic, AOCs may maintain a short cache of previous VCC replies or a more efficient indexing scheme to minimise communication costs.

```
for each VCC V:
    L_total += L_V
    if C_V is TRUE
        insert V to candidate list S
    else
        insert V to candidate list S'
end for

L_avg = L_total / (number of VCCs)
for each VCC V:
    if absolute(L_V - L_avg) <= 10
        balance = TRUE
    else
        balance = FALSE
end for

if S has only one member
    return this VCC
else if S has more than one member
    if balance is TRUE
        sort S by D_V from lowest to highest
    else
        sort S by L_V from lowest to highest
    return top member
else
    if balance is TRUE
        sort S' by D_V from lowest to highest
    else
        sort S' by L_V from lowest to highest
    return top member
```

C indicates the ability to cache a file without requiring a corresponding eviction when adequate free space is available. D serves as an indication of network proximity (as noted for example in [Obraczka, 2000]) as well as achievable throughput[39], which in turn may be translated into download performance for the peer. Finally, measuring VCC load ensures that performance bottlenecks due to overloaded VCCs can be avoided and load is distributed evenly throughout the infrastructure. For simplicity, L can be estimated from the number of concurrent file transfers served by each VCC (indicating link load indirectly as well) assuming each one of them consumes a determined amount

[39] The (inverse) relationship between latency and throughput has been reported extensively in the literature (for instance in [Padhye, 1998]).

of CPU, memory and I/O (input/output) resources, or a more detailed method can be used to represent both hardware and network load.

The algorithm maximises cache capacity utilisation and number of cached objects by overruling load and delay factors when a file can be cached without requiring a corresponding eviction. In any other case, the algorithm calculates the average load currently experienced in the deployment and examines whether any of the VCCs has more than 10 units of load difference from the average. This signifies load imbalance in the deployment and is addressed by nominating the less loaded VCC as the destination cache. When VCC load levels are comparable indicating balance, the algorithm selects the VCC with the lowest round-trip delay to the inquiring peer, optimising for proximity.

What this approach achieves is the distribution of caching load in different parts of the overlay, avoiding the case of a single VCC becoming a bottleneck. Furthermore, assuming that the ability to host EEPs inside the ISP network is not limited by external factors, there is high flexibility in where to place VCCs both for upgrading and maintaining the infrastructure. Indeed, the AVP architecture offers the ability to activate or deactivate VCCs with a few administrative commands. Since the available caching capacity is treated as a single unified virtual cache and replication is avoided, the cumulative performance is estimated to be close to that of a single cache of equal size, factoring in some losses due to each VCC applying replacement policies independently and imperfect coordination.

The second approach is to enable a closer cooperation between VCCs in caching high-value files. In the context of web caching, the term cooperative caching denotes the formation of a hierarchy of caches where each consecutive miss leads to interrogating a cache further up the hierarchy until the content is found or it is determined it needs to be fetched from its original source. For the AVP, cooperative VCC caching describes the replication of very popular objects amongst VCCs with the aim of exploiting locality advantages inside the ISP network.

While the independent VCC approach maximises capacity utilisation by avoiding replication of cached objects amongst the caches, it does not explicitly deal with flash crowds or activity hotspots that might emerge inside the network. A particularly popular object is still served from a single VCC, which may prove a performance bottleneck for the time period required for enough copies to be naturally replicated on local peers to assist distribution. This issue can be addressed by creating replicas of such objects on other VCCs. Furthermore, if the very popular content is

brought closer to demand, traffic can be localised and proximity advantages such as improved throughput can be achieved. Sizeable ISPs therefore may wish to employ cooperative VCC caching as part of their broader P2P traffic engineering strategy.

Under cooperative VCC caching, each VCC contributes a fraction of its total capacity for the replication of the currently most popular objects cached throughout the infrastructure. Specifically, each VCC is periodically asked to report the following information for the top `n%` (typically 1-5% but this can be adjusted to reflect cache size, number of VCCs, etc) most popular files in its cache at that moment:

- File identifier information
- Total number of hits attracted by file
- Number of hits attracted by file since last update
- Current total VCC load
- Portion of load experienced due to serving the particular file (e.g. ratio of total load over number of active transfers of that file)
- Change in load since last update

The queries are typically made by a designated AOC, but for redundancy more than one may be assigned that role. The results from all the VCCs are collected and ranked into a global list using the following high-level algorithm:

1. If a VCC is nearing overload, relieve it by prioritising replication of its heaviest hitters. The number of "urgent" replicas is determined by the amount of load reduction needed to bring the VCC to either a typical non-overloaded state (for instance, no more than 65% load) or to average VCC load as reported in that round.
2. Replicate the top hitters from each VCC in a round-robin fashion.

The processed list is then forwarded to all VCCs, which attempt to cache as many items as their contributed space can hold. It is possible that some of these entries may be already cached in VCCs from an earlier round. In that case, the VCC selects the next suitable entry until the space is filled or the list exhausted.

5.3 Implementation of the VCC prototype

Similarly to the AOC, the VCC component is developed as a proxylet that runs on the ALAN funnelWeb platform [ALAN]. As such, it is written in the Java programming language. Although the VCC is designed so that multiple P2P protocols can be handled, the prototype implementation supports the Gnutella protocol only. However, no protocol-specific features are used in any of the core functions, maintaining extensibility.

The VCC proxylet design can be separated into the following internal components, as illustrated in Figure 28:

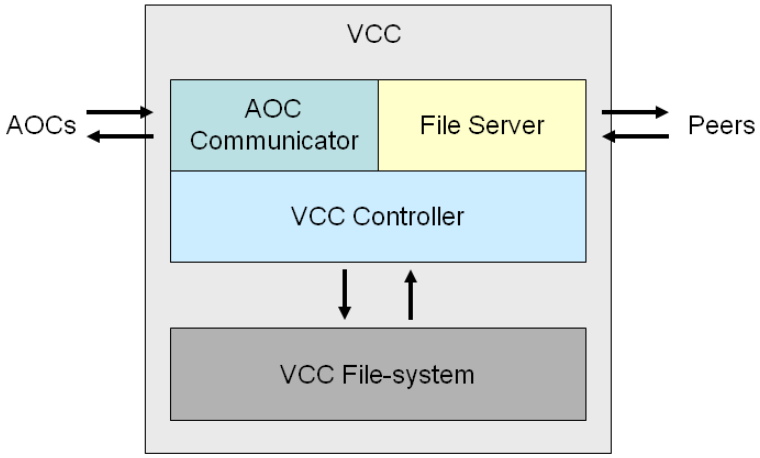

Figure 28: VCC component diagram.

- An *AOC Communicator* which provides the communication interface with an AOC. The AOC Communicator listens for incoming requests from an AOC, interprets them and carries out the requested action by calling the appropriate functions of the VCC Controller.
- A *File Server* which handles the serving of files to remote peers. The file server incorporates the following:
- A Download Request Handler, which listens for and interprets requests by peers and responds with an appropriate HTTP (or other protocol-specific) response. Both full and partial file requests can be interpreted. For Gnutella 0.6 protocol support, HTTP is used.
- Functionality for uploading the requested file or file range to the requesting party.
- The *VCC Controller*. This component contains the basic functional elements of the VCC. These are:

- File Download Manager: Manages the request and download of a specific file from a peer for storage in the cache. The prototype supports the Gnutella 0.6 protocol.
- File Replacement: Handles the process of cache file replacement according to the replacement policy in place. The replacement strategies for the prototype were presented in the previous section.
- File Indexer: Indexes and maintains information on all the files stored in the cache. An index containing file-specific information (e.g. file name, size, number of requests etc.) as well as any available metadata is stored locally.

5.4 Legal considerations of caching P2P traffic

The work presented in this chapter as part of the AVP framework began in 2002 and was one of the first that proposed caching of P2P traffic [Koulouris, 2003] to alleviate its impact on the network, when most ISPs were still uncertain of how to respond to this then new class of network applications. In many occasions when the work was presented to the research community, the issue of ISP liability due to caching copyrighted material as part of P2P traffic was raised. The author believes that the discussion of this matter falls outside the scope of this chapter, which is the discussion of the design and technology of the VCC, as it is predominantly an issue of interpreting copyright law, which not only is a matter of ongoing discussion in the face of rapid social change brought by technological progress, but also requires an extensive legal background. Nevertheless, it is useful to acknowledge these concerns and briefly examine their roots.

P2P file-sharing networks entered the public eye in a rather negative light, due to allegations by organisations representing copyright holders that the former facilitated intellectual property infringement and were responsible for large financial losses. Specifically, it was claimed that "the transmission of a file containing copyrighted works from one person to another results in a reproduction, a distribution, and possibly a public performance (in the world of copyright law, "public performance" includes the act of transmitting a copyrighted work to the public)" [Von Lohmann, 2003]. Initially the closing-down of Napster and other non-fully decentralised P2P networks and later the much publicised legal action against end-users (and academic institutions) polarised the public and perplexed many of the stakeholders on what their rights and responsibilities are. ISPs in particular appeared to be uncertain whether by

implementing network engineering and optimisation in regards to P2P traffic, they could be held liable of copyright infringement.

From a technical standpoint, P2P traffic caching is no different than caching of web traffic, as most of the content present in a web page (e.g. images, text, multimedia etc) is copyrighted by some person or organisation. Since P2P networking can be reduced to a family of networking protocols (like HTTP), it can be presumed that ISPs are eligible to the same protection they enjoyed for years for web caching. Indeed, and again to the author's best interpretation, in the European Union [European Union, 2000] and countries like the United States [Online Copyright Infringement Liability Limitation Act, 1998] and Australia [Australian Copyright Act, 1968], ISPs are exempted from liability for caching copyrighted content, if that is a result of user activity. This "safe harbour" is provided if the ISP does not explicitly target, select or modify copyrighted material, none of which is made possible by the VCC.

The fact that a number of ISPs in the UK (e.g. NTL [Anon, 2006]) and elsewhere (e.g. "True Internet" [Anon, 2007]) currently implement P2P caching solutions indicates that caching of P2P traffic, including as performed by the VCC, does not constitute copyright infringement on behalf of ISPs.

5.5 Summary

This chapter presented the content caching capabilities of the AVP, as provided by the VCC component. The rationale behind offering such a capability was discussed along with a basic deployment scenario explaining how such a system operates. Then, a number of existing, customised and novel cache replacement strategies for the VCC were examined. This set the tone for the discussion of different caching approaches such as full or partial file caching as well as advanced VCC capabilities like multiple-VCC deployment. The implementation of the VCC prototype proxylet followed. The chapter was concluded with a brief examination of potential legal considerations from the use of P2P caching, and how the VCC is affected by them.

During the discussion of VCC replacement policies and advanced features, a number of assumptions were made on cache design and performance, which need to be validated. The AVP simulator, presented in the next chapter, will provide the necessary tool to achieve that goal.

6. THE AVP SIMULATOR

6.1 Introduction

Modern science employs simulations to examine systems that cannot be practically examined physically, due to sheer complexity, size or other obstacles; usually with the help of computers. Central to simulation is the concept of "model". A model is the portrayal of the interrelationships of parts of a system in precise terms. This portrayal can be interpreted in terms of some system attributes and is sufficiently detailed to permit study under a variety of circumstances [Law, 2000]. Simulation is the "execution" of such a model, so that valuable information can be collected about the system and its future behaviour predicted. Many types of models and simulation approaches exist, some of which will be discussed later on. Ideally, simulations complement results or observations gathered from experiments, as by their very nature they use models - approximations of systems' attributes - to understand the behaviour of these systems. Therefore, it is a common approach to use simulation to account for the universal behaviour of a system and run experiments where possible to complement the simulation results for subsets of the system's overall behaviour [Floyd, 2001].

There are five necessary conditions for obtaining credible results from a simulation: (i) having an adequate understanding of the problem to be solved, (ii) having a correct model, (iii) using a valid simulation program, (iv) executing a valid simulation experiment, and (v) making a correct interpretation of the results [Page, 1994]. A simulation program is valid, if it is a verified computer program (i.e. the program performs as intended) capable of accurately representing the simulation model and handling every parameter. Consequently, it is important that a P2P simulator is flexible and adaptable enough for expressing models of P2P behaviour, and especially those characteristics that are not encountered in more traditional types of networks.

This chapter presents the AVP simulator, a software simulator developed to evaluate the AVP concept. First, the rationale behind the development of a purpose-built simulator is explained. The methodology and design decisions behind the software

follow. Next, the simulator architecture and basic functions are presented. Finally, the chapter is concluded with the examination of P2P and AVP protocol simulation models.

6.2 Why develop yet another simulator?

A number of simulators, either purposefully developed for (e.g. P2PSim [P2Psim], PeerSim [PeerSim], GPS [Yang, 2005]), or extended to support P2P networks (e.g. ns-2 [ns2, 2007; Fall, 2006], Narses [Narses]) exist. Additionally, the Overlay Weaver overlay emulator has been developed to assist in the design and testing of P2P protocols [Shudo, 2006]. The question which thus arises is why choose to develop another simulator from scratch when alternatives are readily available. The answer lies with customisation: While all but few claim flexibility and ability to simulate different types of P2P protocols, the fact remains that each tool is influenced by its developer's view of what is important to model in a P2P system. Consequently, most simulators are limited in the flexibility they provide in specifying or modifying the types of statistics to be collected besides the built-in ones, and the variety of parameters that can be specified in the system model. Ting [Ting, 2002] adds the inability to customise the initial network state (connections between the simulated computers and the network delay) of many current P2P simulators as an additional limitation to the level of model detail they can support. Further limitations are introduced when the basic topology mechanisms are fixed. P2PSim and Overlay Weaver, for instance, support DHT-based systems exclusively, making them unsuitable for simulating unstructured overlays. More importantly however, scalability, which is a crucial component of P2P network evaluation, varies widely with a number of simulators incapable of practically handling more than a few thousand nodes[40]. Finally, as pointed by [Naicken, 2007], the poor documentation accompanying many of the available simulators presents a further obstacle in using them more widely.

Special mention needs to be made to ns-2, due to its wide use by the research community. ns-2 is a general packet-level network simulator, with many extensions to

[40] For example, [Naicken, 2006] reports that P2PSim supports a maximum of 3000 nodes while Overlay Weaver cannot practically support more than 2700 nodes.

support a variety of network types and access technologies (e.g. wireless networks). ns-2 is heavily focused on accurately modelling the network, link and physical layers, which is not always critical for P2P applications that are mainly concerned with the application layer. Apart from the additional complexity in specifying the system model, this results in lower scalability and performance since a lot of data, processing power and memory is dedicated to simulate functions that in many P2P models can be reasonably abstracted. The considerable learning curve of ns-2 can also in some cases be a deterrent. For that reason, while ns-2 can generally support highly detailed models, it is not always the optimal choice for simulating any P2P overlay.

Consequently, unless the intended system model is such that it can be adequately described by the built-in parameters of one of the available simulators, and the properties of interest measured by the supported statistics, one can either attempt to modify an existing simulator to implement the missing functionality or develop a suitable simulator from scratch. The code structure and overall design decisions taken by the original developer, in combination with the lack of necessary documentation may in many cases lead to the latter being a "cleaner" and more effective route than the former. In the case of the AVP framework, the need to specify the observable properties of interest and express the AVP system model at the desired level of detail indicated that existing simulators needed to be heavily modified, while not fully addressing scalability concerns. As a result, the development of the AVP simulator was considered a more appropriate solution.

6.3 AVP Simulator design

6.3.1 General principles

The AVP simulator (AVPsim hereafter) was written in the C programming language [Kernighan, 1988] to take advantage of the high performance and portability the language enjoys. Furthermore, it was determined that the broader user base a generic programming language has compared to a simulation-oriented language (for example such as GPSS [Stahl, 1990]) would give the resulting code greater chances of being reused by the scientific community or have its validity examined.

All functions, data structures and software components were designed to be as extensible as possible while at the same time being robust and following good structured programming principles (e.g. principle of least privilege, portable design,

modularity, extensive run-time error checking etc.). As a result, AVPsim is capable of robustly managing very large data sets avoiding the usual "traps" that come with them (memory corruption, unexpected behaviour or termination etc). At the same time, a large number of parameters can be portably set, and different configurations can be used to control the granularity of the simulation, allowing for simulations of a large array of different scenarios.

When developing new simulation software, the designer almost inevitably faces the dilemma of choosing between performance and simplicity of design. Data structures, for example, which are fundamental in simulator design, can either be expressed in a simple manner such as arrays or in the more complex forms of linked lists, trees etc [Kruse, 1998]. Design choices like these can affect the effectiveness of the simulator in many and unforeseen ways [Watkins, 1993].

Apart from the evident trade-off of performance (execution speed or memory usage) and complexity, an unsound design decision can lead to code that is difficult to maintain and review, and ultimately risk the validity of the simulator. Such code will almost certainly contain logical bugs due to the complexity of verifying that all components operate as expected, which will affect the validity of the results [Handley, 2005].

For the AVPsim, having a reliable design which minimises the possibility of logical bugs was considered a priority. Consequently, all components and interactions between them were kept as simple as possible at the expense of possibly lower performance (even when "clever" alternatives existed), in order to make certain that the software is as bug-free as possible and behaves accurately. Code optimisations were incorporated, but only in parts where overall complexity would not increase and after extensive testing to ensure optimisations do not alter operation. This approach, complemented with exhaustive functional- and system-testing rounds gives considerable certainty on the validity of the AVPsim.

6.3.2 Methodology

Two basic approaches generally exist for simulating a system: The stochastic approach, which involves the use of statistical (probabilistic) system models, and the deterministic approach, which corresponds to the use of mechanical (deterministic) system models.

Stochastic simulations generally have a probabilistic component and are used when the local behaviour of a system is only understood in statistical terms (e.g. some of the quantities involved vary in an unpredictable or random fashion), or where it is convenient to model many of its components using statistical or probabilistic models due to extreme system complexity. Because of that probabilistic component, stochastic simulations must be performed a number of times until there is an adequate sample of results to evaluate, since any single "run" of a simulation will have one of many possible outcomes.

Deterministic simulations on the other hand are ideal when one solution set exists for a given input situation. There, all data and relationships are given with certainty. They are primarily used to extrapolate and evaluate outcomes given hypothetical inputs, or to examine the interaction of a number of interdependent deterministic models.

Simulation models are also differentiated on the basis of the granularity of their treatment of time. Models of systems that involve clearly distinguishable events such as the arrival of a customer or the transmission of a packet, are called discrete models. In discrete models, the notion of the "event" is fundamental. It is a significant point in the course of a simulation, where the system state changes. Models where it is impossible to distinguish between specific events taking place are called continuous models. There, time is considered to be an unbroken flow and events cause a marginal change in the system attributes.

Finally, simulations can be categorised as discrete event or trace-driven simulations. In a discrete event simulation, a model is represented as a "box" that has an internal source of random numbers. The random numbers drive the components of the simulation model: They are used to determine when events are to take place, branching probabilities and so on. The essential feature is that the model is self-contained and requires no external inputs to operate.

Trace-driven simulations on the other hand require the use of trace data, generated from a real system to control the input sequences. The trace data are a profile of the system dynamics observed. As such, trace-driven simulations have an advantage over discrete event simulations, since much of the statistical work and any shortcomings that come with it are avoided. Nevertheless, trace-driven simulations usually have a limited scope as it not always easy to obtain a suitable trace for the system properties under examination. For that reason, they are usually confined to performance modelling with the aim of making moderate changes to an already running system.

AVPsim is a discrete event simulator based on a stochastic model. Events in AVPsim represent changes in the state of the system and the entities that comprise it. The primary entities are the peers that join or leave the P2P overlay and with their behaviour affect other peers and the overlay in general. Changes in the system are signified by discrete events that take place during the simulation, ordered by time of occurrence. Table 3 presents the most representative. Events may create other events upon execution, immediately or at a later time. For instance, when a peer exits the overlay in real life, all its connections to other peers are terminated. In the same way, when a "peer exit" event occurs during a simulation, it triggers "end connection" events for each connection the peer in question maintained with its neighbours. These events are inserted in the event list, to be encountered some - simulated - time later.

Event Name	Event Description
New peer	A new peer entity is created and inserted into the overlay
Peer exit	A peer leaves the overlay
New connection	A new connection between two peers is created
End connection	A connection between two peers is terminated
Manage search	File search and download functions such as searching for a file, gathering responses, selecting sources, etc are triggered
Start download	A download session between two peers is started
End download	A download session between two peers is ended
Update	Snapshots of the simulator state are created, consistency checks are performed etc

Table 3: Basic AVPsim events.

A master simulation timer is used to keep track of simulated time. At each value of the timer, the events that are scheduled to take place at that time are executed. Similarly to reality, many different events may occur at the same time. These are processed in the order they were inserted (i.e. in a first-in, first-out basis). When no more events are to occur at a particular time, the simulator timer is incremented to the next value. To summarise, the processing of events consists of the following steps:

1. Determine the first event e_i (the event with the smallest timestamp) of the event list.
2. Set the simulation timer to the timestamp t_i of e_i.
3. Change the state variables according to the effects of e_i.

4. Schedule new events and insert them in the event list if necessary.
5. Remove e_i from the event list.
6. Repeat steps 1 to 5 until no more events remain in the event list or the simulation timer has reached its preset target.

As no event is allowed to have an impact on any event in the past, always simulating the event with the smallest timestamp guarantees causality [Lüthi, 1994].

6.3.3 Random number generation

Using a probabilistic component in a computer simulation requires special attention to be paid on the random number generation regime employed by the simulator. Leaving the paradox of using precise and deterministic machines to produce "random" numbers aside[41] [Knuth, 1981; Press, 1992], it is very important to verify that the random number generator and the statistical distributions used in the simulator software are validated and do not produce correlation errors or other artifacts [Park, 1988]. Use of an improper random number generator, either directly or indirectly by "feeding" a probability distribution function, can significantly impair the validity of the results. Unfortunately, that is the case with most generators provided as part of contemporary programming language standard libraries, including the C programming language "rand" function [Kerninghan, 1988; Press, 1992][42].

For AVPsim, third-party random number generation libraries (Ranlib [ranlib] and Gnu Scientific Library [GSL]) were used for all distribution functions utilised. These libraries were successfully validated using statistical tests [Watkins, 1993; Knuth, 1981] to verify that results gathered by the simulator are not contaminated by random number generator artifacts. Furthermore, all such random number functions are accessed

---

[41] John Von Neumann's quote that "Anyone who considers arithmetical methods of producing random digits is, of course, in a state of sin" sums it up quite nicely.

[42] Computer random number generation is a large research topic requiring extensive discussion of its specifics, which fall outside the scope of this thesis. Interested readers are kindly directed to the bibliography mentioned throughout the section, which should provide a starting point.

from the main program via “wrapper” interfaces to allow the use of a different library if deemed necessary, with minimal changes to the source code.

6.3.4 Topology generation and BRITE

As part of a P2P overlay simulation, the simulator usually has to either generate or import a router-level network topology upon which to build the P2P overlay network. Depending on the granularity of the simulation model and the design decisions taken during the development of the simulation software (e.g. flexibility to add new data structures/functions, memory usage etc.) the latter can be a more desirable approach, as it allows the use of real-world traces or topologies generated by well-known and reviewed software. While AVPsim is designed so that it can be easily extended to generate its own router-level topology information, the preferred approach is to create such a topology using the BRITE topology generator [Medina, 2001b] and import it to AVPsim using the built-in import function.

BRITE (Boston university Representative Internet Topology gEnerator) [Medina, 2001a; Medina, 2001b] is an Internet topology generation framework, developed at Boston University, that focuses on reflecting many aspects of the actual Internet topology (e.g. hierarchical structure, degree distribution etc.) with accuracy, providing many generation models in a single tool and being interoperable with other widely-used simulation applications such as ns-2 [ns2, 2007; Fall, 2006], SSF [SSF] and OMNet++ [OMNet++] as well as visualisation applications (e.g. Otter [Huffaker, 1999]). It is written in both Java and C++, and as such is very portable.

BRITE supports degree-based (i.e. power-law inspired) models as well as older models like Waxman and Transit-stub. Thus, by supporting BRITE, AVPsim can readily employ a large array of network topology models and interoperate with other tools (e.g. GT-ITM [Calvert, 1997]) without the need to implement this functionality from scratch and re-invent the wheel. In any case, acknowledging the arguments against power laws [Chen, 2002] and degree-based methods [Li, 2004] being the final word in representing large-scale Internet structure, the decoupling between Internet and overlay topology in AVPsim allows for new models to be readily incorporated as they appear.

6.3.5 Basic AVPsim data structures

Having imported a router-level network topology, the AVPsim then creates the basic data structures necessary for the simulation of a P2P overlay. Figure 29 illustrates

the basic relationships between the different entities of both the router-level and overlay topologies, and how they correspond to the internal AVPsim data structures.

The *node table*[43] and *edge table* hold information on the network nodes (routers) and edges (links between them) respectively, as inferred from the topology file. For nodes, such information includes location on the network plane, degree, AS correspondence, role (e.g. border or access router), etc. The edge table stores link capacities, minimum propagation delay figures, endpoint node IDs, etc. The *peer list* holds information about all peers present in the overlay at any given time. Each time a peer joins the P2P overlay, a new entry is created in the peer list containing its full profile; peer ID, list of resources it is sharing, on which router in the network plane it corresponds to and other attributes are stored there. Some of these entries, like the list of resources (e.g. files shared), may change over time based on peer activity. Additionally, each peer maintains a record of its active connections to other peers, known as its neighbour list. This list corresponds to entries in the "master" *connection list* (as shown in Figure 30) which holds detailed information on all active overlay connections. This information includes the mappings of overlay connections onto physical links (stored in the edge table) along with their static (e.g. source/destination pair, internal/transit categorisation, etc) and dynamic (e.g. current throughput and RTT, etc) properties. Transfer properties are specific to the payload transferred over the connection, such as file name and size. The relationships of the peer list with the other data structures of the simulator are illustrated in Figure 31.

Finally, connections are categorised as originating and terminating within the "home" ISP (local connections), originating and terminating outside the ISP as well as traversing the ISP domain (differentiating between inbound and outbound transit traffic).

---

[43] Under the adopted naming convention "tables" store data related to the *largely static* router-level topology and "lists" data related to the *constantly changing* overlay topology. This differentiation is used to highlight the existence of two separate layers but does not dictate specific implementation decisions.

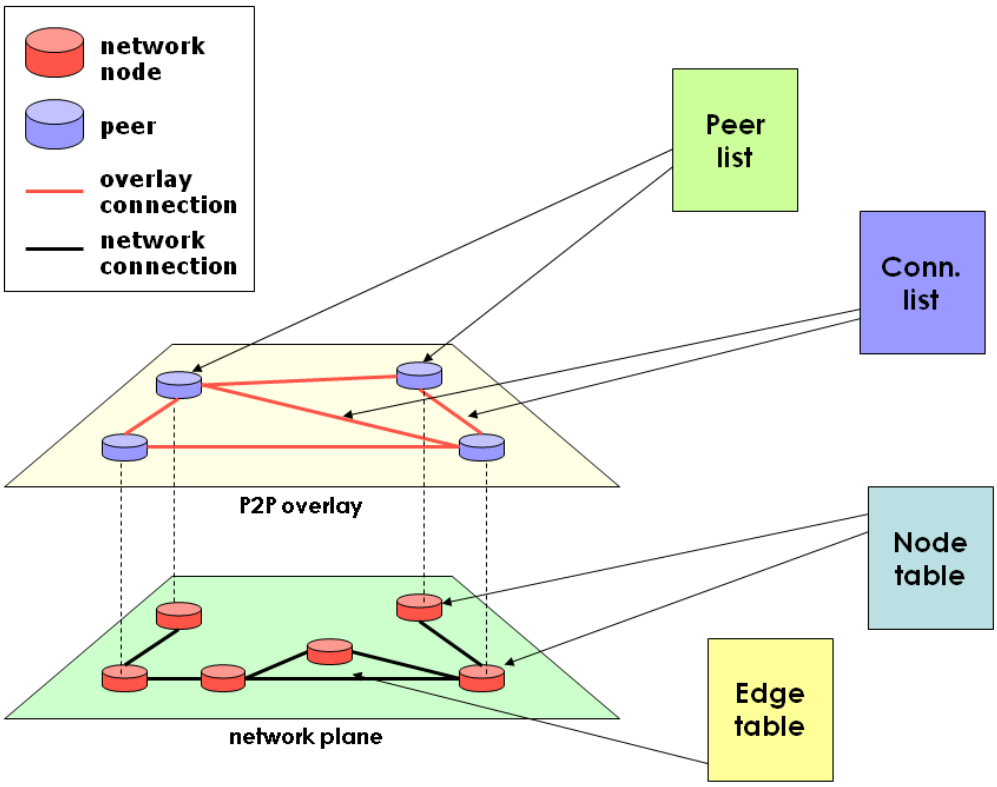

Figure 29: Basic AVPsim data structures and their relationship to the network.

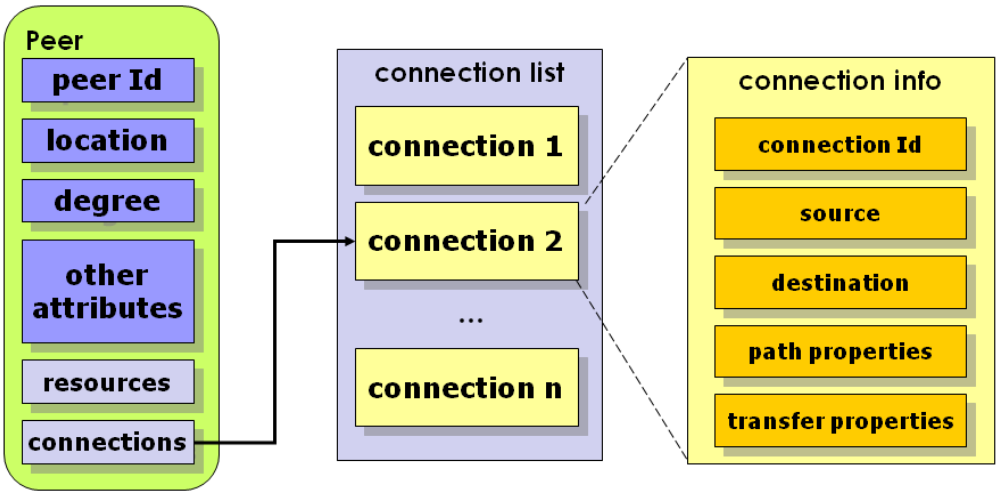

Figure 30: Looking-up overlay connection details.

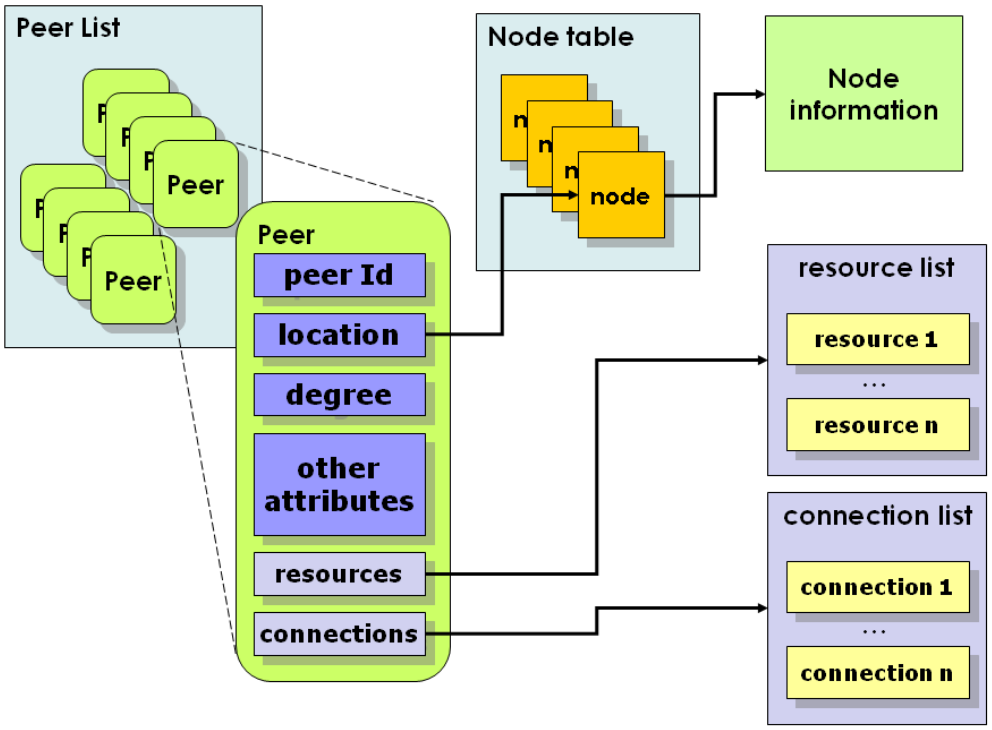

Figure 31: The peer list and its association to other data structures.

