UNIVERSITY OF SOUTHAMPTON

FACULTY OF ENGINEERING AND APPLIED SCIENCE

School of Electronics and Computer Science

DDLS: Extending Open Hypermedia Systems into Peer-to-Peer Environments

by

Jing Zhou

Thesis for the degree of Doctor of Philosophy

September 2004

UNIVERSITY OF SOUTHAMPTON

ABSTRACT

FACULTY OF ENGINEERING AND APPLIED SCIENCE
SCHOOL OF ELECTRONICS AND COMPUTER SCIENCE

Doctor of Philosophy

DDLS: EXTENDING OPEN HYPERMEDIA SYSTEMS INTO PEER-TO-PEER ENVIRONMENTS

by Jing Zhou

Peer-to-peer (P2P) computing is primarily characterised by decentralisation, scalability, anonymity, self-organisation and ad hoc connectivity. It attracted considerable attention in open hypermedia research due to its potential for supporting collaboration among a community of people sharing similar knowledge background. The aim of this research is to investigate the feasibility and potential benefits of incorporating the P2P paradigm in open hypermedia systems to support resource sharing-based collaboration. This is accomplished by utilising a distributed dynamic link service (DDLS) as a test bed, addressing issues that arise from implementing the paradigm, and demonstrating the efficiency of proposed techniques through simulation.

This research begins with the development of a prototype DDLS using the open hypermedia paradigm for storing and presenting resources and a centralised P2P model which adopts a central service directory for publishing and discovering resources in a well-arranged environment. This is enhanced by an operational analysis and feature comparison between prototypes based on the traditional client-server and the centralised P2P models. Various P2P models are analysed to identify the key characteristics of and requirements for the DDLS using an unstructured P2P model which empowers collaboration in an ad hoc environment.

The second phase of this research concentrates on overcoming the challenges of resource description, publishing and discovery posed by the unstructured P2P DDLS: using RDF to encode information about resources, developing a clustering technique to group resources and form the information space; and creating a semantic search mechanism to discover resources; respectively. Finally, this research proposes re-organisation techniques based on the exponential decay function and the naive estimator to enhance the performance of resource discovery in resource sharing-based collaboration.

Contents

List of Figures

List of Tables

Acknowledgements

First and foremost, I would like to thank my supervisor, Professor Wendy Hall, for her time in revising reports, papers and thesis, for her support in attending workshops and conferences, and for her efforts in helping me overcome difficulties throughout my PhD. My PhD experience would be significantly less memorable without her.

I would like to thank Professor David De Roure for his technical support in my research, his feedback on papers and thesis, and his confidence in my capability. I am also grateful to Vijay Dialani for the numerous thought provoking conversations that kept me on the right research path.

My sincere appreciation also goes to Muan Hong Ng, whose friendly smile helped me settle down in the Intelligence, Agents, Multimedia Group when I first arrived; Zhuoan Jiao and her family who have always been concerned about me and made my life more enjoyable; Christopher Bailey for proof-reading my thesis and supplying me with very helpful comments; Norliza Mohamad Zaini for her time and patience to familarise me with SoFAR; Danius Michaelides for his thoughts and experience on mobile link services; Mark Thompson for his academic papers which were impossible to obtain online!; and Georgia Roidouli and Nor Aniza Abdullah for their good company.

This thesis is dedicated to my parents, sister and husband
Lean Zhou, Jiana Huang, Jian Zhou and Wei Deng

Abbreviations Used

API	Application Program Interface
ASCS	Agent Semantic Communication Service
ASM	Association Set Manager
CAN	Content Addressable Network
CB-OHS	Component-Based Open Hypermedia System
CGI	Common Gateway Interface
CNS	Context Name Service
COHSE	Conceptual Open Hypermedia Services Environment
COM	Component Object Model
CPU	Central Processing Unit
CRI	Compound Routing Index
CSCW	Computer-Supported Cooperative Work
DAML	DARPA Agent Markup Language
DBMS	Database Management System
DCS	Document Control System
DDLS	Distributed Dynamic Link Service
DHT	Distributed Hash Table
DLS	Distributed Link Service
DMS	Document Management System
EDFSR	Exponential Decay Function Supported Re-organisation
FIFO	First In First Out
FMS	Filter Management System
FTP	File Transfer Protocol
HBMS	HyperBase Management System
HCM	Heterogenous Communication Model
HCMT	HOSS Communications Model Toolkit
HPMT	HOSS Process Model Toolkit
HR	Hierarchy of Resemblance
HTML	HyperText Markup Language

HTTP	HyperText Transfer Protocol
IR	Information Retrieval
IRIS	Institute for Research in Information and Scholarship
JVM	Java Virtual Machine
LAN	Local Area Network
LFU	Least Frequently Used
LRU	Least Recently Used
LSI	Latent Semantic Indexing
LSS	Link Server System
Microcosm TNG	Microcosm: The Next Generation
NESR	Naive Estimator Supported Re-organisation
NLS	oNLine System
NNTP	Network News Transfer Protocol
OHS	Open Hypermedia System
OHSWG	Open Hypermedia Systems Working Group
OWL	Web Ontology Language
P2P	Peer-to-Peer
RDF	Resource Description Framework
RI	Routing Indices
RMI	Remote Method Invocation
RPC	Remote Procedure Call
SCM	Segregated Communication Model
SDE	Software Development Environment
SHOE	Simple HTML Ontology Extensions
SM	Storage Manager
SMTP	Simple Mail Transfer Protocol
SOA	Service-Oriented Architecture
SOAP	Simple Object Access Protocol
SoFAR	Southampton Framework for Agent Research
SSA	Semantic Search Agent
STA	Semantic Translation Service
TCP/IP	Transmission Control Protocol/Internet Protocol
TTL	Time-To-Live
UDDI	Universal Description, Discovery and Integration
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
UUID	Universally Unique Identifier
VOM	Versioned Object Manager

WSDL	Web Service Description Language
WWW	World Wide Web
W3C	World Wide Web Consortium
XML	eXtensible Markup Language

Chapter 1

Introduction

1.1 The Origins of Open Hypermedia and the Web

Information is the extraction and interpretation of raw data, with the ultimate goal of being further analysed and utilised. When receiving information from the external world, human memory associates different pieces of information and establishes complex information structures that reflect such relationships to facilitate comprehension and review. These structures, because of the way human memory works, are therefore rarely linear.

This essential feature of human memory was captured in the 'Memex', a device for individual use conceived by Vannevar Bush, President Roosevelt's science advisor during the Second World War. He depicted the system in his article, 'As We May Think' (Bush 1945), that supported rapid and natural non-linear access to information by means of associative indexing. Bush also envisioned the concept of trails in the Memex, which associated related information under specific themes that could be retrieved subsequently for a review.

Ted Nelson embodied his enthusiasm for Bush's original ideas and coined the term *hypertext* in 1965 when describing his Xanadu system. He stated in his book 'Dream Machine' that hypertext meant 'non-sequential writing' (Nelson 1987) and ascribed the non-sequential nature of hypertext to the non-sequential structure of ideas. As a model, Xanadu was intended to incorporate a universal repository for all information and literature ever published with all versions of the documents coexisting. Relationships between information were allowed to be instantiated as hyperlinks, or links, which not only connected chunks of information, but also provided a structuring mechanism to be

utilised to navigate through the vast information space. Xanadu employed transclusions as the fundamental mechanism to realise the virtual presence of the same material in distinct contexts through embedded shared instancing (Nelson 1995), therefore effectively managing information reuse and other issues. However, the implementation of Xanadu has yet to be a reality.

The vision of the Memex as 'an enlarged intimate supplement' to human memory was profoundly shared by Douglas Engelbart who endeavoured to augment 'the human intellect' in view of mankind's inability of coping with the ever increasing complexity and urgency of challenges in human situation (Englebart 1986). Engelbart gave a public debut of the oNLine System (NLS) at the Fall Joint Computer Conference in San Francisco in December 1968. The 90 minute presentation successfully demonstrated, among others, innovative hypertext features which included cross-references and hyperlinking, and illustrated the use of NLS in collaborative group work.

The concepts and philosophy initiated by Bush, Nelson and Engelbart led the way for the hypertext system research and development that followed. The first working hypertext system, named the Hypertext Editing System running on an IBM 360/50 mainframe computer, was built at Brown University by Andries van Dam, Ted Nelson and several Brown graduate students in 1967 with most of interface being text-based (Carmody et al. 1969). The first working hypermedia system appeared on the horizon when Aspen Movie Map was developed at the Massachusetts Institute of Technology by Andrew Lippman in 1978 (Nielsen 1990). The term *hypermedia*, also coined by Nelson, extends the notion of hyperlinks to include links among any set of multimedia objects, including the area of a picture, sound, motion video sequences and virtual reality.

The advent of the World Wide Web (also known as WWW, W3 and the Web) (Berners-Lee et al. 1992), the most widely used and successful hypermedia system to date, is considered an important milestone that has demonstrated the possibility of effecting hypermedia across the Internet. The Web is nothing more (or less) than a universe of network accessible information connected by an enormous number of hyperlinks. It achieves universal readership by using a number of essential concepts including that of hypertext, and uses data formats (e.g. HTML[1]) and Internet protocols (e.g. FTP[2], NNTP[3] and HTTP[4]) that make it open, extensible and standard.

The Web is open in the sense that information from a variety of sources can be incorporated in the Web and can be accessed with a Web browser running on computer

[1]HyperText Markup Language
[2]File Transfer Protocol
[3]Network News Transfer Protocol
[4]HyperText Transfer Protocol

platforms supported by heterogeneous hardware and software. However, the way in which the Web is open differs significantly from the concept of 'open hypermedia' as defined by the hypermedia community. The Web can not be thought of as an *open* hypermedia system because it is unable to support an open set of clients that can enjoy the services provided by the Web and to support an open set of data model formats. The concept of an open hypermedia system (OHS) predates the Web and dates back to the mid 1980s when Sun's Link Service (Pearl 1989) was shipped with Sun's programming in the Network Software Environment. The Link Service advocated a loose coupling between the management of data and the management of links with links stored separately from data. Such an external link model is in contrast to the embedded link model (Davis 1998) that the Web employs in which links are stored in documents. The idea of separating hypermedia link facilities from data storage and display functionality is characteristic and crucial in achieving the objectives of open hypermedia research and development.

The external link model has been recognised and encapsulated into a wide range of OHSs, for example Microcosm (Fountain et al. 1990), DHM (Grønbæk et al. 1993, Grønbæk and Trigg 1994), Chimera (Anderson et al. 1994), Hyper-G (Andrews et al. 1995) and HyperDisco (Wiil and Leggett 1997). The link service[5], the term of which was foremost used by Pearl (1989), acts as a middleware component of the client's computing environment. By accessing and manipulating hyperlinks separately from the document, the link service allows hypermedia link facilities to be accessed by an open set of applications, thereby enhancing their performance with hypermedia linking functionality without a rewrite of the applications themselves.

1.2 Motivation: When Open Hypermedia Meets P2P Computing

OHSs have employed distributed architectures since the late 1980s because access to computer networks potentially became commonplace and empowered both a variety of distributed online information to be utilised and groups of distributed teams to collaborate. The prevalent model is that of client-server and three-tier (see Section 3.2). The client-server model has been adopted by a number of OHSs and the three-tier model is more commonly observed in the integration of these systems and the Web. By inte-

[5]The link service belongs to one of the OHS categories in the Flag taxonomy (Østerbye and Wiil 1996), whereas every OHS requires a link service of one form or another to support the hypermedia linking functionality of the system.

grating with the Web, OHSs allow collaboration on a global scale. Example systems include Chimera, Hyper-G, DHM and HyperDisco in which users can appreciate the benefits from both systems.

The client-server and three-tier architectures assume explicit logical separation between the roles of a server and that of a client, and neglect the circumstances in which both the functionality of a server and a client is desirable possession of a single program. For instance, users of a distributed hypermedia system may anticipate to exchange information. Under such a circumstance, an information requestor (client) may also be an information provider (server). The distinction between the server and the client blurs as multiple roles are required of a program. The enabling technologies, such as peer-to-peer (P2P) computing[6] (Clark 2001), is a promise that can be used to support such a system.

Suppose there is a research community in which people sharing similar knowledge background maintain network accessible resources, or documents, for resource sharing-based collaboration within the community. Upon the insertion of documents into their storage, people capitalise on their knowledge to analyse, categorise and annotate the documents. The associated intellectual products, such as categorisation and annotation information, are intended to be used by people in the community to discover and acquire documents of interest from other peers. Semantically related documents are organised and described by a concept hierarchy: the most abstract concept (as well as the documents it is used to annotate) sits at the top of the hierarchy while the most concrete concepts (as well as the documents they are used to annotate) reside at the bottom. The hierarchy bears a tree structure[7]. Because people may have different viewpoints on the same document, sharing resources, including documents, categorisation and annotation information, enables people to understand other peers' opinions on the same concept a document conveys by means of the way the document is categorised and annotated. For instance, if a person anticipates obtaining documents with a topic on the history of hypermedia, he/she can submit a query to other peers. A recipient peer compares the query against the annotation information about the documents he/she maintains and returns the matches if possible. Some of the documents in the result set may be directly related to the history of hypermedia, while others may be associated with the enabling technologies closely involved in the evolution of hypermedia. Nonetheless, the result reflects the different opinions of peers on the same topic, and due to the similar knowledge background peers possess, the query result may be of help for understanding a

[6] Although the term P2P computing is new, the basic P2P technology can date back to at least 1979 when USENET was originally implemented.

[7] In each concept hierarchy, there is a single root concept and each concept (except the root concept) in the hierarchy must be a child of (at least) one parent concept. Moreover, the hierarchy must be cycle-free.

topic from different perspectives.

The scenario above exhibits the following notable properties: equal capability and dispersed autonomy of individuals, and a collaborative relationship implied among individuals. People in the research community can simultaneously be resource requestors and providers. Each of them is independent in making decisions and performing actions, whereas collaboration based on resource sharing becomes crucial when peers' knowledge can assist in accommodating information needs or broadening individual's view. To turn the preceding scenario into reality, several key issues should be taken into account which include how resources can be discovered among multiple providers under certain circumstances, for instance in one well-arranged environment and in another with ad hoc properties, how resources should be organised and manipulated to potentially facilitate their presentation to requestors, and how resource discovery can be expedited.

The use of P2P technologies is to effect and enhance collaboration among people in the research community. The primary functionality of the technologies in the scenario lies in their support for efficient resource discovery and acquisition. A centralised P2P (Lv et al. 2002) approach would typically establish a central directory for resource publishing and discovery, and allow the subsequent resource acquisition to occur directly between peers. In contrast, an unstructured P2P solution would preclude the existence of any form of central authority and realise resource discovery through some search mechanism and routing protocol that needs to be investigated.

Meanwhile, the responsibility of organising, manipulating and presenting resources falls back upon some other orthogonal technologies, one of which that attracts the attention of this work is the DLS. The DLS (see Section 3.4) is a Web-based OHS which satisfies a user's information needs by providing hyperlinks that refer to the documents of interest. The paradigm of information provision, acquisition and presentation exhibited by the DLS fits in with the scenario of resource sharing-based collaboration within a research community.

The concept of grounding an OHS on a P2P architecture is not new. Microcosm (Fountain et al. 1990) and Microcosm TNG (Goose 1997) maintained some form of central repository to facilitate resource discovery in the systems. Microcosm was deployed across a set of workstations and peers could share their resources with others of interest. This enabled all available resources to be extensively utilised by the user community. However, Microcosm was never developed into a P2P system.