6.3.6 Presentation of results

At the end of a simulation run, AVPsim presents on screen and in a text file a summary of the results. The following are the most representative:

- Duration of simulation (in simulated seconds)
- Number of unique peers simulated
- Number of AVPs/VCCs simulated
- Total P2P traffic volume generated
- Internal traffic volume measured
- Transit traffic volume measured (for the "home" ISP)
- Traffic volume completely external to the ISP (originating and terminating on foreign peers) measured
- Total number of downloads (by all peers)
- Number of downloads by local peers
- Number of downloads by local peers from foreign peers

- Number of downloads by foreign peers from local peers
- Number of downloads by foreign peers from foreign peers
- VCC hits[44]
- VCC misses
- Total number of downloads from VCC
- Number of fetches by VCC from local sources[45]
- Number of fetches by VCC from foreign sources
- Amount of bytes served by VCC
- Amount of bytes requested by VCC
- Number of cache replacement operations
- Unsuccessful searches (queries with no successful responses received) made by local and by foreign peers
- AOC redirection operations to local sources

Furthermore, three additional files are created in comma-separated value format (CSV) to allow easier manipulation using third-party software. The first contains information on every file transfer session that took place during the simulation run. Amongst the data logged are the peer identifiers of the two endpoints, average connection delay and rate of the transfer, transfer duration and amount of data exchanged, and number of hops traversed. The second log contains detailed measurements of each type of traffic (total, internal, inter-ISP inbound, inter-ISP outbound, total inter-ISP, external-to-external) recorded at regular intervals. How often a sample is taken is defined by the user (the default value is every 10 simulated seconds). This file can be used to plot traffic utilisation for any or all of the aforementioned types of traffic with adequate granularity. The third file contains overlay statistics, also sampled at regular user-defined intervals. These include the number of peers present in the overlay at the time of sampling, minimum and maximum

[44] VCC-related information is presented per individual VCC simulated.

[45] This field is only relevant if a VCC is allowed to cache content already available locally to improve availability. Under normal scenarios a VCC will only cache content not already available by other local peers, in which case this quantity will be equal to zero.

number of overlay connections observed, average degree, number of active file downloads and percentage of local and foreign peers present. This file is mostly useful for plotting general overlay statistics.

AVPsim also creates a text file listing the values of all modifiable simulation parameters, useful for determining the initial state of a simulation run when it needs to be repeated.

Additionally, if the high-verbosity mode is selected (by compiling AVPsim with the "DEBUG_OUTPUT" option), AVPsim stores in text files the topology data imported from BRITE, the corresponding routing table created along with bandwidth, delay and path length figures for each link and complete overlay snapshots taken at regular intervals. The snapshots are in text form and contain full information about each peer (e.g. peer id, location on network plane, degree, list of shared resources, list of connections etc.) and overlay connection (e.g. source and destination peers, allocated bandwidth etc.) present at the time of the snapshot. These files are generally useful for debugging purposes but can be used to complement the primary results. In that direction, they are structured so that they can be easily parsed or converted to other formats if further manipulation is desired.

Finally, AVPsim contains a function to export node and edge information in a "dot" [Gansner, 2002] or "otter" [Huffaker, 1999] file format so that a graph can be plotted if desired using the GraphViz suite of tools [Ellson, 2003] or Otter respectively. This provides a graphical representation of the imported router-level network topology. The same function may be used to visualise snapshots of the overlay but because the large number of peers and connections present under typical scenarios render the resulting graphs practically illegible, such visual representations have limited utility.

6.4 Simulation model

6.4.1 Connection model

AVPsim distinguishes between two types of connections: protocol connections and transfer connections. Protocol connections carry signalling traffic used to communicate peer queries, send responses and carry out other protocol-specific functions. In protocols like Gnutella such connections determine the immediate neighbourhood of a peer and ultimately its search scope. AVPsim uses higher level mechanisms to simulate peer and resource discovery more efficiently and, thus, uses a

simplified model for protocol connections. In that, they are principally used to indicate endpoint association and are assumed to have a constant data rate during their lifetime. The rate is derived using a normal distribution with a configurable mean μ. As an indication of scale, Qiao et al [Qiao, 2006] report a rate of 200 B/s for control (e.g. Pings and Pongs) and 5-10 KB/s for query-related messages for a leaf Gnutella peer. The lifetimes of protocol connections are typically short, ranging from a few seconds to a few minutes [Ilie, 2004; Qiao, 2006; Azzuna, 2004], and are derived probabilistically using a Pareto distribution (unless they are terminated prematurely by peer departure). While online, a peer will try to maintain an average, protocol-specific number of such connections to other peers and will replace any terminated connections with new ones to other peers it discovers.

Transfer connections are direct connections between two peers, created for the purpose of exchanging content. These are long-duration bulk TCP transfers whose performance depends on path characteristics such as capacity, latency, congestion and packet loss. Transfers are modelled as flows – unidirectional connections between a sender and a receiver over a path which is static for the duration of the transfer (i.e. multiple routes due to traffic engineering or failure are not considered). Each path has a fixed maximum capacity (equal to the bandwidth of the bottleneck link in the path) and minimum delay (equal to the sum of the propagation delays of all mapped links) Additional queuing delay is added depending on congestion to give the path RTT delay. Abrupt source departure notwithstanding, AVPsim calculates the duration of a transfer based on the amount of data to be exchanged plus overheads and the effective throughput of the connection. Overheads are calculated based on a Maximum Segment Size (MSS) of 1460 Bytes and a Maximum Transmission Unit (MTU) of 1500 Bytes. Because AVPsim is not a packet-level simulator, trade-offs were unavoidable in regards to level of model detail and execution time/scalability and the throughput estimation model is less detailed than, for instance, those described in [Padhye, 1998] and [Cardwell, 2000]. Specifically, the focus is on the steady state behaviour of TCP (which is reasonable for the payload sizes and transfer durations involved) and omits connection establishment and slow-start. It is assumed that each flow tries to maximise its own throughput resulting in flows achieving max-min fairness (e.g. [Mo, 2000]) in sharing link bandwidth. Thus, throughput is calculated based on number of competing flows and link bandwidth, starting with the most congested link (lowest share ratio) and recalculating every time a connection is added or removed.

6.4.2 Ordinary peer model

Without delving into too much detail, the peer model can be described as follows. As in reality, the number of peers present in the overlay varies with time. Peer arrivals are independent events, as are searches and download requests made by different peers. Peer arrivals are therefore modelled as a Poisson process where inter-arrival times are determined through an exponential distribution. Peer session durations are determined using Pareto (heavy-tailed) distributions. Finally, peer requests for content follow a bi-modal Zipf distribution to account for the "flattened" head of P2P file popularity, as discussed in Chapter 3. This ensures that the most popular files have lower popularity than a normal Zipf distribution would predict. Peer placement is random. A new peer may be assigned to any router on the node plane with equal probability.

Before joining the overlay, the characteristics of a new peer instance are determined probabilistically based on the particular simulation parameters in use. These include download budget and number of files shared, as well as ISP correspondence and connectivity. Some random noise is added in this process to ensure that the peer profile assigned will not always define the peer's behaviour in the particular session, to account for variability in user behaviour or other external factors. Peer arrival is then signified by creating a new entry in the peer list, holding all information pertaining to this new peer. Then, similarly to reality, the peer joining the overlay enters a bootstrapping phase where it attempts to find a number of existing peers to connect with. The peer is presented with a list of candidates (e.g. as in GwebCache for Gnutella [GWebCache]) and attempts to create connections to them. Since the maximum number of protocol connections a peer can have is controlled by the P2P protocol, peers that already maintain the protocol maximum number of connections will not accept any further. The parameters that determine the minimum, maximum and average number of connections as specified by the P2P protocol are modifiable to accommodate the use of other protocols or protocol versions.

While online, peers will generally attempt to search for and download a number of files. In AVPsim, this is facilitated through the use of specific "Manage search" events that trigger the necessary functions in a way that mirrors real P2P protocol behaviour without modelling interactions at the packet level. These include transmitting the search to a set of neighbours, collecting responses, setting up connections for content transfer and updating peer resource indexes when downloads are completed. If a

peer leaves the overlay before successfully completing any ongoing content transfers, these connections are terminated and the connection lists of its former neighbours updated. Furthermore, future scheduled download events are cancelled.

Although the length of time a simulated peer will stay in the overlay is predetermined in the sense that it is set when the peer entity is created and a corresponding "exit" event is inserted, no assumptions are made by any simulator function on a peer's availability. This way, the essential characteristics of peer autonomy and service unpredictability are preserved. As a consequence, a peer may create connections or initiate a file download and leave moments later, before these functions complete.

6.4.3 AVP model

In accordance with the actual design of the AVP architecture, the simulated AVPs are to a large extent treated by AVPsim like ordinary peers. When an AVP is scheduled to join the overlay, an entry is created in the peer list holding regular peer information such as location, degree, list of connections to other peers etc. A special flag is set to indicate that this peer is an AVP so that AVP-specific functions can be utilised where applicable. Essentially, inside AVPsim an AVP has a superset of a regular peer's properties and shares a lot of common features and functions with it. This not only makes the real-life AVP concept more evident in the simulator design, but allows ordinary peers and AVPs to be treated identically when it comes to regular P2P protocol functions such as searching, creating connections etc.

AVPs differ from ordinary peers in that they are typically assumed to have much longer lifetimes, often as long as the duration of the simulation. Thus, session durations are configured manually, unless the simulation scenario calls for AVP failure. Furthermore, AVP location is determined based on the locations of installed EEPs on the node plane. Typically EEPs are provisioned with higher capacity links. Finally, AVPs will attempt to maintain protocol connections with a larger variety of foreign peers, specifically in diverse ASes in order to ensure good global overlay connectivity.

A consistent approach was followed for the VCC. Since a VCC component communicates directly only with other AOCs and is invisible to regular peer searches, it exists as a separate entity in the simulator memory space and an entry is not created for it in the peer list. AOCs can access the VCC and its data through querying, but ordinary peers cannot discover nor connect to it independently. Ordinary peers may connect to a

VCC only after an AOC re-direction operation. Depending on the simulation scenario, the AOCs monitor peer requests for content and signal the VCC to download and cache content according to demand. Similarly to reality, the simulated VCC monitors its capacity utilisation and enforces cache replacement policies if necessary, independently of the AVPs. The algorithms used to implement the cache replacement policies for the prototype and for the simulator are identical.

6.5 Summary

This chapter presented the AVP simulator. After a brief survey of available P2P network simulators and the rationale behind the decision to develop a new one, the design of AVPsim was outlined. Various design decisions and features were examined, including internal data structures, result presentation and the fundamental characteristics of the connection, AVP and P2P protocol models.

The use of AVPsim features heavily in next chapter, where the evaluation of the AVP framework is presented.

7. EVALUATION OF THE AVP

7.1 Introduction

What traffic savings can be expected from applying topology control? Will peers experience any positive or negative effects from said control? Will caching parts of a contemporary P2P workload make a considerable difference to overall traffic? Which cache strategy performs best for a given set of requirements? What are the implications of deploying multiple VCCs? This chapter examines these and many more questions with the help of a comprehensive simulation model and provides extensive analysis of the results.

The chapter is structured as follows: In the next section, the simulation setup used throughout the chapter is described. Then, the effect of local source promotion on traffic minimisation and application performance is examined. The establishment of the theoretical maximum cache performance follows. Next, the performance of different caching strategies in a single-cache deployment is evaluated using two distinct workload scenarios. Having established the necessary background to caching strategy performance and effects, multiple AVP deployments are evaluated next. The differences between autonomous and cooperative caching are examined using a variety of metrics. This is followed by the investigation of the effect of AVP placement on network and peer performance, along with an exploration of the economic considerations of AVP deployment. Finally, the chapter is completed with the discussion of component testing of the AVP prototype.

7.2 Simulation setup

The impact of the AVP on P2P application operation as well as the performance of different cache replacement strategies and AVP configurations were evaluated using the AVPsim software presented in the previous chapter. For consistency with the prototype implementation, the Gnutella protocol was chosen as the basis of the simulation. As discussed in Chapter 3, most contemporary file-sharing applications

share close similarities in the content download phase and mainly differ in the ways they form their overlay topologies and locate peers and resources. Even when downloads from multiple sources are supported (as is the case with most contemporary applications) the differences lay in the number of concurrent connections maintained or algorithms employed to prioritise file chunk selection (e.g. [Legout, 2005]), but ultimately facilitate the exchange of the entire file (user aborts notwithstanding). For that reason, any observations made using this simulation setup are to a very large extent applicable to a number of different file-sharing protocols besides Gnutella.

A workload of 600,000 unique files[46] was simulated. Files occupied a range of different sizes normally distributed around the values of 2 MB, 5 MB, 80 MB, 350 MB and 700 MB which are common sizes for single non-media files (e.g. photos, text documents etc), single media files (e.g. mp3 files etc), archives of files (e.g. zip compressed files etc), short-duration video content and longer-duration video content respectively. The popularity distribution and replication characteristics of files based on type and size were modelled after the recent data presented in [Stutzbach, 2007] after being cross-examined with an older similar study [Chu, 2002]. Peer session times, inter-arrival times and signalling connection lifetimes were modelled after the data and traces presented in [Qiao, 2006], [Stutzbach, 2005], [Ilie, 2004], [Klemm, 2004] and [Saroiu, 2002]. In general, all simulation parameters were chosen after extensive scrutiny and comparison of available studies and traces as discussed in Chapter 3. Table 4 presents the values assigned to some of the most critical during the runs featured in this chapter.

The underlying network topology was constructed using the BRITE tool, as discussed in the previous chapter. A hierarchical top-down approach was used wherein the top level describes the AS-level topology (i.e. each node represents an AS with edges representing inter-AS links) and the bottom level describes the router-level topology inside each AS. Both the "home" and "foreign" ISPs/ASes were constructed following the same basic router-level topology model with variations in size/number of

[46] A workload of 600,000 unique files is considered adequate for evaluating VCC caching. While many P2P networks claim millions of unique shared files a large percentage of them are duplicates that return a different hash value, have no sharing value (e.g. thumbnail files, metadata files, other operating system-related files etc that are shared along with other content) or are fakes/malware. Including such files in the simulation workload offers no direct advantages while inflating computational resource requirements.

routers. As illustrated in Figure 32, this model contains a number of access (or edge) routers located at each Point-of-Presence (PoP) handling the access network traffic handed-over from the local loop[47], interconnected by backbone routers. Border (or gateway) routers provide connectivity with the rest of the Internet via different ASes for resilience. Aggregation (or distribution) routers are also included to realistically reflect contemporary ISP designs, as for instance described in [Cisco, 2005] and gathered from the findings of the Rocketfuel project [Sprint, 2002]. All internal links are assigned 2.488 Gbps (e.g. OC-48) capacity and transit links 1 Gbps (Gigabit Ethernet). For readability, most link and router redundancy is omitted from the figure but is present in the synthetic topology. Last-mile link capacities between 256/512 Kbps and 1/8 Mbps (uplink/downlink) were assumed, to reflect current consumer offerings by ISPs[48] for ADSL broadband access.

Parameter	Value
Peer arrival rate (Exponential)	λ=4
Peer session duration (Pareto)	α=1.09, x_m=0.85
Protocol connection duration (Pareto)	α=1.88, x_m=0.87
File popularity (Zipf)	head=0.65, body=1.18
Number of global routers	10,000
Number of ASes	990

Table 4: Key simulation parameters of the featured runs

Each simulation run was repeated three times with different random number generator seeds to eliminate the possibility of bias. As such, unless stated otherwise, all results are given as the arithmetic mean calculated from the three runs along with their standard deviation. Where that would affect the legibility or ability of the reader to easily analyse the results or graphs, Pearson correlation coefficients are instead provided to indicate the degree of correlation of simulation results from the three runs. These

[47] No distinction is made between ISPs employing LLU (Local Loop Unbundling) or using a wholesale provider to reach the customer premises.

[48] Based on broadband deployment data collected from http://www.thinkbroadband.com and http://www.broadbandperformance.co.uk/availabilitymap.aspx.

coefficients, denoted by R^2, were calculated using Formula (1) below, where x_i is the measurement, m_x the sample mean, x_i' the value of the measurement minus the mean and N the sample size:

$$R^2 = 1 - \frac{\sum_{i=1}^{N} (x_i')^2}{\sum_{i=1}^{N} (x_i - m_x)^2} \quad (1)$$

Finally, although all the results presented in this chapter were generated using a single network topology for consistency, numerous simulation runs were performed with different topologies to ensure that bias was not introduced due to BRITE topology artifacts.

In order to give a sense of the scale of the system simulated, for the specific set of simulation parameters and topology used 345,534 unique peers were simulated over a period of 24 hours. In normal operation they exchanged 55.3 TB of data over 1,361,265 download sessions globally, with peers belonging to the "home" ISP transferring approximately 3 TB over transit links and 324.8 GB internally.

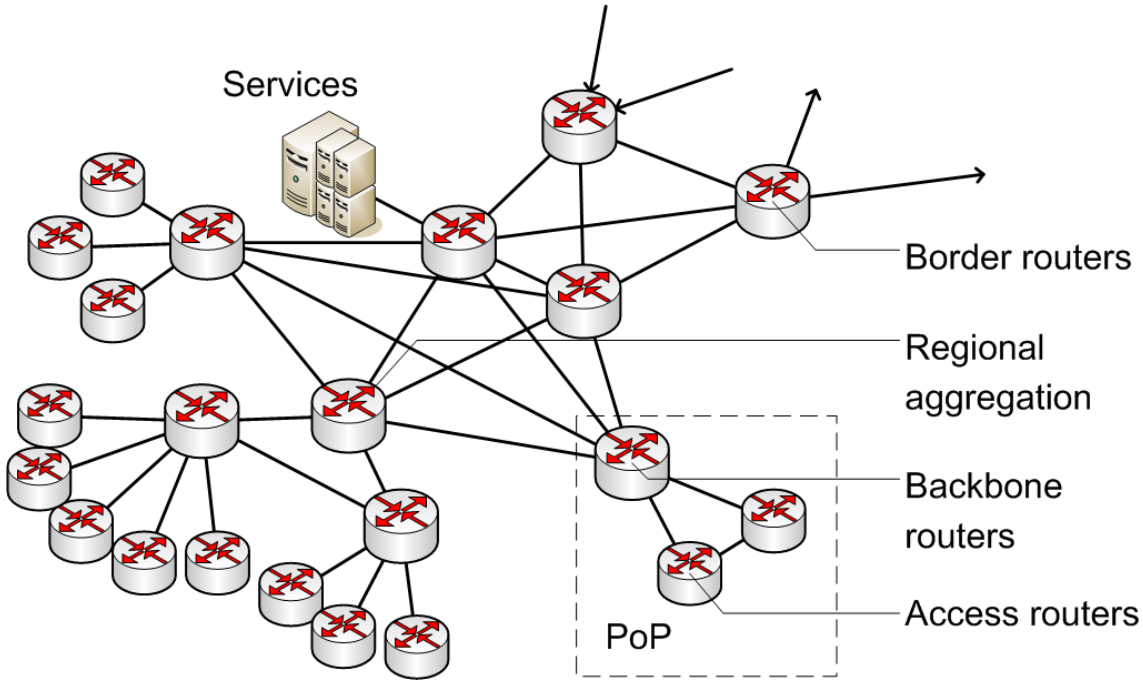

Figure 32: Basic ISP router topology model.

7.3 Evaluation of AOC routing/topology control and application performance improvement

As discussed in Chapters 4 and 5, the AVP accomplishes P2P content caching through the combination of protocol message manipulation and peer redirection carried out by the AOCs and the provision of object storage/serving capabilities in the form of

VCCs. The redirection mechanism in particular is not limited to simple VCC support but instead is a core AVP routing/topology control capability which operates regardless of VCC presence. By leveraging the ISP's privileged knowledge of the underlying network infrastructure and their ability to obtain accurate measurements much more effectively than peers on their own, AOCs can direct peers towards optimal paths and appropriate neighbours and thus help them avoid costly and inefficient peering decisions.

This section examines the extent of P2P application transfer-related performance improvement brought by AOC peer selection biasing compared to regular, random peering. Towards that, the same synthetic Internet topology was simulated for a period of 24 hours under normal P2P application operation (i.e. no AVPs present) as well as when an AVP consisting of a single AOC component (i.e. no VCC caching) was deployed in the "home" AS. All other parameters (e.g. number of peers, location of peers, query workload, capabilities, shares etc.) were identical to allow for direct comparison of results. Source evaluation was based solely on RTT measurement between inquiring peer and available sources and local/foreign peer identification, to capture the general case without assuming advanced AVP capabilities such as the presence of a NOC. As a result, in the case of foreign sources only RTT was taken into account regardless of AS hop distance.

Table 5 highlights some of the most representative results collected from both scenarios. Specifically, it presents the total volume of transit traffic measured throughout the 24-hour period, the number of queries made by local peers that were resolved by other local peers (also given as a percentage of all successful locally-initiated queries), the number of queries made by local peers that were not replied to by any local or foreign peer, the average (arithmetic mean) round-trip delay experienced on a connection initiated by a local peer, the average number of IP-level hops a connection initiated by a local peer corresponded to[49], as well as the average time needed for a

[49] In the network model used in this thesis, an IP hop is defined as a direct link between two distinct routers. By that definition, routing a packet between two separate interfaces in the same router does not constitute a hop. In addition, IP hops are measured from the ISP PoP onwards (i.e. the part of the network connection from the user premises to the local exchange and, where applicable, over a wholesale provider's ATM-based backhaul network is not considered). As a corollary, peers that correspond to the same PoP are considered to be separated by one hop.

locally-initiated transfer of any 350MB file to be completed[50]. The standard deviation of results is provided in square brackets.

	No AVP	With AVP	Difference %
Total transit traffic (Gb)	25,038.65 [±0.3601]	19,989.18 [±0.3226]	- 20.17 %
Queries resolved locally	8,819 (11.82%) [±6.9761]	39,041 (52.34%) [±28.3314]	+ 342.69 %
Unsuccessful queries by local peers	11,204 [±7.2572]	11,100 [±9.4163]	- 0.93 %
Average connection latency	331.19 ms [±0.3459]	200.67 ms [±0.3771]	- 39.41 %
Average IP hops per connection	10.76 [±0.0047]	7.18 [±0.0082]	- 33.27 %
Average transfer completion duration for a 350 MB file	6,873.58 s [±0.5889]	5,010.29 s [±0.9622]	- 26.72 %

Table 5: Traffic and connection characteristics of no AVP/single AOC deployments.

In addition to the "big picture" presented in Table 5, Figure 33, Figure 34 and Figure 35 illustrate Cumulative Distribution Functions (CDFs) of the connection path length (in hops), delay and transfer completion times respectively for both scenarios. Table 6 presents the Pearson correlation coefficients of the data plotted in these figures, calculated from the three separate simulator runs.

	Normal	With AVP
IP hops (Figure 33)	0.999529	0.999375
RTT (Figure 34)	0.996973	0.995321
Transfer time (Figure 35)	0.993242	0.991897

Table 6: Degree of correlation of connection characteristics data between runs.

[50] As noted earlier, the file workload modelled consisted of files of various sizes (2MB – 700MB) and types. For brevity, transfer duration-related results are presented for a single file type only.

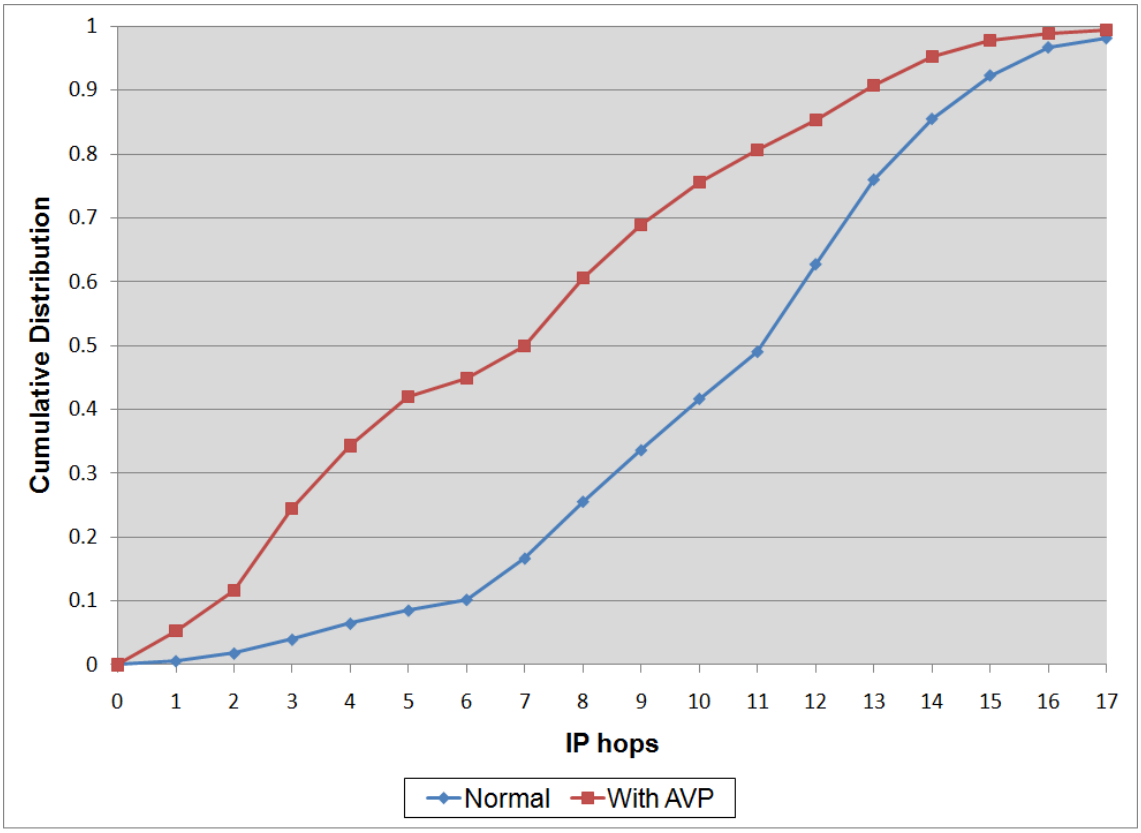

Figure 33: CDF of number of IP hops between source and destination peers under normal operation and when an AVP is present.

It becomes evident from these results that the introduction of peer selection biasing results in the decisive reduction of the number of IP hops between source and destination peers, with almost 35% of connections terminating within 4 hops from source for the simulated topology in the presence of an AVP (cf. Figure 33). In comparison, connected peers are at most 4 hops away in only 6.5% of all cases under random neighbour selection. Crucially, the AVP manages to keep more than half (52.34%) of all transfers within the AS, compared to 11.82% when sources are selected randomly. This has a clear effect in the reduction of transit traffic, which in this case was reduced by approximately 20.17%. Moreover, even when overlay connections terminate outside of the ISP's network due to lack of local sources, they correspond to shorter path lengths compared to regular P2P protocol behaviour. With AVP mediation, 75% of all peer connections correspond to 10 hops at most. In the un-biased scenario, the same happens for less than 42% of connections, while 75% of connections correspond to up to 13 hops.

It has to be noted that this reduction in path lengths and, critically, in transit traffic volume, was achieved by simply promoting local sources where possible, without interfering with peer query criteria or reducing a peer's chance of locating a resource

when it was only available outside the ISP network. This is demonstrated by the number of queries made by local peers that were not successfully resolved, which was not affected by source promotion. The slightly fewer unresolved queries (0.93%) in the presence of the AVP can be attributed to the latter's broader search horizon due to maintaining more connections to foreign peers and having almost permanent uptime compared to an ordinary peer.

Shorter path lengths are important not only because they lead to less inter-domain traffic but also because they translate to less network load This is especially true of P2P bulk transfers which when carried out over unnecessarily long paths (when they could be served over shorter ones equally well) lead to congestion both in the backbone as well as transit links. This point is revisited later in the chapter.

The promotion of local sources has a distinctive effect on peer transfer performance. Figure 34 demonstrates that AVP-promoted connections suffer from drastically less latency than connections formed under normal, un-biased P2P protocol behaviour[51]. While peers experience less than 100 ms of round-trip delay in only 4% of their connections under regular protocol behaviour, with AVP assistance this figure grows to 38%. Additionally, in the latter case half of all connections experience no more than 210 ms of delay, compared to up to approximately 330 ms experienced under random peering. Finally as the CDF reveals, under random peering the majority of connections experience between 200 ms and 450 ms of round-trip delay, while 17% achieves good performance (less than 250 ms delay) and another 7.5% suffers from more than 450 ms of latency. With AVP assistance in contrast, four distinct modes are observed: 35% of connections achieve very low latency (less than 80 ms) due to being terminated within the AS and in most cases very close to the originating peer's PoP. Then, another group of connections experiences relatively low latencies of up to 250 ms. These are connections that terminate in one of the nearby ASes. The third group includes almost 40% of all connections and corresponds to latencies between 250 ms and 400 ms. The linearity of this part of the curve indicates that these connections comprise the bulk of longer inter-AS connections; generally a result of fetching non-

[51] For clarity, the graph illustrates the CDF for up to 700 ms of latency. This covers 98% of all connections, excluding outliers.

adequately replicated objects. Finally, 10% of connections suffer from upwards of 400 ms of latency. These are connections that traverse multiple ASes or paths affected by severe congestion.

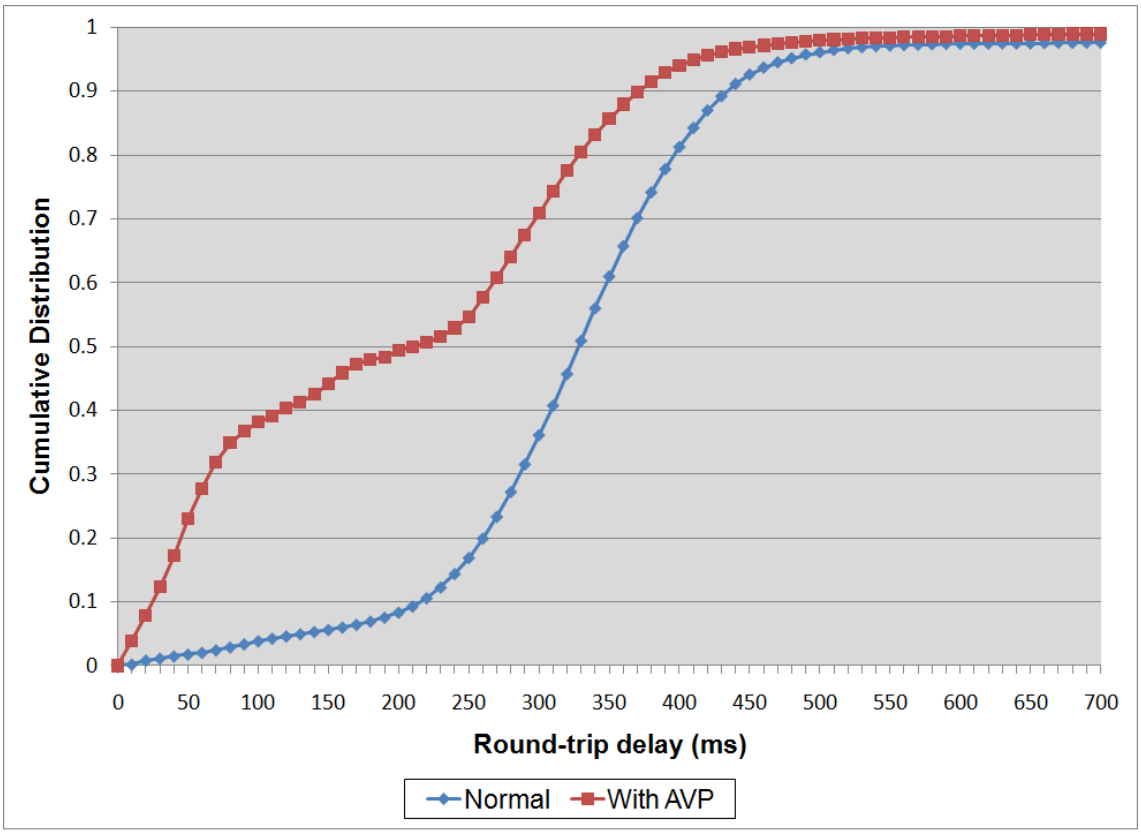

Figure 34: CDF of connection round-trip delay between source and destination peers under normal operation and when an AVP is present.

The effect AVP source promotion has on latency minimisation is reflected on the effective connection throughputs achieved. Figure 35 displays the CDF of the time needed for a local peer to complete a 350 MB transfer for the first, and most significant, 3.5 hours (12,600 seconds). As illustrated in the figure, local peers experience a healthy decrease across the spectrum of the time needed to fully transfer a 350 MB file compared to when sources are selected randomly. It should be stressed that instead of using a "special" download file as in many studies, any file of that size from the entire workload was considered, regardless of popularity or level of replication in the network. Since most files are not as well replicated as those of high popularity, the improvement in download times is often small enough to be "smoothed out" by the 5-minute bins used to calculate the CDF. In addition, source promotion alone cannot guarantee a faster transfer rate when sources are few and similar. Furthermore, the large size of a 350 MB file makes it more susceptible overall to packet loss and congestion which penalise effective throughput. Nevertheless, this ensures a realistic setting and demonstrates a

clear improvement in download times with AVP mediation. Therefore, the redirection of peers to local sources when they are available and closer sources in general has a positive effect on the download performance experienced by most users, because it creates the conditions to achieve higher connection throughputs and thus reduced download times.

The simulation results collected from three separate runs have demonstrated very little variation between them, as indicated by their Pearson coefficients presented in Table 6. This eliminates the possibility of bias and ensures that the observations and conclusions drawn in this section describe the system accurately in its entirety, while the plots drawn using the mean values from the three runs reflect system behaviour without "hiding" salient details.

To summarise, AVP-assisted neighbour selection achieves two important goals:

- It reduces costly transit traffic by a significant amount where possible without otherwise limiting peer search and transfer functions.
- It reduces average path length and delay which has a positive effect on download times.

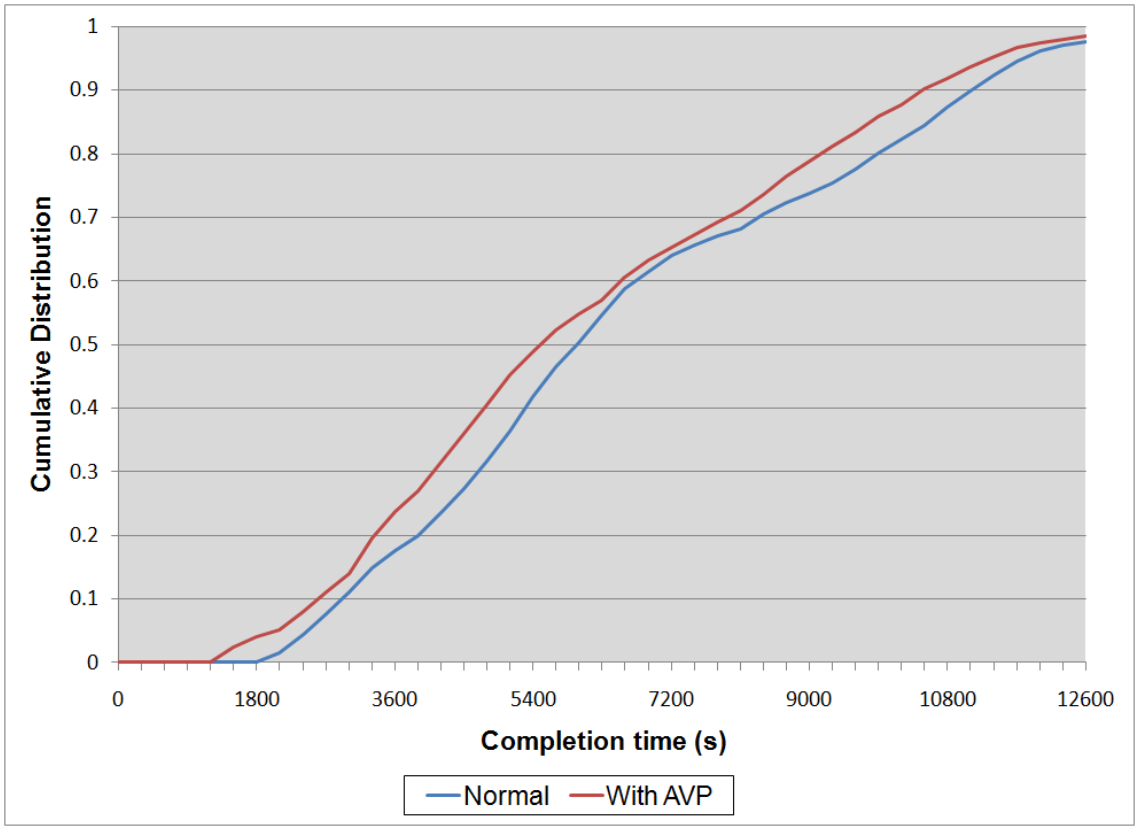

Figure 35: CDF of transfer completion time under normal operation and when an AVP is present.

7.4 Evaluation of VCC caching

Having established the areas and magnitude of P2P application performance improvement achieved solely through AVP peer selection biasing, it is now time to investigate whether VCC caching, which essentially adds dedicated, high-availability local sources to the existing AVP overlay control capability, can offer any significant additional benefits to the network provider and P2P application users.

Because VCC content caching encompasses numerous techniques and addresses some issues not typically encountered in traditional web caching approaches, the evaluation will be broken down into three parts. First, the viability of a VCC-based caching scheme will be examined from a transit traffic-minimising perspective. Next, the various caching strategies presented in Chapter 5 will be evaluated against each other in a single VCC configuration. Finally, the multiple-VCC distributed caching capability will be assessed.

7.4.1 Ideal cache performance

The performance of a cache depends on a variety of factors such as the number of requests seen by the cache, the diversity of these requests (i.e. the slope and shape of the query distribution), the size of the cache and the efficiency of the replacement strategy used. Before examining the performance of VCCs under different strategies it is thus important to establish certain fundamentals for the particular simulation parameters (e.g. number of peers, query workload, replica distribution, file size distribution, peer session times, etc.) employed, as no useful conclusions can be drawn from simple comparison to other studies[52]. The first step is to measure the maximum theoretical traffic savings achievable through VCC operation within a defined period of time. Ideal VCC performance is established by having the VCC apply no cache replacement policies, effectively leading to an infinite size cache.

[52] For example, in [Leibowitz, 2002] a 67% maximum hit rate is reported whereas Dunn in the same year [Dunn, 2002] reports a figure of over 80%. This is because for infinite-sized Web proxy caches the hit ratio grows logarithmically with the client population of the proxy and the number of requests seen by the proxy [Breslau, 1999; Cao, 1997], making any observations only relevant to the particular work.

The same overlay was simulated for 24 hours (after a 6-hour simulator warm-up[53]) under normal P2P operation and when an AVP containing an AOC/VCC pair was present. Figure 36 illustrates the inter-domain P2P traffic volume measured over this time period for these two cases (sampled every 120 seconds). For the largest part of the warm-up period (not shown) traffic volumes for both cases were comparable, as the empty cache was being populated with foreign content. Gradually however, an increasing number of requests was served from cached copies instead of foreign peers leading to the reduction of transit traffic, as evidenced from the graph. This resulted in approximately 200 Mb/s less transit traffic than normal towards the end of the measurement period. Overall, the effect of the introduction of a VCC on the reduction of external traffic is significant. Specifically, for the selected simulation parameters the maximum achievable cache object hit-ratio (or hit-rate) was 59.4% while the byte hit ratio was 56.3%. Of course, these figures represent the upper limit of cache efficiency for the particular workload when cache capacity limitation is not factored in. The introduction of cache replacement policies will inevitably reduce cache performance below these ceiling values.

Table 7 presents overlay statistics collected during the aforementioned run along with the set discussed in the previous section. According to these results, the introduction of a VCC brings improvements in all relevant areas. Specifically, over the measurement period of 24 hours the addition of a VCC reduced transit traffic by 4,738.2 Gb or 23.7% compared to only AOC neighbour selection biasing, and by 9,787.67 Gb or 39.09% compared to regular operation. Given the large peer set and file workload considered, even distribution of replicas and very modest cache warm-up time, this reduction is very substantial. A longer simulation would allow the modelling of flash-crowd dissemination of new objects and uneven spread of replicas in the network which would pronounce the differences between simply redirecting to local sources when they are available and ensuring local copies exist in the cache in regards to transit savings.

[53] All cache simulations described in this chapter are kept "conservative" by starting with empty caches. While this unavoidably introduces compulsory (cold start) misses which hurt the maximum hit rate, it helps avoid biased results due to selecting possibly favourable initial states.

Modelling these traits within a 24-hour window however is unrealistic and would bias the results in the VCC's favour.

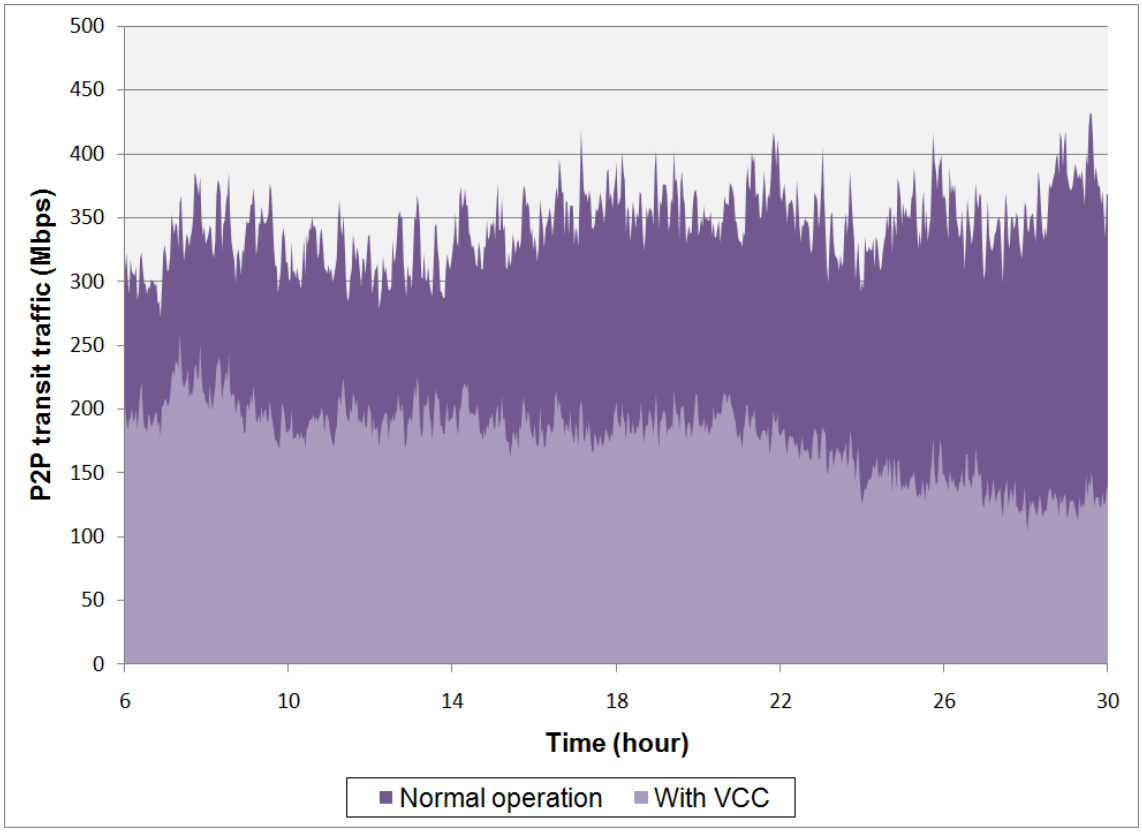

Figure 36: Transit traffic volume with and without a VCC present.

	No AVP	Only AOC	Infinite VCC
Total transit traffic (Gb)	25,038.65 [±0.3601]	19,989.18 [±0.3226]	15,250.98 [±0.6874]
Unsuccessful queries by local peers	11,204 [±7.2572]	11,100 [±9.4163]	9,434 [±19.5959]
Average connection latency	331.19 ms [±0.3459]	200.67 ms [±0.3771]	45.74 ms [±0.2776]
Average IP hops per connection	10.76 [±0.0047]	7.18 [±0.0082]	2.99 [±0.0047]
Average transfer completion duration for a 350 MB file	6873.58 s [±0.5889]	5010.29 s [±0.9622]	2529.25 s [±0.5573]

Table 7: Peer traffic and connection characteristics in "no AVP", "single AOC" and "AOC+VCC" deployments.

Crucially, the big difference comes in those characteristics that constitute an end user's perception of service quality. Both connection delay and path length are minimised, leading to almost halving the average download time for a 350 MB file,

compared to only relying on an AOC. Specifically, the addition of a VCC led to less than half the average path length and reduced average round-trip delay by 77.21%.

The introduction of a high-availability local source in the form of the VCC had the effect of reducing the number of unsuccessful queries due to once-available files disappearing from the network (i.e. due to peer departure). Given that all but the least popular files would achieve a degree of natural replication and thus be available at alternative locations, this signifies searches for rare files. In practice, under any cache replacement strategy such files would be purged from the cache in favour of storing more valuable files, therefore a reduction of this magnitude is attributed to the cache's infinite capacity and unlike the rest of the measured characteristics it is not claimed as an advantage of the VCC.

In short, by ensuring local copies of popular content exist, VCC caching builds upon the improvements brought by AOC source promotion by:

- Offering a further significant reduction of transit traffic, independent of regular local source churn or load.
- Reducing latency and path length many-fold, thus leading to faster downloads and improved service quality perception.

7.4.2 Cache replacement strategy performance

After establishing the theoretical caching maximum, the effect of the different cache replacement policies on cache efficiency was evaluated. First, the performance of each replacement policy at different cache capacities was examined. A single VCC deployment of 100 GB, 200 GB, 400 GB, 600 GB, 800 GB and 1000 GB (1 TB) capacity was simulated for a period of 24 hours for each of the replacement policies implemented. These are LRU (Least Recently Used), LFU (Least Frequently Used), LRUSS (Least Recently Used of Similar Size), LFUSS (Least Frequently Used of Similar Size), ILR (Intelligent Least Requested), ALR (Averaged Least Requested), LFUTS (Least Frequently Used Threshold Smaller) and LFUTL (Least Frequently Used Threshold Larger) as presented in Chapter 5. Again, all other factors remained unchanged between simulations.

A cache hit is noted when the VCC stores and can readily serve the requested file range (or entire file) to the relevant peer. On the other hand, a hit is not awarded when the requested file is available in another local peer and the redirection to it was facilitated by an AVP, despite the mediation. A cache miss occurs when the requested

file is not present in the cache (or other local peers). Upon a miss, the VCC downloads and stores the requested file, unless the strategy employed performs additional checks first (e.g. ILR, ALR, LFUTS and LFUTL). In practical implementations a third possibility exists: Unlike many simulation-based studies which imply temporary capacity overflows or unlimited active transfer "scratch" space (as for instance in [Wierzbicki, 2004]), in the AVPsim model the necessary space is reserved beforehand for each file admission, as would happen in a practical deployment. This is space that most of the times will need to be released by a replacement operation. It is possible that the volume of requests is such that a VCC is simultaneously transferring too many files for storage in the cache and replacement operations need to occur so often that files are evicted shortly after their admission. Not only this can reduce cache operation to simple store-and-forward with no practical benefits, but it can also affect the availability of the physical device due to overloading.

In order to avoid such a case, all policies apart from ILR and ALR implement a download transfer threshold. Specifically, if the amount of space reserved at any time by active transfers exceeds 50% of the total cache capacity, then the VCC suspends the admission of any further files and allows their unmediated transfer from foreign sources until the volume drops below the threshold. This event is denoted as a cache *rejection*. Depending on the query workload and cache size, all policies experience varying numbers of cache rejections. Because ILR and ALR apply replacement operations on a subset of cached files and employ cache rejection by design, they are the only two policies of the group not employing a threshold in this manner.

Figure 37 illustrates the object hit-ratio (OHR) performance of the aforementioned policies, averaged over three simulation runs. For clearer comparison, these results are also presented in tabulated form in Table 8. Pearson coefficients indicating the degree of correlation between the simulation results from each run are presented in Table 9.

Focusing on LRU, LFU, LRUSS and LFUSS for the moment, it becomes evident that policies which operate based on the volume of file requests such as LFU and LFUSS achieve higher object hit ratios throughout compared to LRU and LRUSS respectively, which focus on file last-request times. In other words, long-term behaviour (i.e. frequency of requests over time) proves to be a more effective metric to judge caching suitability than short-term behaviour (i.e. "age" of last request). This indicates that the effect of temporal locality which endows LRU and similar policies with good performance for web content or processor memory caching is not the most dominant

factor in P2P workloads. The dynamics of popularity in current P2P usage profiles change over longer timescales (i.e. days instead of hours). Furthermore, unlike web content which often builds its popularity through frequent modification and updates that lead to repeated access, P2P content is immutable and can generally only attract a "flash crowd" upon its initial insertion in the network. If the typical P2P query workload was characterised by predominantly flash crowd behaviour, then LRU-based policies would have an advantage over LFU-based ones. For present workloads however, LFU-based policies are clearly more suitable.

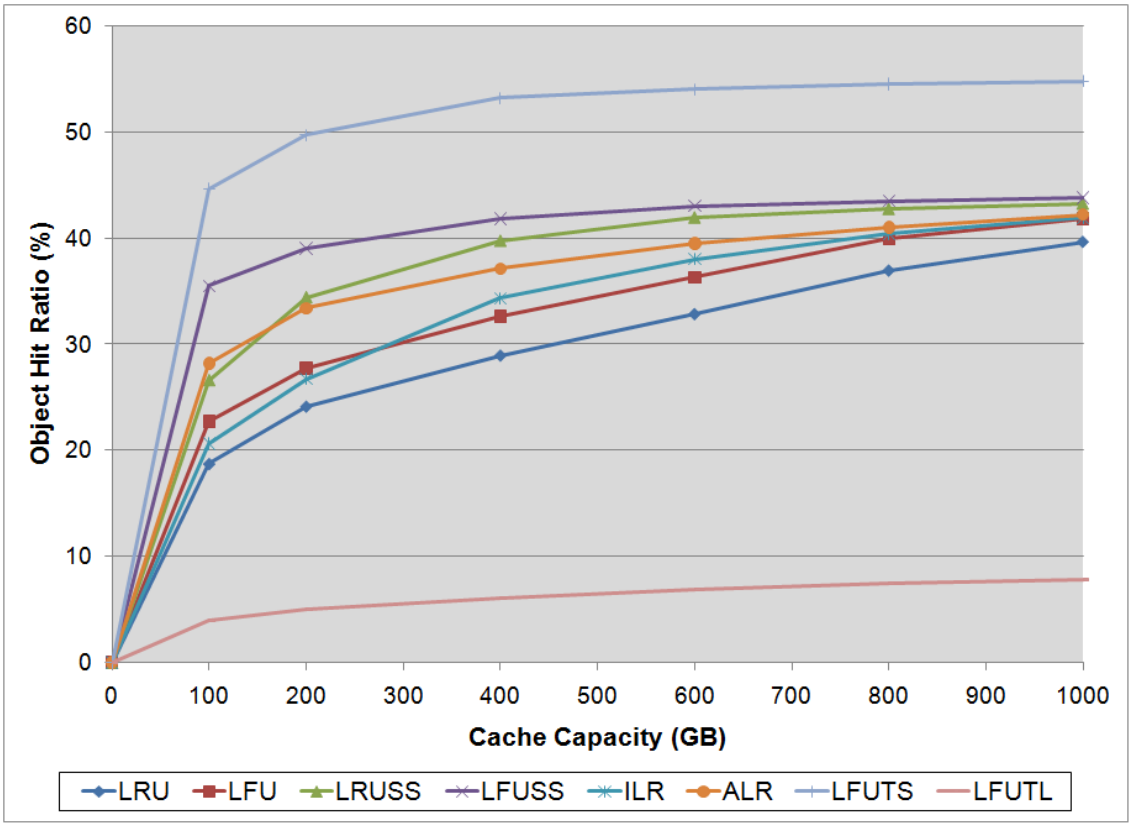

Figure 37: Object hit ratio performance per cache capacity.

A quick observation of the full results reveals that LFUTS offers the best OHR performance throughout, approximately 10% higher than the second best policy (LFUSS) at each capacity. On the other end of the spectrum, LFUTL attains the worst hit ratios, never reaching more than 10% at any capacity. These results are not surprising. Both policies employ a selective caching model where LFUTS caches only small objects (smaller than 80 MB) while LFUTL caches only large objects (larger than 80 MB). Because LFUTS avoids the caching of large objects that would naturally occupy a lot of space at the expense of cached object count, it can use the available capacity to cache many small objects and return a hit in many more cases compared to

other policies. Conversely, because LFUTL caches only large objects, it can hold in the cache only relatively small numbers of them at any time, resulting in low object counts and thus hit rates.

Size	LRU	LFU	LRUSS	LFUSS	ILR	ALR	LFUTS	LFUTL
100 GB	18.7 %	22.7 %	26.6 %	35.5 %	20.6 %	28.2 %	44.6 %	3.9 %
200 GB	24.1 %	27.7 %	34.4 %	39.0 %	26.7 %	33.4 %	49.7%	5.0 %
400 GB	28.9 %	32.6 %	39.7 %	41.8 %	34.3 %	37.1 %	53.2 %	6.1 %
600 GB	32.8 %	36.3 %	41.9 %	43.0 %	38.0 %	39.5 %	54.0 %	6.9 %
800 GB	36.9 %	39.9 %	42.7 %	43.5 %	40.4 %	41.0 %	54.5 %	7.4 %
1000 GB	39.6 %	41.8 %	43.2 %	43.8 %	41.9 %	42.2 %	54.8 %	7.8 %

Table 8: Replacement policy object hit ratios per cache capacity.

LRU	LFU	LRUSS	LFUSS	ILR	ALR	LFUTS	LFUTL
0.99969	0.999804	0.999572	0.999258	0.999754	0.999558	0.999203	0.997639

Table 9: Degree of correlation of OHR simulation results between runs.

From the 600 GB capacity point upwards the object hit rates LFUTS achieves stabilise close to that of the ideal, infinite cache (59.4%). This is a result of the file workload simulated, based on the most recent data available ([Stutzbach, 2007]), which is largely composed of small-sized files (e.g. 60% of all files are smaller than 10 MB). Big caches can store a large number of such files and reach a state where most of the truly popular files remain cached while the replacement policy operates on the edge of the "long tail"; where selecting one file over another does not affect overall hit-ratio significantly. Of course, in real life a workload is not static as new files are gradually made available while unpopular ones may be removed altogether. For the 24-hour period simulated, however, the use of a static workload is not unrealistic, especially since a large number of distinct objects (600,000) was simulated.

Throughout the 100 GB to 1 TB capacity region, the superiority of LRUSS and LFUSS over LRU and LFU respectively is clearly identifiable. These policies combine a file's request age or frequency with its size and demonstrate a considerably higher object hit rate performance than their single-metric counterparts (LRU and LFU). Due

to factoring in size in the eviction decision, LRUSS and LFUSS manage to keep a stable number of cached objects by evicting exactly one file for each admission. The effects of this approach are visible in Figure 38 which illustrates the number of cache replacement operations performed by each policy during the simulation campaign. As the available capacity increases, the number of necessary operations drops for all policies but overall two distinct groups can be observed. The file size-aware policies (LRUSS and LFUSS) require noticeably fewer replacement operations than the simpler LRU and LFU at each capacity point, indicating better utilisation of available space per operation. Crucially, cases where a significant number of small files is evicted to make room for a single large file are avoided. This keeps the object count as high as possible at any given time and contributes to the considerably higher overall OHR.

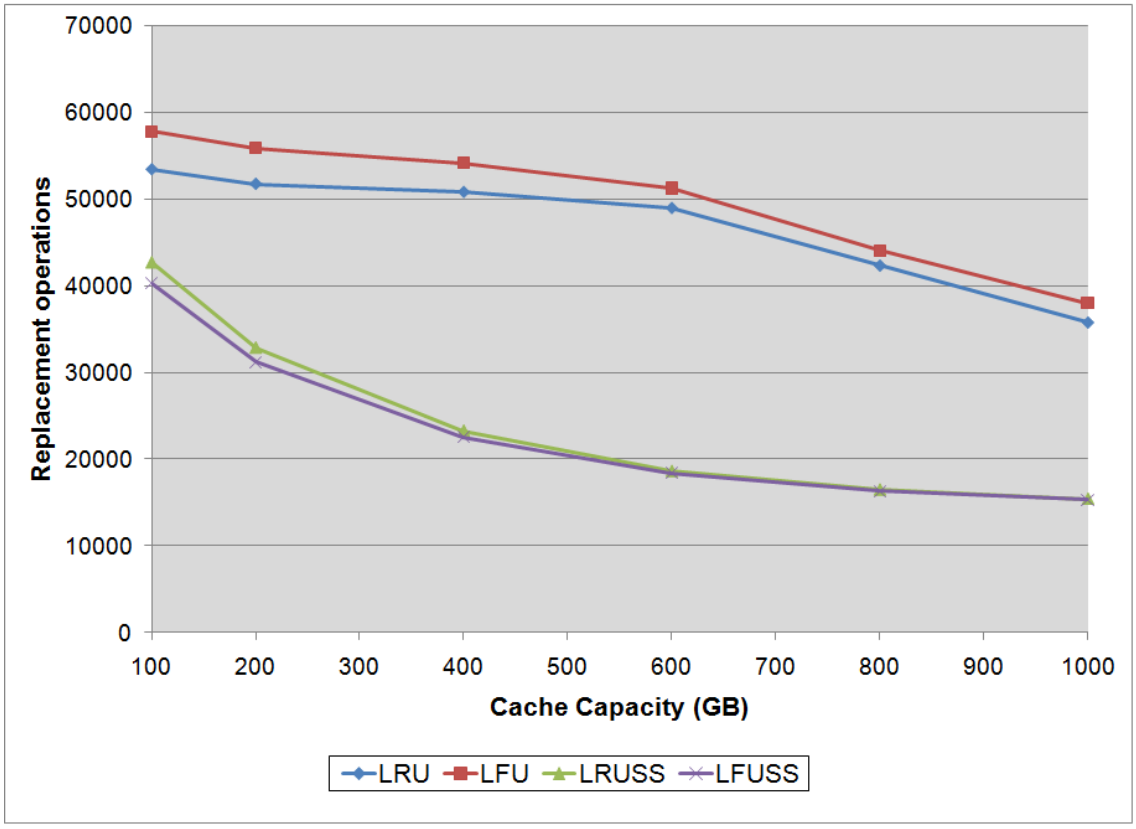

Figure 38: Cache replacement operations per policy and cache capacity.

ILR and ALR comprise two selective caching strategies where a requested file is admitted into the cache only if it fulfils the conditions dictated by the strategy. Their object hit ratio performance lies roughly in the middle of the group, being higher than LRU, LFU (for 300 GB and higher) and LFUTL but lower than LRUSS, LFUSS, and LFUTS. Between the two, ALR shows higher object hit ratio performance, which is more pronounced in smaller caches. Finally, despite being the only two policies of the

group that do not employ a download volume threshold to deal with overwhelming query rates, the policy intelligence manages to identify and keep really valuable files stored while allowing all files enough time in the cache to let them "prove" their popularity.

In general, the simulation results collected from the three runs demonstrate a very high degree of correlation (very close to the "ideal" value of 1), as shown on Table 9. This indicates that the variation of OHR results between the different runs was minimal, eliminating the possibility of bias and ensuring that they are described accurately by the mean values plotted in Figure 37 and tabulated in Table 8.

While valuable, object hit ratios alone cannot provide a definitive indication of replacement policy performance as they do not necessarily reveal the extent of bandwidth savings achieved. For that reason, VCC cache replacement policies were also evaluated according to *byte hit ratio* performance. Byte hit ratio (BHR) denotes the number of bytes transferred from the cache as a percentage of the total number of bytes resulting from all (local) peer requests. Figure 39 illustrates BHR performance for the aforementioned policies and cache capacities. The results are also presented in Table 10.

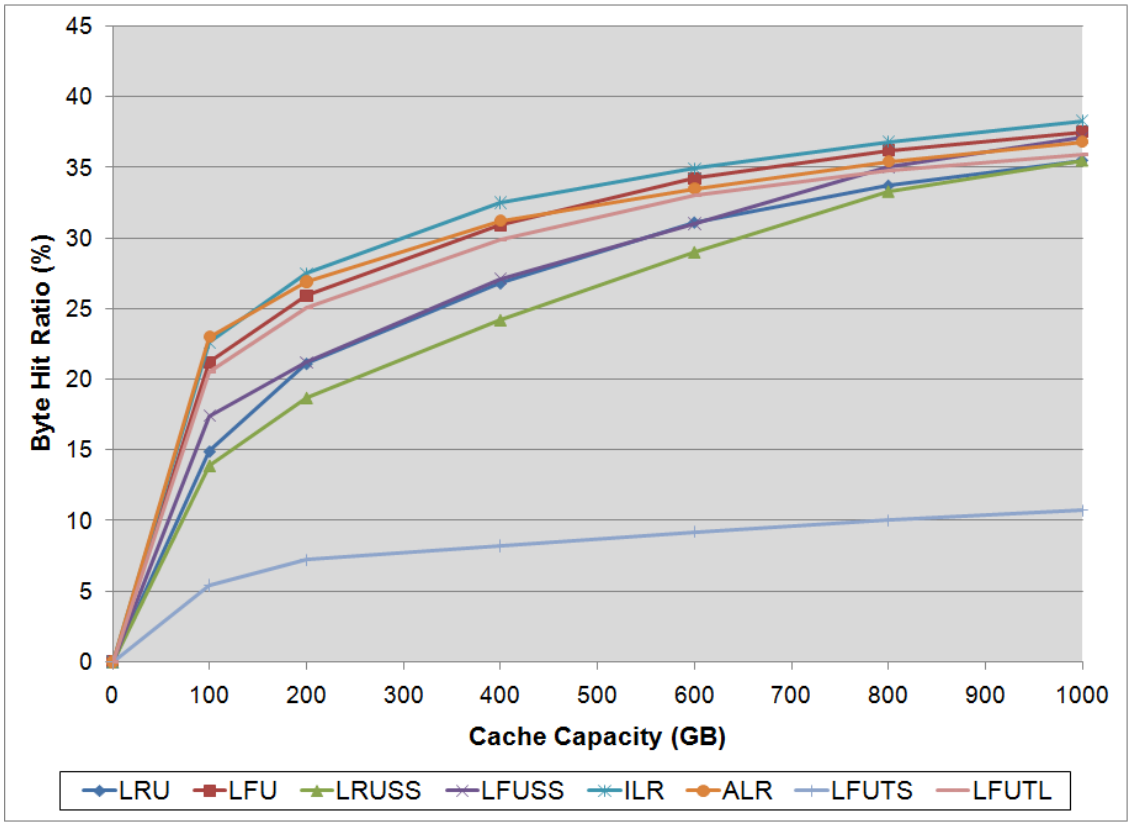

Figure 39: Byte hit ratio performance per cache capacity.

Byte hit ratios reveal a different side of the story to object hit ratios. With the exception of the 100 GB size cache, ILR achieves the highest byte hit ratios across the board. ALR appears as second best in smaller caches, but for caches larger than 600 GB it is outdone by LFU. As with object hit ratios, LFU retains an advantage over LRU and this superiority of request frequency over age is also maintained between LRUSS and LFUSS. Furthermore, as with the OHR analysis, the Pearson coefficients estimated from the three separate simulation runs demonstrate a very high correlation of BHR results, giving confidence of their validity. These coefficients are presented in Table 11.