It is recognised by this work that the use of a centralised P2P model in which com-

munication and management of computing tasks rely on central servers is no longer feasible for an environment with an ad hoc nature (see Section 5.4.1). This work therefore attempts to identify the requirements for a Web-based OHS, more specifically an open link service, based on an unstructured P2P model which serves to enable resource sharing-based collaboration in ad hoc settings. Other efforts this work makes are to discover a solution that supports the realisation of such an open link service and to explore techniques that can enhance the performance of resource discovery for better collaboration.

1.3 Objectives and Scope

The ultimate objective of this work is to explore how the open hypermedia approach can be augmented by P2P technologies to continuously function in a collaborative environment in which a dispersed resource (in the form of links and linkbases[8]) repository is available for sharing. The primary enabling technologies upon which this work is built are open hypermedia and P2P computing. They complement the mission of each other's in fulfilling the goal of this work - open hypermedia deals with storage, manipulation and presentation of resources while P2P technologies provide solutions to publishing, discovery and acquisition of resources for sharing in distributed environments. There are some critical issues that neither is able to address, for instance, the mechanisms for resource description and clustering that facilitate efficient resource discovery. This calls for other technologies to be involved. As will be described, the Semantic Web (Berners-Lee et al. 2001) and Information Retrieval (IR) (Belkin and Croft 1992) provide promising technologies that satisfy the requirements of this work.

The Semantic Web is an extension to the current Web in which information is made understandable for machine consumption, and is based on the Resource Description Framework (RDF) standards (Lassila and Swick 1999) and other standards to be defined. The RDF provides a simple graph-based model for representing information about resources on the Web. The mechanism for defining groups of related resources and the relationship between these resources is missing in the RDF and is instead provided by RDF's vocabulary description language, RDF Schema (RDF-S). The RDF and RDF-S layer in the Semantic Web infrastructure effects support for some basic querying and reasoning. van Ossenbruggen et al. (2002) envisioned the potential of RDF-enabled search mechanisms to yield a significant improvement over the traditional keyword-based search mechanisms. This bears much implication for open hypermedia research

[8] A linkbase is a collection of links.

because of the following reasons. Firstly, the hypermedia community has long recognised the need for good query and search mechanisms (Halasz 1988). Moreover, the external link model adopted by OHSs is inherently able to make the RDF-enabled query and search mechanism possible. Semantic relationships between links and linkbases could be encoded and stored externally by link servers, thus allowing a search for links to be conducted at both link and linkbase levels. This work primarily explores the way that resources (links and linkbases) should be expressed and maintained at link servers to facilitate querying and searching in environments with varying degrees of decentralisation of control.

To realise resource discovery in ad hoc environments, an overlay network which comprises all peers running the open link service should be established. Ideally, the way that the overlay network is constructed should take into account certain relationships between peers or resources that peers maintain to assist discovery. Related work on the overlay network can be seen in CAN (Ratnasamy et al. 2001), Chord (Stoica et al. 2001) and Pastry (Rowstron and Druschel 2001). Because of the exclusive requirements for the open link service (see Section 5.4.1) this work investigates, the clustering techniques (Theodoridis and Koutroumbas 1999) which have been extensively studied in IR will be employed to organise the overlay network in a way that facilitates resource discovery by grouping peers on the basis of similarity of certain features of their resources.

This work will begin with extending the original DLS into a centralised P2P link service which encapsulates a central service directory for resource discovery in a well-arranged distributed environment. For explanatory and comparison purposes, another link service prototype that adopts a client-server architecture - the common architecture shared by many OHSs, will also be developed. This work will then identify the unique characteristics of and requirements for an unstructured P2P link service that supports resource sharing-based collaboration in an ad hoc environment. The absence of a central service directory in the unstructured P2P link service entails improvements on the the centralised P2P link service. Firstly, this work will need to devise mechanisms for describing, maintaining and manipulating resources which aim to facilitate resource discovery. Secondly, a search algorithm should be available for resource discovery in an ad hoc environment. Finally, techniques that expedite resource discovery should also be explored. Because these techniques are particularly intended to enhance the discovery performance of the link service, this work will conduct an evaluation of gains in the discovery performance that consists of an analysis and a series of simulation to demonstrate the proposed techniques.

To distinguish it from the original DLS, the extended version will be referred to as

the DDLS (Distributed Dynamic Link Service) throughout this thesis.

1.4 Contribution

This work differs from others in the sense that it presents a collective effort which primarily takes advantage of technologies of both open hypermedia and P2P computing to support resource sharing-based collaboration within a community in which people share similar knowledge background. The main contribution of this work attributes to its awareness of a fact that has not attracted sufficient attention before, i.e. the inability of current P2P technologies to help address the problems that an OHS, as required by the scenario mentioned earlier, faces. To circumvent the arising challenges, this work

- devises a way of modelling the information (about resources) space and of establishing the overlay network.
- develops a search algorithm which utilises the semantic relationship between (resources of) peers to discover the resources of interest on the overlay network.
- investigates the application of enabling techniques to enhance discovery for improved collaboration by re-organising the peer network.

Although a number of OHSs, e.g. DHM, HyperDisco and Construct (Wiil and Nürnberg 1999), provide support for different modes of collaboration by means of mechanisms such as the awareness service, event notification, concurrency control and access control, their solutions are centralised and therefore incapable of supporting collaboration in an ad hoc context. This work focuses on how to support collaboration with the absence of a centralised service directory in a distributed environment. The DDLS approach explored by this work leaves issues such as which collaboration services are necessary in a specific OHS and how to implement the services untouched. Instead, it revolves around mechanisms that describe, publish and discover resources to make collaboration in an ad hoc setting a reality. In essence, the DDLS approach is not limited to facilitating only resource sharing-based collaboration as described in Section 1.2. Any OHS that needs collaboration support (Reich et al. 1999) in an ad hoc environment will benefit from utilising the methodology that this work has explored. As a consequence, this work can be regarded as a supplement to research on collaboration support in the open hypermedia community.

Moreover, this work is one of the few practices that implement open hypermedia services, or likewise, based on the Semantic Web infrastructure. By using RDF to encode information about resources and a semantic search mechanism to support resource discovery, this work provides a manifestation of the potential of the Semantic Web to promote open hypermedia research.

Finally, this work proposes the adoption of the unstructured P2P model in the DDLS to support collaboration in ad hoc settings, and investigates all the related issues that collectively enable the realisation of the DDLS. This enriches the research on unstructured P2P systems as described in Section 5.3.2 by providing experience of implementing the unstructured P2P paradigm in a different application domain - the OHS which, due to its characteristics, poses specific challenges to the DDLS approach (see Section 5.4.1).

1.5 Thesis Structure

The remainder of the thesis is structured as follows:

Chapter 2, Open Hypermedia Systems, provides an in-depth look into the field of open hypermedia systems, describes the core concepts and philosophies in detail, and documents the research themes in the area by providing a selection of influential systems.

Chapter 3, Distributed Hypermedia Systems, presents a review of distributed computing models and examines their development and evolution from the author's perspective. Moreover, a selection of distributed hypermedia systems with various software architectures are described.

Chapter 4, Requirements Analysis and Design of the DDLS, presents the motivation and design of extending the original DLS into a truly distributed dynamic link service. In particular, several aspects of software engineering are involved. Two prototypes that adopt a client-server architecture and a centralised P2P architecture are described, respectively. An analysis and comparison between the architectures of both are performed.

Chapter 5, Rethinking the P2P Paradigm, presents an overall review of contemporary P2P solutions from the architectural perspective. Also, the characteristics of and requirements for the DDLS are examined in detail, which explains why the existing approaches do not fit the conceived scenario for this work (see Section 1.2).

Chapter 6, Evolution of the DDLS into an Unstructured P2P System, explores the extension of a centralised P2P DDLS (see Chapter 4) to an unstructured P2P system. The approach to resource description at the linkbase level is proposed and a distance-based semantic search algorithm is presented. Simulation is conducted which helps understand the behaviour and performance of the algorithm, with varying distributions of potential resources and different query profiles involved.

Chapter 7, Re-organising the DDLS Peer Network, introduces the concept of re-organisation and its supporting data structure, query history, upon which re-organisation of the peer network primarily depends. The exponential decay function and naive estimator are proposed to support re-organisation, and their different effectiveness and applicability are demonstrated by a series of simulation.

Chapter 8, Conclusions and Future Work, summarises this work and presents possible future directions of the research.

1.6 Declaration

This thesis represents the author's personal view of the field. It is all the author's independent work, with the exception of that described in Section 6.4 which was conducted in conjunction with Vijay Dialani.

Chapter 2

Open Hypermedia Systems

2.1 Introduction

This chapter presents open hypermedia systems (OHSs), a field that offers significant concepts, philosophies, experience and lessons potentially benefiting the entire hypermedia community. A selection of influential OHSs, chosen on the basis of the Flag taxonomy (Østerbye and Wiil 1996), are described to present the core concepts and philosophies, and to document the research development in the area. This is followed by a description of the close relationship between open hypermedia research and that of the emergent Semantic Web.

2.2 Concept and Features

An important consideration in the field of hypermedia systems is the distinction between structure and content. A hypermedia system that imposes a data model (structure and data formats) on its hypermedia enabled applications is considered to be closed, since applications have to be custom-made to participate in the hypermedia environment. An OHS, however, only imposes a structure format on its hypermedia enabled applications and allows content to be stored outside the system. The applications can store content in different formats, which encourages the integration and use of third-party applications.

An OHS is typically a middleware component in the computing environment offering hypermedia functionality to applications independent of its storage and display functionality. It enables the client applications to create, edit, delete and activate links

which are maintained and manipulated separately in linkbases. Hypermedia services can also be used by other third-party applications, programs and services in the computing environment (Wiil 1997).

The term *open* means that OHSs allow an open set of clients (applications) of the hypermedia services provided by the systems and support an open set of data model formats. In contrast to closed hypermedia systems, such as the Web, OHSs exhibit a variety of advantages and circumvent the limitations of closed hypermedia systems due to the inherent weaknesses in their architecture, data model, protocol and enabling technologies. The advantageous properties that OHSs possess are highlighted as follows.

1. *Separation of links from documents*
   The open hypermedia model enables the maintenance and manipulation of hypermedia links separately from the documents they describe, which fundamentally differs from the model of closed hypermedia systems. The separation of links from documents enables links to be applied to documents in any format (Carr et al. 1998a).

2. *No imposition of mark-up on data*
   OHSs allow more types of data model formats which are not limited to either HTML or image formats. New data model formats can be supported in OHSs by enabling the applications capable of handling the required data model formats. Applications are extended to make hypermedia functionality available in the hypermedia environment with minimum efforts while the data content remains unaltered.

3. *Integration with third-party applications*
   Unlike the closed hypermedia system, an OHS provides a linking protocol between applications and the OHS that allows any application to participate in the hypermedia service. Through the linking protocol, applications are loosely integrated into the hypermedia environment with various levels of hypermedia awareness.

4. *Context-specific query*
   The task-specific query context can be specified by a user, or inferred by a user interface agent. Therefore, link following is affected not only by the selection of the link source but also the user's dynamic context that indicates the kind of resources they would like to follow up.

5. *Easy to add new functionality*
   A component-based approach was suggested by Wiil and Nürnberg (1999) for the

design of OHSs. The component technology empowers the specification of the services of a component in an implementation independent manner. Each component provides descriptions of services in a separate interface serving a domain model, concealing the concrete implementation. The approach enables an extensible architecture in which a new hypertext domain can be added with ease by defining an interface for the description of services presented in that domain.

6. *Distributable hypermedia structure processing*
   OHSs adopt different software architectures, spanning from traditional centralised client-server systems with a central storage server running on a LAN[1] to multi-layer systems with multiple storage servers operating within different Internet domains. The decentralisation of storage relieves the burden on a single server and enables hypermedia structures to reside in the vicinity of where they may be requested.

Links are first-class entities in an OHS. The central feature of an OHS is that it can make use of link specifications stored in linkbases and manage the links separately from documents. By allowing links to be manipulated separately, an OHS decreases document maintenance efforts as there is no need for a document to be revised in order to change its links. An open link service allows the performance of any application to be enhanced with hypermedia functionality without a rewrite of the application (Carr et al. 1995), which is also a minimal requirement for an OHS.

Hall et al. (1996), based on their experience gained from the design and development of Microcosm (see Section 2.4.3), summarised the essential properties in defining a truly open hypermedia system as follows.

1. *Size*: no limitations, with regard to either the size of objects or the maximum number of such objects, should be imposed by the hypermedia system.
2. *Data formats*: the system should allow import and use of data in any format, including temporal media.
3. *Applications*: the hypermedia system should provide facilities for any application to access the hypermedia service in order to participate in the hypermedia environment.
4. *Data models*: the system should be configurable and extensible to allow new hypermedia data models to be incorporated. Furthermore, interoperability is an important property that the system should possess.

[1] Local Area Network

5. *Platforms*: it should be possible to implement the system across a variety of platforms.

6. *Users*: the system should support multiple users and allow the users to maintain their private views of the objects in the system.

To fully satisfy such a definition there is still a long way to go in OHS research. Examining the OHSs documented later in this chapter demonstrates that the efforts from the OHS community have only achieved part of the entire specification. However, the definition of openness as described will serve as a signpost, showing future directions for open hypermedia research.

2.3 The Flag Taxonomy

The Flag taxonomy (Østerbye and Wiil 1996) was built on the terminology of the Dexter hypertext reference model (Halasz and Schwartz 1994) which was a popular model covering ideas and experience from advanced hypermedia research at that time. The taxonomy aimed to serve as a framework to describe, classify and compare different hypermedia systems in a system independent way.

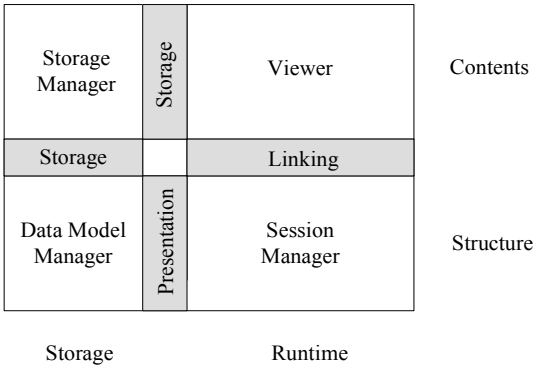

FIGURE 2.1: The Flag of Hypermedia Systems
(Østerbye and Wiil 1996)

Figure 2.1 reveals that the taxonomy does not represent hypermedia systems by a layered architecture as that in the Dexter model. Rather, it distinguishes between the storage aspect and the runtime aspect on the one hand, and structure and contents on the other, leading to four functional modules (illustrated by the white rectangles) and

four protocols (illustrated by the grey rectangles). The Storage Manager corresponds to the within-component layer in the Dexter model, and the Data Model Manager responsible for storing the structure of a hypertext maps to the storage layer in the Dexter model. The taxonomy divides the Dexter runtime layer into the Viewer module and the Session Manager module, which bridge the Dexter within-component layer and the runtime layer to reflect the exclusive property of the OHS that contents can be stored by applications outside the hypermedia system.

Each functional module provides functionality to neighbouring modules by means of available protocols. The Storage protocol encapsulates the Storage Manager from both the Viewer and the Data Model Manager because the Flag taxonomy divides storage into contents and structure aspects. The Linking protocol provides necessary operations which bind the Viewer and Session Manager functionality together. The Presentation protocol defines the operations in the Data Model Manager and the Session Manager available to each other.