Size	LRU	LFU	LRUSS	LFUSS	ILR	ALR	LFUTS	LFUTL
100 GB	14.9 %	21.2 %	13.9 %	17.4 %	22.6 %	23.0 %	5.4 %	20.5 %
200 GB	21.1 %	25.9 %	18.7 %	21.2 %	27.5 %	26.9 %	7.2 %	25.1 %
400 GB	26.8 %	30.9 %	24.2 %	27.1 %	32.5 %	31.2 %	8.2 %	29.9 %
600 GB	31.1 %	34.2 %	29.0 %	31.0 %	34.9 %	33.5 %	9.2 %	33.0 %
800 GB	33.7 %	36.2 %	33.3 %	35.0 %	36.8 %	35.4 %	10.0 %	34.8 %
1000 GB	35.5 %	37.5 %	35.5 %	37.1 %	38.3 %	36.8 %	10.7 %	35.9 %

Table 10: Replacement policy byte hit ratios per cache capacity.

LRU	LFU	LRUSS	LFUSS	ILR	ALR	LFUTS	LFUTL
0.999837	0.999802	0.999725	0.9996	0.999581	0.999403	0.998498	0.999682

Table 11: Degree of correlation of BHR simulation results between runs.

By estimating the value of a candidate object before admission, ILR manages to avoid most non-optimal caching decisions, which are costly, and this is reflected in the superior returns per cached byte it achieves. The similarly operating ALR is also in the top performers but, despite demonstrating better object hit ratio performance, is not as effective from a byte hit rate perspective compared to ILR. The reason is that because it enforces stricter admission criteria to ILR, ALR does not cache newly-appeared objects as early and may cause multiple downloads from external sources before finally caching them in high churn conditions (i.e. when the local source disappears before the object is further replicated naturally). This incurs a byte hit rate penalty which becomes more pronounced as capacity increases and the cost per non-optimal decision becomes

smaller. This is the reason why in larger caches LFU is competitive to, or even surpasses ALR.

Unlike their object hit rate performance, the byte hit rates achieved by LRUSS and LFUSS are rather mediocre. This is because the size differentiation they employ mainly reduces cache operations and maintains a stable number of objects but ultimately, like LRU and LFU, operates on files and not bytes. Thus, while the byte hit rates they achieve improve steadily with each cache capacity increase, both policies replace byte-for-byte, not always identifying those files with the best byte returns in the process.

LFUTS demonstrated the worst byte hit ratio performance of the group. While by selectively caching only small files this policy achieved superior object hit rates, it stored files that because of their size offered minimal transit traffic savings despite their repeated requests. In other words, the collective bytes served by the set of cached files could not offset the penalty incurred by fetching large files from foreign sources.

On the other hand, LFUTL achieved high byte hit rates despite its inferior object hit rate performance for the same reason. While the average number of cached files at each configuration was an order of magnitude smaller than that of all other policies, the bytes served amounted to a large percentage of total requested bytes.

Clearly, from the strategies examined those that favour the caching of smaller objects achieved high object hit rates but low byte hit rates. This is because from a cached object count standpoint it is preferable to replace one large object and miss it if it is requested again, than to replace many small objects to reclaim the same amount of space and suffer misses on many of them. On the other hand, in order to match the byte savings gained by serving a large file even once, a small file will need to be served numerous times. In other words, unless small files are universally more popular than large files, any policy that favours the storing of the former is going to suffer from reduced byte hit rate.

In essence, a trade-off exists: Due to the size and dynamics of P2P workloads, an ISP deploying a P2P caching infrastructure has to decide whether to optimise for discrete object availability (i.e. to ensure a high object hit rate) or transit traffic minimisation (i.e. aim for a high byte hit rate). High cached object counts have the effect of improving the ISP subscribers' service quality perception as more searches can be successful and more objects can be served from the cache, which is expected to benefit from good connectivity, lower latency and high availability. On the other hand, transit traffic is expensive and minimising it by caching those objects that cost the most

to transfer over inter-AS links will have a direct effect on the ISP's operational costs. In most cases it is expected that ISPs will select the second option, as its effects are more up-front and affect their bottom-line. However, if the improved perception of network quality can be translated into a value-added service and more subscribers are thereby attracted, the bottom-line will also be positively affected, albeit indirectly.

The file workload simulated up to this point was based on the most recent data available at the time ([Stutzbach, 2007]), in order to provide a realistic scenario of VCC performance evaluation at present time. As already stated, this workload is largely composed of small-sized files. The advances in storage technology and broadband connectivity, the uptake of new high-definition media formats (e.g. MPEG-4 H.264 [ITU-T, 2005]) as well as user demand for richer content (a trend highlighted by the comparison of the aforementioned study with a similar but older one [Chu, 2002]) support the hypothesis that future P2P workloads will shift towards larger files and in particular video. This will have a direct impact on traffic generation and cache performance. To complement the earlier results and investigate the operation of the proposed cache replacement policies on future usage scenarios, the simulation campaign was repeated with a new file workload where video files occupied 20% of the file population compared to the earlier 6%. This increase is analogous to the increase of video files from 2.1% in 2002 to 6% in 2007 as reported by the aforementioned studies and is intended to approximate how P2P file workloads may evolve in the next five years. Furthermore, the higher proportion of large files in the workload will stress replacement policies harder, allow a finer examination of their performance and pronounce their differences.

The simulation campaign was repeated with the new workload for cache capacities of 100 GB, 200 GB, 400 GB, 600 GB, 800 GB and 1000 GB. The rest of the simulation parameters remained the exactly same, to allow for a direct comparison of cache performance between the two workloads. Figure 40 provides a plot of object hit rate performance at the aforementioned cache capacities. Additionally, Table 12 presents the results in tabulated form. Pearson correlation coefficients of the results are presented in Table 13.

As expected, by assuming a heavier workload without increasing cache sizes accordingly, replacement strategies have to operate a lot more frequently and their differences are pronounced. LFUTS achieves the highest object hit rates for caches up to approximately 900 GB, beyond which it is surpassed by LFUSS. ALR and LRUSS are competitive, reaching almost similar performance. LFU maintains a steady

advantage over LRU, while ILR performs poorly in the smaller caches improving only from 800 GB and above. LFUTL, again, trails behind the rest.

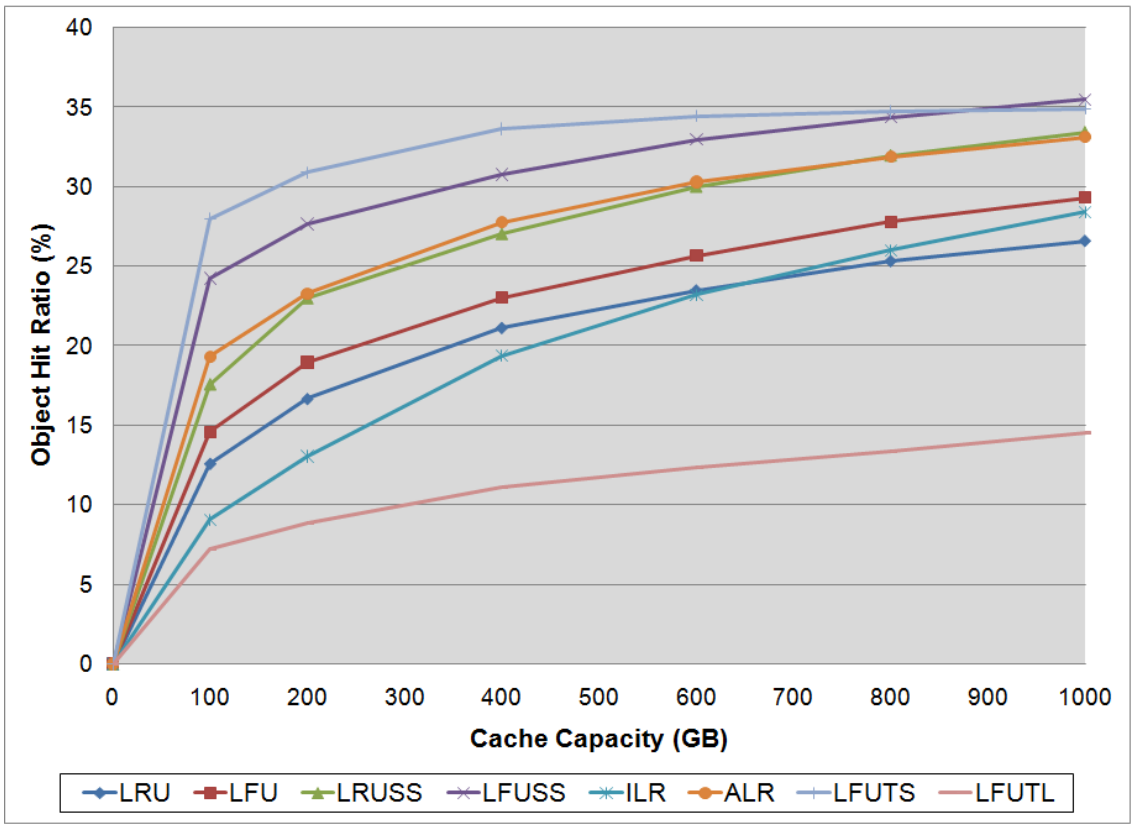

Figure 40: Object hit ratios per cache capacity (2nd workload).

Size	LRU	LFU	LRUSS	LFUSS	ILR	ALR	LFUTS	LFUTL
100 GB	12.6 %	14.6 %	17.6 %	24.2 %	9.1 %	19.3 %	28.0 %	7.2 %
200 GB	16,7 %	19.0 %	23.0 %	27.7 %	13.0 %	23.3 %	30.9 %	8.9 %
400 GB	21,1 %	23.0 %	27.0 %	30.8 %	19.4 %	27.8 %	33.6 %	11.1 %
600 GB	23.5 %	25.7 %	30.0 %	33.0 %	23.2 %	30.3 %	34.4 %	12.3 %
800 GB	25.3 %	27.8 %	32.0 %	34.4 %	26.0 %	31.9 %	34.7 %	13.3 %
1000 GB	26.6 %	29.3 %	33.4 %	35.5 %	28.4 %	33.1 %	34.9 %	14.5 %

Table 12: Object hit ratios per cache capacity (2nd workload).

Byte hit ratios (plotted in Figure 41 and tabulated in Table 14) demonstrate the superiority of ALR up to approximately 700 GB. Above 700 GB ILR achieves the highest byte hit ratio. LFU does not lag far behind, especially in small caches, but as

more space becomes available it cannot compete with ILR and ALR. The rest of the policies essentially demonstrate the same pattern as with the original workload, with LFUSS returning similar byte hit ratios to LRU, LRUSS being 2% to 4% lower and LFUTS exhibiting abysmal performance. The Pearson correlation coefficients of the BHR results, presented in Table 15, demonstrate very high correlation.

LRU	LFU	LRUSS	LFUSS	ILR	ALR	LFUTS	LFUTL
0.999545	0.999546	0.999445	0.998847	0.999714	0.999187	0.997331	0.996591

Table 13: Degree of correlation of 2nd workload OHR simulation results between runs.

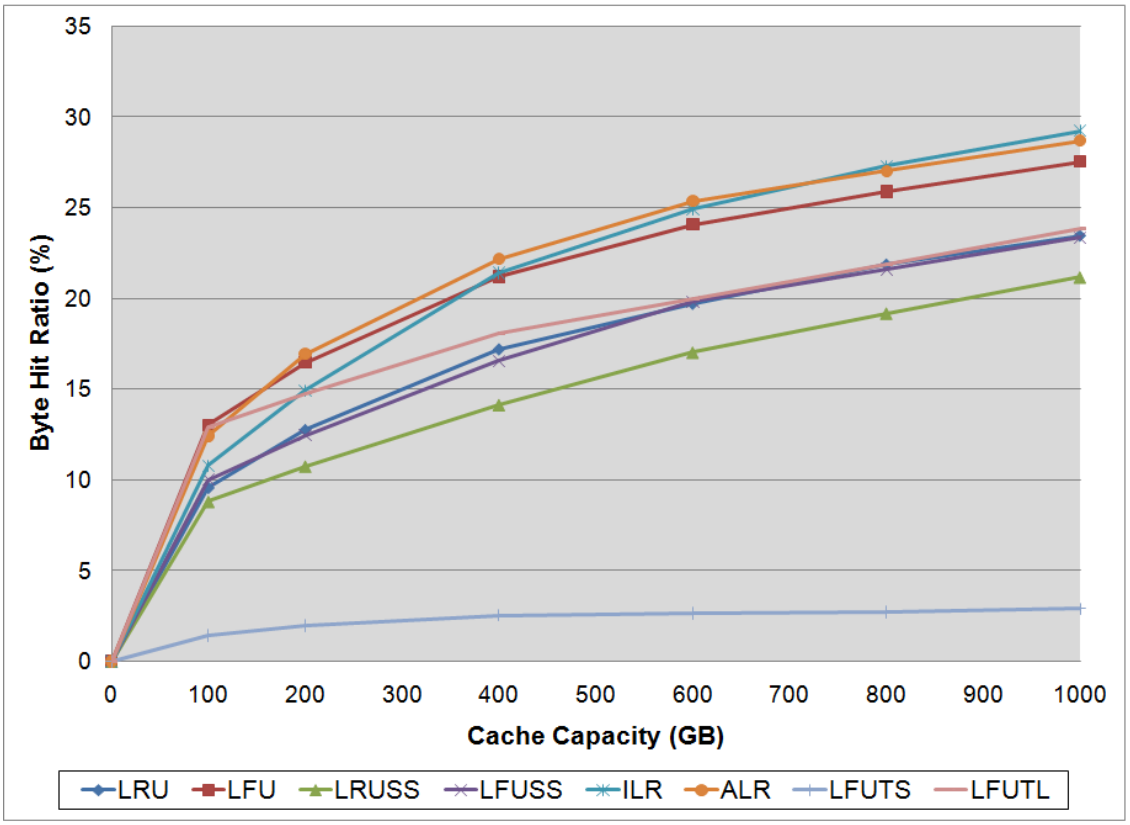

Figure 41: Byte hit ratios per cache capacity (2nd workload).

It is clear that all policies have to operate very frequently and the resulting hit ratios suggest that caching such a workload with very small caches is not effective. Files do not stay long enough in the cache to attract further hits while long transfer times (due to file size) make bad cache decisions even more costly. Of course, a 100 GB cache is modest even by today's standards. While the cache sizes remained the same to enable direct comparison with the previous workload, it is certain that storage capacities will increase along with the inflation of the workload. Assuming however that the workload evolves in such a way that available cache capacity cannot keep up (even temporarily),

some policies appear better suited than others. ALR is the most balanced policy achieving both high object and byte hit rates. LFU is not as effective but achieves a similar balance. The rest either need more "breathing room" to operate effectively (like ILR) or manage to hold their own in regards to object hit rates but at the cost of low byte hit rates (i.e. LRUSS and LFUSS).

Size	LRU	LFU	LRUSS	LFUSS	ILR	ALR	LFUTS	LFUTL
100 GB	9.6 %	13.0 %	8.8 %	10.0 %	10.8 %	12.4 %	1.4 %	12.9 %
200 GB	12.8 %	16.5 %	10.8 %	12.5 %	14.9 %	16.9 %	2.0 %	14.8 %
400 GB	17.2 %	21.2 %	14.1 %	16.6 %	21.4 %	22.2 %	2.5 %	18.1 %
600 GB	19.7 %	24.1 %	17.0 %	19.8 %	24.9 %	25.4 %	2.6 %	20.0 %
800 GB	21.9 %	25.9 %	19.2 %	21.6 %	27.3 %	27.0 %	2.7 %	21.9 %
1000 GB	23.5 %	27.6 %	21.2 %	23.4 %	29.3 %	28.7 %	2.9 %	23.9 %

Table 14: Byte hit ratios per cache capacity (2nd workload).

LRU	LFU	LRUSS	LFUSS	ILR	ALR	LFUTS	LFUTL
0.999355	0.999465	0.999589	0.999318	0.999584	0.999482	0.963359	0.999152

Table 15: Degree of correlation of 2nd workload BHR simulation results between runs.

In conclusion, the following observations were made while assessing cache replacement policy efficiency in single VCC deployments:

- The frequency and diversity of queries and available cache capacity in relation to file workload indicate that request frequency-based metrics are more accurate than request time-based metrics.
- Replacement policies that take into account more than one metric utilise available cache capacity more effectively than their single-metric counterparts.
- Most policies are efficient towards either object hit rate or byte hit rate but not both. An inverse proportionality exists in their performance in these two metrics.
- ALR demonstrated a good balance between object and byte hit rates, and is the proposed policy for the VCC.

- ILR is also competitive, especially if transit traffic reduction is the caching objective. In imbalanced workload-capacity situations its object hit rate performance may suffer.
- Although not a top performer, LFU showed good balance between object and byte hit rate performance in both workload configurations. If implementation simplicity is a requirement and ample capacity is available, LFU can be considered as a "good enough" replacement policy.

7.4.3 Multiple-VCC deployments

In the previous section, the performance of the various replacement strategies considered for the VCC was assessed in a single VCC deployment. This allowed the identification of the best-performing strategies of the group as well as the establishment of performance benchmarks for each given capacity. This section builds upon these findings and examines how P2P applications are assisted when more than one VCC is provided. Two deployment scenarios (previously described in Chapter 5) are examined: First, when multiple VCCs operate autonomously in a cluster, followed by a cooperative caching scenario.

The simulation set-up and parameters used in both scenarios were identical to the single-VCC experiments (using the "current" workload) for consistency of results. The cache replacement strategy applied in all scenarios was LFU.

7.4.3.1 Autonomous VCC operation

In this scenario each VCC manages a different set of cached files with no replicas shared between different VCCs. The existence of more than one caching location allows the focus to shift from pure cache hit performance-related metrics to transfer, and ultimately application, performance. In particular, the number of IP hops separating a peer from a file source and the latency experienced in the resulting transfer connections are measured. In addition, the load experienced on VCCs from serving files and managing caching and replacement operations in different deployments can now be compared.

Table 16 presents the results from simulating a varying number of VCCs while keeping the combined cache capacity constant. A single-VCC run was included as a point of reference. The standard deviation values are presented within square brackets.

Cumulative object hit ratios were calculated by dividing the overall number of cache hits (i.e. the sum of each participating VCC's hits) with the total number of requests made by local peers. Since in this scenario the total available cache capacity is treated like a single virtual cache, comparable hit rates were expected in all configurations accounting for a small performance penalty due to the decentralisation of the caching infrastructure (e.g. imperfect inter-VCC coordination, communication overheads etc). Indeed, the simulation results show all configurations achieving very close OHR performance to the reference, with no (mean) result deviating more than 0.05% from it. This indicates that cached object availability inside the ISP is not affected by the distribution of the cache capacity in different locations.

	1 VCC	**2 VCCs**	**3 VCCs**	**4 VCCs**	**5 VCCs**
Capacity per VCC	1x1TB	2x500GB	1x200GB 2x400GB	4x250GB	5x200GB
OHR (cumulative)	41.81 % [±0.01%]	41.80 % [±0.01%]	41.75 % [±0.01%]	41.77 % [±0.01%]	41.76 % [±0.01%]
Average VCC load	72.68 % [±0.05%]	36.47 % [±0.04%]	24.65 % [±0.02%]	18.54 % [±0.03%]	14.81 % [±0.03%]
Average IP hops to file source	3.01 [±0.02]	2.76 [±0.01]	2.73 [±0.02]	2.69 [±0.01]	2.65 [±0.01]
Average connection RTT (ms)	46.26 [±0.15]	41.86 [±0.13]	41.43 [±0.09]	40.51 [±0.08]	39.31 [±0.09]

Table 16: Variation between different VCC deployment configurations.

In order to verify that the closeness of the multi-VCC results to the - effectively centralised - single-VCC performance is not a product of simulation model oversimplifications or other omissions, the simulations were run again, this time assuming a 5% message loss rate in inter-VCC coordination. Unlike network packet loss which in most cases can be detected and dealt with by TCP or higher layer protocol intelligence, message loss in this case is used to denote that the entire protocol message does not reach its destination. In this case the destination party has no indication that someone attempted to contact it and unless a retransmission occurs, the intended action is never carried out. In reality such a set of circumstances is very rare and indicative of severe network problems that would affect the overlay in much more direct ways anyway. Furthermore, AVP protocol messages need to be acknowledged and would normally be retransmitted on error. Nonetheless, simulations were repeated using a model where 1 every 20 messages on average between AOCs and VCCs would be

ignored and no retransmissions would occur, resulting in a sensitivity test of sorts. This affected operations by having one of the VCCs not receive a look-up query or one of the AOCs never receive a reply. The results of this second round of simulations showed between 1% and 3% lower hit rates, indicating that the system can cope even in such disadvantageous conditions since (i) even if a message is not delivered to some peers or AVPs it may reach others that can respond to it, and (ii) even if all 5% of file requests are for files that are in the cache and are all undelivered, the volume of processed requests is such that in the long term those losses will have a negligible effect on the overall hit rates. In other words, the slight hit rate difference between single and multiple VCC deployments is more due to the change in workload patterns and replacement operation frequency *per VCC* (due to the cache size difference) than due to communication errors. This is also evident when examining the penalty trend with respect to deployment size. The 3-VCC deployment, which is the only configuration where VCCs do not all have equal capacity, is penalised more than the 4-VCC and 5-VCC deployments despite having fewer VCCs. This is because the performance of the 200 GB VCC drops the cumulative hit rate average.

Deploying more than one VCC has a profound effect on reducing the load experienced in each VCC. Naturally, being the only cache in an ISP network a single VCC will need to handle a large number of concurrent requests and transfers and thus operate under high load. In the simulations under discussion, the single VCC operated at approximately 73% average load, while peak load reached 91% of available capacity. While within operational limits, this is close to full capacity indicating that an influx of new peers or change in peer usage patterns will threaten the cache's availability. The addition of just one more VCC lowers individual load substantially (to 36.47%), while 3 VCCs bring load to a manageable 24.65%. It becomes clear that each subsequent addition offers a smaller incremental improvement with the transition from 4 to 5 VCCs only lowering individual load by 3.73% units for the specific network simulated.

The existence of more than one VCC means that there are more than one cache locations at different distances network-wise from a particular peer. It is interesting to examine if this has an effect on the quality of the resulting connections between peers and VCCs. Here, the term "quality" is used to describe connection characteristics under normal conditions (i.e. excluding drastic factors such as link failure) which can be largely determined by the throughputs achieved, as higher throughputs mean faster downloads. As in the preceding sections, connection quality will be quantified by

measuring connection round-trip delay and path length for connections initiated by peers of the "home" ISP.

As discussed in section 7.2, the simulated network topology represents a medium- to large-sized "home" ISP with access, backbone, distribution as well as border routers providing connectivity with the rest of the Internet via different ASes for resilience. In the resulting topology the simulation results for the single-VCC deployment[54] show that all local peer-to-local peer or local peer-to-VCC connections terminate within an average of 3.01 hops. Adding a second VCC resulted in a shorter average path length of 2.76 hops. The deployment of further VCCs demonstrates a progressive reduction of hop count, with each additional VCC translating into fewer average hops than in the single VCC case as more of the local peers find their content of interest cached closer. While the rate of reduction depends on the particular network's size and VCC placement, in this setup 5 VCCs resulted in an 11.96% reduction of average path length. This indicates a further localisation of traffic beyond that already offered by the placement of a single VCC and hints at the consequent reduction of congestion due to long flows.

The connection delay measurements lead to similar conclusions. The average round-trip delay for intra-ISP connections with one VCC present was 46.14 milliseconds (ms). As with path length, each further increase of the number of VCCs contributed towards its reduction. Ultimately, the 5-VCC deployment offered a 15.02% delay reduction compared to the single-VCC one.

Both the IP hop and delay reductions are interesting considering that the main goals of the multiple autonomous VCC caching strategy are individual cache load minimisation and avoiding the single point of failure a sole VCC poses. While link latency plays a significant role on target VCC selection, the fact that each VCC holds a different set of files means that it cannot be guaranteed that the initial VCC selection decision based on load and network proximity to the peer making the first request (see Chapter 5) will remain optimal in terms of future demand. In other words, bringing

[54] It is assumed that VCC placement is strategic and reflects the particular ISP topology. If VCCs are placed randomly or inconsistently, it is possible for path length and especially latency to certain parts of the network to actually increase. Such an outcome has been demonstrated in simulation and is avoided in the present discussion as it goes against standard network engineering practices.

something close to one group of users may mean placing it further away from another group. However, it was demonstrated that increasing the number of participating VCCs resulted in shorter average path lengths and round-trip latencies than those attained with smaller deployments for the entire local peer base.

The path length reduction can be mainly attributed to VCC placement following standard network engineering principles. In the case of a single VCC it makes sense from a network engineering and management perspective to place the cache in the core of the network so that no part of it is disadvantaged compared to others. When more VCCs are available, the ability to place them closer to the edge of the network becomes viable (and indeed is central to the AVP concept). In the present scenario, any additional VCCs were placed on PoPs (i.e. while retaining one at the original network core location), starting with those serving the most subscribers and moving to the smaller ones depending on the number of available VCCs. The fact that in many cases hop distance essentially does not increase between a PoP-to-core and a PoP-to-PoP connection due to redundant PoP interconnection (as for instance discussed in [Iannaccone, 2004]) along with natural replication on local peers and the subsequent AOC redirection to them, results in lower average path length. This does not mean that in individual cases peers don't end up fetching some cached object from further away compared to the single central VCC. On average however, the majority is served over shorter paths.

At the same time, cache load, which in the single-VCC scenario could be an ongoing concern, is decisively reduced even when deploying only two VCCs. Besides ensuring optimal VCC load levels from an operational perspective, by spreading the load over a number of them the conditions for less response time (e.g. as noted in [Zegura, 2000]) and less congestion are presented. This, along with the lower propagation delay of shorter paths (and in the case of passing through fewer routers, less queuing and processing delays) results in the observed reduction of latency and the improvement of throughput.

More importantly, both the primary (load reduction) and secondary (latency reduction) goals are achieved at no practical cost in terms of hit rate performance. The measured reduction due to cache separation was in the worst case less than 0.1%.

7.4.3.2 Cooperative VCC operation

In this scenario, in addition to its normal caching tasks each VCC periodically reports the top n% most popular files stored in its cache which can then be replicated in

other VCCs which dedicate a portion of their storage capacity for that purpose. The intended result is for very popular files to be made available in more than one location, and in particular closer to localised demand, so that cases of cache and link overloading due to high demand can be avoided.

Table 17 contains the simulation results for a cooperative VCC deployment scenario where each VCC dedicates 0.5% of its total capacity to inter-VCC replication. Since the only difference to the earlier, autonomous VCC scenario is the periodic replication, it is interesting to see how this affects the results.

	1 VCC	**2 VCCs**	**3 VCCs**	**4 VCCs**	**5 VCCs**
Capacity per VCC	1x1TB	2x500GB	1x200GB 2x400GB	4x250GB	5x200GB
OHR (cumulative)	41.81 % [±0.01%]	41.76 % [±0.02%]	41.58 % [±0.02%]	41.45 % [±0.03%]	41.28 % [±0.01%]
Average VCC load	72.68 % [±0.05%]	28.60 % [±0.03%]	13.91 % [±0.03%]	10.60 % [±0.04%]	7.88 % [±0.02%]
Average IP hops to file source	3.01 [±0.02]	2.71 [±0.01]	2.63 [±0.01]	2.58 [±0.01]	2.49 [±0.01]
Average connection RTT (ms)	46.26 [±0.15]	41.08 [±0.12]	39.04 [±0.10]	36.86 [±0.09]	35.78 [±0.06]

Table 17: Variation between different cooperating VCC configurations.

As in the autonomous VCC scenario, there is a hit rate reduction when increasing the number of deployed VCCs. However, in this case it is more apparent. The reservation of some cache space for replicating existing objects reduces the total available space for caching unique objects, thus reducing object hit rate. As each additional VCC brings an increase of the total capacity reserved for replication, deploying more VCCs results in lower cumulative object hit-rates compared to the single-VCC scenario. Therefore, the 5-VCC deployment reaches a 0.53% smaller object hit ratio compared to the single-VCC deployment.

Individual load is again progressively reduced with each VCC addition. Compared to the autonomous VCC mode, load reduction is more dramatic at each configuration. For example, while the average load experienced by the 3-VCC deployment in autonomous operation was 24.65%, it was dropped to 13.91% with VCC cooperation. This can be attributed to the fact that the most “heavy-hitting” objects of each VCC, which are now replicated, are responsible for a considerable part of the total load experienced by each VCC. Thus, requests for a particular popular object which in

the earlier scenario would all be exclusively served by one VCC are now shared between VCCs, dropping individual load further. Consequently, in terms of load reduction the cooperative-VCC mode is clearly superior to using a single VCC, but also better than the autonomous-VCC mode.

IP hop and delay reduction between local peers and sources follow the same pattern as in the autonomous VCC operation. Increasing the number of VCCs reduces the average connection delay and hop count compared to a central cache. The availability of highly-popular sources in more than one VCC, however, results in more evident reductions, especially when comparisons are drawn between the deployments of equal number of VCCs in the two scenarios. Comparing the cooperative 5-VCC to the single-VCC deployment, the former brings a 17.27% average hop reduction and a 22.65% RTT reduction. In contrast, the autonomous 5-VCC deployment reached an 11.96% path length reduction and a 15.02% delay reduction. Therefore, by ensuring that the "hottest" objects are cached at the network edge and for most peers at the same PoP, average delay and path length can be decisively reduced. This is particularly important given that requests for non-popular content that still result in connections over long and possibly congested paths act as outliers, inflating these averages. In conclusion, for peers whose query workload focuses on popular objects, the latency improvements will effectively be even more substantial.

The amount of cache capacity dedicated by each VCC to store replicas affects system performance in a number of ways. Naturally, the more capacity is dedicated to inter-VCC replication, the less is made available to cache unique objects. This will reduce the object hit rate and may in turn affect negatively the volume of transit traffic and local peer transfer performance. Table 18 illustrates how the variation of replication capacity affects the metrics of interest for a 4-VCC deployment simulated over 24 hours.

It becomes evident that increasing the capacity available for inter-VCC replication drives individual VCC load down, as it allows for larger numbers of "heavy-hitters" to be replicated. With the same workload, moving from 0.3% replication capacity to 10% resulted in a reduction of average individual load from 12.02% to 1.90%. However, not all system properties benefit equally from more replication. Because total capacity is limited, liberally allocating portions of it for inter-VCC replication will have adverse effects on locally initiated overlay connections and the cumulative object hit rate. With less space available to cache individual objects, VCCs are forced to run replacement operations more frequently, expel valuable content

prematurely and suffer many more cache rejections. All these factors not only increase transit traffic volumes but translate into many more transfers over long, multi-hop paths. Put differently, dedicating resources beyond a certain point to replicating elements from the "head" of the query distribution will not improve download performance by a useful amount (as it is ultimately hard-limited by the path characteristics of the "last mile"), but at the same time the cost associated with obtaining any other file will increase and worsen overall performance. The rapid diminishing returns of placing more mirrors in terms of latency and server load balance were also reported in the context of web mirror placement in [Cronin, 2002]. For current workloads it was found that an allocation of between 0.5% and 1% of total capacity offers a good compromise between hop/latency/load reduction and acceptable hit rate penalty.

Level of replication	0.3%	0.5%	1%	1.5%	2%	10 %
OHR (cumulative)	41.62% [±0.02%]	41.45% [±0.03%]	41.37% [±0.03%]	41.30% [±0.03%]	41.15% [±0.04%]	39.98% [±0.06%]
Average VCC load	12.02% [±0.06%]	10.60% [±0.04%]	7.50% [±0.05%]	6.20% [±0.04%]	5.03% [±0.03%]	1.90% [±0.04%]
Average IP hops to file source	2.63 [±0.02]	2.58 [±0.01]	2.56 [±0.01]	2.63 [±0.02]	2.89 [±0.02]	4.12 [±0.04]
Average connection RTT (ms)	36.98 [±0.11]	36.86 [±0.09]	36.26 [±0.10]	37.27 [±0.13]	39.08 [±0.11]	90.90 [±0.19]

Table 18: Variation of overlay metrics at different levels of replication.

7.5 Evaluation of the AVP as a system

7.5.1 Effect of AVP placement on network utilisation

It was demonstrated in the previous section that a distributed VCC deployment results in lower individual load for each VCC as well as improved path length and latency characteristics for the majority of peer transfers compared to a central VCC. These findings hint at the increased localisation of P2P traffic within the ISP network, which is beneficial not only for participating peers but for all users of the network. While the ISP backbone is typically over-provisioned, the lifetimes and greedy nature of TCP-based bulk P2P transfers increase queuing delay and can cause congestion during peak times. Avoiding unnecessary multi-hop transfers across the network reduces

backbone load and protects inelastic and interactive traffic (e.g. real-time streams, web) which due to its server-based nature is carried over it. Therefore, pushing the VCCs away from the core and placing them at the edge of the network may prove beneficial from a traffic engineering perspective.

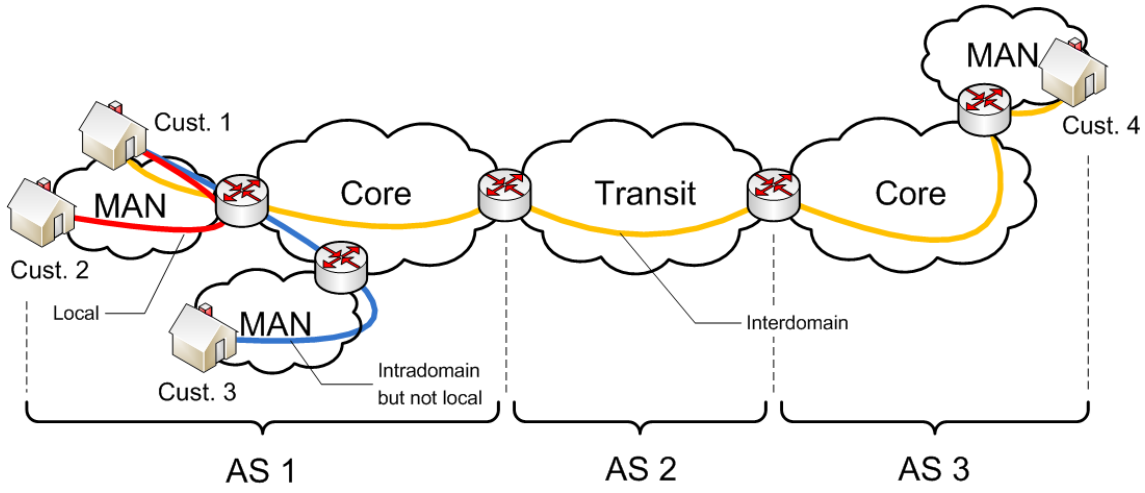

Figure 42: Different degrees of P2P traffic localisation.

While so far the separation of interest was between internal (often denoted local) versus inter-domain traffic for simplicity, it is now necessary to differentiate between different degrees of traffic localisation within the AS, as illustrated in Figure 42. With customers 1 and 2 corresponding to the same PoP the connection between their peer clients is local, consuming only access network and PoP resources. The connection between customers 1 and 3 is still intra-domain since they belong to the same ISP but has to additionally be carried over the ISP's backbone. Finally, the connection between customers 1 and 4 is inter-domain, with AS 2 providing transit service.

Employing the same simulation configuration used throughout this chapter, 10 new scenarios were simulated. These were:

- Normal peer operation without any AVPs.
- AVP operation with a single 1000 GB VCC placed at the core network.
- 4 multiple-AVP scenarios where the number of AVPs, each holding a 200 GB VCC, was varied between 2 and 5 in autonomous caching mode.
- 4 multiple-AVP scenarios where the number of AVPs, each holding a 200 GB VCC, was varied between 2 and 5 in cooperative caching mode with a replication factor of 0.5%.

Each scenario was simulated for a 24-hour period of operation after a 6-hour simulator warm-up. All AVPs consisted of one AOC/VCC pair each. For the single-VCC configuration, the AOC/VCC pair was placed at the network core identically to that discussed in section 7.4.3. In the multiple-AVP scenarios the AVP placement strategy was to place each AOC/VCC pair at a PoP, prioritising by PoP size. Unlike the previous section, there is no VCC placed at the core in this case.

In order to examine the effect different AVP deployment configurations have on traffic localisation and backbone load, the number of P2P transfer flows at the ingress queue of each backbone router was measured at frequent intervals. By comparing how much of the local peer workload is carried over the backbone in the aforementioned scenarios, the ability of different AVP configurations to localise P2P traffic at the PoP or regional aggregation level can be assessed. Figure 43 shows the number of P2P flows carried over the backbone as measured at four aggregation points, averaged over the 24-hour measurement period. Measurements were taken every 120 seconds to allow for adequate granularity and capture transfers of small files. For clarity, only results from normal P2P operation, the centralised VCC and the 5-VCC configurations (in both caching modes) are plotted.

It becomes clear that placing AVPs at the network edge not only matches the fundamental peer focus on edge resources, but succeeds in keeping a considerable portion of peer transfers local to the PoP or regional PoP cluster and away from the core. AOC local source promotion alone is able to reduce non-local flows by a substantial amount compared to normal P2P operation, as seen in the case of the central VCC where cached objects are still fetched from the core of the network. This is notable because, as discussed earlier, the AVP concept does not sacrifice peer performance in the pursuit of extreme localisation of peer traffic. These connections were kept local because appropriate local sources were found over paths that returned better latency estimates, even in the presence of the central VCC at the core. The 5-VCC edge deployment, however, increased localisation further by having an additional 7% (approximately 222) peer transfers on average satisfied locally at any given time for the workload examined. This is a notable improvement given that under autonomous caching mode the content of interest is in many cases not cached in the VCC local to the peer but in another, placed at a different PoP. Thus, many transfers still had to be routed over the backbone. Still, as the average hop reduction noted in section 7.4.3.1 indicated, placing VCCs at the edge increases the amount of valuable content that can be found one hop away, which in this case meant that more queries were satisfied at the regional

level. Clearly, however, the largest improvement in regards to P2P traffic localisation is unsurprisingly achieved by employing cooperative caching. With the most-requested files available in every edge VCC, the number of connections reaching the backbone at any time was 23% lower than the central VCC scenario.

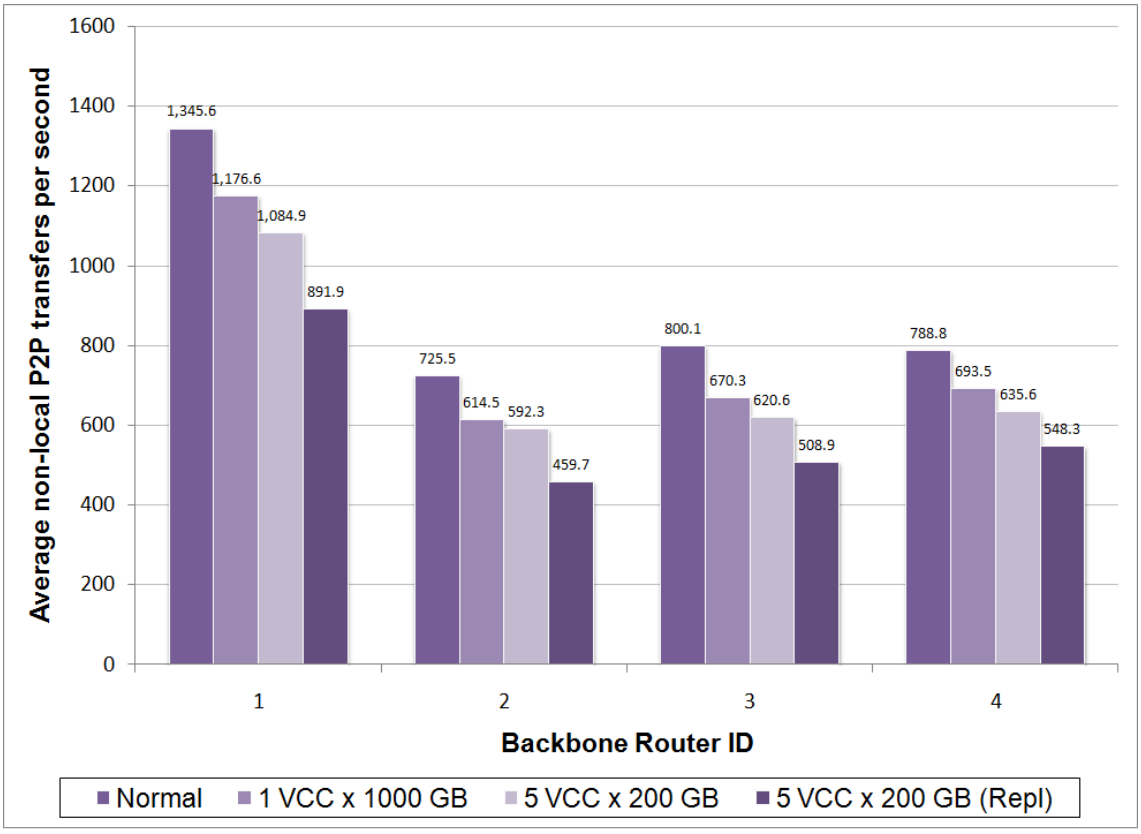

Figure 43: Effect of AVP placement on localisation of P2P transfers.

As a side note, it is evident from the graph that not all backbone routers receive the same amount of traffic. Specifically, router-1 handles the most flows while router-2 the least. Furthermore, unlike the rest, router-2 does not benefit as much from the transition from a central VCC to the autonomous 5-VCC deployment. This is a result of router-1 aggregating traffic from more PoPs, while PoPs themselves vary in size due to the number of subscribers served.

Figure 44 plots the cumulative distribution of the time needed for a local peer to fully download a 350 MB file in normal conditions, when a large central VCC is provided and when 5 smaller replicating VCCs are deployed instead at the edge. Pearson correlation coefficients of the data collected from the three separate simulation runs are presented in Table 19.

Comparing normal P2P operation with any of the two AVP deployments illustrates the importance of P2P content caching and intelligent source promotion in

optimising P2P operation. Crucially however, the superior localisation of transfer traffic, achieved by decentralising the caching infrastructure and placing the replicating VCCs at the edge of the network, has a very measurable effect on the download performance experienced by local peers.

In short, the provision and placement of AVPs at the edge of the network and particularly the use of cooperating VCCs offers clear benefits in terms of high traffic localisation inside the AS, reduction of backbone P2P load and improvement of local peer transfer performance.

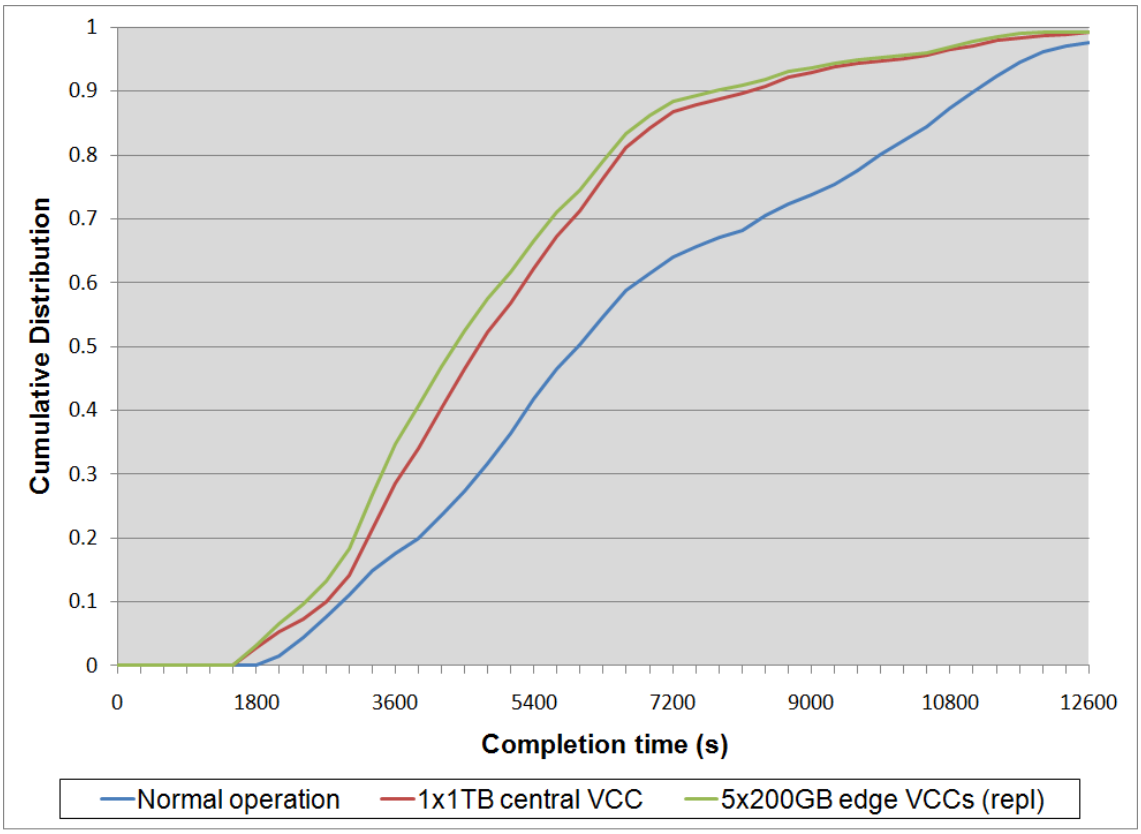

Figure 44: Effect of AVP placement on peer transfer performance.

Normal operation	Central VCC	Edge VCCs
0.993242	0.998315	0.997654

Table 19: Degree of correlation of completion time data between runs.

7.5.2 Economics of AVP deployment

So far, all evaluation of multiple-VCC deployments assumed a fixed amount of available cache capacity, varying only the number of active VCC components. This permitted the examination of how decentralised VCC caching compares to an effectively centralised cache for properties such as cache load and peer connection

latency. This examination however did not take into account one very critical factor: cost. ISPs have to optimise their network provisioning given a finite set of resources, while the market realities of this decade indicate that it is rather unlikely that an ISP will invest in any support infrastructure unless there is a strong expectation of profit or at least cost recovery. Given the level of flexibility afforded by the AVP architecture in creating deployments of different sizes, capabilities and degrees of decentralisation, it is therefore important to examine the practical relationship between AVP deployment costs and gains.

If T_t represents the transit P2P traffic generated (measured either in volume or rate), T_i the internal P2P traffic and C_{AVP} the cost of the AVP infrastructure which corresponds to the measurement period of interest, the P2P-related cost to an ISP over that period can be calculated using the following formula (where α and β serve as price scaling factors):

$$C_{total} = \alpha T_i + \beta T_t + C_{AVP} \quad (2)$$

For clarity, it is assumed that the hardware supporting each AVP execution environment (EEP) minus any storage devices is identical. That way the cost of the AVP infrastructure can be separated into a flat cost for the EEP hardware platform times the number of available EEPs, and the total cost of storage provided for VCC caching throughout the infrastructure (as a cost per GB). It is further assumed that cooling, power, installation/maintenance and administration costs for each EEP are included in the EEP cost. In other words, for n EEPs and m gigabytes of cache capacity, C_{AVP} can be calculated as follows:

$$C_{AVP} = nC_{EEP} + mC_{VCC} \quad (3)$$

While a multiple-AVP approach allows for relatively inexpensive hardware to be used without any mandatory fault-tolerance requirements (which is, after all, one of its immediate advantages), in the case of a deployment built around a central VCC the execution environment supporting the latter needs to be adequately provisioned to handle the combined workload of all local peers and ideally have load-balancing/failover capabilities. For simplicity, it is assumed that the increased hardware requirements of such a configuration can be expressed as multiples of the aforementioned EEP unit hardware cost C_{EEP}. In other words, although in practice a

central VCC will be considered to run on top of a single EEP, the higher specification hardware used can be represented by assigning a value higher than one to the n factor.

In summary, the cost function (2) can be calculated as follows:

$$C_{total} = \alpha T_i + \beta T_t + nC_{EEP} + mC_{VCC} \quad (4)$$

Employing the same simulation configuration used throughout this chapter, T_i and T_t pairs were collected for 26 different scenarios. These were normal peer operation and all 25 combinations of 1-5 AVPs with 200-1000 GB of VCC capacity each added in 200 GB increments, operating in autonomous caching mode. Each scenario was simulated for a 24-hour period of operation after a 6-hour simulator warm-up. All AVPs consisted of an AOC/VCC pair and AVP placement was identical to that described in section 7.5.1.

To gain an insight into real-world costs and scale the simulation results accordingly, a survey of ISP wholesale pricing for the year 2008 was carried out. While transit traffic can be billed based on volume (e.g. bits transferred), the predominant charging model is based on the 95-th percentile of peak rate (e.g. [Odlyzko, 2001b]). Specifically, bandwidth use is measured at regular intervals (typically every 5 minutes) and at the end of the billing period (typically a month) collected samples are ordered from highest to lowest. The top 5% of sorted samples is then ignored and usage is billed at the rate of the immediate next sample. Current transit prices for UK-based ISPs are widely reported to be in the region of £10 per Mbps per month [Evans-Pughe, 2009]. Hence, assuming a uniform daily traffic pattern so that the results from the 24-hour long simulations can be extrapolated to monthly use (i.e. no highest usage in any other day in the same billing month), β is £10 per Mbps per month. Storage was estimated at £0.20 per GB with an effective lifetime of 1 year, leading to a 200 GB disk amounting to approximately £0.11 per day. Finally, C_{EEP} was calculated to £2.19 for 24 hours of operation (capital and operational costs of £800 per year with a lifetime of 3 years).

Internal bandwidth cost is widely regarded to be effectively zero because the network represents a fixed cost where, as long as demand does not exceed capacity, carrying more traffic costs nothing extra to the provider. This capacity is normally not the full capacity installed but rather the maximum utilisation level, as designated by the provider, which maintains the QoS (Quality of Service) targets set. The prevalent over-provisioning strategy is to have in place at least twice the capacity consumed during peak utilisation in order to maintain low delay, jitter and loss rates and account for

failures or other events that may shift additional traffic over specific links. Thus, as long as the rate of aggregate traffic is not such that noticeable congestion is caused, the practical cost of traffic to the provider is zero.

For day-to-day operation it is often valuable however to be able to put a price on internal traffic and estimate operational costs. Personal communication with a large UK-based ISP revealed that (as of 2008) for practical purposes they calculate the ratio of transit to internal bandwidth cost as 4:1. In the interest of assessing costs with both models, the cost function was calculated for $\alpha=0$ and $\alpha=\beta/4$ (i.e. internal traffic cost of £2.5 per Mbps per month).

Figure 45 and Figure 46 plot the ISP cost associated with P2P operation C_{total} over a period of 24 hours[55] for different configurations of AVP deployments[56] when $\alpha=0$ and $\alpha=2.5$ respectively. The ability of the AVP to reduce transit traffic, both through caching and to a smaller extent with localisation alone, has been demonstrated already earlier in the chapter. The plots however show that for current prices transit traffic reduction not only offsets the cost of the AVP infrastructure even when numerous EEPs are involved, but results in substantial savings for day-to-day operation. These savings are significant even when internal traffic has a non-zero cost, as illustrated in Figure 46.

[55] Traffic costs were calculated on the monthly rate and divided by 30 to estimate daily cost.

[56] For consistency, in single-AVP configurations a value of n=1 is used.

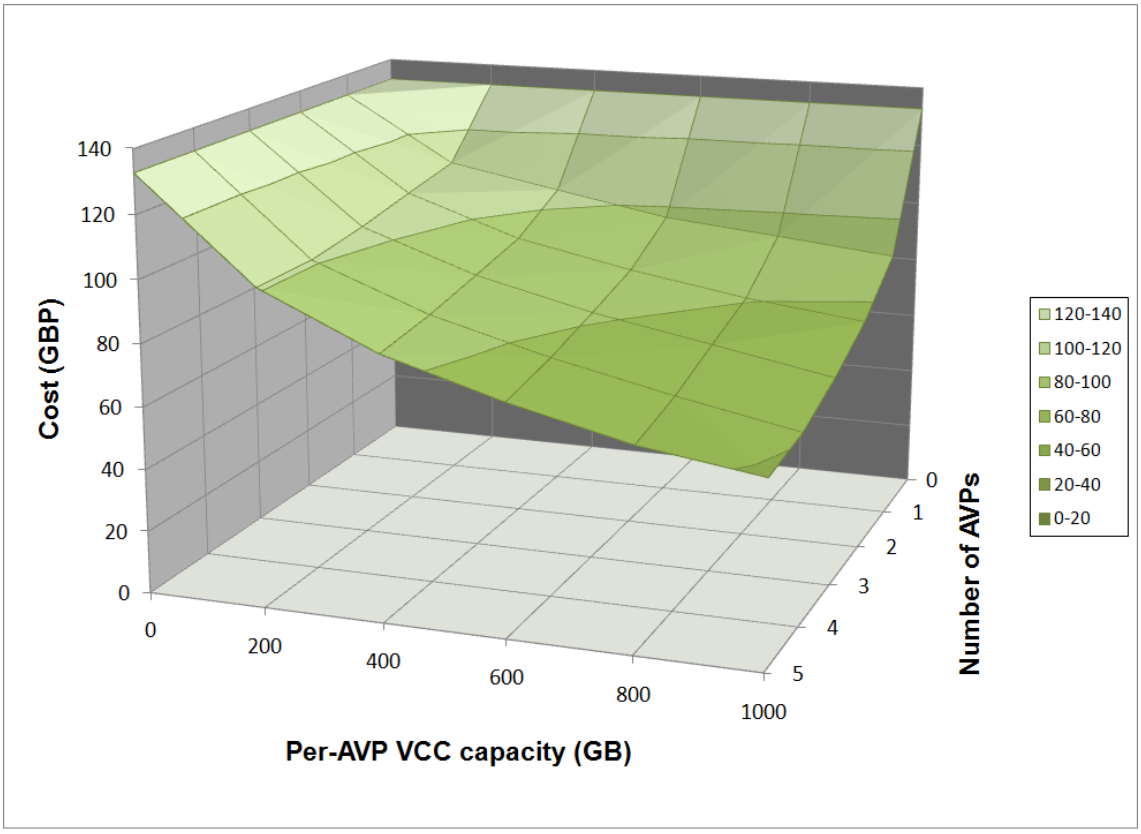

Figure 45: Costs over a 24-hour period for α=0, β=10.

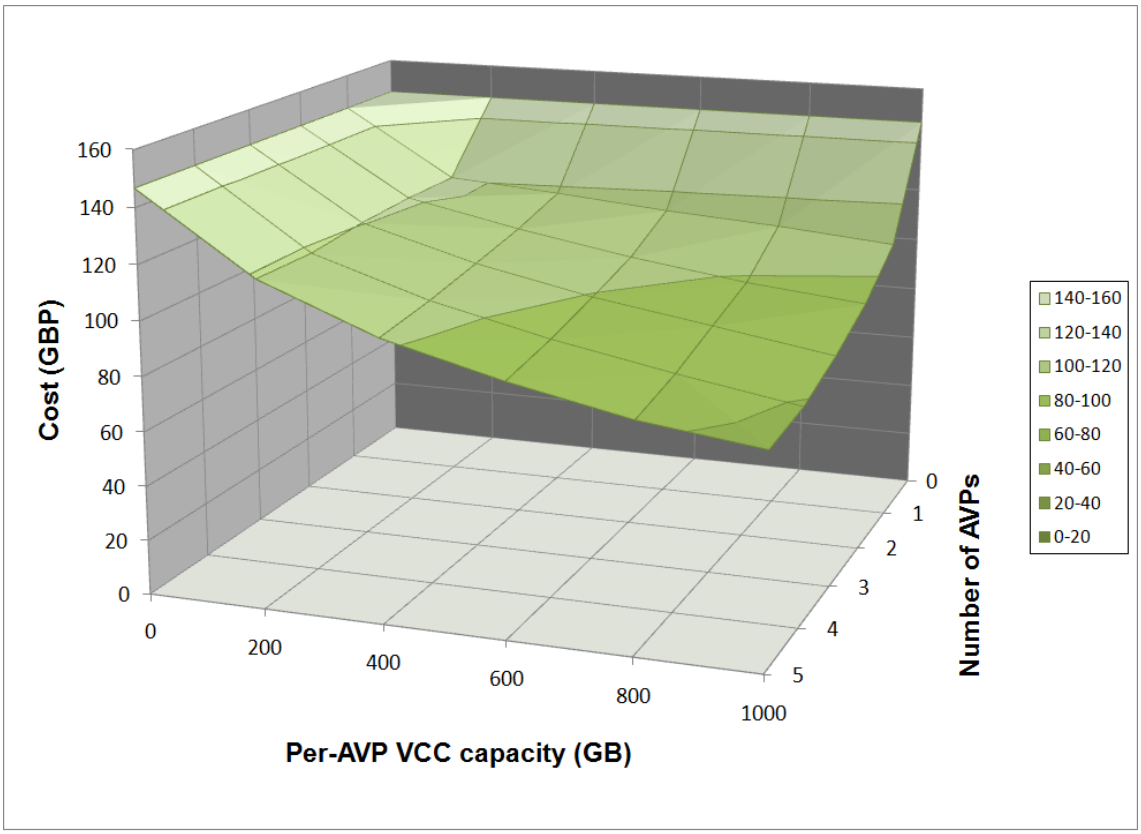

Figure 46: Costs over a 24-hour period for α=2.5, β=10.

Because prices for transit billed by volume (bits transferred) are not openly available, the price of transferring 1 GB of data over a transit link will be estimated as follows. Assuming that the transit link has a maximum capacity of 1 Gbps then in a 30-day month the maximum amount of data transmitted is 2,592,000 Gb (86,400 seconds times 30) or 324,000 GB (8-bits per byte), regardless of useful payload/overheads. At £10 per Mbps per month the maximum cost for the link is £10,000. Thus, 1 GB of data costs the ISP approximately £0.031 in transit fees. The minimum AVP configuration examined in this section of an AOC/VCC pair with 200 GB capacity costs £2.3 per day. Therefore even at the least cost-efficient configuration (from a useful cache capacity to EEP cost ratio standpoint) of those examined, the AVP needs to only serve 74.19 GB each day to pay for itself. Depending on the deployment size, each AVP served between 5 and 10 times as much data over 24 hours in the aforementioned scenarios.

Essentially, the AVP presents both direct and indirect gains for a network provider. Namely, an AVP deployment offers:

- Direct gains from the reduction of operational costs due to the decrease of P2P-generated transit traffic.
- Indirect gains from the enhancement of overall P2P application performance which in turn improves customer-perceived quality of service and satisfaction.
- Indirect gains from the improvement of network load conditions due to the management of P2P traffic which reduces the possibility of congestion.

The degree of transit traffic reduction was investigated for a variety of AVP deployment configurations and it was demonstrated that it can be translated into a tangible monetary gain. Furthermore, the simulation scenarios examined revealed improved transfer characteristics that can lead to an increase of P2P application performance. Finally, evidence of better management of incurred traffic load on the network infrastructure was found, especially with larger AVP deployments.

The improvement of P2P application performance offered by the AVP is of vital importance to an ISP because it affects the service quality perception of end-users, not only in regards to the P2P application in use but for the network service as a whole. This is crucial because even with a well-provisioned network and sensible QoS mechanisms in place, an ISP may have trouble communicating the level of service quality provided to its users, especially for overlay-based applications. Customers do not care about the challenges associated with service provision. More importantly, traditional QoS

mechanisms focus on the technical aspects of service provision, expressed in terms which describe well the requirements applications have from the network (i.e. throughput, delay, jitter, loss, etc), but have little meaning to an end-user. "Quality of Experience" (QoE) is a term increasingly used to bridge this gap [Kilkki, 2008], which describes how a customer perceives the usability of a service and how satisfied he/she is with the service he/she receives. The distinction is that QoS has become more relevant for the interactions between application and network while QoE focuses on the human dimension. As a result, QoE is subjective and while it is clearly dependent on QoS factors (i.e. packet loss will degrade application performance and ultimately affect negatively user experience), it is also affected by price (e.g. a customer paying a low price X for a service is more likely to tolerate congestion effects that if he/she paid 2X), public perceptions (e.g. "this level of service is acceptable because this is what others receive and deem acceptable"), ease of use and other factors.

QoE is intimately linked to customer churn and ARPU (Average Revenue Per User) which are important for the "ISP as a business" because they are indicators of its ability to extract profit from its given infrastructure investment. Given the frequent reports that providers are largely unable to extract value from their networks and be profitable [Crowcroft, 2003] (in part due to the lack of an appropriate pricing model [Byun, 2004; Odlyzko, 2001b]), the ability to stabilise customer churn (gain customer loyalty), attract new customers (gain competitive advantage) and improve ARPU (translate customer satisfaction into selling additional/higher value services and bundles) is crucial. The AVP provides an ISP with an additional tool to achieve these goals, which is particularly important given that its primary goal is the minimisation of costs due to inefficient peerings and operations. Specifically, the AVP improves customer QoE in two ways:

- It improves P2P application performance: The AVP optimises overlay structure and guides peers to sources which offer a higher probability of sustained high throughput. Although the AVP cannot guarantee that every single transfer will complete faster, since it does not control the availability of the necessary conditions (e.g. adequate number of sources, replicas over appropriate paths, etc), popular or semi-popular content which comprise the bulk of peer workload meet these criteria by definition. Furthermore, P2P traffic localisation ensures a consistent level of QoS as intra-AS traffic remains end-to-end under the control of one provider. Finally, source balancing and caching means that there is lower

possibility that a peer becomes overloaded with requests, preserving peer stability and responsiveness.

- It helps protect other traffic classes from resource starvation: P2P traffic localisation leads to less long-lived bulk transfers carried over the backbone and competing for resources with other traffic. Distributed caching and load-balancing makes link utilisation levels due to P2P traffic more stable and predictable, and helps an ISP apply more effective traffic engineering mitigating congestion. This allows other network services to operate unhindered.

7.6 Proxylet testing and trials

This section describes the component testing and trials performed on the AVP prototype. Four test scenarios were devised and carried out to test and assess the operation of the AOC prototype. The test apparatus consisted of a number of networked computers serving as Gnutella peers and EEPs. These computers were part of the UCL Networks and Services Research Lab (NSRL) research network which serves as a dedicated testbed for networking experiments undertaken by NSRL. The computers were arranged in two privately-addressed subnets representing an ISP network and the rest of the Internet respectively. All computers ran the Linux operating system with a 2.6 version kernel. The funnelWeb software (version 2.1.5) provided the ALAN EEP functionality. For the Gnutella client, the 'gnut' command-line based client [gnut] was chosen for the ability to directly access low-level Gnutella protocol functions (e.g. open connection, query etc) via its command line shell and the ease of remotely managing numerous clients via an SSH or telnet shell. A separate web server provided the proxylet repository.

Configuration and management of the AOC proxylets was performed by remotely connecting to their administrative interface via telnet. For EEP management, the control and monitor interfaces of funnelWeb were used. The Gnutella clients could also be remotely controlled at any time via SSH. The following test procedures are presented in high-level description. Appendix D contains a short guide of 'gnut' client commands, used during testing.

7.6.1 Controlled Domain creation and signalling restriction

The aim of this test is to assess the ability of the AOC to create a controlled domain (CD) and suppress the signalling traffic between the peers belonging in that

domain and the global Gnutella overlay. This is achieved by capturing, modifying and blocking P2P protocol messages to and from the CD as necessary, so that replies to the "internal" peers always originate from it. The desired outcome of the test is for Ping messages originating from within the CD to be answered only by other peers belonging in the CD or the AOC. No direct replies from peers outside the CD should reach the inquiring peers.

For this test, three PCs running Gnutella clients and one serving as an EEP are used. Two of the clients (Gnut1 and Gnut2) represent peers belonging to the CD and the third (Gnut3) a peer of the global overlay. The AOC proxylet runs on the EEP. The 'gnut' Gnutella client is run on all three PCs and a set of test files are shared.

The test procedure is the following:

1. Configure AOC with CD data (i.e. set IP address range)
2. Connect Gnut1 with AOC
3. Connect Gnut2 with AOC
4. Connect Gnut3 with AOC
5. Try to connect Gnut1 with Gnut3
6. Try to connect Gnut3 with Gnut2
7. Both attempts should fail
8. Try to connect Gnut1 with Gnut2
9. The connection should be successful
10. Issue query "testfile1.txt" from Gnut1
11. Wait for reply from Gnut2

The above procedure demonstrates two points. Firstly, in step 7, it is demonstrated that connections between the controlled domain and the rest of the Gnutella overlay are blocked by the AOC. Secondly, queries issued from within the controlled domain are only answered by other peers of that domain (step 11). This test was completed successfully.