The taxonomy emphasises the distinction between structure and contents as the major criterion to identify an OHS, and excludes any approach to OHSs that imposes a data model on their hypermedia enabled applications. The categories of OHSs, according to the Flag taxonomy, include two fairly independent domains: link server systems (LSSs) and open hyperbase management systems (HBMSs) (Wiil and Leggett 1997). LSSs (e.g. Microcosm and Chimera) revolve around the development of middleware components that allow existing tools to use hypermedia linking functionality, whereas open HBMSs (e.g. DHM and HyperDisco) focus on the development of middleware components that not only provide linking but also storage functionality to be utilised. This thesis is only considering LSSs.

2.4 Link Server Systems

The link service, a term first used by Pearl (1989), allows hypertext (or hypermedia) facilities to be accessed by an open set of applications, thereby acting as a middleware component of the user's computing environment in which links are allowed to connect objects in any media without any restrictions on the data format. Link server systems prefer an external link and reference model in which a link service stores links and content references while leaving the original document intact since no mark-up or anchor tables are embedded in the document. With all the information about links being kept in a separate linkbase, links to and from a document can be traced by querying against the linkbase. As links become independent objects in terms of management, the revision

of links in a changed document is unnecessary. Besides, links may be applied to serve other legacy systems with data on read-only media.

2.4.1 Intermedia

Intermedia (Meyrowitz 1986, Yankelovich et al. 1988), developed at Brown University's Institute for Research in Information and Scholarship (IRIS) from 1985 to 1990, was the first advocate for an open hypermedia philosophy. It was a multi-application hypermedia system designed to support teaching and research in educational settings. Intermedia was intended to model how hypermedia functionality should be handled at the system level to provide linking capabilities integrated into the desktop environment.

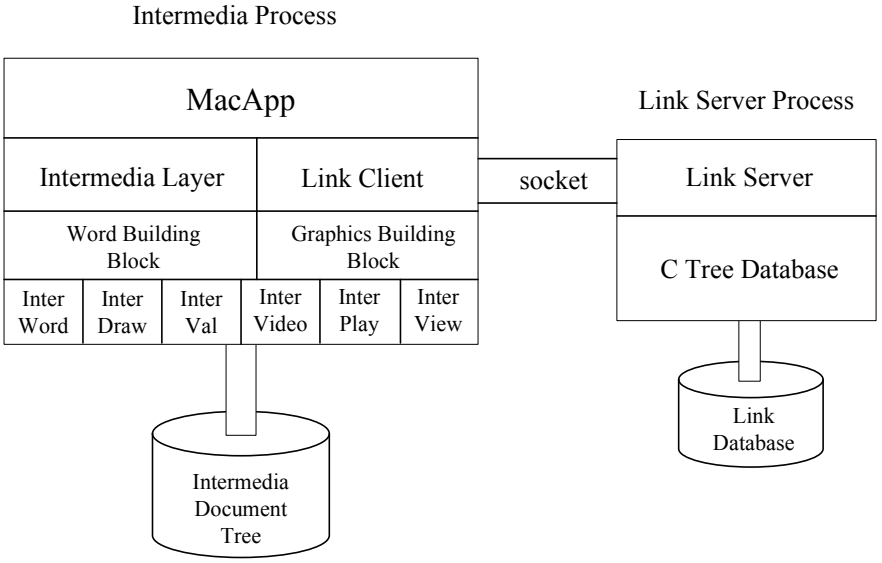

FIGURE 2.2: Intermedia Architecture
(Haan et al. 1992)

The overall architecture of Intermedia, based on the client-server model (Section 3.2.1), is shown in Figure 2.2. From the operating system's perspective, the Intermedia system appeared as two distinct processes, the Intermedia process and the link server process that communicated via sockets. The first bottom layer of the Intermedia process consisted of applications sharing functionality defined in its immediate upper layer. The word and graphics building blocks in the second layer were implemented for encapsulating important end-user functionality and providing a consistent user interface. The Intermedia layer, providing classes for implementing the core link-

ing functionality, extended MacApp's classes and added the functionality necessary to specific applications. The link client was a library bound with the Intermedia layer. The link server was also implemented as a library associated with a database management system (DBMS) running as a separate process. Intermedia was only implemented for Apple's version of the UNIX operating system. It presented to users customised integrated applications operating on their own document types that conformed to user interface standards for consistency purposes. Thus, users would encounter quite identical implementations of features seen elsewhere across multiple applications.

Intermedia allowed users to create bidirectional links between specific locations in documents (created by its dedicated applications shown in Figure 2.2) of different types. This was a distinctive property unseen in other systems, such as HyperCard (Apple Computer Inc. 1989), NLS (Englebart 1986) and Notecards (Halasz et al. 1986). Intermedia named these specific locations 'anchors'. Information about anchors and links between the anchors were stored separately from documents they described. Collections of anchors and links were partitioned into webs. Users could alter their working context by switching from one web to another, and therefore a different set of anchors and links in the web were superimposed on the documents that users were browsing. To maintain data consistency, the deletion of an Intermedia document would lead to the deletion of all the anchors and links within and to the document throughout the Link Database. Concurrency control was implemented in Intermedia to help manage multiple users sharing a network of hypermedia material. Intermedia supported multiple users with granted access rights to read and annotate (anchors and links of) a single document simultaneously, but only one user to write a document at a time.

Intermedia was a pioneering hypermedia system[2] that achieved academic success and aroused great research interest. However, due to the failure of convincing other people to adopt its protocol and the lack of funding to upgrade to a new operating system, Intermedia fell into disuse in the early 1990s.

2.4.2 Sun's Link Service

Sun's Link Service (Pearl 1989) was not a hypertext system in its own right but provided a protocol and linking functionality for integrating linking mechanisms into existing applications in a distributed workstation world. The link data and object data were stored

[2]Intermedia was considered a monolithic hypermedia system because its architecture tightly controlled data, hypermedia structures and the user interface of the system. Also, Intermedia allowed only dedicated applications to access its hypermedia services and therefore it might be best described as a partially open hypermedia system.

and managed separately. Manipulating and storing objects were undertaken by independent editing applications. A linking protocol was designed to facilitate communication between the Link Service and the integrated applications.

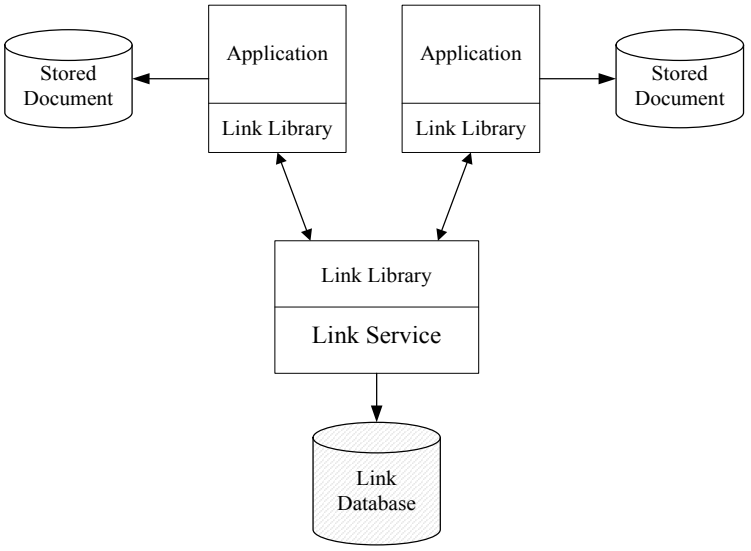

FIGURE 2.3: Sun's Link Service Architecture
(Pearl 1989)

The Link Service (see Figure 2.3) comprised the protocol specification, a link server program, a library that defined the protocol for integrating with the Link Service and utilities for managing the link databases (i.e. linkbases). Integrating with the Link Service required little change of applications. Each integrated application included a link library which was part of the link server process to communicate with the link server, and therefore became part of an extensible and loosely coupled frontend interface to the hypertext system. Applications registered their availability and capability with the Link Service so that they could be called to handle objects upon users' requests.

The manager of the object data provided an interface and application specific functionality for users to manipulate objects. Similarly, the Link Service offered an interface and functions for users to create and modify links between data objects. By separating the user interface for linking from that for editing, the Link Service introduced as minimal impact as possible on the appearance of integrated applications and the burden on the cognition of users.

The Link Service addressed the link maintenance issue by means of two mecha-

nisms: implicit and explicit. When a user attempted to traverse from the valid end of a link to an invalid node, the Link Service informed the user of the dangling link and suggested deletion of the link. Or, the Link Service utilised a link garbage collection mechanism to check the validity of links by querying their managing applications. The Link Service left the versioning of data objects to individual applications and maintained the consistency of link objects to a limited degree. This is because the versioning of data objects was implemented by integrated applications and the Link Service was unable to establish any connection between the versioning of the link objects and data objects.

Although some issues required further investigation, such as the granularity of an identifiable object in structureless documents and the extension of link types available between objects, Sun's Link Service made an initial attempt at integrating linking functionality into existing applications and the accompanying protocol was crucial for an extensible and loosely coupled open hypertext system.

2.4.3 Microcosm

The development of Microcosm (Fountain et al. 1990), which was one of the first OHSs, predates the Web. The initial motivation for Microcosm arose from constructing fully cross-referenced versions of very large electronic archives and information repositories (Lowe and Hall 1999). While building resource-based applications using hypermedia systems, the Microcosm team found that the main issues for hypermedia designers and authors were the heavy load from working with a large number of documents and links, and the increasingly highly multimedia nature of electronic information. Besides, different users would access different parts of the information repositories and try to understand them from different perspectives. Surrounded by collections of unstructured information, it was hard, even impossible sometimes, to find the beginning or end. For users who came across applications for the first time, no assumptions could be made about their preferences, knowledge, beliefs or information seeking goals, based on which some kind of assistance could be provided to support their interaction with the systems.

Microcosm was best thought of as a number of autonomous processes that communicated with each other by a message passing mechanism, see Figure 2.4. The Segregated Communication Model (SCM) employed in the original Microcosm supported both viewer processes and filter processes. The Document Control System (DCS) maintained a record of each viewer and coordinated message routing between viewers and

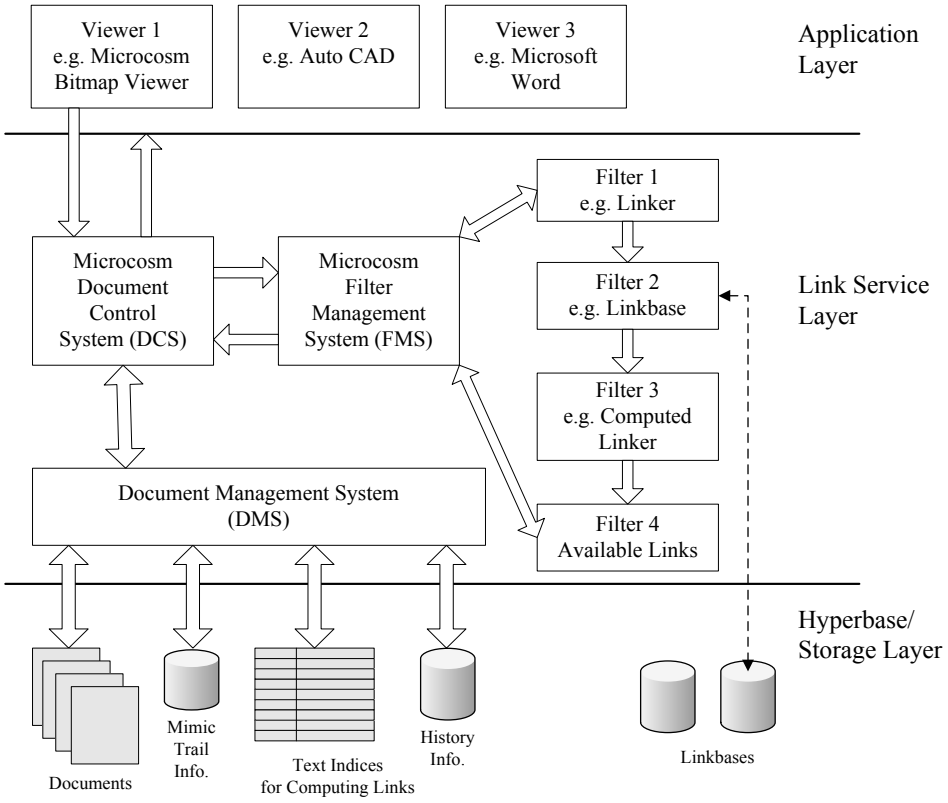

FIGURE 2.4: Microcosm Architecture
(Lowe and Hall 1999)

the rest of the system. Viewers were identified by means of an identifier allocated by the DCS. The Filter Management System (FMS) coordinated the serial chain of filters. Filters in Microcosm were among those independent processes which were connected in a chain topology and might be dynamically installed, removed or even reordered. The hypermedia link service was embodied within filters[3]. When an action was initiated by a user, a message that contained the details of relevant information would be sent to the filter manager which arranged the message to be passed through all registered filters. Filters declared their interests in handling specific messages from users' actions and they might block, ignore, alter messages or create new ones (Hall et al. 1996).

The sequence of some filters was fixed. For instance, the linker filter, a process which primarily dealt with the start link and end link messages, was usually positioned in the first place along the filter chain. Following it was the linkbase filter that was responsible for creating, following and resolving links. The available links filter was

[3]Very little of the functionality that users see was 'hard coded' in the core of Microcosm, which was one of the flaws in the architecture.

typically positioned at the end of the filter chain to present the result of a link traversal or search action. This filter could not precede the linkbase filter since there would be no links for it to display before the linkbase filter processed a message and gave the references to destination documents. For the linker filter and the linkbase filter, the exchange of their positions in the sequence would not incur such a problem and therefore was permissible.

The bottleneck between the DCS and the FMS was one of the limitations on the performance and scalability of Microcosm. In addition, the serial and uni-directional filter chain yielded major communication overhead because all messages are propagated to every filter in the chain. Such limitations were resolved in the design of filters in Microcosm TNG (see Section 3.3.3) and linkbases in the DLS (see Section 3.4) in which no chain topology was used in organising filters and linkbases. In both systems, individual filters and linkbases could be dynamically added or removed from the process of link resolution and retrieval, and consequently different views of link data were presented upon different user requests.

The Microcosm model allowed users to create three primitive link types: specific links, local links and generic links. A specific link may be followed from the source selection at a specific location in a specific document. A local link may be followed when the source selection occurs at any place in a specific document. A generic link may be followed from wherever the source selection occurs. The Microcosm team identified two distinct link integrity problems - the *editing problem* and the *dangling link problem* - that might occur in Microcosm and explored their solutions. Hall et al. (1996) provided more details.

In applications such as the delivery of teaching materials, Microcosm required that all shared resources, such as documents, linkbases and the Document Management System (DMS), should be made read-only so as to solve the concurrency problem. However, for other applications involving a small number of changes to shared resources, Microcosm could adopt a crude user controlled locking and notification scheme which allowed locks on each document and linkbase (but not the DMS), so that only one single user might edit shared documents or links at a time, and the edit would be notified to other users before they carried out any further update. In order to support Microcosm working in a large scale cooperative environment, it was envisioned that using a client-server architecture which had only one database to host all linkbases and DMSs might produce the ease of concurrency control.

2.4.4 Chimera

Chimera (Anderson 1997) was an open hypermedia system developed at the University of California, Irvine. It primarily aimed to provide hypertext services in heterogeneous software development environment (SDE).