7.6.2 VCC operation

The test procedure for assessing the operation of the VCC proxylet is very similar to the previous test. The aim of the test is to demonstrate the manipulation of P2P protocol messages and redirection of "local" peer queries so that file transfers are served by the VCC instead of "external" peers. This test presupposes the existence of a controlled domain, set-up as described earlier. The desired outcome is for successful queries made by peers within the CD to be served by other "local" peers and the VCC only. If the requested file only exists outside of the CD, the VCC should cache it and serve it to the requestor transparently.

For this test, a PC serving as an EEP and four PCs running the 'gnut' client are used. Two of the clients (Gnut1 and Gnut2) represent peers belonging to the CD. The rest (Gnut3 and Gnut4) are peers of the global Gnutella overlay. An AOC and a VCC proxylet run on the EEP. The test files to be requested are named "testfile1.txt" and "testfile2.txt". The former is shared by peers Gnut2 and Gnut3. The latter is shared by Gnut4 only.

The test procedure is the following:

1. Configure AOC with CD data.
2. Connect Gnut1 with AOC.
3. Connect Gnut2 with AOC.
4. Connect Gnut3 with AOC.
5. Connect Gnut4 with Gnut3.
6. Optional: Try to connect Gnut1 with Gnut3 to ensure CD operates as intended. Connection should fail.
7. Issue query "testfile1.txt" from Gnut1.
8. Confirm that Gnut2 replies. The query is blocked at the AOC and does not reach Gnut3. The AOC "sees" that the query was satisfied locally by Gnut2 (by its reply) and does not need to cache the file from Gnut3.
9. Issue query "testfile2.txt" from Gnut1.
10. Confirm that the AOC modifies the query and Gnut4 replies to VCC.
11. Confirm that VCC replies to Gnut1, offering the file.

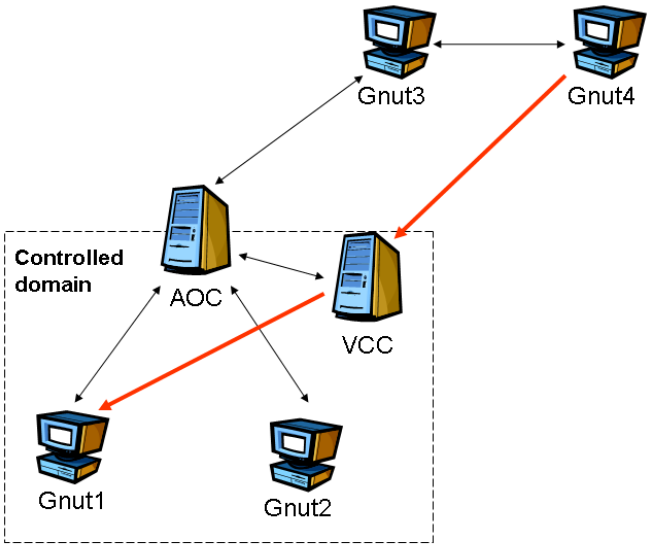

Figure 47: VCC operation test set-up.

This test demonstrates that the AOC modifies queries before forwarding them outside the CD and redirects file transfers to the VCC when a query cannot be satisfied from "local" peers. As a result, future queries can be served from the VCC and inter-domain traffic is reduced. The test was carried out successfully.

7.6.3 Routing control

The aim of this test is to demonstrate the ability of the AOC proxylet to suppress the forwarding of messages to a neighbour if it senses path state deterioration while forwarding them normally to other neighbours. By avoiding the passing of additional load over the overlay connection which maps onto the degraded path, it expedites the recovery of that path from congestion.

For this test, the EEP and three additional PCs running the 'gnut' client are used. In lack of a better way to control overlay link congestion or delay in real time for this test, the threshold parameters are manually passed to the AOC proxylet to indicate the link's state degradation. The test file "testfile1.txt" is shared by both Gnut2 and Gnut3.

The test procedure (set-up illustrated in Figure 48) is as follows:

1. Connect AOC to Gnut1.
2. Connect AOC to Gnut2.
3. Connect AOC to Gnut3.
4. Issue query for "testfile1.txt" from Gnut1.

5. Confirm replies from both Gnut2 and Gnut3.
6. Degrade link state to Gnut2 (manually increase threshold value of the connection between AOC and Gnut2).
7. Issue query for "testfile1.txt" from Gnut1.
8. Repeat step 7 a number of times.
9. Confirm that a reply to a query is always received by Gnut3 but rarely by Gnut2.

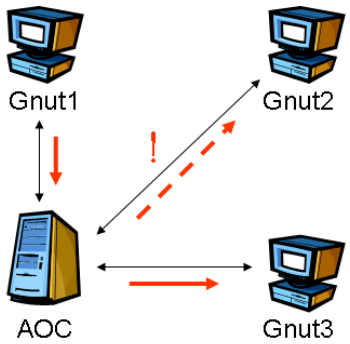

Figure 48: Routing control test set-up.

This test demonstrates that once the threshold value assigned to a connection by the AOC is increased, fewer messages are forwarded over it. Because the aim of probabilistic routing control is to relieve an overloaded link but not isolate it completely (which incurs the danger of fragmenting the overlay), whether a message will be forwarded to it depends on the comparison of the overlay connection's assigned threshold to a random value. The higher the threshold value, the lower the probability that a message will be forwarded over it. Thus, queries are expected to reach Gnut2 but their number should be noticeably smaller than these reaching Gnut3. Table 20, below, presents the variation in the number of replies from Gnut2 in response to queries, when different threshold values are used. For each run, ten queries were made.

Threshold	Replies from Gnut2	Replies from Gnut3
0.3	7	10
0.4	6	10
0.5	6	10
0.6	3	10
0.8	2	10
1.0	0	10

Table 20: Reduction in traffic to 'Gnut2' with different threshold values.

This test was completed successfully.

7.6.4 Protocol message tunnelling

The aim of this test is to demonstrate the message tunnelling capability of the AOC which can be used as part of overlay topology control. Overlay connections will be initiated and terminated between AOC proxylets to re-configure the mapping of overlay to underlay.

For this test, three PCs running Gnutella clients and two EEPs are needed. An AOC proxylet runs on each EEP. The test file "testfile4.txt" is shared by Gnut3.

The test procedure is as follows:

1. Connect Gnut1 with AOC1.
2. Connect Gnut2 with Gnut3.
3. Connect Gnut2 with AOC1.
4. Connect AOC1 with AOC2.
5. Issue query "testfile4.txt" from Gnut1.
6. Wait for response from Gnut3.
7. Connect AOC2 to Gnut3.
8. Wait for AOC2 to advertise route to Gnut3, to AOC1.
9. Disable Gnut2 (stop the client).
10. Issue query "testfile4.txt" from Gnut1.
11. Wait for response from Gnut3 via AOC tunnel.

This test demonstrates the ability of AOC proxylets to create alternative connections and tunnel P2P protocol messages in order to maintain the overlay in a good state. In this test, once Gnut2 was disabled indicating a failure or unexpected exit from the overlay, messages between Gnut1 and Gnut3 were tunnelled through the AOCs.

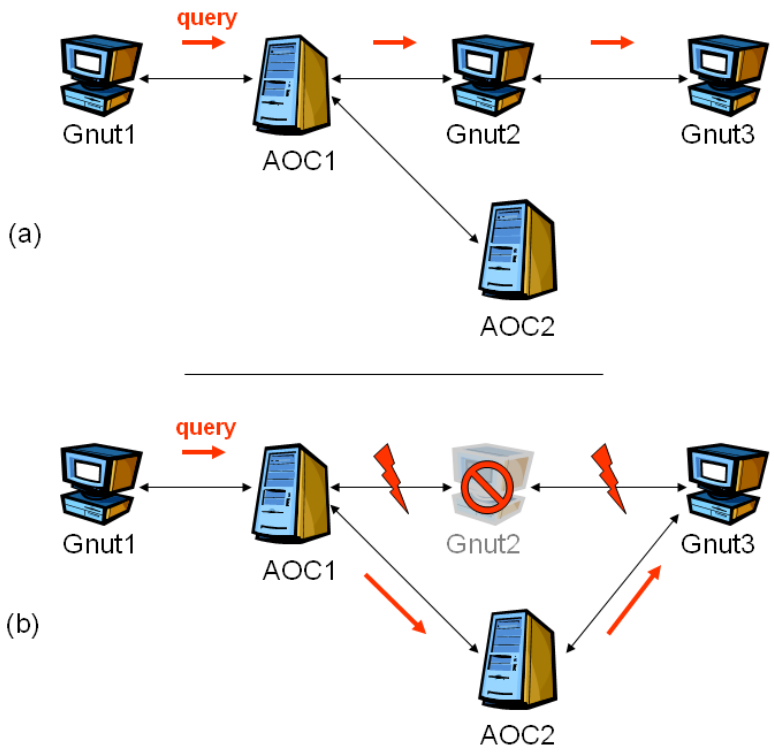

Figure 49: Topology control test set-up.

7.7 Summary

This chapter presented a comprehensive evaluation of the effectiveness of the AVP concept under numerous scenarios. It was demonstrated that local source promotion as enabled by AVP routing and topology controls can facilitate the preference of local sources over foreign ones by local peers and reduce average connection path length and RTT delay, contributing in faster downloads. The reliance on local source availability inherent to passive local source promotion is overcome with the introduction of VCCs, which apart from reducing path length and latency further achieve significant savings by eliminating redundant transit traffic. VCC performance was examined under various replacement strategies and deployment configurations. It was found that while a central VCC may seem attractive from a short-term investment perspective, a distributed deployment at the network edge achieves important traffic engineering objectives and can improve customer service perception. The chapter was concluded with a discussion of four test scenarios applied to the prototype implementation of the AOC and VCC proxylets.

8. CONCLUSIONS AND FUTURE WORK

8.1 Summary and contributions

As discussed in earlier chapters, current P2P services do not fit well in the present Internet, leading to increased costs for network providers, disruption of non-P2P Internet services and inefficient operation for the services themselves. While imposition of bandwidth caps, port blocking, traffic shaping and even capacity over-provisioning have been employed to address these issues, it is now becoming widely accepted that none of these approaches can fully solve the problem in the long term. In addition such approaches can be costly and, with the exception of over-provisioning, unpopular with the customer base of each provider.

The work described in this thesis developed from the position that instead of treating P2P applications as unwelcome applications whose use needs to be restricted, ISPs need to find ways to ensure their smooth integration within the Internet ecosystem and encourage their fair and considerate use of network resources. To meet these goals, the ability to manage how P2P services operate inside their networks in a transparent and flexible manner is needed.

The Active Virtual Peer (AVP) provides such a framework for the flexible management of P2P overlays. Application Level Active Networking (ALAN) was chosen as a natural vehicle to enable evolutionary adaptation on the application layer. The incorporation of ALAN maintains the decoupling of network and application layers and provides operational support. Importantly, the ALAN infrastructure enables the AVP to respond to changes in network conditions rapidly and on timescales that match native P2P application behaviour.

The presented AVPs for Gnutella implement means for overlay control with respect to access, routing, topology formation, and application layer resource management. The AVP concept not only allows the combination of separate algorithms and techniques with proven merit to address individual issues, but allows them to

operate over a flexible and adaptive framework. The significance of this approach lies with the expandability and adaptability of the system as P2P services evolve. The AVP can offer solutions suited to a particular P2P service when and where needed, and can be expanded as new P2P services appear.

The flexibility of the AVP architecture does not end with the ability to deploy different AVP configurations, add or take offline proxylets on-the-fly, employ various caching strategies or select between autonomous and cooperative VCC caching. Crucially, the AVP architecture provides for incremental deployment of the infrastructure, which is a practical means to address scalability issues, accommodate budget considerations as well as attend to new needs that did not exist or were not identified upon initial deployment. Incremental deployment can come in the form of both upgrading existing EEPs with more capacity (e.g. more disk space for VCC operation or more CPU power/memory to deal with service load) and installing new EEPs in available PoPs (thus enlarging the deployment). As a result, the architecture allows for varying degrees of infrastructure homogeneity both in the hardware capabilities of EEPs and in the service roles of AVPs. It is quite possible, for example, to create hybrid deployments where a "core" of VCCs performing autonomous caching is supplemented with a few, replicating VCCs in selected locations.

Furthermore, the AVP framework allows the provision of value-added services on top of the core management functions. For example, the AVP architecture allows for VCCs to be operated as a content delivery platform for the subscribers of an ISP. The ISP can deploy specialised VCCs that accelerate delivery of specific content for a fee alongside its regular deployment with minimal additional effort due to the support for customised proxylet versions running side-by-side by ALAN and the manageability afforded by AVP policies. Additional components, intended to improve specific P2P application facilities or offer some service differentiation may also be developed.

Presently, a lot of effort goes into developing better techniques and protocols to address issues of current P2P systems. Significant work is devoted to ensuring peers are well-behaved, stay available for as long as possible and do not free-ride. However, users are disinclined to offer private resources to support other, anonymous users and in general contribute for as long as they receive something in return. This dimension is a major cause behind uneven P2P service performance and overlay volatility. Operated by the ISP, the AVP overcomes many of these issues by providing the support infrastructure necessary to address performance and stability issues without imposing any burden to end users. AVPs serve as selfless peers of high availability and good

connectivity, which organise into a stable foundation upon which the local overlay can evolve. Crucially, transparent operation ensures that built-in protocol incentives or mechanisms do not clash but rather improve AVP scope.

While the primary objective of the AVP infrastructure is to eliminate inefficient peerings and operations due to the costs incurred for the ISP, it was demonstrated that the promotion of local sources over shorter paths, the caching of highly-popular files at the edge and the load-balancing of the local overlay results in noticeably faster downloads, improved resource availability and increased overlay stability for the customers of the ISP. This improves their Quality of Experience and provides the right incentive for both users and application developers to cooperate with the AVP for mutual benefit. This presents a long-term goal towards native P2P application AVP-awareness which will further improve efficiency and expand the scope of beneficial guidance peers can receive from the network.

In conclusion, the work described in this thesis made the following contributions, restated here for the benefit of the reader:

- The aspects of peer behaviour that can be managed effectively without imposing fundamental limits to P2P service operation were identified. Realistic scenarios of how these controls can be applied on existing protocols were formulated.
- A novel framework architecture was designed to implement the proposed management functions. The architecture was designed to meet numerous requirements and provides for high flexibility in deploying, managing and extending the framework components and their capabilities.
- A policy model for the automated configuration and management of the framework elements was defined.
- A prototype implementation of the framework was developed to investigate and evaluate the validity of the proposed management functions through experiment.
- Caching strategies which are suitable and effective for P2P workloads, match peer behaviour and take advantage of capabilities specific to the AVP were developed and evaluated.

Finally, the comprehensive evaluation of various functions and capabilities of the framework provided valuable insights into large-scale peer behaviour and its effect on network operation.

8.2 Related work

The work described in this thesis began in 2002 when P2P computing in its current form was in its early stages. Between that period and the completion of this thesis, P2P has evolved significantly, becoming a thriving research area. At the same time, this rapid evolution meant that any research was essentially applied on a moving target. Within a few years numerous protocols appeared only to later disappear after a brief period of popularity, while user habits also evolved. More importantly, along with the steady growth of P2P traffic, came the realisation of the limitations of earlier systems and the need for better scalability and performance. As discussed earlier, scalability and performance issues were predominantly addressed in a protocol-centric manner, while network resource utilisation, which is more pressing to ISPs, received less attention. Below, is a list of research that can be considered related to the AVP.

The inability of most early unstructured P2P applications to maintain topology and membership information in an efficient manner as they grow larger has been partly acknowledged by the developers of second-generation P2P protocols (e.g. Limewire [Singla, 2002], Kazaa [Kazaa] etc), who employed the concept of "super-peers". As discussed in Chapter 3, this concept suggests the creation of a two-level hierarchy inside the overlay network, where the super-peers, i.e. peers possessing better networking capabilities and processing power, undertake message handling and routing on behalf of ordinary peers. The decreased number of peers responsible for these protocol functions reduces the signalling traffic significantly, making the network more scalable. The AVP concept shares some minor similarities in the sense that it also involves peers with additional functionality and responsibilities and it contains functions that reduce signalling traffic. AVPs, however, have a much broader scope than merely aggregating signalling traffic and search results. They are primarily designed to manage the overlay and can allow or prohibit access to peers or resources selectively, intelligently adapt to changes in the network, actively improve performance (i.e. not as a by-product of reduced signalling traffic) and support multiple P2P protocols. Finally, AVPs are provided by the ISPs as dedicated infrastructure to manage and improve the state of a P2P overlay. Super-peers in contrast, are run by their respective users and as such consume resources primarily for own use, offer no guarantees on availability and quality of service and operate with no consideration for the service's impact on the network.

Resilient overlay networks (RONs) [Andersen, 2001] were proposed as a way to improve end-to-end reliability and performance by offering control and choice on how data can be transmitted for end hosts and applications. Much like P2P networks, a RON is an application-layer overlay on top of the existing Internet routing substrate. RON nodes aggressively measure and exchange information about the quality of available routes and use it to provide alternative routes in the face of a problem, faster than standard lower-layer mechanisms (e.g. BGP). The ability of AVPs to monitor and restructure areas of a P2P overlay to improve stability and performance is not, however, comparable to the RON concept. Firstly, AVPs are transparent to peers, which remain unaware of any overlay control operations. Unlike RON which provides a client API (Application Programming Interface) an application must explicitly call to use the service, AVPs encapsulate peer messages and tunnel them to their destination without any change to the P2P application. More importantly, RON and overlay routing networks in general can be said to violate routing polices. Nodes provide transit in violation of inter-domain routing principles [Gao, 2001] while selfish routing [Qiu, 2003] can undermine traffic engineering decisions. On the other hand, AVPs are ISP-managed infrastructure designed to operate within the confines set by the provider, avoiding conflicts with any inter-domain routing policies and traffic engineering practices applied.

Caching of P2P traffic has gained considerable traction in recent years. A number of companies such as Cachelogic [Cachelogic] and Sandvine [Sandvine] offer commercial products to ISPs in the form of network elements that reside at the gateways of the ISP network and redirect P2P download traffic to a local cache. Being commercial solutions, not adequate information is publicly available (to the best of the author's knowledge) for a fair and detailed assessment of their performance and capabilities. From what is known however, they show considerable differences to this work. The VCC, as part of the AVP framework, offers incomparable flexibility both in operating as well as in evolving the system. Unlike the aforementioned, generally static, solutions, under the VCC architecture a number of VCCs may be deployed on demand to address changes in network conditions in timescales that match P2P operation. Extending the system can also be as straightforward as adding or upgrading an execution environment, with generally no downtime. More importantly, the AVP/VCC is part of the P2P network, interacts with peers directly and gathers information from within the application overlay. This compatibility with the P2P service paradigm

strengthens the adaptability of the concept to future types of P2P services that may operate differently than the current generation.

In [Wierzbicki, 2004] the authors apply existing cache replacement policies, developed for web caching, and compare them with others they propose for use with P2P traffic. They examine the characteristics of traffic generated by the Kazaa file-sharing application and for that reason focus on the potential of policies that operate on file chunks to reflect Kazaa protocol behaviour, although they are also evaluated on caching entire files. Other work in this direction includes [Leibowitz, 2002], [Saleh, 2006] and [Dunn, 2002]. All four concentrate on policy design and imply a traditional single cache deployment at the entry point of the network. The AVP research work similarly proposes and evaluates a number of policies applicable to P2P workloads, but more importantly describes a particular architecture and deployment method, and replacement policies which reflect the VCC mode of operation and capabilities. These notably include multiple P2P protocol support and dynamic multi-cache deployment. Furthermore, the VCC is only one component of the larger AVP framework which aims to improve P2P application performance beyond caching. The aforementioned works instead appear to overspecialise on particular protocol features (e.g. file chunk caching for Kazaa), sacrificing transparency, forgoing the ability to support multiple protocols from the same cache and diminishing the relevance of the proposed techniques on future P2P applications.

Towards the very completion of this work, the idea that the impact caused by P2P applications on ISP networks needs to be addressed holistically and that ISPs can achieve much by encouraging "good" application behaviour in various ways compatible to them, which is central to the AVP concept, started appearing elsewhere. Aggarwal et al [Aggarwal, 2007] acknowledge the impact P2P applications have on ISP networks and propose the use of "oracle" services, provided by ISPs, to reduce inefficient and costly inter-ISP connections. The oracle service can propose appropriate neighbours to a peer using information like customer connection characteristics (capacity, congestion level, etc.) or geographical information (PoP, city, etc), that is readily available to an ISP but harder for a peer to infer on its own. Peers can then use these suggestions offered by the oracle instead of selecting neighbours independently. The claimed benefits are that this way an ISP can influence P2P routing decisions and "regain" its ability to perform traffic engineering, without degrading P2P application performance.

Xie et al [Xie, 2007] similarly attribute the lack of input from ISPs, especially in relation to lower-layer traffic engineering, to the inefficient and unfair utilisation of

network resources by P2P applications. They propose a system that allows network providers to explicitly provide more information to P2P applications, with the aim of achieving better traffic control in cooperation with the latter. Again, much importance is given in the value and breadth of information an ISP can readily provide but peers cannot easily gather on their own. The proposed P4P framework consists of a control plane and a data plane component. The control plane comprises of portals called "iTrackers" that provide three kinds of ISP information to applications: network status/topology, provider guidelines/policies and network capabilities. The data plane provides P2P applications with feedback from routers, which can be used for the adjustment of peer flow rates. The end result is, like in the aforementioned oracle service, to allow P2P applications to use information provided by the ISP and make more informed decisions regarding overlay formation and routing, with the aim of limiting costly and inefficient uses of the available network resources.

Ono [Choffnes, 2008] operates on the same premise as the aforementioned oracle service but proposes the inference of peer proximity and path state information from large Content Delivery Networks (CDNs) which, as part of their operation, have already deployed extensive sensor infrastructures to perform such measurements. The goal is to avoid having the ISP compile such information through its own means, which the authors consider a major obstacle in deploying the oracle service. The "hints" inferred from the CDN infrastructure are used to direct peers to suitable neighbours with the aim of reducing transit traffic and improving transfer characteristics.

All three works bear close similarities to the AVP concept, as they select as their starting point the realisation that the best way for ISPs to deal with the negative effects of P2P applications running on their networks is to provide infrastructures that assist the latter in treating network resources more considerably. AVP, P4P and the oracle service base a lot of their functionality on the privileged position of the ISP to obtain current and accurate information on the network and use it for P2P traffic optimisation, which is something most P2P-related research overlooked for years. Ono is equally dependent on such information but attempts to acquire it indirectly through CDNs. Finally, all concepts share the view that their respective control is beneficial not only to the ISP but also to the end-user who, in general, will experience improved application performance.

The AVP concept has however significant differences from the other proposals. The first is transparency: The AVP does not require any deliberate peer cooperation or changes to P2P protocols in order to interact with the overlay. All actions are carried out by message manipulation and control of peer groups. The other three concepts instead

require for the P2P applications to actively query their respective infrastructures, which implies that changes to P2P protocols and applications are necessary[57]. In the case of Ono this was made explicit by the development of an Ono plug-in module to work with a popular BitTorrent client. However, while the particular client supports such additions through an open architecture (and it is assumed it was selected for that feature), the vast majority of contemporary applications do not. The support of P2P application developers is, thus, not simply desirable (as in the AVP) but crucial and actively sought after. This, in return, requires for a significant number of ISPs to adopt any of these solutions, in order for a critical mass to be formed and convince developers to modify their applications. By losing transparency and having the peer consult the infrastructure, issues arise about how peers will discover infrastructure nodes and interpret the information provided. In such a scenario, is the infrastructure and communication protocol standardised and the various P2P applications expected to adapt, or multiple interfaces built to work with each protocol? Clearly, this leads to a chicken-and-egg situation further complicated by the existence of three competing approaches.

The loss of transparency raises robustness and scalability issues. Since the AVPs appear like ordinary peers, the loss of an AVP will not affect the P2P network more than the departure of any well-connected peer. To ordinary peers this will look like another peer departure amongst the many they deal with in each session. The loss of the oracle service or of the iTracker in contrast means total service unavailability. If peers are not granted the ability to operate independently in the face of oracle or iTracker unavailability (for instance in order to enforce exclusive use of the system, as otherwise it may be ineffective) the P2P network will be seriously impaired. For that reason both concepts suggest the deployment of more than one server for scalability and fault-tolerance. Essentially, both P4P and oracles trade off transparency for a more straightforward type of control. This is particularly true of the P4P which is more coupled to the application than the oracle service and, as such, less flexible.

Ono forgoes the fundamental problem of obtaining current and valid network information by “piggybacking” on the infrastructure employed by an unrelated party, a

[57] P4P also requires modifications to DNS servers to support a P4P-specific record type.

CDN provider. However inventive, this approach results in a parasitic relationship which raises two main concerns: Firstly, the ISP which is described as the primary stakeholder (i.e. Ono primarily aims to reduce transit traffic which is purely an ISP concern) in Choffnes et al [Choffnes, 2008] does not have any control on the accuracy of the information inferred. Ono "hints" indicate peer proximity with some degree of success (e.g. peers that demonstrate similar CDN redirection behaviour are considered close) but are not accurate. More importantly, path RTT is a bad indicator of AS hop distance. To an ISP interested in minimising its own transit traffic, it is little comfort to know that most locally-originating peer connections terminate a few IP hops after leaving its own border routers. Secondly, with Ono the owners of the CDN end up providing an additional service without their consent or compensation. While in the aforementioned paper the authors are careful to stress that, from their assessment, Ono does not place a large burden on CDNs, it is not clear if the other side agrees. Even if utilisation due to Ono is small compared to legitimate CDN traffic, it is possible that CDN providers will seek reimbursement for the service they provide as they are profit-oriented and bear the costs of the infrastructure. Essentially, Ono trades off the technical and economic issues involved with deploying a support infrastructure by relying on less accurate but seemingly free measurements obtained indirectly.

With AVP, the cost of providing the necessary infrastructure "buys" the ISP much better control over transit traffic minimisation (as local and foreign sources are identified with certainty) and the ability to customise peer management to the specific needs and realities of its network. Unlike Ono, with AVP (as well as with P4P and oracles if similarly configured) the ISP has the ability to ensure that lower layer traffic engineering decisions are not invalidated by overlay paths.

Crucially, the AVP comprises a framework of technologies where locality-biasing or ISP-provided network information are not utilised in isolation but reinforce each other and complement other techniques. The AVP combines locality-biasing with routing controls that ensure overlay and peer load-balancing, and VCCs which ensure that the content most likely to cause transit costs and link congestion *is* local.

The caching capability of the AVP not only makes for a more pragmatic but also for a more effective solution. On their own, localisation mechanisms are unlikely to offer the claimed benefits. Factors such as the size and dispersion of the swarm (i.e. peer population actively involved in the exchange of particular content at a given time), its maturity (i.e. number of available complete sources versus downloaders) or the rate of source churn are unique and mean that blindly pursuing localisation carries the danger

of breaking the overlay into smaller, weakly linked “islands” or ultimately causing slower transfers. On the other hand, having P2P applications purposefully mix P4P, oracle or Ono suggestions with random “noise” to ensure overlay robustness and scope limits both the amount of transit savings attained and the appeal of the aforementioned proposals. Furthermore, as Karagiannis et al have noted [Karagiannis, 2005], a P2P locality-aware solution can be expected to generate five times more traffic on average than a “perfect” caching solution. More importantly, because last-mile access technologies are still notoriously asymmetric (especially shared mediums like cable), localisation may have adverse effects on uplink last-mile congestion. Caching is ideal for asymmetric access technologies because it alleviates peer upstream load and congestion in the part of the network where upgrades are practically unfeasible. Finally, as discussed before, VCC caching is transparent to peers, requires no protocol modifications and can accommodate different P2P protocols from the same content.

At any rate, the fact that recent work adopts the same ideas that sparked the AVP research some years earlier can only testify to the concept’s validity and potential. More importantly, the AVP concept is not necessarily incompatible or competing with such approaches. While the need for transparent operation allows only for specific control mechanisms to be implemented, the ability of peers to query the infrastructure for information and act accordingly creates numerous possibilities for broadening the scope and capabilities of the AVP. Cooperation between P2P application developers and ISPs is a win-win situation which can benefit all parties immensely. Moreover, VCC caching can work particularly well with P2P protocols which support locality awareness; the latter providing another layer of efficient source selection and content distribution functionality within the ISP domain, once brought locally by the VCC from foreign sources.

8.3 Future directions

The AVP architecture can be supplemented, extended or modified in many ways. The author has identified the following as promising starting points:

Although planned from the beginning as a core AVP component and capability [Koulouris, 2003], the Network Optimisation Component (NOC) was not fully specified or prototyped during the duration of this research. The separation and componentisation of AVP functions led to the AOC and VCC components taking priority along with the AVP simulator, as they encompassed necessary functionality both from a research and

practical prototype operation perspective. The NOC remained as a planned expansion, intended to provide the AVP with direct traffic engineering capabilities beyond what could be achieved by the AOC through manipulation of the overlay.

Because P2P traffic is elastic, long-lived, edge-to-edge and, crucially, bandwidth-consuming, alleviating its impact on the backbone through localisation is only part of a broader traffic engineering goal. Ensuring traffic load is evenly spread across the network can help an ISP utilise its existing infrastructure more efficiently, smooth-out demand peaks, avoid congestion hotspots and delay a capacity upgrade. In that direction, many providers already apply such traffic engineering at the packet or flow level by distributing traffic over equal cost paths, either via MPLS or IP interior routing (for example link weight tuning in OSPF [Fortz, 2000]). The connection endpoints, however, are determined by the application, limiting the effect of management to the choice of intermediate hops. Clearly, the ability of the AVP to influence peer source selection (and thus the location of one endpoint of a TCP transfer) is powerful. With the NOC providing underlay information, the AVP can implement targeted and increasingly effective distribution of traffic based on, for instance, peer-PoP association which real-time RTT probes cannot reveal. Furthermore, the NOC can provide the necessary bridge between the AVP deployment and any lower-layer traffic engineering or QoS mechanisms enabling a synergistic relationship between underlay and overlay management.

The recent appearance of similarly-themed work (i.e. P4P, oracle service) presented both a benefit and a challenge. From one hand it communicated the urgent need for a tighter coupling of overlay and underlay to a wider audience. On the other, it limited the room for original contribution. Still, future work can avoid duplicating effort and, where applicable, incorporate P4P/oracle contributions to the AVP or vice-versa.

Another direction for future work is the implementation and integration of additional P2P protocol modules in the AVP. The simultaneous handling of multiple P2P application overlays and support of several protocols in caching scenarios is valuable for two reasons: First, the ability to bridge together different overlays may reveal important gains in terms of resource utilisation and localisation of traffic (e.g. exploiting proximity of peers which use different protocols but exchange the same object). Second, the utilisation of the same cache by different protocols may give valuable insights in terms of query workload overlap and evaluate how cache hit rate and replacement strategy performance are affected.

The security-related aspects of the AVP architecture are another area that can benefit from further study. As part of the ISP infrastructure, an AVP deployment must be secure from unauthorised access which could be used for performing denial of service attacks or distribution of malware. A basic level of security is afforded by the ALAN architecture and the sand-boxing features of the Java programming language in executing proxylets. Further work could start from strengthening the AVP architecture (for instance, by adding SSH protocol support in the administrative console, apply cryptographic techniques for the run-time authentication of AVPs and encryption of any sensitive communications, etc) to equipping AVPs with facilities to identify peers that transmit malware so that corrective action can be taken.

Further improvements can be made in the area of VCC caching. In particular, the investigation of additional caching strategies that can take advantage of the flexibility afforded by the AVP framework and of techniques to eliminate inefficient caching operations or maximise utilisation of caching capacity are two areas that can be explored further. Examination of how natural replication of content on peers can be leveraged in novel ways to optimise cache performance is another attractive prospect.

Finally, many other aspects of the AVP concept and architecture can be extended or improved. The author hopes that the present thesis succeeds in providing the inspiration (or, in case of disagreement, motivation) needed.

REFERENCES

Australian Copyright Act of 1968, [online], available at: http://www.austlii.edu.au/au/legis/cth/consol_act/ca1968133/, [Accessed 6th October 2009]

ADAR, E., HUBERMAN, B.A., 2000, "Free riding on Gnutella", First Monday, Vol. 5 (10)

AGGARWAL, V., FELDMANN, A., SCHNEIDELER, C., 2007, "Can ISPs and P2P Users Cooperate for Improved Performance?", ACM SIGCOMM Computer Communication Review, Vol. 37 (3), pp. 29-40

ALAN, UTS ALAN/funnelWeb website, [online], available at http://dmir.it.uts.edu.au/projects/alan/, [Accessed 6th October 2009]

ALBERT, R., JEONG, H., BARABASI, A., 2000, "Error and attack tolerance of complex networks", Nature 406, p. 378

ANDERSEN, D.G., BALAKRISHNAN, H., KAASHOEK, M.F., MORRIS, R., 2001, "Resilient Overlay Networks", in proceedings of 18th ACM Symposium on Operating Systems Principles (SOSP '01)

ANDROUTSELLIS-THEOTOKIS, S., SPINELLIS, D., 2004, "A Survey of Peer-to-Peer Content Distribution Technologies", ACM Computing Surveys, Vol. 36 (4), pp. 335-371

ANON, 2006, "NTL, BitTorrent and CacheLogic announce joint technology trial", [online], available at: http://www.bittorrent.com/pressreleases/2006/02/09/ntl-bittorrent-and-cachelogic-announce-joint-technology-trial/, [Accessed 6th October 2009]

ANON, 2007, "Thai ISP Using Oversi Grid to Manage P2P Growth", [online], available at: http://www.gridtoday.com/grid/1701050.html, [Accessed 6th October 2009]

AIM, AOL Instant Messenger, [online], available at http://dashboard.aim.com/aim, [Accessed 6th October 2009]

ARLITT, M.F., CHERKASOVA, L., DILLEY, J., FRIEDRICH, R. J., JIN, T.Y., 2000, "Evaluating content management techniques for Web proxy caches", in proceedings of the Workshop on Internet Service Performance (WISP '99)

ARNOLD, K., GOSLING, J., 1996, "The Java programming language", Addison-Wesley, ISBN 0201634554

AZZUNA, N.B., GUILLEMIN, F., 2004, "Experimental analysis of the impact of peer-to-peer applications on traffic in commercial IP networks", European Transactions on Telecommunications: Special Issue on P2P Networking and P2P Services

BARABASI, A., ALBERT, R., 1999, "Emergence of scaling in random networks", Science 286, p.509

BARABASI, A., BONABEAU, E., 2003, "Scale-free networks", Scientific American 288, pp. 50-58

BBC iPlayer, [online], available at www.bbc.co.uk/iplayer/, [Accessed 6th October 2009]

BERGMAN, M.K., 2001, “The deep web: surfacing hidden value”, The journal of electronic publishing, vol. 7 (1), [online], available at http://www.press.umich.edu/jep/07-01/bergman.html

BERNERS-LEE, T., BRAY, T., CONNOLLY, D., et al, 2004, “Architecture of the World Wide Web, Volume One”, version 20041215, W3C consortium

BERNERS-LEE, T., HENDLER, J., LASSILA, O., 2001, “The semantic web”, Scientific American, May 2001

BERNSTEIN, P., GIUNCHIGLIA, F., KEMENTSIETSIDIS, I., et al, 2002, “Data management for peer-to-peer computing: A vision”, in proceedings of the Workshop on the Web and Databases (WebDB’02)

BHAGWAN, R.., SAVAGE, S., VOELKER, G. M., 2003, “Understanding availability”, in proceedings of 2nd International Workshop on Peer-to-Peer Systems (IPTPS’03)

BHARGAVA, A., KOTHAPALLI, K., RILEY, C., SCHEIDELER, C., THOBER, M., 2004, “Pagoda: a dynamic overlay network for routing, data management and multicasting”, in proceedings of the ACM Symposium on Parallelism in Algorithms and Architectures (SPAA ’04)

BitTorrent, [online], available at http://www.bittorrent.com, [Accessed 6th October 2009]

BOLLOBAS, B., 2001, “Random graphs”, 2nd ed., Cambridge University Press, ISBN 0521797225

BOLLOBAS, B., 2002, “Modern graph theory”, Springer Verlag, ISBN 0387984887

BRAY, T., et al, 2004, “Extensible markup language (XML) 1.0 3rd edition specification”, W3C, [online], available at http://www.w3.org/TR/2004/REC-xml-20040204/

BRESLAU, L., CAO, P., FAN, L., PHILLIPS, G., SHENKER, S., 1999, “Web caching and Zipf-like distributions: Evidence and implications”, in Proceedings of IEEE INFOCOM 1999

BRISCOE, B., ODLYZKO, A., TILLY, B., 2006, “Metcalfe’s law is wrong”, IEEE Spectrum, July 2006, pp. 26-31

BUSTAMANTE, F., QIAO, Y., 2003, “Friendships that last: peer lifespan and its role in p2p protocols”, in proceedings of International Workshop on Web Content Caching and Distribution (IWCW ’03)

BYERS, J., CONSIDINE, J., MITZENMACHER, M., 2003, “Simple Load Balancing for Distributed Hash Tables”, in proceedings of 2nd International Workshop on Peer-to-Peer Systems (IPTPS’03)

BYUN, J., CHATTERJEE, S., 2004, “A Strategic Pricing for Quality of Service (QoS) Network Business”, in proceedings of the 10th Americas Conference on Information Systems (AMCIS ’04)

Cachelogic, Cachelogic Inc, [online], http://www.cachelogic.com, [Accessed 6th October 2009]

CALVERT, K., DOAR, M., ZEGURA, E.W., 1997, “Modelling Internet topology”, in IEEE Communications magazine, Vol. 35 (6), pp. 160-163

CAO, P., IRANI, S., 1997, "Cost-aware WWW proxy caching algorithms," in proceedings of USENIX Symposium on Internet Technologies and Systems (USITS ’97), pp. 193-206

CASTRO, M., DRUSCHEL, P., KERMARREE, A.M., ROWSTRON, A., 2002, “Scribe: A large-scale and decentralized application-level multicast infrastructure”, IEEE Journals Selected Areas Communication, Vol. 20 (8)

CASTRO, M., DRUSCHEL, P., HU, Y.C., ROWSTRON, A., 2002, “Exploiting network proximity in peer-to-peer overlay networks”, in proceedings of the International Workshop on Future Directions in Distributed Computing (FuDiCo’02)

CASTRO, M., DRUSCHEL, P., HU, Y.C., ROWSTRON, A., 2002, "Topology aware routing in structured peer-to-peer overlay networks," Microsoft Research, Technical Report, MSR-TR-2002-82

CHAWATHE, Y., RATNASAMY, S., et al, 2003, “Making Gnutella-like systems scalable”, in proceedings of ACM SIGCOMM ’03

CHEN, Q., CHANG, H., GOVINDAN, R., JAMIN, S., SHENKER, S., WILLINGER, W., 2002, “The Origin of Power Laws in Internet Topologies Revisited”, in proceedings of IEEE INFOCOM 2002

GHIRITA, P.A., IDREOS, S., KOUBARAKIS, M., NEJDL, W., 2004, “Publish/subscribe for RDF-based p2p networks”, in proceedings of the 1st European Semantic Web Symposium (ESWS ’04)

CHO, K., FUKUDA, K., ESAKI, H., KATO, A., 2006, “The impact and implications of the growth in residential user-to-user traffic”, in proceedings of ACM SIGCOMM ‘06

CHOFFNES, D.R., BUSTAMANTE, F.E., 2008, “Taming the torrent: A practical approach to reducing cross-ISP traffic in Peer-to-Peer systems”, in proceedings of ACM SIGCOMM ‘08

CHOLVI, V., FELBER, P., BIERSACK, E., 2004, “Efficient search in unstructured peer-to-peer networks”, in proceedings of the 16th annual ACM symposium on Parallelism in algorithms and architectures (SPAA ’04)

CHU, J., LABONTE, K., LEVINE, B.N., 2002, “Availability and locality measurements of peer-to-peer file systems”, in proceedings of Scalability and Traffic Control in IP Networks II Conference (SPIE ITCom ’02)

CISCO, 2005, “ISP network design”, Cisco systems, [online], available at: ws.edu.isoc.org/data/2005/41363436042fc2b563533b/d1-6up.pdf, [Accessed 6th October 2009]

CLARKE, I., SANDBERG, O., WILEY, B., 2000, “Freenet: A distributed anonymous information storage and retrieval system”, in proceedings of the Workshop on Design Issues in Anonymity and Unobservability 2000

CLAY, K., MILLER, G., THOMPSON, K., 1998, “The nature of the beast: recent traffic measurements from an Internet backbone”, in proceedings of INET conference (INET ’98)

COHEN, E., SHENKER, S., 2002, “Replication strategies in unstructured peer-to-peer networks”, ACM SIGCOMM Computer Communication Review, Vol. 32 (4)

CONTI, J.P., 2007, “Couldn’t care net – Internet Regulation”, IET Communications Engineer, Vol. 5 (3), pp. 14-19

CRONIN, E., JAMIN, S., JIN, C., KURC, A.R., RAZ, D., SHAVITT, Y., 2002, “Constrained mirror placement on the Internet”, IEEE Journal on Selected Areas in Communications, Vol. 20 (7), pp. 1369-1382

CROWCROFT, J., HAND, S., MORTIER, R., ROSCOE, T., WARFIELD, A., 2003, “QoS's downfall: at the bottom, or not at all!”, in proceedings of the ACM SIGCOMM Workshop on Revisiting IP Qos (RIPQoS '03), pp. 109-114

DABEK, F., COX, R., KAASHOEK, F., MORRIS, R., 2004, “Vivaldi: a decentralized network coordinate system”, in proceedings of ACM SIGCOMM ’04

DE MEER, H., TUTSCHKU, K., TRAN-GIA, P., 2003, “Dynamic Operation in Peer-to-Peer Overlay Networks’’, Praxis der Informationsverarbeitung und Kommunikation, Special Issue on Peer-to-Peer Systems

DRUSCHEL, P., ROWSTRON, A., 2001, “Past: A large-scale, persistent peer-to-peer storage utility” in proceedings of the Eighth Workshop on Hot Topics in Operating Systems (HotOS VIII)

DUNN, R.J., 2002, “The Effectiveness of Caching on a Peer-to-Peer Workload”, Masters Thesis, University of Washington

EASTLAKE, D., JONES, P., 2001, “US secure hash algorithm 1”, RFC 3174, IETF, [online], available at http://www.ietf.org/rfc/rfc3174.txt, [Accessed 6th October 2009]

ECONOMIDES, N., 1996, “The economics of networks”, International Journal of Industrial Organization 14, pp. 673-699

EDELSTEIN, H., 1994, “Unraveling Client/Server architecture”, DBMS, Vol. 7 (5), pp. 34

eDonkey, eDonkey2000 article, [online], available at http://en.wikipedia.org/wiki/EDonkey_network, [Accessed 6th October 2009 –original source was removed due to legal action]

ELLSON, J., GANSNER, E.R., KOUTSOFIOS, E., NORTH, S.C., WOODHULL, G., 2003, “Graphviz and Dynagraph – static and dynamic graph drawing tools”, in Graph Drawing Software, Jünger M., Mutzel P. (Eds.), Springer

ERDŐS, P., RÉNYI, A., 1959, “Publ. Math”, Vol. 6, pp. 290-297

European Union, 2000, European Union Directive 2000/31/EC on electronic commerce, [online], available at: http://ec.europa.eu/internal_market/e-commerce/directive_en.htm, [Accessed 6th October 2009]

EVANS, T.S., 2004, “Complex networks”, Contemporary physics, Vol. 45 (6), pp. 455-474

EVANS-PUGHE, C., 2009, "Traffic cops and bottlenecks", IET Engineering & Technology, Vol. 4 (1), pp. 78-81

evolution@home, [online], available at: http://evolutionary-research.net/, [Accessed 6th October 2009]

FALL, K., VARADHAN, K., eds, 2006, “The ns manual (formerly ns Notes and Documentation)”, The VINT project, [online], available at http://www.isi.edu/nsnam/ns/doc/ns_doc.pdf , [Accessed 6th October 2009]

FALOUTSOS, M., FALOUTSOS, P., FALOUTSOS, C., 1999, “On power-law relationships of the Internet topology“, in proceedings of ACM SIGCOMM ’99

FERGUSON, T., 2007, “Broadband adoption passes halfway mark in U.S.”, [online], available at http://news.cnet.com/2110-1034_3-6160422.html, [Accessed 6th October 2009]

FESSANT, F.L, HANDURUKANDE, S., KERMARREC, A.M., MASSOULIE, L., 2004, “Clustering in peer-to-peer file sharing workloads”, in proceedings of the 3rd International Workshop on Peer-to-Peer Systems (IPTPS’04)

FIAT, A., SAIA, D., 2002, “Censorship resistant peer-to-peer content addressable networks”, in proceedings of 13th ACM SIAM symposium on Discrete Algorithms (SODA ’02)

FIGUEIREDO, D., SHAPIRO, J., TOWSLEY, D., 2004, “A public good model of availability in peer to peer systems”, MSU Technical Report, MSU-CSE-04-27, [online], available at: http://www.cse.msu.edu/~jshapiro/papers/tr-04-27.pdf

flickr, flickr photo sharing, [online], available at: http://www.flickr.com, [Accessed 6th October 2009]

FLOYD, S., PAXSON, V., 2001, “Difficulties in Simulating the Internet”, IEEE/ACM Transactions on Networking, Vol. 9 (4), pp. 392-403

FORTZ, B., THORUP, M., 2000, “Internet Traffic Engineering by Optimizing OSPF Weights”, in proceedings of IEEE INFOCOM 2000

FOSTER, I., 2002, “The Grid: A New Infrastructure for 21st Century Science”, Physics Today, Vol. 55 (2), pp. 42-47

FOSTER, I., IAMNITCHI, A., 2003, "On death, taxes and the convergence of Peer-to-Peer and grid computing", in proceedings of the 2nd International Workshop on Peer-to-Peer Systems (IPTPS'03)

FRY, M., GHOSH, A., 1999, "Application level active networking", Computer Networks, Vol. 31 (7), pp. 655-667

GANESAN, P., SUN, Q., GARCIA-MOLINA, H., 2004, "Apocrypha: making P2P overlays network-aware", technical report, [online], available at http://newdbpubs.stanford.edu/pub/2003-70, [Accessed 6th October 2009]

GANESAN, P., GUMMADI, K., GARCIA-MOLINA, H., 2004, "Canon in G-major: designing DHTs with hierarchical structure", in proceedings of the 24th International Conference on Distributed Computing Systems (ICDCS '04)

GANSNER, E.R., 2002, "The DOT language", [online], available at: www.research.att.com/~erg/graphviz/info/lang.html/, [Accessed 6th October 2009]

GAO, L., 2001, "On inferring autonomous system relationships in the Internet", IEEE/ACM Transactions on Networking, Vol. 9(6), pp.733-745

GARCES-ERICE, L., BIERSACK, E., et al, 2003, "Hierarchical peer-to-peer systems", in proceedings of 9th European Conference on Parallel Processing (Euro-Par '03)

GDF, GDF - Gnutella Developer Forum, "The Gnutella protocol v0.6", [online], available at http://groups.yahoo.com/group/the_gdf/, [Accessed 6th October 2009]

GHOSH, A., FRY, M., CROWCROFT, J., 2000, "An architecture for application layer routing", Active Networks, Springer, LNCS 1942, pp. 71-86

GHOSH, A., FRY, M., MACLARTY, G., 2001, "An infrastructure for application level active networking", Computer Networks, Vol. 36 (1), pp. 5-20

gnut, gnut POSIX Gnutella client, [online], available at http://schnarff.com/gnutelladev/source/gnut/gnut-0.4.21/, [Accessed 6th October 2009]

Gnutella, Gnutella P2P network, [online], available at http://en.wikipedia.org/wiki/Gnutella, [Accessed 6th October 2009 – lack of original source]

GODFREY, B., LAKSHMINARAYAN, K., SURANA, S., KARP, R., STOICA, I., 2004, "Load Balancing in Dynamic Structured P2P Systems" in proceedings of IEEE INFOCOM '04

GOLDMAN, A., 2004, "Company releases real world data on file sharing", [online], available at http://www.isp-planet.com/research/2004/cachelogic_data.html, [Accessed 6th October 2009]

Groove, Groove networks, [online], available at http://www.groove.net, [Accessed 6th October 2009]

GSL, Gnu Scientific Library, [online], available at http://www.gnu.org/software/gsl/, [Accessed 6th October 2009]

GUMMADI, K., GUMMADI, R., GRIBBLE, S., et al, 2003, "The impact of DHT routing geometry on resilience and proximity", in proceedings of ACM SIGCOMM '03

GUMMADI, K., DUNN, R., SAROIU, S, GRIBBLE, S., LEVY, H., ZAHORJAN, J., 2003, "Measurement, modeling, and analysis of a peer-to-peer file-sharing workload", in proceedings of the 19th ACM Symposium on Operating Systems Principles (SOSP '03)

GUO, L., MATTA, I., 2001, "The war between mice and elephants", proceedings of 9th International Conference on Network Protocols (ICNP '01)

GWebCache, Gnutella Web Caching System – GWebCache, [online], available at http://www.gnucleus.com/gwebcache/, [Accessed 6th October 2009]

HALEVY, A., IVES, Z., MORK, P., TATARINOV, I. 2003, “Piazza: Data management Infrastructure for semantic web applications”, in proceedings of the 12th International Conference on World Wide Web (WWW ’03)

HANDLEY, M., 2005, “So you think you want to simulate a network?”, presentation at the NGN ns2 workshop - Microsoft Research Cambridge Dec. 2005, [online], available at http://www.informatics.sussex.ac.uk/ngn/slides/ns205talks/mark-ns-tutorial.pdf

HARDIN, G., 1968, "The Tragedy of the Commons", Science, Vol. 162, No. 3859, pp. 1243-1248

HARREN, M., HELLERSTEIN, J., et al, 2002, “Complex queries in DHT-based peer-to-peer networks”, in proceedings of 1st International Workshop on Peer-to-Peer Systems (IPTPS ’02)

HEWLETT-PACKARD, 1999, “A primer on policy-based network management”, Hewlett-Packard Co., [online], available at http://www.openview.hp.com/Uploads/primer_on_policy-based_network_mgmt.pdf, [Accessed 6th October 2009]

HILL, M. D., 1990, “What is scalability?”, ACM SIGARCH Computer Architecture News, Vol. 18 (4), pp. 18-21

HUEBSCH, R., HELLERSTEIN, J., LANHAM, N., THAU LOO, B., 2003, “Querying the internet with PIER”, in proceedings of the 29th International Conference on Very Large Data Bases (VLDB ‘ 03)

HUBERMAN, B.A., ADAMIC, L.A., 1999, Nature, 401, p. 131

HUFFAKER, B., NEMETH, E., CLAFFY, K., 1999, “Otter: A general-purpose network visualization tool”, in proceedings of INET conference (INET ’99)

IANNACCONE, G., CHUAH, C.N., BHATTACHARYYA, S., DIOT, C., 2004, “Feasibility of IP restoration in a Tier-1 backbone”, IEEE Network, Vol. 18 (2), pp. 13-19

ICQ, [online], available at http://www.icq.com, [Accessed 6th October 2009]

ILIE, D., ERMAN, D., POPESCU, A. NILSSON, A, 2004, "Measurement and Analysis of Gnutella Signaling Traffic", in proceedings of IPSI Internet Conference (IPSI ’04)

ITU-T, 2005, “H.264: Advanced video coding for generic audiovisual services”, International Telecommunication Union, [online], available at: http://www.itu.int/rec/T-REC-H.264-200503-I/en, [Accessed 6th October 2009]

IPOQUE, 2007, “Internet study 2007”, [online], available at: http://www.ipoque.com/resources/internet-studies/internet-study-2007, [Accessed 6th October 2009]

IZAL, M., URVOY-KELLER, G., BIERSACK, E.W., FELBER, P.A., AL HAMRA, A., GARCES-ERICE, L., 2004, “Dissecting Bittorrent: 5 months in a torrent’s lifetime”, in proceedings of the 5th Passive and Active Measurements Workshop (PAM ’04)

Jabber, [online], available at: http://www.jabber.org/, [Accessed 6th October 2009]

JANAKIRAMAN, R., WALDVOGEL, M., ZHANG, Q., 2003, “Indra: A peer-to-peer approach to network intrusion detection and prevention”, in proceedings of 2003 IEEE WET ICE Workshop on Enterprise Security

JOVANOVIC, M., ANNEXSTEIN, F., BERMAN, K., 2001, “Modelling peer-to-peer network topologies through ‘small-world’ models and power laws”, in proceedings of IX Telecommunications Forum TELFOR ‘01

KARAGIANNIS, T., BROIDO, A., BROWNLEE, N., CLAFFY, K.C., FALOUTSOS, M., 2004, "Is P2P dying or just hiding?" in proceedings of IEEE Global Communications Conference (Globecom ‘04)

KARAGIANNIS, T., RODRIGUEZ, P., PAPAGIANNAKI, K., 2005, "Should Internet Service Providers Fear Peer-Assisted Content Distribution", in proceedings of Internet Measurement Conference 2005 (IMC '05)

KARBHARI, P., AMMAR, M., DHAMDHERE, A., RAJ, H., RILEY, G., ZEGURA, E., 2004, "Bootstrapping in Gnutella: A measurement study", in proceedings of 5th Passive and Active Network Measurement workshop (PAM '040

Kazaa, [online], available at http://www.kazaa.com, [Accessed 6th October 2009]

KERALAPURA, R., TAFT, N., CHUAH, C.N., IANNACONNE, G., 2004, "Can ISP's Take the Heat from Overlay Networks?", in proceedings of 2nd ACM Hot Topics in Networks Workshop (HotNets '04)

KEROMYTIS, A., RUBENSTEIN, D., 2002, "SOS: Secure overlay services", in proceedings of the ACM SIGCOMM '02

KERNIGHAN, B.W., RITCHIE, D.M., 1988, "The C programming language", 2nd ed., Prentice-Hall, ISBN 0131103628

KILKKI, K., 2008, "Quality of Experience in Communications Ecosystem", Journal of Universal Computer Science, Vol. 14 (5), pp. 615-624

KLEINBERG, J., 2000, "The small-world phenomenon: an algorithmic perspective", in proceedings 32nd ACM Symposium on Theory of Computing (STOC '00), pp. 163–170

KLEIS, M., LUA, E.K., ZHOU, X., 2005, "Hierarchical Peer-to-Peer networks using lightweight superpeer topologies", in proceedings of 10th IEEE Symposium on Computers and Communications (ISCC '05)

KLEMM, A., LINDEMANN, C., VERNON, M.K., WALDHORST, O.P., 2004, "Characterizing the query behavior in peer-to-peer file sharing systems", in proceedings of the 4th ACM SIGCOMM conference on Internet Measurement (IMC '04)

KNUTH, D.E., 1981, "The art of computer programming", 2nd ed., vol. 2, Addison-Wesley, ISBN 0201038226

KOULOURIS, T., HENJES, R., TUTSCHKU, K., DE MEER, H., 2003, "Implementation of Adaptive Control for P2P Overlays", in proceedings of 5th International Workshop on Active Networks (IWAN '03), Springer, Volume 2982/2004, pp. 292-304

KRUSE, R.L., RYBA, A.J., 1998, "Data structures and program design in C++", Prentice-Hall, ISBN 0137689950

KUBIATOWICZ, J., BINDEL, D., CHEN, Y., EATON, P., et al, 2000, "Oceanstore: An architecture for global-scale persistent storage", in proceedings of the 9th ACM International conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS '00)

LARSON, S., SNOW, C., PANDE, V., 2003, "Folding@Home and Genome@Home: Using distributed computing to tackle previously intractable problems in computational biology", In "Modern Methods in Computational Biology", GRANT, R., ed, Horizon Press

LAW, A.M., KELTON, D.M., 2000, "Simulation Modelling and Analysis", 3rd edition, McGraw-Hill, ISBN 0070592926

LAW, C., SIU, K.Y., 2003, "Distributed Construction of Random Expander Graphs", proceedings of IEEE INFOCOM 2003

LEGOUT, A., URVOY-KELLER, G., MICHIARDI, P., 2005, "Understanding Bittorrent: an experimental perspective", Technical report, INRIA-EURECOM

LEIBOWITZ, N., BERGMAN, A., BEN-SHAUL, R., SHAVIT, A., 2002, "Are File Swapping Networks Cacheable? Characterizing P2P Traffic", in proceedings of 7th Web Content Caching and Distribution Workshop (WCW '02)

LELAND, W.E., TAQQU, M.S., WILLINGER, W., WILSON, D.V., 1994, "On the self-similar nature of ethernet traffic", IEEE/ACM Transactions on Networking, vol. 2 (1), pp. 1-15

LEONARD, D., RAI, V., LOGUINOV, D., 2005, "On lifetime-based node failure and stochastic resilience of decentralized peer-to-peer networks", in ACM SIGMETRICS Performance Evaluation Review, Vol. 33 (1), p. 26-37

LI, L., ALDERSON, D., WILLINGER, W., DOYLE, J., 2004, "A First-Principles Approach to Understanding the Internet's Router-level Topology", in proceedings of ACM SIGCOMM '04

LIANG, J., KUMAR, R., ROSS, K., 2004, "The Kazaa overlay: a measurement study", in proceedings of the 19th IEEE Annual Computer Communications Workshop (CCW '04)

LIBEN-NOWELL, D., BALAKRISHNAN, H., KARGER, D., 2002, "Analysis of the Evolution of Peer-to-Peer Systems", in proceedings of 21st Symposium on Principles of Distributed Computing (PODC '02)

LIOGKAS, N., NELSON, R., KOHLER, E., ZHANG, L., 2006, "Exploiting BitTorrent for fun (but not profit)", in proceedings of 5th International Workshop on Peer-to-Peer Systems (IPTPS '06)

LOGUINOV, A., KUMAR, A., RAI, V., GANESH, S., 2003, "Graph-theoretic analysis of structured peer-to-peer systems: routing distances and fault resilience", in proceedings of ACM SIGCOMM '03

LOO, B.T., HUEBSCH, R., STOICA, I., HELLERSTEIN, J., 2004, "The case for a hybrid P2P search infrastructure", in proceedings of 3rd International Workshop on Peer-to-Peer Systems (IPTPS '04)

LUA, E.K., CROWCROFT, J., PIAS, M., 2004, "Highways: Proximity clustering for scalable peer-to-peer networks", in proceedings of 4th IEEE International Conference on P2P Computing (P2P '04)

LUA, E.K., CROWCROFT, J., PIAS, M., SHARMA, R., LIM, S., 2005, "A survey and comparison of Peer-to-Peer overlay network schemes", IEEE Communications Surveys and Tutorials, Vol. 7 (2), pp.72-93

LÜTHI, J., 1994, "Distributed discrete event simulation: optimistic protocols with probabilistic time window adaptation", Diploma Thesis (summary), University of Vienna

LV, Q., RATNASAMY, S., SHENKER, S., 2000, "Can Heterogeneity Make Gnutella Scalable?", proceedings of 1st International Workshop on Peer-to-Peer Systems (IPTPS '02)

LV, Q., CAO, P., COHEN, E., LI, K., SHENKER, S., 2002, "Search and Replication in Unstructured Peer-to-Peer Networks", in proceedings of 16th ACM International Conference on Supercomputing (ICS '02)

MALKHI, D., NAOR, M., RATAJCZAK, D., 2002 "Viceroy: A scalable and dynamic emulation of the butterfly", in proceedings of the 21st Symposium on Principles of Distributed Computing (PODC '02)

MARK, R., 2004, "Workers find IM a mixed blessing", [online], available at http://www.internetnews.com/stats/article.php/3403221, [Accessed 6th October 2009]

MAYMOUNKOV, P., MAZIERES, D., 2002, "Kademlia: a peer-to-peer information system based on the XOR metric", in proceedings of 1st International Workshop on Peer-to-Peer Systems (IPTPS '02)

MEDINA, A., LAKHINA, A., MATTA, I., BYERS, J., 2001, "BRITE: Universal topology generation from a user's perspective", [online], available at http://www.cs.bu.edu/brite/publications/usermanual.pdf, [Accessed 6th October 2009]

MEDINA, A., LAKHINA, A., MATTA, I., BYERS, J., 2001, "BRITE: An approach to universal topology generation", in proceedings of the International Workshop on Modeling, Analysis and Simulation of Computer and Telecommunication systems (MASCOTS '01)

MENNECKE, T., 2005, "DSL Broadband Providers Perform Balancing Act", [online], available at http://www.slyck.com/news.php?story=973, [Accessed 6th October 2009]

MILGRAM S., 1967, "The small world problem", Psychology Today, May 1967, pp.60-67

MINAR N., 2001, "Distributed systems topologies", O'Reilly network, [online], available at: http://www.openp2p.com/lpt/a/1431, [Accessed 6th October 2009]

MO, J., WALRAND, J., 2000, "Fair end-to-end window-based congestion control", IEEE/ACM Transactions on Networking, Vol. 8 (5), pp. 556-567

MOORE B., et al, 2001, "Policy core information model – version 1 specification", RFC 3060, IETF, [online] available at http://www.ietf.org/rfc/rfc3060.txt, [Accessed 6th October 2009]

MOORE G., 1965, "Cramming more components onto integrated circuits", Electronics magazine 19

MSN, MSN messenger, [online], available at http://get.live.com/messenger/overview, [Accessed 6th October 2009]

NAICKEN, S., BASU, A., LIVINGSTON, B., RODHETBHAI, S., 2006, "A Survey of Peer-to-Peer Network Simulators", in proceedings of 7th Annual Postgraduate Symposium (pgnet 2006)

NAICKEN, S., LIVINGSTON, B., BASU, A., RODHETBHAI, S., WAKEMAN, I., CHALMERS, D., 2007, "The State of Peer-to-Peer Simulators and Simulations", in ACM SIGCOMM Computer Communication Review, Vol. 37 (2), pp. 95-98

NARAINE R., 2004, "Microsoft Hooks into AOC, Yahoo for Business IM", [online], available at http://www.internetnews.com/bus-news/article.php/3381551, [Accessed 6th October 2009]

Narses, Narses network simulator, [online], available at http://sourceforge.net/projects/narses, [Accessed 6th October 2009]

NEJDL, W., WOLF, B., QU, C., DECKER, S., SINTEK, M., et al, 2003, "Edutella: A p2p networking infrastructure based on RDF", in proceedings of the 12th International Conference on World Wide Web (WWW '03)

netflow, Internet2 Netflow weekly reports, [online], available at http://netflow.internet2.edu, [Accessed 6th October 2009]

NEWMAN M.E.J., STROGATZ S.H., WATTS D.J., 2001, "Random graphs with arbitrary degree distributions and their applications", Phys. Rev. E 64, 026118

NG, T., CHU, Y., RAO, S., SRIPANIDKULCHAI, K., ZHANG, H., 2003, "Measurement-based optimization techniques for bandwidth-demanding peer-to-peer systems", in proceedings of IEEE INFOCOM '03

NORTON, W.B., 2003, "The evolution of the US Internet peering ecosystem", [online], available at: http://www.nanog.org/mtg-0405/pdf/norton.pdf, [Accessed 6th October 2009]

ns2, "The network simulator – ns-2", 2007, [online], available at http://www.isi.edu/nsnam/ns/, [Accessed 6th October 2009]

OBRACZKA, K., SILVA, F., 2000, "Network latency metrics for server proximity", in proceedings of IEEE Globecom 2000

ODLYZKO, A., 2001, "Content is not king", [online], available at http://www.firstmonday.org/issues/issue6_2/odlyzko/#o11, [Accessed 6th October 2009]

ODLYZKO, A., 2001, “Internet pricing and the history of communications”, Computer Networks, Vol. 36 (5/6), pp. 493-517

OMNet++, [online], available at http://www.omnetpp.org/, [Accessed 6th October 2009]

Online Copyright Infringement Liability Limitation Act: Limitations on liability relating to material online - 17 USC 512, 1998, [online], available at: http://www.bitlaw.com/source/17usc/512.html, [Accessed 6th October 2009]

ORAM, A., ed., 2001, “Peer-to-Peer: harnessing the power of disruptive technologies”, O’ Reilly, ISBN 059600110X

O’ REILLY, T., 2005, “What is Web 2.0 - Design Patterns and Business Models for the Next Generation of Software”, [online], available at http://www.oreillynet.com/pub/a/oreilly/tim/news/2005/09/30/what-is-web-20.html?page=1, [Accessed 6th October 2009]