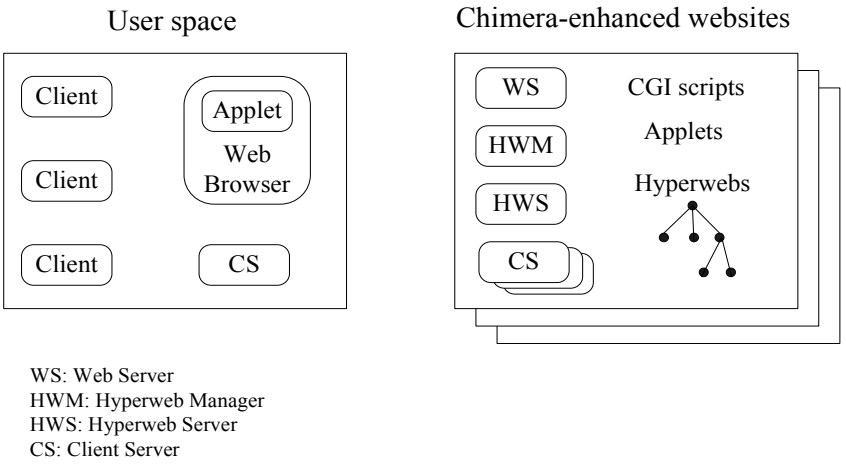

FIGURE 2.5: Chimera 2.0 Architecture
(Anderson 1997)

The Chimera architecture adopted a client-server approach (see Figure 2.5) and was separated into two environments: a user space and Chimera-enhanced websites. A user space consisted of Chimera clients, a client server and a Java-enabled Web browser, while a Chimera-enhanced website contained a Web server, a hyperweb manager, a hyperweb server, a set of hyperwebs and a set of client servers.

Chimera clients were applications that provided end users with hypermedia services. They interacted with the local client server which connected to the hyperweb server to provide its clients with access to hyperwebs. Java applets could also be Chimera clients in support of Chimera's integration with the Web. However, such clients would face obstacles that normal clients did not have to. This is because Java imposes security restrictions on applets, and the Web server, the hyperweb manager and the client server must all execute on the same machine in order that applets could communicate with them. A CGI[4] script was used to append HTML code to include an applet at the end of a Web page. The Java applet was downloaded by the Web browser in the user space and it provided pervasive access to Chimera's services to the Web page a user was visiting. Therefore, Chimera users were free to manipulate links or hyperwebs and initiate link traversal from within an applet's interface.

[4]Common Gateway Interface

In Chimera, a hyperweb that was manipulated by the hyperweb manager referred to a database file containing groupings of related hypermedia concepts. The hyperweb manager was responsible for the creation and deletion of hyperwebs. Besides, it provided connection information of Chimera related servers. For instance, the hyperweb manager could reveal the contact information of the hyperweb server on the Chimera site of interest to the client server in the user space. The hyperwebs of a Chimera site were stored and managed by the hyperweb server.

In addition to hyperwebs, Chimera had a set of hypermedia concepts, including viewers, objects, views, anchors and links along with their attributes. Objects were named, persistent entities in Chimera and were displayed, created and edited by viewers. A view associated an object with the viewer that displayed it. Anchors were created and managed by a viewer with respect to the particular view of the object being displayed, rather than the object itself, while a link consisted of a set of anchors. Each instance of a Chimera hypermedia concept could be described by an arbitrary number of attribute-value pairs specifying run-time semantics or behaviour. These concepts enabled the hypermedia needs of an application to be easily modelled and implemented by means of invoking appropriate calls to Chimera's API[5] to create persistent instances of the concepts. Therefore, the integration of an application with Chimera was facilitated.

Chimera could not be completely modelled in the Dexter model since the former could handle the presence of links with zero or one anchor (dangling links) (Anderson et al. 1994). In this case, Chimera was similar to DHM, see Section 3.3.2. Moreover, the concept of a view in Chimera could not be modelled by a composite component in Dexter, because a view contained information about the object being displayed and the view that displayed the object, but a composite component included references to atomic components which contained only data. Finally, a Chimera viewer would be able to define anchors on the view of an object which exists only at run-time. However, Dexter was unable to specify the same type of anchors.

2.4.5 Hyper-G

Hyper-G (Andrews et al. 1995), a multi-user, multi-protocol, structured and distributed hypermedia information system, was the product of a group of researchers and developers at Graz University of Technology, Austria. It integrated all the functionality of the Web with a set of facilities, resulting in much additional functionality, and therefore was seen as an extension of the Web (Maurer 1996).

[5]Application Program Interface

Like most information management systems, Hyper-G was client-server based (see Figure 2.6). The Hyper-G server consisted of a number of modules implemented as concurrent Unix processes. The document server maintained local documents of the Hyper-G server as well as cached documents from remote servers. The full text server was responsible for storing an inverted index of all text documents for searching. The link server, which was later renamed the object server, primarily stored a database of objects and the relationship between objects. Every Hyper-G object, including documents, anchors and collections (defined later), could be searched for. Hyper-G objects were typically described by a set of attributes and the search on attributes was supported. In particular, documents and collections in a Hyper-G server were automatically indexed upon the insertion into the database and could be subsequently accessed by a full-text search.

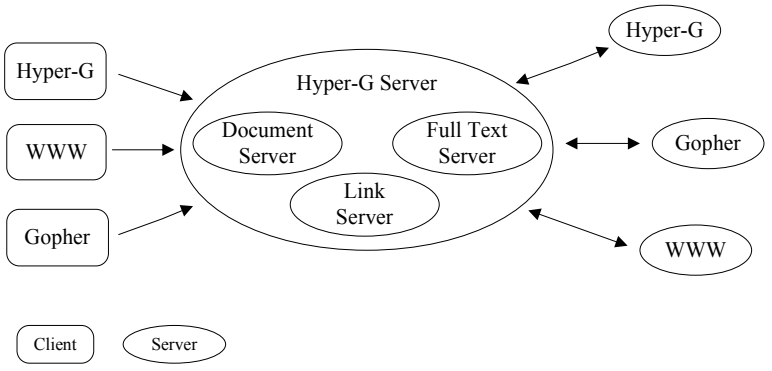

FIGURE 2.6: Hyper-G Architecture
(Andrews et al. 1995)

Users could access Hyper-G using a standard WWW browser, a naive Hyper-G client, or a Gopher client. A WWW gateway program was developed and installed with every Hyper-G server so as to provide the user of a WWW client with an interface to the Hyper-G functionality. When accessing the Web with Hyper-G clients, a Hyper-G server was typically used as a gateway. Hyper-G was also designed to be fully interoperable with Gopher. Hyper-G servers could present themselves to Gopher clients by means of the Gopher gateway. When accessing Gopher servers with Hyper-G, a Hyper-G server, which cached Gopher directories and documents, acted as a proxy.

The concept of *collections* in Hyper-G implemented a hierarchical structuring mechanism. A collection contained documents or other collections. This definition led to the generation of a *collection hierarchy*. While every document or collection (except the Hyper-G server's root collection) must be a member of at least one collection

(its parent collection), the collection hierarchy must be cycle-free. The collection hierarchy could be used for a number of occasions. For instance, the logical structure of the information was made explicit to users by means of the collection hierarchy, which facilitated users' navigation through the information space. Also, documents and collections might have multiple parent collections in a collection hierarchy, and therefore users could have multiple views of the same available information. The collection hierarchy could be attached with access permissions which supported multiple users to simultaneously use a single Hyper-G server. Finally, Hyper-G allowed the use of the collection hierarchy to define the search scope.

Hyper-G implemented links as objects containing attributes. Links could be assigned keywords for searching and permissions for access restrictions. A link in Hyper-G connected a source anchor within one document and a destination anchor within another document, an entire document, or groups of documents. Links were not stored within documents but in separate link databases. Hence, Hyper-G supported bidirectional linking and link consistency within servers and across server boundaries were guaranteed.

2.5 Open Hypermedia Systems and the Semantic Web

The Semantic Web, the next generation Web infrastructure as envisioned by its inventor Tim Berners-Lee, is designed to provide information in the Web in a more machine understandable manner. Recent initiatives at the World Wide Web Consortium (W3C) have produced multiple specifications, such as that of XML[6], RDF and OWL[7] (Smith et al. 2004). While XML defines customised tagging schemes and RDF enables a flexible approach to representing information, OWL provides more vocabulary for formally describing the meaning of terms in a Web document and the relationship between those terms. They form part of the growing stack of the W3C recommendations related to the Semantic Web.

The need of the Semantic Web to capture and represent the semantic relationship between resources on the Web, invalidates the embedded linking model as used in the Web. Moreover, downloading bulky semantic annotations together with Web documents deteriorates the performance of hypermedia applications, especially when annotations are even not required. A potential approach to these issues involves externally encoding information about resources and the semantic relationship between the

[6] eXtensible Markup Language
[7] Web Ontology Language

resources, and employing dedicated servers to maintain and manipulate these semantics (van Ossenbruggen et al. 2002). OHSs inherently possess the capability to deal with similar problems. The most significant feature that empowers OHSs to facilitate, and also enjoy, the emergent Semantic Web technologies is that links are stored and managed separately from the documents they describe. Capturing the semantic content (concepts) of documents, modelling it as metadata, and authoring links between related concepts to construct hypertext, are therefore feasible. This is also applicable to annotations - the ability to annotate documents of others has been an important feature in many hypertext/hypermedia systems. Storing annotations externally to the documents that are annotated and accessing them over some protocol, can produce an OHS-like annotation service. Example systems include COHSE (Carr et al. 2001) and Annotea (Kahan et al. 2001).

The main objective of the COHSE (Conceptual Open Hypermedia Services Environment) project is to produce an ontological reasoning service which provides a conceptual model for describing document terms and the relationship between the terms, and a Web-based open hypermedia link service to deliver link-providing facilities in a scalable and non-intrusive manner. The conceptual information of Web documents is represented as metadata[8]. Metadata can be reasoned over to classify documents by using a predefined ontology in COHSE - a thesaurus consisting of concepts related by different relations. Documents are considered to be similar in some way if they share metadata. The COHSE link service authors links between associated concepts, and therefore corresponding documents are also linked for navigation.

The Annotea project, part of the Semantic Web efforts, aims to enhance collaboration via sharing metadata-based annotations, bookmarks and their variants. Annotea is a Web-based annotation system built on top of an open RDF infrastructure through combining RDF with XPointer, XLink and HTTP. Annotations are modelled as a class of metadata. They are described with an RDF schema and are stored inside generic RDF databases hosted by annotation servers. XPointer and XLink associate metadata with part of the document that is annotated. By interacting with an annotation server over HTTP, users can perform different operations on annotations, such as retrieval, addition, modification and deletion.

Both practices of COHSE and Annotea reveal that the concepts and philosophies of OHSs can be utilised to satisfy the requirements of the Semantic Web. On the other

[8]This is an idea borrowed by this work on resource description presented in Section 6.3.2. Although both COHSE and the DDLS employ metadata to describe the concepts that documents are associated with, the former intends to construct hypertexts and build links for navigation, while the latter aims to assist resource discovery.

hand, it is also demonstrated that the Semantic Web can, and definitely will, augment open hypermedia research and enrich its potential area of application by providing state-of-the-art and standardised technologies.

2.6 Summary

This chapter examined the core concepts and features of OHSs and described the Flag taxonomy which divides OHSs into two categories: link server systems (LSSs) and open hyperbase management systems (HBMSs). In particular, a selection of representative link server systems was chosen and studied. Examining contemporary research on OHSs and its close relationship with the Semantic Web identified that both parties are mutually beneficial - the Semantic Web provided advanced technologies to augment open hypermedia research, and meanwhile, open hypermedia presented many years of experience and lessons to facilitate research on the Semantic Web.

The next chapter will present the field of distributed hypermedia systems. Following a review of distributed computing paradigms, it will document a variety of closed and open hypermedia systems which adopt distinct distributed models.

Chapter 3

Distributed Hypermedia Systems

3.1 Introduction

This chapter presents a review of distributed computing in terms of the software architecture, ranging from the client-server to the contemporary service-based model. The review aims to provide a prerequisite knowledge for understanding the concepts and backgrounds in the following presentation of selected distributed hypertext and hypermedia systems along the path of development.

3.2 Models of Distributed Computing

Bass et al. (2003) define the software architecture of a program or computing system as 'the structure or structures of the system, which comprise software elements, the externally visible properties of those elements, and the relationships among them'. The definition implies that software architecture is concerned with the encompassing elements that interact with one another. The behaviour of each element is also part of the architecture.

Distributed computing is one of the software architectures in which processing occurs among teams of collaborative computers over a network. The architecture allows geographically distributed computers to work together with responsibility being partitioned among multiple parties. Distributing the computational load among appropriate computers can improve the system performance.

3.2.1 Client-Server Architecture

Client-server, a term first used in the 1980s, is a distributed computing model in which client applications request services from server processes. A client application is a process or program that sends messages to the server and requests the latter to perform specific tasks. The server process or program listens to the client requests, receives messages and performs corresponding actions. A single machine can be both a client and a server depending on software configuration. However, clients and servers typically run on differently machines interconnected by a computer network.

The central server runs on powerful personal computers, workstations or mainframe computers which host the majority of computing resources, for example, files, devices and processing power. Clients are separate and subordinate to the server. They log into the server and make a search request on the server that will immediately return the search result. The subsequent requests to the server for services will also be tracked.

The client-server model gained its wide acceptance in the 1990s. The reason is simple because it adheres to software modularity and usability requirements very well. The model simplifies the administration and management tasks by monitoring user access at the central control point (the server). Also, it enables systems to be easily extended by adding new services in the form of new servers.

One of the main drawbacks of the client-server model is its heavy dependence on the central server, which results in a system being vulnerable to server failure and being plagued by the dramatic and exponential growth of online service requests. Therefore, the expenditure in the maintenance of distributed components and others is increased.

3.2.2 Three-tier/Multi-tier Architecture

The client-server architecture (also referred to as the two-tier architecture) assumes that clients access servers that run on the same operating system or use the same database engine. Otherwise, clients must be equipped with matching drivers for such configurations. Any update to an application needs to be deployed for all users of the application. Also, it is shown that beyond 100 users, the performance of the two-tier design is exceeded. The three-tier architecture, which emerged in the 1990s, addresses the problem by introducing a middle tier between the front-end client environment and the back-end server environment to support application logic and common services.

Systems based on the three-tier model can be split into three logic tiers: the user

interface tier, the business logic tier and the database access tier. The user interface tier is responsible for accepting user requests and forwarding them to the business logic tier. The business logic tier acts as both a client and a server because it processes requests from the user interface tier and sends them further to the database access tier which provides database management functionality. According to the user request, the connectivity between components of these tiers can be dynamically changed and established. In some cases, the middle tier consists of two or more units with different functions. Therefore, the three-tier model is also referred to as the multi-tier model.

The three-tier model addresses the issues that the two-tier model is incapable of dealing with while hiding the complex distributed processing from users. It can accommodate more than 100 users. By centralising process logic at the business logic tier, the model promises improved performance, flexibility, reliability and scalability.

3.2.3 Peer-to-Peer Architecture

The peer-to-peer model (Clark 2001) refers to a class of systems and applications that employ resources in a distributed environment to perform a critical function without central servers. Each node, or peer, plays the role of both a client and a server. Examples include instant messaging systems and document sharing applications, which have exploded in popularity and transformed the way users interact with one another over the network. P2P networks allow a group of online users with the same networking program to connect with one another and directly access files from others' physical storage. In the P2P model, each peer has equivalent capabilities and responsibilities.

Peers are autonomous, free from the control of any other party. Therefore, a network administrator is unnecessary in the P2P model, which cuts down the cost of administration and maintenance by spreading control and expenditure across all peers. Moreover, the absence of a centralised control authority yields a robust system against the single point of failure. Search results, as a consequence of the direct contact with information providers, keep fresh when they are requested.

However, spreading the overall control complicates many issues. First, there is no accurate view of the entire system since each peer holds a partial picture. Also, it becomes difficult to know the state of a component or to locate specified resources of a component through interrogating any single component. The query and search mechanisms need to rely on techniques specifically designed for P2P computing, which should involve as few as possible of peers to avoid the heavy use of bandwidth and low search efficiency.

3.2.4 Component-based Architecture

A component-based architecture comprises an architecture and a set of APIs which define modular and reusable software components that can be deployed and assembled into larger systems. A software component is a piece of code that encapsulates certain functionality and publishes the operations to access the functionality at the interface between components. Each component conforms to a prescribed behaviour common to all the other components in the same architecture. Large software systems can be built by assembling and integrating the existing software components.

The assumption underlying the use of component-based architecture is that certain parts of a large system are used regularly. Rather than being written many times, the code which encapsulates the common parts into components should only be written once. The component-based model provides system designers with a mechanism to develop applications by composing existing software components through their well-defined interfaces without developing new components or changing the existing ones.

The component-based model reduces software development and maintenance costs, and increases the flexibility of the system. As the software units of change, components are easy to evolve and upgrade. New requirements from the changing environment can be satisfied by developing compatible components and plugging them into the system. Furthermore, components can be easily tested independently of the larger system.

3.2.5 Service-based Architecture

A service-based architecture[1] is essentially a collection of loosely coupled services that involve various means of connection for communication between one another. A service is a location transparent and network addressable unit of software logic offered by service providers to achieve the intended functionality. Compared to components, services are coarser grained software elements that satisfy a particular requirement. They are well-defined and self-contained, independent of the context or state of other services. Services have published interfaces that define operations with generic semantics encoded at the interfaces. They communicate via standard protocols and data formats.

Figure 3.1 demonstrates a basic service-based architecture which embraces three parties: a service provider, a service broker and a service requestor. To be accessible, a service provider describes its services and publishes the specification to a service

[1] A service-based architecture is also known as a service-oriented architecture (SOA).

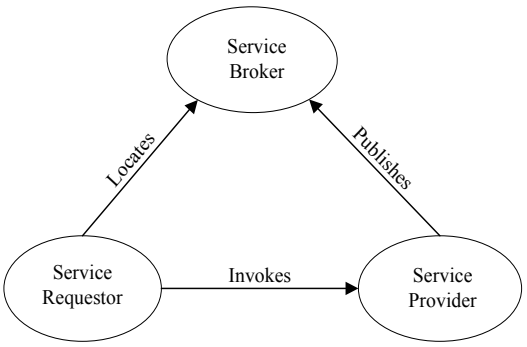

FIGURE 3.1: Service-based Architecture

broker which is an intermediary between the service provider and the service requestor. The service requestor locates a service of interest and determines how to communicate with the service by issuing queries to the service broker. The service broker looks up for the compatible service and sends the published interface description back to the requestor. Upon the receipt of the information of the service required, the service requestor formulates a request according to the specification and poses it to the service provider. Subsequently, the service provider offers the expected response to the service requestor.

Employing a service-based architecture in software development brings many benefits. The facilities of dynamic discovery and binding to a service enable the service providers to run their services at the location they prefer according to the infrastructure and technical support. Multiple service components allow developers to specialise in the task they are experienced in. The independent development realises better parallelism, resulting in rapid delivery of the product. Effecting services as smaller pieces of logic simplifies the location of errors and defects, as well as the modification to accommodate new commands and requirements.

Web services are an example of implementing a service-based architecture. They are at the heart of the service-based architecture because they are built on top of many well-known and platform independent protocols, such as XML, WSDL[2], SOAP[3] and UDDI[4], which fulfill the requirements of the service-based architecture. XML provides a cross-platform approach to data encoding and formatting. WSDL supplies a model and an XML format for describing Web services. SOAP, built on top of XML, defines a way to package XML-based information for exchanging structured and typed

[2]Web Service Description Language
[3]Simple Object Access Protocol
[4]Universal Description, Discovery and Integration

information across system boundaries. RPCs[5] can be encapsulated in SOAP messages, through the SOAP HTTP binding, and dispatched across systems to invoke the target services. UDDI specifies how to publish and discover information about Web services via distributed Web-based information registries.

3.3 Development Traces

This section will document and analyse a cross-section of hypermedia systems that adopt the models illustrated in the previous section, with the exception of systems having a service-based architecture. This is because service-based hypermedia systems have yet to be implemented. As the most popular distributed hypermedia system, the World Wide Web will be presented in the first place. In the following subsections, distributed examples selected from the open hypermedia community are presented which demonstrate how the open hypermedia community has made efforts to overcome the limitations existing in closed hypermedia systems, such as the World Wide Web.

3.3.1 The World Wide Web (client-server)

The World Wide Web (also known as WWW, W3 and the Web) was defined by Berners-Lee (1996) as the universe of global network accessible information. As of today, it has become the most widely used and successful distributed heterogeneous hypermedia system. The Web was developed in 1989 at CERN (European Laboratory for Particle Physics) as a project led by Tim Berners-Lee and was intended to serve as a hypertext system for international cooperation between physicists. Although hypertext systems had been a reality for many years, there were no global systems to facilitate the desired cooperation. In particular, the variety of network information retrieval protocols and workstations with varying display capabilities hindered researchers in achieving the goal.

The Web aimed to manage a widely distributed set of computers running different applications that employed different data formats. To this end, the Web provides a common naming scheme, the Uniform Resource Identifiers (URIs), to point to any document of any kind available via a number of different Internet protocols. For instance, a core network access protocol, the HyperText Transfer Protocol (HTTP), has been developed to support references between information in a hypertext system. The

[5]Remote Procedure Calls

Web also comprises an important document format, the HyperText Markup Language (HTML), which enables the creation of documents that are portable from one platform to another. These three important specifications laid the foundation for the success of the Web.

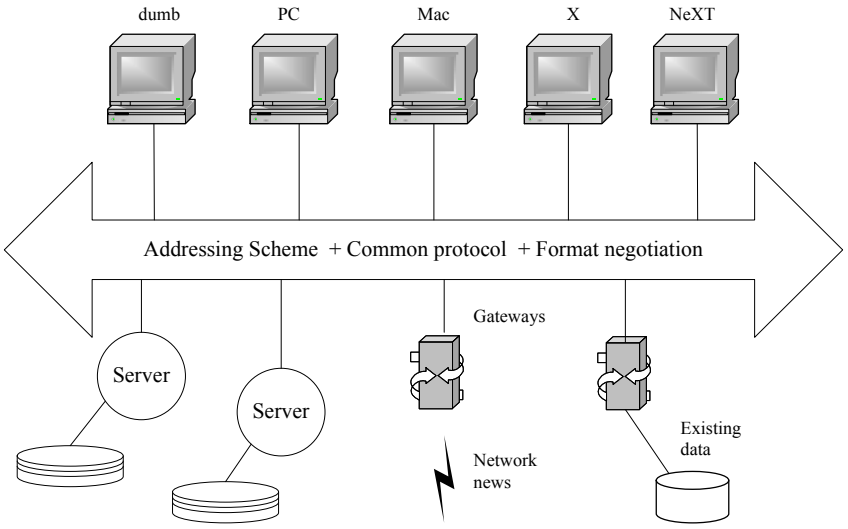

FIGURE 3.2: Web Architecture
(Berners-Lee et al. 1992)

The Web, like many other applications with a global scale, employs a client-server model, see Figure 3.2. The clients, primarily in the form of Web browsers for interactive use, are responsible for collecting requests for documents from users and sending them to Web servers. The servers, upon the receipt of requests, retrieve documents and send back the answers that may be in any other format. During each transaction, the server establishes a connection between the client and itself. The connection can be terminated by either the client or the server, or both.

From the point of view of OHSs, the Web is a closed hypermedia system. In the first place, a link on the Web is bound to a particular object in a source document in the form of mark-up within the content data, and its destination is described through the aid of a URL (Uniform Resource Locator)[6] or a script[7]. The embedded link model makes the movement of data and the editing of links a very convenient procedure. However, it is difficult to maintain the referential integrity of links if the movement of data breaks

[6] A URL is an example of the URI that identifies a resource by means of a representation of its primary access mechanism.

[7] In the context of the Web, script languages are often written to handle forms of input or other services for a website and are processed by either the Web server or the Web browser.

the binding of the links to their associated objects, raising the dangling link problem (Davis 1998). Secondly, data are restricted to be imported into a proprietary format, for example the HTML format. Links on the Web may be created in document formats other than HTML or image formats. The traversal of such links in a document will reach dead ends with no links to follow if the application that displays the document has not been enabled to support hypermedia linking. Moreover, links on the Web can not be applied to data stored in other applications since they are embedded within the content data and belong exclusively to the owner of the document. Finally, as a universe of electronic information, the Web has no full-text search facilities of its own and relies on external search engines. These search engines index much of the Web document's content but lack mechanisms for providing either user's context or the document's context to aid in comprehension.

3.3.2 DeVise Hypermedia/Webvise (client-server/three-tier)

The DeVise Hypermedia (DHM) framework, developed as part of the DeVise project at Aarhus University, Denmark, was an object-oriented environment for developing advanced hypermedia systems (Grønbæk et al. 1993, Grønbæk and Trigg 1994). The designers took the Dexter hypertext reference model (Halasz and Schwartz 1994) as the starting point and turned it into an object-oriented design and prototype implementation.

DHM was based on the client-server architecture, see Figure 3.3. The application layer, which corresponded to the within-component layer in the Dexter model, represented the diverse space of applications and viewers that could be integrated with DHM. The data objects were stored in the hypermedia database (the physical storage layer), or otherwise maintained by the applications and viewers. The communication, runtime and storage layers captured the DHM class hierarchies. The communication layer provided a uniform interface to the applications and viewers in the application layer. The hypermedia services runtime classes defined the generic behaviour of hypermedia systems. The storage classes represented the conceptual schema for data objects in the object-oriented database. The object-oriented database primarily served as a permanent storage for hypermedia data objects. The object distribution mechanisms aimed to facilitate linking between hypermedia data objects stored by different object-oriented databases.

DHM supported bidirectional links with multiple endpoints. Links were stored in a central link database. In contrast to the Dexter model, DHM allowed *dangling links*

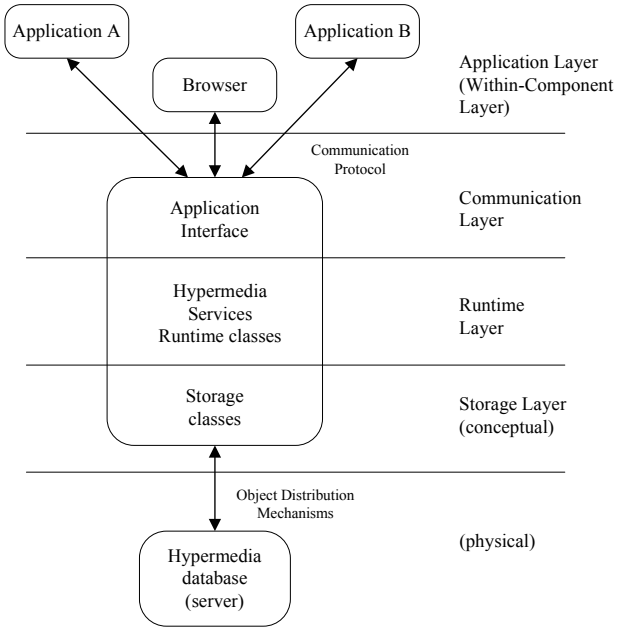

FIGURE 3.3: DHM Architecture
(Grønbæk et al. 1997)

to exist when components[8] were deleted. Options as to dealing with the dangling links when link following occurred, were provided to users. The users might choose to either delete the link or the endpoint, or enable re-linking to another destination. Furthermore, DHM motivated the awareness of link directionality. The designers considered three kinds of directionality of links. Semantic direction revolved around the semantic relationship between components connected by links. Creation direction emphasised the sequence of the creation of link endpoints, whereas traversal direction specified the way a link should be traversed. DHM supported selection of both creation direction and traversal direction by means of an attribute mechanism that recorded direction values. Support for semantic direction implicitly relied on the same attribute mechanism.

DHM tackled two integration related problems which the Dexter model did not address. First, Dexter does not distinguish between components whose contents are managed by the hypermedia and those whose contents are managed by third-party applications. DHM addressed this issue by introducing a component 'wrapper' for applications and their data objects. If stored by the hypermedia system, data objects became part of the content of an atomic component[9]. Otherwise, they would be separately stored and referenced by the content of the component. Therefore, DHM was able to

[8] In the context of the Dexter model, a component is the fundamental entity and basic unit of addressability in the storage layer.

[9] Atomic components are the primitive in the storage layer.

link to documents that were created by third-party applications. Second, Dexter only proposes the use of composite components[10] to model application documents having internal structure, while leaving how to utilise the composite component's structure to model the internal structure of an application document unspecified. DHM allowed composites to directly refer to data objects. Meanwhile, the internal structure of a data object could be modelled by its encapsulating data objects. Therefore, a composite and its nested components could refer to both the enclosing object and its internal structure.

Implicit and asynchronous collaboration modes were supported by the object-oriented database at the physical storage layer which provided transactions, locking and event notification facilities. A transaction in DHM could be of arbitrary length, as called for by Halasz (1988). For concurrency control, read and write locks were available to clients from the object-oriented database server, and a flexible read/write locking protocol which specified a set of rules followed by all transactions when requesting and releasing these locks was also developed. Clients could subscribe to a variety of events on shared hypertext. If any changes to the shared hypertext occurred, a notification would be sent from the object-oriented database server to all clients who had opened the hypertext with read permission and subscribed to notifications about changes.

DHM was extended to Webvise, an open hypermedia service which augmented the Web by providing hypermedia structures such as links, contexts, annotations and guided tours[11]. Hypermedia structures were stored in a hypermedia database and manipulated via Java applets and a proxy server. The Webvise proxy server checked the Webvise server for every document being viewed in the browser and tried to find potential external structures to be compiled into the document (Grønbæk et al. 1999). Microsoft Internet Explorer, Microsoft Word and Microsoft Excel were extended with menus or toolbar extensions to support the integration with Webvise clients via the COM[12] interface. For each application augmented with hypermedia services, the Webvise client designed and implemented an application wrapper responsible for communicating with the integrated application. In addition to linking to/from HTML Web pages, Webvise also supported open hypermedia linking of multimedia contents.

[10] In contrast to atomic components, composite components are those constructed out of other components.

[11] Webvise could be accessed via an ordinary URL.

[12] Component Object Model

3.3.3 Microcosm TNG (peer-to-peer)

Microcosm TNG (The Next Generation) was a framework for an open and extensible distributed hypermedia system. It aimed to facilitate distributed information sharing and organisation and provide extensible distributed services (Goose 1997). Microcosm TNG inherited the core philosophies of Microcosm, whereas its design demonstrated a significant departure from the original Microcosm architecture (see Figure 2.4). Figure 3.4 depicts the architecture of Microcosm TNG which exposed a centralised P2P (see Section 5.3.1) nature. Peers appeared in the form of user sessions in Microcosm TNG. A user session discovered services of interest by interrogating the message router that kept a record of registered service providers both inside and outside (by exchanging registration information between message routers) the domain.