P2Psim, P2Psim – a simulator for peer-to-peer protocols, [online], available at http://pdos.csail.mit.edu/p2psim, [Accessed 6th October 2009]

PADHYE, J., FIROIU, V., TOWSLEY, D., KRUSOE, J., 1998, “Modeling TCP throughput: A simple model and its empirical validation”, in proceedings of ACM SIGCOMM 1998

PAGE, E.H., NANCE, R.E., 1994, “Parallel discrete event simulation: A modelling methodological perspective”, in proceedings of the 8th Workshop on Parallel and Distributed Simulation (PADS ’94)

PARK, K., PAI, V.S., PETERSON, L., WANG, Z., 2004, “CoDNS: Improving DNS Performance and Reliability via Cooperative Lookups”, in proceedings of the 6th Symposium on Operating Systems Design and Implementation (OSDI '04)

PARK, S.K., MILLER, K.W., 1988, “Communications of the ACM”, vol. 31, pp. 1192-1201

PeerSim, PeerSim P2P simulator, [online], available at http://peersim.sourceforge.net, [Accessed 6th October 2009]

PIETZUCH, P.R, BACON, J., 2003, “Peer-to-peer overlay broker networks in an event-based middleware”, in proceedings of the 2nd international workshop on Distributed Event-Based Systems (DEBS ’03)

PITKOW, J., RECKER, M., 1994, “A simple yet robust caching algorithm based on dynamic access patterns”, in proceedings of the 2nd International World Wide Web Conference (WWW2)

PLAXTON C., RAJARAMAN R., RICHA A., 1997, “Accessing nearby copies of replicated objects in a distributed environment”, in proceedings of 9th Annual ACM Symposium on Parallel Algorithms and Architectures (SPAA ’97)

PODLIPNIG, S., BOSZORMENYI, L., 2003, “A survey of web cache replacement strategies”, ACM Computing Surveys, Vol. 35 (4), pp. 347–398

PORTMANN, M., SOOKAVATANA, P., ARDON, S., SENEVIRATNE, A., 2001, “The cost of peer discovery and searching in the Gnutella peer-to-peer file sharing protocol”, in proceedings of the 9th IEEE International Conference on Networks (ICON 2001)

POSTEL, J., ed., 1981, “Transmission control protocol”, RFC793, IETF, [online], available at http://www.ietf.org/rfc/rfc793.txt, [Accessed 6th October 2009]

POSTEL, J., ed., 1981, “Internet Protocol”, RFC791, IETF, [online], available at http://tools.ietf.org/html/791, [Accessed 6th October 2009]

PRESS, W.H., FLANNERY, B.P., TEUKOLSKY, S.A., VETTERLING, W.T., 1992, “Numerical recipes in C: The art of scientific computing”, 2nd ed., Cambridge University Press, ISBN 0521431085

QIAO, Y., LU, D., BUSTAMANTE, F., DINDA, P., 2004, “Looking at the server side of peer-to-peer systems”, Technical Report NWU-CS-04-37, Department of Computer Science, Northwestern University, March 2004

QIAO, Y., BUSTAMANTE, F.E., 2006, “Structured and unstructured overlays under the microscope - a measurement-based view of two P2P systems that people use”, in proceedings of 2006 USENIX Technical Conference (USENIX ’06)

QIU, L., YANG, Y.R., ZHANG, Y., SHENKER, S., “On selfish routing in Internet-like environments”, in proceedings of ACM SIGCOMM 2003

RABINOVICH, M., SPATSCHAK, O., 2001, “Web caching and replication”, Addison-Wesley, ISBN 0201615703

RAGHAVAN, S., GARCIA-MOLINA, H., 2001, "Crawling the Hidden Web", in proceedings of the 27th International Conference on Very Large Data Bases (VLDB ’01), pp. 129-138

RAMASUBRAMANIAN, V., SIRER, E.G., 2004, “Beehive: O(1) Lookup Performance for Power-Law Query Distributions in Peer-to-Peer Overlays”, in proceedings of Networked System Design and Implementation 2004 (NSDI’04)

ranlib, Library of C routines for random number generation, [online], available at: http://orion.math.iastate.edu/burkardt/c_src/ranlib/, [Accessed 6th October 2009]

RATNASAMY, S., FRANCIS, P., HANDLEY, M., KARP, R., SCHENKER, S., 2001, “A Scalable Content-Addressable Network”, in proceedings of ACM SIGCOMM 2001

REARDON, M., “Skype hits 100M subscriber mark”, news.com article, [online], available at http://news.com.com/2061-10785_3-6066399.html, [Accessed 6th October 2009]

REED, D., SALTZER, J., CLARK, D., 1998, “Commentaries on active networking and end-to-end arguments,” IEEE Network, Vol. 12 (3), pp. 69-71

REED, D.P., 1999, “That sneaky exponential – Beyond Metcalfe’s law to the power of community building”, [online], available at http://www.reed.com/Papers/GFN/reedslaw.html, [Accessed 6th October 2009]

REKHTER, Y., LI, T., HARES, S., 2006, "A Border Gateway Protocol 4 (BGP-4)", RFC 4271, IETF, [online], available at: http://www.ietf.org/rfc/rfc4271.txt, [Accessed 6th October 2009]

RHEA, S., GEELS, D., ROSCOE, T., KUBIATOWICZ, J., 2004, “Handling Churn in a DHT”, in proceedings of USENIX 2004

RIPEANU, M., FOSTER, I., IAMNITCHI, A., 2002, “Mapping the Gnutella network: properties of large scale peer-to-peer systems and implications for system design”, IEEE Internet Computing Journal, Vol. 6

RITTER, J., 2000, “Why Gnutella can’t scale. No, really”, [online], available at http://www.tch.org/gnutella.html, [Accessed 6th October 2009]

ROWSTRON, A., DRUSCHEL, P., 2001, “Pastry: scalable, decentralized object location and routing for large-scale peer-to-peer systems”, in proceedings of IFIP/ACM International Conference on Distributed Systems Platforms (ICDSP ’01)

ROWSTRON, A., KERMARREC, A.M., CASTRO, M., DRUSCHEL, P., 2001, “SCRIBE: The design of a large-scale event notification infrastructure”, in proceedings of 3rd Workshop on Networked Group Communication (NGC 2001), pp. 30-43

SAFFRE, F., GHANEA-HERCOCK, R., 2003, “Beyond anarchy: self-organized topology for peer-to-peer networks”, Complexity, Vol. 9 (2), pp. 49-53

SALEH, O., HEFEEDA, M., 2006, “Modeling and Caching of Peer-to-Peer Traffic”, in proceedings of the 14th IEEE International Conference on Network Protocols (ICNP ’06)

SALTZER, J.H., REED, D.P., CLARK, D.D., 1984, “End-to-end arguments in system design,” ACM Transactions on Computer Systems, Vol. 2 (4), pp. 277–288

Sandvine, Sandvine Inc., [online], available at: www.sandvine.com, [Accessed 6th October 2009]

SANDVINE, 2002, “P2P file sharing: the impact of file sharing on service provider networks”, Sandvine Inc., [online], available at http://www.sandvine.com/, [Accessed 6th October 2009]

SANDVINE, 2003, “P2P file sharing: port hopping and challenges to traffic control methodology”, Sandvine Inc., [online], available at http://www.sandvine.com/, [Accessed 6th October 2009]

SANDVINE, 2004, “Meeting the challenge of today’s evasive P2P traffic”, Sandvine Inc., [online], available at http://www.sandvine.com/, [Accessed 6th October 2009]

SAROIU, S., GUMMADI, K., GRIBBLE, S., 2002, “A Measurement study of peer-to-peer file sharing systems”, proceedings of 9th Multimedia Computing and Networking conference (MMCN ’02)

SAROIU, S., GUMMADI, K.P., GRIBBLE, S.D., 2003, “Measuring and analyzing the characteristics of Napster and Gnutella hosts”, Multimedia Systems 9, pp.170-184

SCHLOSSER, M., SINTEK, M., DECKER, S., NEJDL, W., 2001, “HyperCup – hypercubes, ontologies and efficient search on P2P networks”, in proceedings of 1st International Workshop on Agents and Peer-To-Peer Computing (AP2PC 2001)

SCHLOSSER, M., SINTEK, M., DECKER, S., NEJDL, W., 2002, “HyperCuP – Shaping Up Peer-to-Peer Networks”, draft, available from http://www-db.stanford.edu/~schloss/ hypercup, [Accessed 6th October 2009]

SCHODER, D., FISCHBACH, K., 2003, “Peer-to-Peer prospects”, Communications of the ACM, Vol. 46 (2), pp. 27-29

SEN, S., WANG, J., 2002, “Analyzing Peer-to-Peer traffic across large networks”, in proceedings of 2002 ACM SIGCOMM Internet Measurement Workshop (IMC ’02)

SHAPIRO, C., VARIAN, H.R., 1999, “Information Rules”, Harvard Business Press, ISBN 087584863X

SHI, S., YANG, G., WANG, D., YU, J., QU, S., CHEN, M., 2004, "Making Peer-to-Peer keyword searching feasible using multi-level partitioning", in proceedings of 3rd International Workshop on Peer-to-Peer Systems (IPTPS '04)

SHIRKY, C., 2000, “What is P2P and what isn’t”, O’ Reilly, [online], available at http://www.openp2p.com/pub/a/p2p/2000/11/24/shirky1-whatisp2p.html, [Accessed 6th October 2009]

SHIRKY, C., TRUELOVE K., et al, 2001, “2001 P2P Networking Overview: The emergent P2P platform of presence, identity and edge resources”, O’Reilly, ISBN 0596001851

SHUDO, K., 2006, “Overlay Weaver”, [online], available at http://overlayweaver.sourceforge.net, [Accessed 6th October 2009]

SIMEONOV, S., 2006, “Metcalfe’s Law: more misunderstood than wrong?”, Web 2.0 Journal, [online], available at: http://web2.sys-con.com/node/259624, [Accessed 6th October 2009]

SINGLA, A., ROHRS, C., 2002, “Ultrapeers: another step towards Gnutella scalability”, Gnutella developer’s forum [online], available at http://groups.yahoo.com/group/the_gdf/files/Proposals/Ultrapeer/Ultrapeers_1.0_clean.html/, [Accessed 6th October 2009]

SIT, E., MORRIS, R., 2002, “Security Considerations for Peer-to-Peer Distributed Hash Tables” in proceedings of 1st International Workshop on Peer-to-Peer Systems (IPTPS ’02)

Skype, [online], available at http://www.skype.com, [Accessed 6th October 2009]

Sprint, Sprint IP monitoring project data, [online] available at http://ipmon.sprintlabs.com, [Accessed 6th October 2009]

SPRINT, N., MAHAJAN, R., WETHERALL, D., 2002, "Measuring ISP topologies with Rocketfuel", in proceedings of ACM SIGCOMM '02

SRIPANIDKULCHAI, K., MAGGS, B., ZHANG, H., 2003, "Efficient Content Location Using Interest-Based Locality in Peer-to-Peer Systems", in proceedings of IEEE INFOCOM '03

SRIVATSA, M., GEDIK, B., LIU, L., 2004, "Scaling Unstructured Peer-to-Peer Networks With Multi-Tier Capacity-Aware Overlay Topologies", in proceedings of the 10th International Conference on Parallel and Distributed Systems (ICPADS '04)

SSF, "Scalable Simulation Framework (SSF)", [online] available at http://www.ssfnet.org/homePage.html, [Accessed 6th October 2009]

STAHL, I., 1990, "Introduction to simulation with GPSS", Prentice-Hall, ISBN 0134832310

STOICA, I., MORRIS, R. et al, 2001, "Chord: A scalable P2P lookup service for Internet applications", in proceedings of ACM SIGCOMM 2001

STOICA, I., ADKINS, D., ZHUANG, S., SHENKER, S., SURANA, S., 2002, "Internet indirection infrastructure", in proceedings of ACM SIGCOMM 2002

STUTZBACH, D., REJAIE, R., SEN, S., 2005, "Characterizing unstructured overlay topologies in modern P2P file-sharing systems", in proceedings of the 2005 Internet Measurement Conference (IMC '05)

STUTZBACH, D., REJAIE, R., 2006, "Understanding Churn in Peer-to-Peer Networks", in proceedings of the 2006 Internet Measurement Conference (IMC '06)

STUTZBACH, D., ZHAO, S., REJAIE, R., 2007, "Characterizing files in the modern Gnutella network", Multimedia Systems, Vol. 13 (1), pp. 35-50

SULLIVAN, W., WERTHIMER, D., BOYWER, S., COBB, J., GEDYE, D., ANDERSON, D., 1997, "A new major seti project based on project serendip data and 100,000 personal computers", in proceedings of the 5th International Conference on Bioastronomy

TENENHOUSE, D., WETHERALL, D., 1996, "Towards an active network architecture", ACM SIGCOMM Computer Communication Review, Vol 26 (2)

TING, N.S., 2002, "A generic peer-to-peer network simulator", in proceedings of 2002-2003 graduate symposium, University of Saskatchewan, [online], available at http://bistrica.usask.ca/MADMUC/Pubs/ting880.pdf, [Accessed 6th October 2009]

VANRENESSE, R., BIRMAN, K., BOZDOG, A., DIMITRIU, D., SINGH, M., VOGELS, W., 2003, "Heterogeneity-aware peer-to-peer multicast", in proceedings of the 17th International Symposium on Distributed Computing (DISC '03)

VIXIE, P., ed., et al, 1997, "Dynamic updates in the Domain Name System (DNS UPDATE)", RFC2136, IETF, [online], available at http://tools.ietf.org/html/2136, [Accessed 6th October 2009]

VLACHOS, V., ANDROUTSELLIS-THEOTOKIS, S., SPINELLIS, D., 2004, "Security applications of peer-to-peer networks", IEEE Computer Networks, Vol. 45 (2), pp. 195–205

VON LOHMANN, F., 2003, "Peer-to-peer file sharing and copyright law: a primer for developers", in proceedings of 2nd International Workshop on Peer-to-Peer Systems (IPTPS '03)

WANG, X., YAO, Z., LOGUINOV, D., 2007, "Residual-Based Measurement of Peer and Link Lifetimes in Gnutella Networks", in proceedings of IEEE INFOCOM '07

WATERS, G., ed., et al, 1999, "Policy framework architecture", IETF Internet-Draft draft-ietf-policy-arch-00, [online], available at: http://tools.ietf.org/html/draft-ietf-policy-arch-00, [Accessed 6th October 2009]

WATKINS, K., 1993, "Discrete event simulation in C", McGraw-Hill, ISBN 0077077334

WATTS, A., STROGATZ, S., 1998, "Collective dynamics of 'small-world' networks", Nature 393, p. 440

WESSELS, D., 2001, "Web caching", O'Reilly, ISBN 156592536X

WIERZBICKI, A., LEIBOWITZ, N., RIPEANU, M., WOZNIAK, R., 2004, "Cache Replacement Policies Revisited: The Case of P2P Traffic", in proceedings of 4th International Global and P2P Computing Workshop (GP2P '04)

Wikipedia, Wikipedia the free encyclopedia, [online], available at: http://en.wikipedia.org/ , [Accessed 6th October 2009]

WILLINGER, W., PAXSON, V., 1998, "Where mathematics meets the Internet", Notices of the American Mathematical Society, Vol. 45, pp. 961-970

XIE, H., KRISHNAMURTHY, A., SILBERSCHATZ, A., YANG, Y.R., 2007, "P4P: Explicit Communications for Cooperative Control Between P2P and Network Providers.", P4PWG Whitepaper, [online], available at: http://cs-www.cs.yale.edu/homes/yong/publications/p4p.pdf, [Accessed 6th October 2009]

XU, J., 2003, "On the fundamental tradeoffs between routing table size and network diameter in peer-to-peer networks", in proceedings of IEEE INFOCOM'03

yahoo, Yahoo! Messenger, [online], available at http://messenger.yahoo.com/, [Accessed 6th October 2009]

YANG, B., GARCIA-MOLINA, H., 2001, "Comparing hybrid peer-to-peer systems", in proceedings of the 27th International Conference on Very Large Data Bases (VLDB '01)

YANG, A., GARCIA-MOLINA, H., 2002, "Efficient Search in Peer-to-peer Networks", in proceedings of 22nd International Conference on Distributed Computing Systems (ICDCS '02)

YANG, A., GARCIA-MOLINA, H., 2003, "Designing a Super-peer Network", in proceedings of 19th IEEE International Conference on Data Engineering (ICDE '03)

YANG, W., GHAZALEH, N.A., 2005, "GPS: A General Peer-to-Peer Simulator and its Use for Modeling BitTorrent", in proceedings of 13th IEEE International Symposium on Modelling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS '05)

YouTube, [online], available at http://www.youtube.com, [Accessed 6th October 2009]

ZEGURA, E., AMMAR, M., FEI, Z., BHATTACHARJEE, S., 2000, "Application-layer anycasting: A server selection architecture and use in a replicated web service", IEEE/ACM Transactions in Networking, Vol. 8 (4), pp. 455-466

ZHAO, B., KUBIATOWICZ, J., JOSEPH, A., 2001, "Tapestry: An infrastructure for fault-tolerant wide-area location and routing", Technical Report UCB/CSD-01-1141, Computer Science Division, U. C. Berkeley, April 2001

ZHAO, B., DUAN, Y., et al, 2002, "Brocade: landmark routing on overlay networks", proceedings of 1st International Workshop on Peer-to-Peer Systems (IPTPS '02)

APPENDICES

Appendix A: Advanced Chord routing scheme

A scalable key location method exists in Chord [Stoica, 2001], which is more efficient than the simple scheme depicted in Figure 50, but requires the nodes to maintain additional routing information apart from information on their successor.

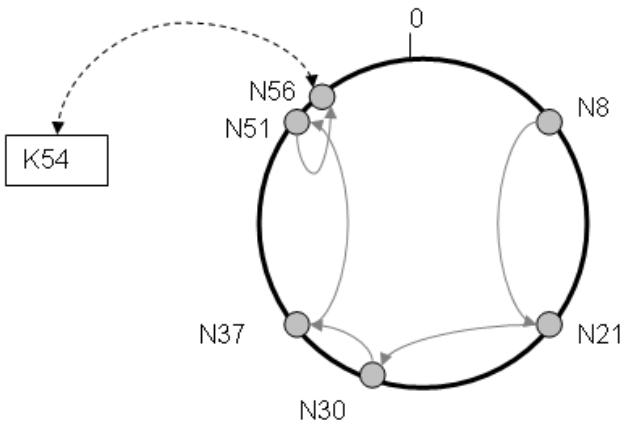

Figure 50: Simple Chord routing scheme.

For faster lookups, each node n maintains a routing table of up to m entries (where m is the length of the Chord identifiers in bits), called the "finger table". The ith entry in the finger table at node n contains the identity of the first node s that succeeds n by at least 2i-1 on the Chord ring. This node s is called the "i^{th} finger" of n. Under this notation, the 1^{st} finger of n is its successor. This is shown in Figure 51, where the finger table of node 8 is presented. As we might expect, the first finger of node 8 is node 14, as node 14 is the first node to succeed $(8 + 2^0) \mathrm{mod} 2^6 = 9$. In a similar manner, the last finger of node 8 is derived from $(8 + 2^5) \mathrm{mod} 2^6 = 40$, and is node 42.

This scheme has two important characteristics: First, each node stores information about a only small number of other nodes and knows more about nodes following close on the identifier ring than about nodes far away. Secondly, a node's finger table does not contain enough information to directly determine the successor of

any arbitrary key. For instance, node 8 in Figure 51 cannot determine the successor of key 28 (node 30) by itself, as node 30 does not appear in its finger table. If, using this scheme, node 8 wishes to locate key 54, it will ask node 42 to resolve the query, since node 42 is its largest finger. Node 42 will in turn check its own finger table for the largest finger that precedes 54, which is node 51. Finally, node 51 will discover its own successor, node 56, and a query reply will be sent to node 8.

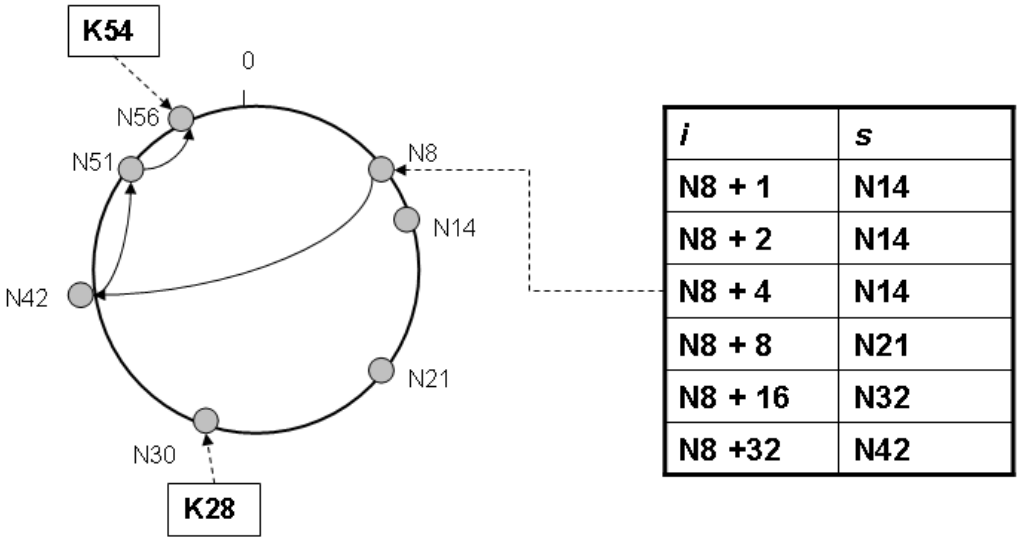

Figure 51: Advanced Chord routing.

Appendix B: AVP Policy XML Schema

```
<?xml version="1.0" encoding="utf-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">

 <xsd:element name="AvpPolicy">
  <xsd:complexType>
   <xsd:sequence>
    <xsd:element name="polId" type="xsd:string" />
    <xsd:element name="polGroupId" type="xsd:string"
     nillable="true" />
    <xsd:element name="polType" type="AvpPolType" />
    <xsd:element name="polDescription" type="xsd:string" />
    <xsd:element name="polPriority" type="PolPriorities" />
    <xsd:element ref="ValidityPeriod" />
    <xsd:element ref="Conditions" />
    <xsd:element ref="Actions" />
   </xsd:sequence>
  </xsd:complexType>
 </xsd:element>

 <xsd:element name="ValidityPeriod">
  <xsd:complexType>
   <xsd:choice>
    <xsd:sequence>
     <xsd:element name="startTime" type="xsd:dateTime" />
     <xsd:element name="endTime" type="xsd:dateTime" />
    </xsd:sequence>
    <xsd:sequence>
     <xsd:element name="endDayOfMnth" type="xsd:gDay" />
     <xsd:element name="endHourOfDay" type="xsd:time" />
    </xsd:sequence>
   </xsd:choice>
  </xsd:complexType>
 </xsd:element>
```

```
<xsd:element name="Conditions">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element name="CondObject" type="CondObject" minOccurs="0"
        maxOccurs="unbounded" />
      <xsd:element ref="CondReq" minOccurs="0" maxOccurs="unbounded" />
      <xsd:element name="EvalParams" type="xsd:string" />
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>

<xsd:complexType name="Event">
  <xsd:sequence>
    <xsd:element name="eventId" type="xsd:string" />
    <xsd:element name="eventType" type="xsd:string" nillable="true" />
    <xsd:element name="eventVariable" type="Variable" minOccurs="1"
      maxOccurs="unbounded" nillable="true" />
  </xsd:sequence>
</xsd:complexType>

<xsd:element name="CondReq">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element name="reqId" type="xsd:string" />
      <xsd:element name="reqType" type="ReqTypes" />
      <xsd:element name="reqObject" type="CondObject" minOccurs="1"
        maxOccurs="unbounded" />
      <xsd:element name="reqParam" type="Param" />
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>


<xsd:complexType name="CondObject">
  <xsd:choice>
```

```
    <xsd:element name="event" type="Event" />
    <xsd:element name="variable" type="Variable" />
  </xsd:choice>
</xsd:complexType>


<xsd:complexType name="Variable">
  <xsd:sequence>
    <xsd:element name="varId" type="xsd:string" />
    <xsd:element name="varType" type="xsd:string" />
    <xsd:element name="varSyntax" type="xsd:string" />
  </xsd:sequence>
</xsd:complexType>

<xsd:element name="Actions">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element name="actionId" type="xsd:string" />
      <xsd:element name="actionType" type="xsd:string" />
      <xsd:element name="actionParam" type="Param" />
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>


<xsd:complexType name="Param">
  <xsd:sequence>
    <xsd:element name="parId" type="xsd:string" />
    <xsd:element name="parSyntax" type="xsd:string" />
    <xsd:element name="parValue" type="xsd:string" />
  </xsd:sequence>
</xsd:complexType>


<xsd:simpleType name="AvpPolType">
  <xsd:restriction base="xsd:string">
```

```
      <xsd:enumeration value="IndvPol" />
      <xsd:enumeration value="SysWdPol" />
    </xsd:restriction>
  </xsd:simpleType>

  <xsd:simpleType name="ReqTypes">
    <xsd:restriction base="xsd:string">
      <xsd:enumeration value="GT" />
      <xsd:enumeration value="LT" />
      <xsd:enumeration value="EQ" />
    </xsd:restriction>
  </xsd:simpleType>

  <xsd:simpleType name="PolPriorities">
    <xsd:restriction base="xsd:string">
      <xsd:enumeration value="besteffort" />
      <xsd:enumeration value="normal" />
      <xsd:enumeration value="critical" />
    </xsd:restriction>
  </xsd:simpleType>

</xsd:schema>
```

Appendix C: AVP Policy example

Starting from a logical IF/THEN statement, let's try to express a simple policy in XML using the format presented in Chapter 4 of the main text:

```
IF (ip_address > 192.168.1.1) AND (ip_address <
192.168.1.128) THEN add_to_CD(3)
```

Such a policy could be used to instruct an AVP to assign any peers whose IP addresses lay in the range between 192.168.1.1 and 192.168.1.128 to the controlled domain with ID "3".

The first part contains the policy name and group identifier, classification (Individual configuration policy), a remark, priority (normal) and validity period:

```
<?xml version="1.0" encoding="UTF-8"?>

<AvpPolicy xmlns:xsi=http://www.w3.org/2001/XMLSchema-
instance
xsi:noNamespaceSchemaLocation="http://www.ee.ucl.ac.uk/~tko
ulour/schemas/AVPSchema1.xsd">

<polId>CDpolicy001</polId>
<polGroupId>testgroup</polGroupId>
<polType>IndvPol</polType>
<polDescription>adds peers to CD3</polDescription>
<polPriority>normal</polPriority>
<ValidityPeriod>
    <startTime>2007-10-04T18:15:00.0Z</startTime>
    <endTime>2007-10-06T18:15:00.0Z</endTime>
</ValidityPeriod>
```

The condition object is an event indicating the arrival of a new peer ("ev1"). Its IP address is referenced by event variable "ipAddr":

```
<Conditions>
    <CondObject>
        <event>
            <eventId>ev1</eventId>
            <eventType>new_peer_arrival</eventType>
            <eventVariable>
                <varId>ipAddr</varId>
                <varType>ev1var</varType>
                <varSyntax>string</varSyntax>
            </eventVariable>
        </event>
    </CondObject>
```

The following requirements (“req1” and “req2”) cover the desired range of IP addresses. Specifically, req1 indicates that the value of “ipAddr” obtained by event “ev1” (i.e. the IP address of the newly-arrived peer) must be greater than the requirement parameter “req1par” which has the value “192.168.1.1”. Similarly, req2 dictates that “ipAddr” must be smaller than “192.168.1.128”. The evaluation parameter field indicates that the ‘AND’ logical operator must be applied on the two requirements:

```
  <CondReq>
    <reqId>req1</reqId>
    <reqType>GT</reqType>
    <reqObject>
        <event>
            <eventId>ev1</eventId>
            <eventVariable>
                <varId>ipAddr</varId>
            </eventVariable>
        </event>
    </reqObject>
    <reqParam>
        <parId>req1par</parId>
        <parSyntax>string</parSyntax>
        <parValue>192.168.1.1</parValue>
    </reqParam>
  </CondReq>
    <CondReq>
        <reqId>req2</reqId>
        <reqType>LT</reqType>
        <reqObject>
            <event>
                <eventId>ev1</eventId>
                <eventVariable>
                    <varId>ipAddr</varId>
                </eventVariable>
            </event>
        </reqObject>
        <reqParam>
            <parId>req2par</parId>
            <parSyntax>string</parSyntax>
            <parValue>192.168.1.128</parValue>
        </reqParam>
    </CondReq>
    <EvalParams>req1 AND req2</EvalParams>
</Conditions>
```

Finally, the prescribed action is included. The action type indicates the operation that should be called (“addToCD”). A parameter (“3”) must be passed to that function, included in the “actionParam” field:

```
<Actions>
    <actionId>act1</actionId>
    <actionType>addToCD</actionType>
    <actionParam>
        <parId>act1par</parId>
        <parSyntax>string</parSyntax>
        <parValue>3</parValue>
    </actionParam>
</Actions>

</AvpPolicy>
```

Appendix D: Basic Gnut commands

This Appendix presents basic usage information for the *gnut* client. The information is compiled from the gnut manual pages (available online at: http://www.schnarff.com/gnutelladev/source/gnut/gnut-0.4.21/doc/gnut.html). For a full list of supported commands, installation instructions and basic and advanced usage tutorials please refer to the above documentation.

List of Commands:

- **info [ctudhqns]** - Displays information about the current status of the client. If given, the switches limit the output to:
- c - list of current Gnutella Net connections.
- t - file transfers in progress (both upload and download).
- u - upload transfers in progress.
- d - download transfers in progress.
- h - host totals.
- q - queries received and replies sent.
- n - network traffic totals.
- s - shared files on this machine.

If **info** is selected with no arguments, the output appears as shown below:

```
gnut> info
HOST STATS:  Hosts: 19       Files: 2.72K      Size: 9.145G
NET STATS:   Msg Received: 20         Msg Sent: 1
             Bytes Rcvd: 740             Bytes Sent: 23
QUERY STATS: Queries: 0        Responses Sent: 0
SHARE STATS: Num Shared: 0    Size Shared: 0
CONNECTION STATS:
-----------------
1)192.168.1.88:6346    Packs:    0:0    0:0    Bytes:
0:0
TID: 7171     Type: OUT   State:   CONN   Rate:   0:0  /sec
```

- **open *host*:*port*** - Open outgoing connection to a host.
- **find** (or **search**) ***string*** - Search the Gnutella network for *string*. **find** or **search** with no argument displays results.

- **list** - Shows all searches currently going on.
- **monitor** ***keywords*** - Monitor incoming search queries. If arguments are supplied, they are used for a boolean AND match (ignoring case) before displaying search queries.
- **mreply** ***keywords*** - Monitor replies to search queries that others have issued. While *monitor* lets you see what people are searching for, *mreply* lets you find out what's actually available on the network. As with *monitor*, arguments can be given to specify a boolean AND match (ignoring case) before displaying filenames.
- **mpush** ***keywords*** - Monitor push requests issued by others. While *mreply* lets you see what files are available on the network, *mpush* lets you find out which of those files are actually in demand.
- **update** - Send out ping packets to all connected hosts.
- **response** ***regexp*** - Show the current query responses which match the given regular expression.
- **get** ***range*** - Start downloading files referenced by *range*. If more downloads are requested than allowed by max_uploads, then the downloads are queued. The *auto_download_retry* flag affects the behavior of **get**, see below.
- **push** ***range*** - Same as **get**, however only a push connection is attempted.
- **stop** ***range*** - Stop the transfers referenced by *range*. The numbers you give the **stop** command come from the **info** or **info t** command.
- **clear** ***range*** - Removes finished transfers from the transfer list. If no range is given, all completed transfers will be removed. You can also set the *auto_clear* variable to make this automatic. Range numbers work the same way as for the 'stop' command.
- **kill** ***range*** - Terminate the Gnutella connections referenced by *range*.
- **hosts** - Displays the current host catcher (this can result in a lot of information).
- **hosts** ***file*** - Reads in a Gnutella-hosts file.
- **share** ***paths*** - Takes a ":" delimited list of paths to share. (";" on Win32)
- **scan** - Rescans the files in the share paths.
- **quit** (or **exit**) - Quit gnut.