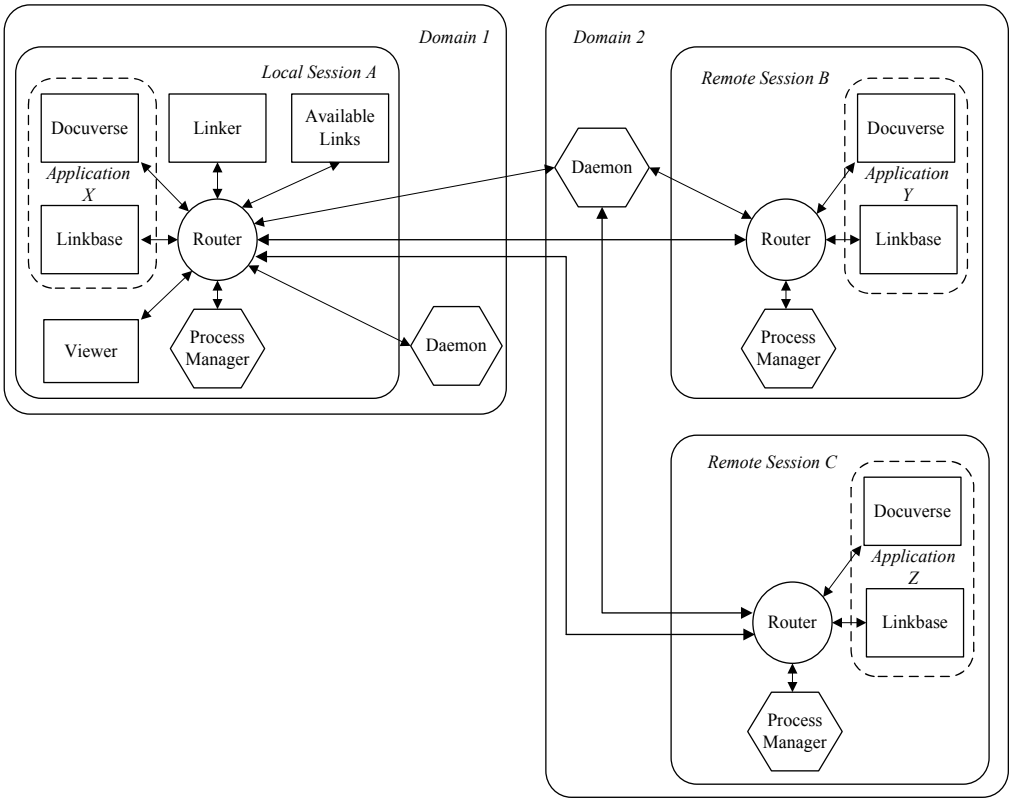

FIGURE 3.4: Microcosm TNG Architecture
(Goose 1997)

When Microcosm executing on a single machine was applied to large scale commercial applications, it yielded a slow performance. The observation therefore entails the distribution of processing load since spreading workload across a number of ma-

chines may lead to a reliable, efficient and flexible system. To divide and distribute the responsibilities of Microcosm involved adapting its two central modules: the Filter Management System (FMS) and the Document Control System (DCS). The FMS was adapted to enable communication with remote instances of Microcosm over the Internet via TCP/IP[13] sockets, thus messages being routed directly to the specific published filters. The DCS was extended to facilitate document retrieval from a remote host by adopting a URL-like format of document identifiers that were uniquely associated with every document under control.

Filters in Microcosm TNG discarded the chain architecture of the ordered and serial nature in the original Microcosm. Each process registered with a message router to gain conversation capability with any other registered process in the system. Messages would not have to encounter all processes in the filter chain but only get involved with those of interest.

A new Heterogenous Communication Model (HCM), utilised as a communication layer, was designed to support the system to function effectively when employed in a heterogeneous and distributed environment. An HCM was characterised by the adoption of a customisable process addressing scheme which covered the user session identifier to which this process belonged, the hypermedia application name to which this process belonged, the name of the Microcosm TNG process, the identifier of the Microcosm TNG process, the document upon which this process operated and the service offered by this process. Message passing between processes on different machines could take place on the basis of a user session, an application, a process, a document, a service or combinations of all.

The message router within a user session was a logical container for all processes registered with the same message router. It served as a mechanism for processes to dynamically advertise and withdraw their services. Moreover, the message router was responsible for coordinating communication between processes that wished to send outgoing messages to other registered service providers. When inter-session communication occurred, a local message router needed to forward all messages to the remote message router in another session in case any process in that session might have interest in them.

Each user session was also accompanied by a process manager which supported distributed process management. The distributed invocation of processes on remote machines was realised through the remote shell (rsh) in Unix.

[13]Transmission Control Protocol/Internet Protocol

A domain denotes a logical set of machines in which the process manager can spawn processes. A domain daemon process was introduced to provide a single point of contact within each domain. Beside, the domain daemon offered a single point of contact to users from other domains. Each message router within a domain is required to register their network address with the domain daemon. Therefore, message routers were freed from pre-defined port numbers for communicating with service providers. Rather, they could dynamically allocate network connections. The HCM enabled processes to act as both a server and a client depending upon the roles they played.

Information regarding hypermedia applications available across different domains was listed by an optional utility called the *app* browser. Once a user selected desirable domains, the *app* browser would contact the chosen domain daemons. The latter, in turn, asked the registered message routers to reply the request by listing all hypermedia applications published from them. The retrieved information would then be presented to the user by the *app* browser.

Although Microcosm TNG exhibits a flexible P2P nature, it is not clear in Microcosm TNG how the remote domain daemons could be located, which is the key element for efficient search in a P2P environment. The design did not address issues such as service (or resource) description and discovery arising in more unexpected and dynamic environments characterised by an ad hoc nature.

3.3.4 HOSS (component-based)

As hypermedia structuring principles had been applied to a broad variety of domains, the hypermedia community realised the necessity of opening the set of structural abstractions supported by an OHS. This gave rise to the component-based open hypermedia system (CB-OHS). A CB-OHS comprises an open set of middleware components with well-defined APIs. Its architecture typically defines three layers: the application layer, the structure model layer and the storage layer. The storage layer (back-end) stores the fundamental structural abstractions and provides them to the structure model layer. The structure model layer (middleware) tailors them to domain specific structural abstractions. The application layer (front-end) consists of various hypermedia enabled applications that edit and display contents and structure from the structural model layer.

Wiil and Nürnberg (1999) divided the evolutionary history of CB-OHSs into two phases: the first generation CB-OHSs and the second generation CB-OHSs. The distinction lies in the component framework they adopt. As one of the first generation CB-

OHSs, HOSS provided its proprietary component framework[14]. HOSS was a structure aware hypermedia operating system prototype developed at the Texas A&M University (Nürnberg et al. 1996). It was advocated in HOSS that the basic structural abstractions of different domains be incorporated into the operating system and therefore issues such as integrity, consistency and semantics locality could be in reach by a structure aware operating system.

There were three basic entities in HOSS: data, structure and behaviour. Data was defined as information associated with the hypermedia system, while structure identified the interrelationship between data. Behaviour was introduced to implement the semantics of structure. HOSS emphasised the separation of these factors, which resulted in more usefulness and flexibility than those of other systems.

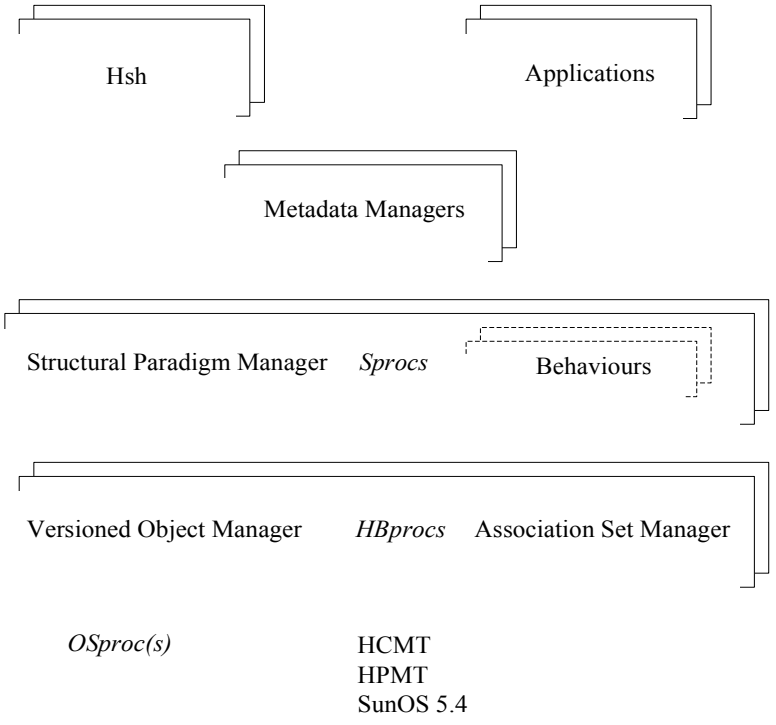

FIGURE 3.5: HOSS Architecture
(Nürnberg et al. 1996)

The architecture of HOSS is shown in Figure 3.5. HOSS provided some toolkits (for example, HCMT[15] and HPMT[16]) for supporting appropriate communication and

[14]As will be mentioned in Section 3.3.6, the second generation CB-OHSs typically adopt existing component technologies and frameworks.
[15]HOSS Communications Model Toolkit
[16]HOSS Process Model Toolkit

process models. These toolkits were constructed on top of SunOS 5.4, thereby utilising facilities provided by the operating system. The HOSS HBprocs comprised the Versioned Object Manager (VOM) and the Association Set Manager (ASM) which implemented data objects and structural services respectively. The VOM was implemented on top of some Storage Manager (SM) outside the hyperbase. The ASM was implemented as a client of the VOM and therefore inherited the VOM's versioning support for its abstract data types. Sprocs (also called structure processors) were clients of the ASM and dealt with different kinds of structure. Metadata Managers provided abstractions to other system processes by resorting to a location services scheme to publish their abstractions and ports. The Hsh (HOSS shell) acted as a command interpreter system which allowed users to manipulate structure from the outside of an application. Applications could take advantage of HOSS once necessary modifications had been made. This typically involved at least an open linking protocol so that the data of the applications could be linked by data from other applications through the Link Services Manager[17]. By replacing the file system in a conventional operating system with a hyperbase, HOSS was enhanced with both data and structure management capabilities and applications could manipulate objects through the hyperbase.

HOSS exhibited some characteristics that could not be found in other operating systems. To begin with, the pre-fetching scheme was modified to rely on the semantic locality, which was based on the awareness of structure rather than that of distance in the logical memory. Furthermore, by threading behaviours in a structure-caching process, communication only occurred inside a process, yielding a gain in efficiency.

3.3.5 HyperDisco (component-based)

HyperDisco (Wiil and Leggett 1997) was a project built on a decade of research on the hypermedia infrastructure at Aalborg University in Denmark started in the late 1990s. Its mission was to work towards innovative hypermedia infrastructures for flexible integration and extension of tools and to use the lessons learnt to serve the future infrastructure for the Internet.

The HyperDisco infrastructure comprised distributed workspaces, tool integrators and participating third-party tools (see Figure 3.6), and was based on a hyperbase management system (HBMS) (see Section 2.3). Tool integrators enabled distributed heterogenous tools (or applications) to participate in hypermedia services, and controlled

[17]The Link Services Manager was one of the Sprocs responsible for structure caching, behaviour loading and inter-application linking.

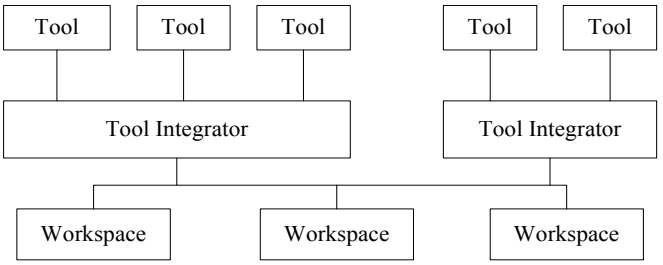

FIGURE 3.6: HyperDisco Architecture
(Wiil and Leggett 1997)

access and operations on workspaces. A workspace was an autonomous HBMS which stored and manipulated a set of multimedia files in addition to providing a wide range of hypermedia services to participating tools. Workspaces could be private, public or belong to a group. A legal user was allowed to create links between documents in different open workspaces and collaborative users could share workspaces. Tool integrators and workspaces respectively implemented two layers of hypermedia functionality in HyperDisco: the integration model layer that supported integration and the data model layer that offered hypermedia storage services for hypermedia objects. The integration model encapsulated tool dependent link services and relied on the data model for essential operations on hypermedia objects.

The data model in HyperDisco was object-oriented and it supported a set of hypermedia object types: anchors, nodes, links and composites (Wiil and Leggett 1996). The Node class, Link class and Composite class were subclasses of the Component class which inherited facilities from its superclasses, such as the Concurrency Control, the Notification Control, the Version Control, the Access Control and the Query & Search classes. The Anchor class was an immediate subclass of the Query & Search class and new subtypes of the Anchor class could be tailored by creating subclasses which inherited related superclasses. Anchors and links were stored separately from documents and would be superimposed on the documents by tools. A link in HyperDisco maintained a number of endpoints with each described by a triple (workspace name, node identifier, anchor identifier). The tool integrator maintained the integrity of a link involving multiple workspaces by ensuring that all workspaces connected by the link were reachable before any operation (creation or deletion) done to it and, in response to the consequence, making a decision about the operation. If the operation involved link following, all workspaces containing the link endpoints would be queried against and as many endpoints as possible would be opened.

HyperDisco differed from Chimera in (at least) the following aspects. Although the concept of workspaces in HyperDisco was analogous to that of hyperwebs in Chimera, workspaces contained not only hypermedia abstractions such as anchors, links and content nodes, but also support for Internet distribution, access control, collaboration and version control. Moreover, HyperDisco used its own naming scheme to allow tool integrators to locate workspaces and dispatch application requests to them, whereas the Chimera's client server contacted the hyperweb manager to retrieve the contact information of the hyperweb server and then directed the end user's request to the hyperweb server for manipulating the hyperweb of interest, thereby avoiding the need for a name service.

3.3.6 Construct (component-based)

Construct, developed at Aalborg University Esbjerg and Aarhus University, was an instantiation of a public domain CB-OHS compliant with OHSWG[18] (de facto) standards, reference models and reference architectures (Wiil and Nürnberg 1999). In contrast to the first generation CB-OHSs which developed its proprietary component framework (e.g. HOSS and HyperDisco), Construct adopted existing component technologies and frameworks and was viewed as one of the second generation CB-OHSs. The three-layered architecture of Construct as depicted in Figure 3.7 comprised an open set of applications, an open set of structure services and an open set of hyperstores.

The hyperstore, which was effected on top of a DBMS, offered persistent storage services. The generic hypermedia service and collaboration services implemented by the hyperstore core were provided to structure services through hyperstore interfaces. Different domains were supported by structure servers which tailored and extended the generic facilities from the hyperstore to domain specific services. In turn, these services were offered to applications through structure server interfaces. Communication between components occurred in many different ways. According to Wiil and Nürnberg (1999), as of November 1998 Construct supported communication via Java RMI[19], TCP/IP and HTTP.

The generic structure in the hyperstore was the basic structural building block. It was tailored to represent all different kinds of domain specific abstractions provided by structure servers, thereby allowing the use of structure across domains. Services pro-

[18] Open Hypermedia Systems Working Group

[19] Remote Method Invocation, which is Java's implementation of RPC for java-object-to-java-object distributed communication.

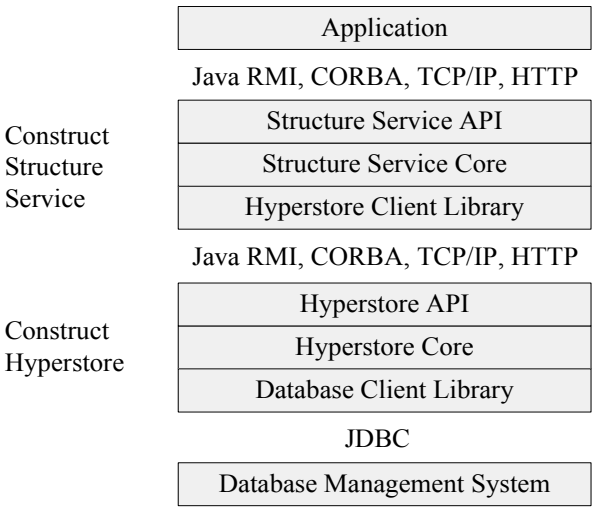

FIGURE 3.7: Construct Architecture
(Wiil and Nürnberg 1999)

vided by the hyperstore were independent of the generic structure abstractions stored, which enabled the hyperstore to provide services with the stored abstractions unaltered.

Construct exhibited enormous extensibility and tailorability. The development and introduction of a new hypermedia domain simply meant defining and implementing a new middleware component to support for abstractions in the new domain. Moreover, by adopting existing component technologies and frameworks, such as Java RMI, interoperability between compliant hypermedia systems was supported, and considerable time and efforts could be saved in both development and maintenance of Construct.

To provide both hypermedia structuring and collaboration facilities, Construct was further developed into a structural computing environment. Structural computing shared its main features with OHSs: separating data and structure, and providing linking functionality to existing desktop applications. In particular, the primacy of structure over data was asserted in structural computing (Nürnberg et al. 1997). Moreover, structural computing aimed to develop environments that support multiple hypermedia domains within a single environment. Wiil et al. (2003) proposed the use of the computer-supported cooperative work (CSCW) technology to augment Construct with collaboration facilities. The hyperstore implemented in the early Construct was extended to a foundation services layer which included structure storage services, concurrency control services, notification control services, access control services and version control services. The Construct structure service shown in Figure 3.7 evolved into the structure services layer which provided a set of structural abstractions for different hypermedia

domains. The infrastructure services layer, including naming and location services, was introduced to enable other services to collaborate in the environment. Furthermore, Construct designers stressed the implications of the awareness service, the session service and the Construct naming service in support of asynchronous and informal collaboration in a conceived application scenario, and implemented these services and related tools to demonstrate their approach.

3.3.7 Callimachus (component-based)

Callimachus was an open distributed hypermedia system with a component-based architecture as shown in Figure 3.8. The objective of the Callimachus system was to provide a framework in which multiple hypermedia domains (e.g. navigational, taxonomic and spatial) co-exist and structure servers of new domains can be developed.

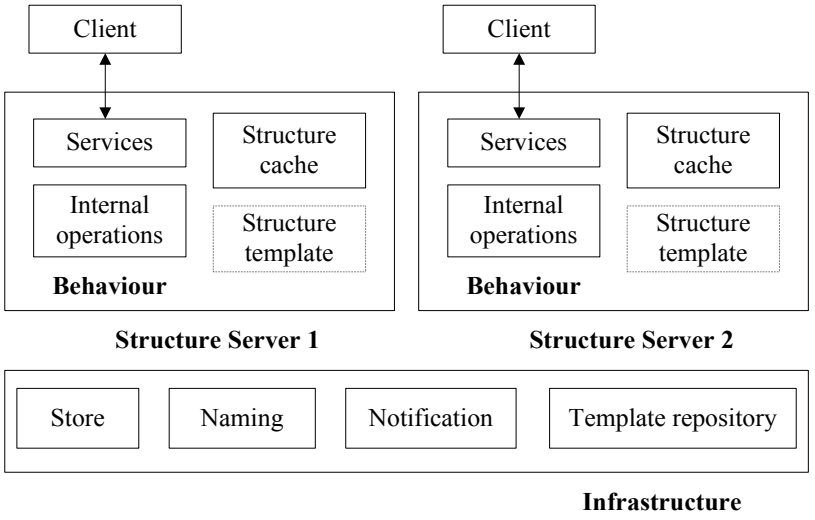

FIGURE 3.8: Callimachus Architecture
(Tzagarakis et al. 2002)

The infrastructure in Callimachus included the fundamental functionality, such as persistent storage, naming and notification. The template repository stored the structure template that defined the structure model upon which structure servers operated. The middle layer comprised an open set of structure servers that delivered domain specific abstractions and services to clients. Each structure server contained a structure template, a structure cache and a set of behaviours which modelled the computational aspect of a domain, a sub-domain or an application. Behaviours were divided into services and internal operations. Services could be accessed by clients using a specific API and pro-

tocol, while internal operations were primarily used by structure servers for consistency purposes. The structure cache was a store for domain specific abstractions of structure servers. Abstractions should be loaded into the structure cache before behaviours performed any operation on them.

The aggregate of the middle layer and the infrastructure originated from the contexts in the early Callimachus (Tzagarakis et al. 1999) which only consisted of distributed contexts and applications requesting hypermedia services. Each context was an HBMS that stored hypermedia abstractions and provided hypermedia services. Contexts provided a data model that described hypermedia objects, such as documents, nodes, links and anchors. Applications obtained hypermedia services from a context by connecting to the context and communicating with it via a common application protocol. To locate hypermedia anchors and services in distributed contexts, Callimachus utilised the Context Name Service (CNS) which generated and managed names of anchors and services.

Each context possessed a CNS that managed the database of bindings between names to attribute values of anchors or services. A CNS name consisted of a handle and a remainder separated by a delimiter. The handle identified the CNS in which the remainder should be resolved. When an application requested operations on an anchor, it needed to provide the anchor name which would be subsequently sent to the CNS of the context to which the application was connected. The anchor name was split into a handle and a remainder by the leftmost delimiter. Looking up the database of bindings of the current CNS identified a new context indicated by the handle. The remainder was sent to the CNS of the derived context to be resolved. This process repeated until the name could not be split further. The remaining name revealed the context in which the anchor of interest was created. The attributes of the anchor were sent back from the target context to the application that requested the information. CNS names were independent of the underlying physical configuration of contexts. Anchor names remained valid even if the related context server[20] was reconfigured, and only the CNS database of bindings in the context in which reconfiguration occurred required an update.

[20]The context server was a process which handled client (application) connections and requests as well as provided hypermedia services.

3.4 Distributed Link Service (DLS)[21]

The first implementation of the Microcosm philosophy on the Web was the DLS (Carr et al. 1995). This was later extended so that link resolution was also distributed around the Web (De Roure et al. 1996). The DLS technology has demonstrated that it is possible to embody the Microcosm philosophy of open linking in the Web to get extra functionality for publishers, authors and readers.

The DLS was a hypermedia service that clients could use to make inquiries against sets of linkbases. It comprised client-side interface tools and the server-side link service CGI scripts. There was a main linkbase for the link server which provided server facilities of the DLS. Additional linkbases existed which allowed the server to offer a range of different sets of links, known as contexts. A personal linkbase in which users might wish to maintain links for their own use was allocated to each user. As described in Section 2.4.3, Microcosm allowed messages to be passed and processed along its filter chain in a sequential way, which was not a satisfactory solution in distributed environments. The DLS abandoned the chain topology of the filters. Multiple messages in the DLS that contained multiple requests to the link service could be processed in parallel, resulting in much more efficiency.

Moving away from its Microcosm roots, the DLS evolved into a proxy server implementation in which the link server was in the form of a link server proxy between the browser and the HTTP server. The reasons are explained as follows. First, users of the original DLS had to learn many new concepts along with the link service. With the proxy server approach, the link service was embedded in the document transport system and links were compiled and inserted into documents when they were delivered to the browser through an adapted Web proxy server. Thus there was no need for an extra client interface for users except a normal Web browser (Carr et al. 1998b). The DLS client might be configured to indicate user preferences for using the link service. A control panel was supplied in the form of an HTML form and the results are sent back to the link server. However, the documents into which links were inserted had to be in a well-understood format, which made the system less open.

Another reason for adopting the proxy server approach lied in the fact that the original DLS had to work with different and constantly changing Web browsers. Furthermore, clients needed to find information such as the current selection and the current document from different applications. The information could be encapsulated as

[21] The DDLS inherits its core concept from the DLS and therefore this section is dedicated to presenting the DLS in its entirety.

an HTTP request and communicated to the Web browser. As Web browsers and inter-application communication evolved, it was difficult to extract such information using the 'standard' software solution. Meanwhile, revisions in operating system implementations also had an impact on the consistent way in which the DLS client worked with the Web environment. The interfaceless proxy server approach required no extra client software for users and thereby able to address the issues.

The simple proxy DLS also brought problems that needed to be tackled. For instance, the generation and processing of a common HTTP request for a document should be as follows. The Web browser built an HTTP request and sent it to the HTTP server. When the document which was returned by the HTTP server passed the link server proxy, related links would be superimposed on the document. The document was then delivered to the Web browser for viewing. During the procedure, link processing and document delivery had to be synchronised with each other before the requested document with newly-inserted links was returned to the user. If link processing cost too much time, the delivery of the requested document to the user's browser would be delayed. Moreover, the newly-inserted links might be viewed as infringing the original document content. These links might also connect to other material, which could be against the author's will. Therefore, the proxy DLS had to solve the problems inherently in its architecture before serving links better.

A possible solution to the above issues could be separating link processing and document delivery and allowing them to take place asynchronously. An HTTP proxy took the place of the link server proxy. The link server was attached to the HTTP proxy with all its components, such as linkbases and link resolvers[22]. The new flow of the HTTP request which was transmitted from the browser to the HTTP server remained the same as before. However, on its return, a copy of the retrieved document would be sent to the link server when it passed the HTTP proxy. As opposed to the previous approach, the links which were relevant to the document would be returned independently of document delivery and displayed in a separate window for recommendation purposes, which made the DLS appear as an advisory service to the user (Carr et al. 1998b).

To be precise, the DLS is more like a dynamic link service than a truly distributed link service. This is because linkbases are maintained on a single server, and the link resolver component is located within the same server. Essentially, linkbase data was regarded just as another document type, so it could be processed and maintained like any other document. Most research on distributed aspects of the DLS initially looked

[22]The link resolver is a component that wraps the function to resolve to links, and is implemented by means of CGI scripts.

for solutions to dealing with decentralised linkbases. For instance, the proxy DLS (Carr et al. 1998a) proposed a DLS network model in which linkbases were scattered around on different servers and more than one link server was introduced. Agents embodied in the proxy enabled queries to be sent from one link server to another. However, no algorithm for query routing was offered, and the details on the way a proxy DLS deals with a query involving linkbases on more than one link server were not clearly defined. In a later paper, De Roure et al. (2000) discussed query processing in a distributed context and investigated off-the-shelf services, such as HTTP, LDAP and Whois++, as candidate technologies for distributed linkbases. It was demonstrated in their study that directory services could be very useful within link service infrastructures.

3.5 Summary

The main objective of this chapter is to serve as an introduction to distributed hypermedia systems, a field that embraces fundamental notions and philosophies in terms of architecture by which this work is inspired. To this end, this chapter described the major models of distributed computing and detailed their architectural features. A cross-section of systems were cited and examined to help understand the various choices of software architecture made in the design of distributed hypermedia systems. In contrast to others, distributed hypermedia systems that adopt a service-based architecture have yet to be implemented.

Studying the distributed hypermedia systems presented in this chapter shows that resources or services can be located in these systems by either posing a query to a central authority, or typing a URL, which is known through other means, pointing to the resource or service. However, none of these systems has provided an approach to resource or service discovery in a more dynamic and ad hoc environment without a central authority. This work aims to enrich research on resource discovery in distributed hypermedia systems by addressing the issue in the unstructured P2P DDLS in which the central control is absent, see Chapter 6.

The next chapter will present the requirements analysis and design of a centralised P2P DDLS which draws on the ideas from both the open hypermedia and P2P computing research areas.

Chapter 4

Requirements Analysis and Design of the DDLS

4.1 Introduction

This chapter begins with an overview of extending the original DLS into a truly *distributed* dynamic link service - the DDLS (Distributed Dynamic Link Service). It then centers around the work conducted in the design of the DDLS which is based on a centralised P2P model that encapsulates a central service directory. Several aspects of software engineering have been involved, ranging from analysis, design and prototyping, to evaluation, with the design of the software architecture being the primary concern. For explanatory purposes, this chapter presents two prototypes that adopt a client-server architecture - a common architecture shared by many OHSs (see Chapter 2 and Chapter 3), and a centralised P2P architecture, respectively. Finally, an evaluation which consists of an operational analysis and feature comparison of both prototypes is presented in detail.

4.2 Overview

It was identified in Section 3.4 that the original DLS was a dynamic link service rather than a truly distributed link service. Linkbases, specified by users as a context for link resolution and retrieval purposes, could be dynamically added or removed from the process of link resolution and retrieval. All linkbases were maintained on a single server, and the link resolver which provided hypermedia link services was located within the

same server. The original DLS did not exhibit any distributed feature in terms of either linkbases or link services.

Decentralising linkbases and link services is driven by the phenomenon that the distinction between the role of a link service provider (a server) and that of a link service requestor (a client) increasingly blurs. In a collaborative environment for example, a single program may be required to act as either a server, or a client, or both. This requirement cannot be fully satisfied by a client-server link service which may use a central link server to manipulate linkbases distributed across the system, or use a central repository to maintain linkbases while deploying link servers across multiple hosts.

Decentralising linkbases and link services is enabled by P2P computing (see Section 5.2). Distributed systems based on the P2P paradigm are characterised by the equal capability of participating nodes and the decentralisation of control, and such a paradigm can be used to support a distributed OHS for collaboration in various settings. A P2P system, in terms of the software architecture, falls into one of the following main categories: the centralised P2P, the unstructured P2P and the structured P2P. The main feature that makes the centralised P2P system different from the rest is its use of a central service directory for resource publishing and discovery. In structured P2P systems, the network topology and the placement of data objects are precisely determined, whereas in unstructured P2P systems they are not. More details can be found in Chapter 5.

The centralised P2P model is closest to the traditional client-server model in the sense that both maintain some form of centralisation[1]. This centralisation is crucial in reducing the complexity of designing and implementing a P2P system. For instance, in a centralised P2P system service/resource discovery involves posing a query to the central authority for locating service/resource providers, whereas in a structured or unstructured P2P system, carrying out the same task requires more complex search mechanisms. More importantly, if the collaborative environment is well-arranged, for instance the presence of a central service directory is allowed, the centralised P2P model will suffice for supporting collaboration. This work therefore starts from designing and exploring the centralised P2P DDLS. For explanatory purposes, two DDLS prototypes based on a client-server architecture and a centralised P2P architecture respectively are described later in this chapter.

The overall aim of this work, as stated in Section 1.3, is to investigate how the open hypermedia approach can be enhanced by P2P technologies to support collaboration in

[1]However, the former requires that the functionality of both a provider and a requestor should be implemented in a single program, while the latter assigns them to different programs.

distributed environments. Such an aim implies that this work also needs to address issues that would arise from environments of an ad hoc nature. An ad hoc environment typically excludes the presence of any central service directory, and together with the specific requirements for the DDLS applicable to such an environment, it invalidates the use of a structured P2P model (see Section 5.4.2.3). Adopting an unstructured P2P architecture poses new challenges to this work, with resource discovery being the most crucial and complex issue to tackle. This work will later investigate the approach to implement the unstructured P2P DDLS and develop mechanisms to enhance its performance of supporting collaboration in ad hoc environments (see Chapter 6 and Chapter 7 for details).

4.3 Requirements Analysis

The DDLS aims to facilitate resource sharing-based collaboration between a community of people with similar knowledge background. It is a tool that empowers any of its users to take advantage of knowledge of peers to assist his/her own activities. This is achieved by discovering related documents from peers and providing them to the user who requests the assistance of peers' knowledge. The peers' knowledge is implicitly conveyed by the way documents are categorised and the information about documents is annotated.

The intended users of the DDLS are professionals in an organisation who rely on computers to deal with their daily work. The DDLS users should be able to use the link service, after suitable training, to store, organise and manipulate links referring to the documents of interest.

The primary requirements for the DDLS are summarised as follows:

- The DDLS shall support users to store links that refer to documents of interest and to maintain them in linkbases.
- Links shall be either manually populated during users' browsing activities, or automatically harvested from other sources.
- The DDLS shall have no restrictions on the content of the documents that are involved in the linkbases - documents can be related to either a specific domain or very diverse domains.

- Users shall be in complete control of their linkbases. For instance, they can either organise and manipulate their own linkbases at will, or place access restrictions on the linkbases.
- When a user decides to obtain documents of interest from peers, all linkbases that have been declared as public across the system shall be visible and accessible to the user.
- The DDLS users shall not be aware of the operational difference between access to public linkbases and access to their own linkbases, with the exception of that caused by the delay of inter-network transmission.
- Every user shall have access to the public linkbases of others if permitted. The operations he/she can conduct on the linkbases, however, include read and search only.
- The DDLS search mechanism shall support document discovery among multiple sources which are decentralised. A search shall be conducted based on various criteria.

4.4 Design

The analysis in the previous section has established an understanding of the requirements for the DDLS from all perspectives. These requirements are to be satisfied by tackling three main issues identified in this section: resource description, organisation and operations, user interface and system functionality, and architecture.

4.4.1 Resource Description, Organisation and Operations

Linkbases are the repositories in which links are stored, organised and manipulated. They are the most essential resources of the DDLS, as they are in the DLS, since the main objective of a link service is to serve links rather than anything else. For each user, different 'context' linkbases should be available for them to select from. By choosing different contextual linkbases, users can specify the link set for their current information needs. Linkbases should be represented in a manner that facilitates associated organisation and operations. To this end, the XML model and syntax were chosen to express linkbases in the DDLS.

More than just a mark-up language such as HTML, XML is a meta-language that can be used for defining new mark-up languages. Instead of replacing HTML, XML is designed to complement it. HTML is currently only used for formatting and displaying data, whereas XML is employed to represent the contextual meaning of the data. XML has a number of advantages over HTML. Firstly, it separates content from presentation of the document. If the content is updated, the document can still be consistently presented. HTML is unable to provide this reusability. Although the export of a document in HTML can be done automatically, it is vulnerable to the change of the data content of the document. Secondly, by specifying different presentation styles, with XML one can give a document a different look upon requests when displaying the content, while HTML usually gives one view of the data. Furthermore, HTML is not extensible so application developers can not create their own tags for specific circumstances, whereas in XML this is easily achieved since application-specific tags can be customised to meet new requirements. Finally, for the description of data in a portable format, XML ensures platform independence and interoperability between hosts.

For instance, in a DDLS linkbase, a link for a particular user as will be mentioned later in this section has the format as in Figure 4.1. Each link is described as a triple (description, source and destination endpoints) and both the source and destination endpoints are further described through a field <data> that contains a field <url> indicating the source and destination anchors of the link.

```
<linkbase>
    <link id="001">
        <description>acamedic-related.research</description>
        <endpoint direction="source">
            <data>
                <url>http://www.rg.cs.university.ac.uk/projects/</url>
            </data>
        </endpoint>
        <endpoint direction="destination">
            <data>
                <url>http://www.cs.university.ac.uk/projects/GIANT/</url>
            </data>
        </endpoint>
    </link>
</linkbase>
```

FIGURE 4.1: An Example of Using the XML model and Syntax to Represent the DDLS Linkbase

Each link entry in the linkbase may potentially have more than one source or destination anchor, which provides flexibility in dealing with all kinds of links. This feature essentially provides a useful manipulation of generic links (see Section 2.4.3). Links

can be searched by their link id (identity), description, source anchor or destination anchor.

Linkbases are usually structured in a hierarchical way to facilitate the organisation of information into categories. A tool that enables users to organise linkbases hierarchically in response to their primary content, is provided in the DDLS. In the hierarchy, the intermediate nodes are analogous to directory listings in the conventional file system and leaf nodes represent the real linkbases which contain links and the associated information about the links. For example, if a user is a member of academic staff at a university, a linkbase can be specified for his/her academic related material and another linkbase for hobbies. In the academic related context, the user may have different linkbases for research, teaching and publication, respectively, similarly for the hobby linkbase.

The DDLS currently support three primitive linkbase types:

- *Private linkbases*: linkbases that belong to their owners (as well as their creators) and are inaccessible to others.
- *Public linkbases*: linkbases that belong to their owners who grant permissions to all the other users for accessing these linkbases.
- *Personal linkbases*: an aggregation of both private and public linkbases of the same owner.

Users have all permissible rights over their personal linkbases, such as creation, deletion and update of links in linkbases, and they grant access permissions for operations on their linkbases to other users upon request. These operations typically include only search or retrieval of a specific or set of links. To increase the interoperability of linkbases among different users, all linkbases in the DDLS may need to conform to the same specification of organisation.

4.4.2 User Interface and System Functionality

Figure 4.2 and Figure 4.3 show the graphical user interface of the DDLS, through which the primary hypermedia functionality available in the DDLS can be accessed and the system can be configured. Two tabs, ‘Link Service’ and ‘Linkbase Config’, were designed to serve the purposes, respectively.

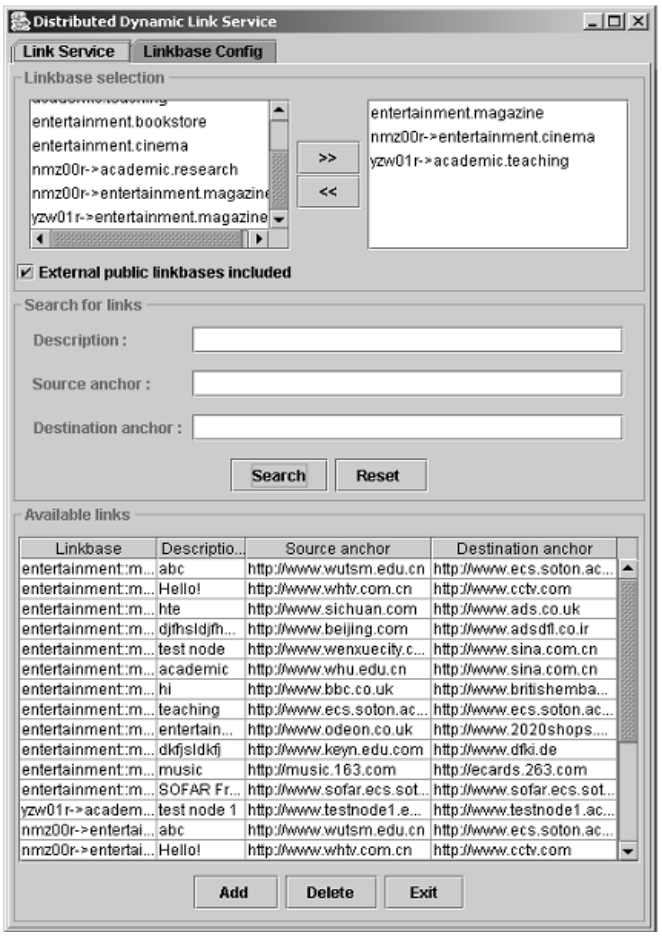

FIGURE 4.2: Screenshot of the DDLS User Interface - the 'Link Service' Tab

Both navigational facilities (link traversal) and authoring facilities (link creation, update and deletion) are provided through the 'Link Service' tab of the user interface, see Figure 4.2. Retrieving documents of interest is transformed into retrieving the links that refer to the documents. In order to search for links, a user first selects from the 'Linkbase selection' list the linkbases that will be involved. Ticking the check box titled 'External public linkbases included' indicates the search will be carried out in an extended scope. The DDLS responds to such an action by locating all the available public linkbases and presenting their description in the list for the user to choose from. Other attributes that can be specified in the search criteria include the description[2], source anchor or destination anchor of target links. The 'Search for links' facility will formulate a query for the search request. The query expression is typically represented

[2]The description of a link refers to an abstraction of the primary content of the document the link refers to, and is not limited to the user's selection based on which the link service will typically initiate an action such as link following.

by conjunctive operations on terms specified in the search criteria[3]. An empty attribute indicates that any link matches the search criteria in terms of the attribute. After a query is formulated, it is submitted to the associated link server. The latter, in turn, retrieves links from related linkbases and displays the result in the 'Available links' table. Users may follow any link in the result and the DDLS will dispatch the related document and display it in a browser.

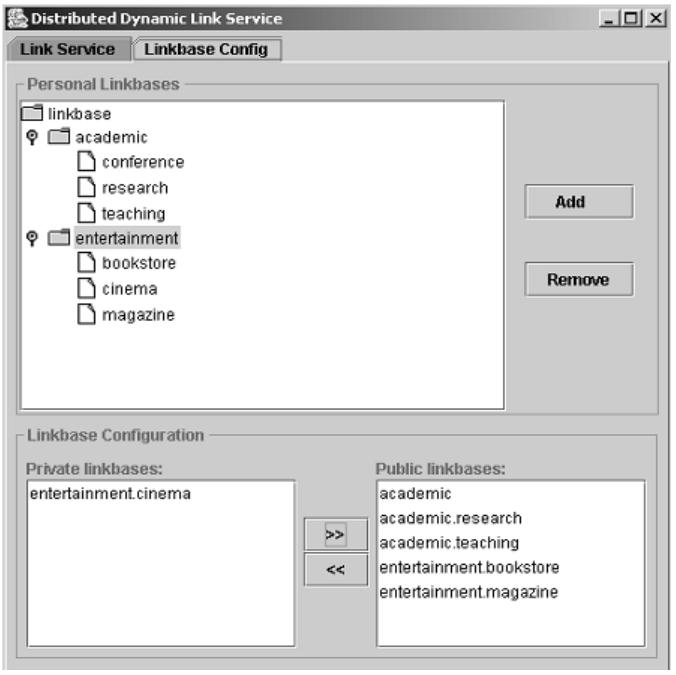

FIGURE 4.3: Screenshot of the DDLS User Interface - the 'Linkbase Config' Tab

A dialogue box is presented for adding new links to personal linkbases. When prompted, a user provides information about the textual description, the source anchor and the destination anchor of the link being added. The user can store the new link in any existing linkbase according to the textual description of the link, or even create a new linkbase for it. As in the DLS, the DDLS can automatically create a linkbase from scratch by extracting sets of keywords from documents and turning them into generic links so as to provide hypermedia functionality with minimal effort. Users can update and remove links only from their personal linkbases.

DDLS users can configure the system, for instance, attaching or detaching personal linkbases in response to their link service requirements, see the 'Linkbase Config' tab in Figure 4.3. Linkbases are created with specified titles which capture the primary

[3]Support for disjunctive queries and others could also be implemented.

content of documents their encompassing links refer to and inserted to certain points of the hierarchy in the 'Personal Linkbases' pane, according to users' personal views. The removal of a linkbase will also lead to all links it maintains being discarded.

The accessibility of personal linkbases, which indicates whether the linkbases are public or private, can also be configured by using the 'Linkbase Configuration' facility the DDLS provides. Once a linkbase becomes public, it will be visible and accessible to all users. Users are then able to select specific linkbases of interest from the 'Linkbase selection' list on the 'Link Service' tab to personalise their query context.

4.4.3 Architecture

The centralised P2P architecture of the DDLS is illustrated in Figure 4.4. Each client, or peer, has an associated link server, called LinkServerAgent. The LinkServerAgent registers with the RegistryAgent, a central service directory not shown in Figure 4.4, and expresses its capability for handling specfic messages. The main responsiblity of the LinkServerAgent is to serve link service requests from peers. The UserAgent[4] captures a link service request, wraps it in an HTTP request, and forwards it to the peer's LinkServerAgent. The LinkServerAgent queries against peer's personal linkbases and also forwards queries involving public linkbases of others to related LinkServerAgents. A peer locates the LinkServerAgents belonging to others through the RegistryAgent. Furthermore, a link following request will be forwarded by the LinkServerAgent to an HTTP proxy which is responsible for sending the request to an HTTP server to fetch the document of interest.

4.4.4 Prototypes

The two DDLS prototypes described in this chapter are based on the client-server and the centralised P2P architectures (see Figure 4.5 and Figure 4.4). Both were implemented using SoFAR[5] (Moreau et al. 2000), a framework enabling rapid prototyping. Prior to proceeding further, an overview of SoFAR is given below.

[4]The UserAgent is a component that simulates the functionality and presence of users. It is responsible for interacting with and configuring the system.

[5]Southampton Framework for Agent Research

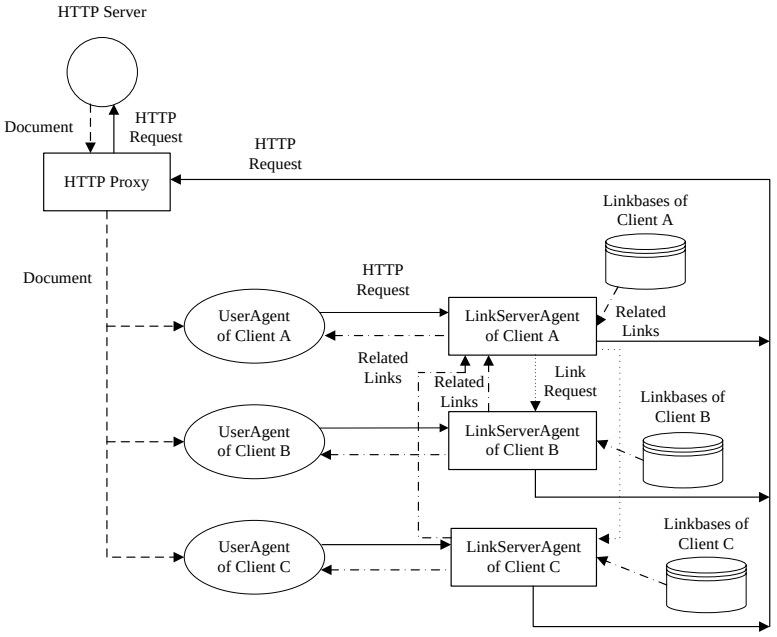

FIGURE 4.4: Centralised P2P Model for the DDLS

4.4.4.1 The Enabling Framework

SoFAR is a Java-based generic tool developed to investigate distributed information management techniques. One of the major characteristics of Java which make it distinct from other programming languages is that applications written using class libraries are guaranteed to be portable across different platforms.

Agents The prime entities in SoFAR are referred to as agents (Jennings et al. 1998) acting on behalf of other entity (or entities). Agents are hosted by platforms. The platform is a Java Virtual Machine (JVM) and it can run a RMI registry listening on a specified port. The responsibility of a platform includes starting up agents and conducting security check on agents' permission to run on the intended location. Agents advertise their services to a built-in matchmaker agent, the RegistryAgent, and express their capabilities for handling specific messages. By matching an agent's need for services with other agents' ability to offer these services, the RegistryAgent can recommend a suitable service provider to the agent requesting the service. The RegistryAgent registers itself with the RMI registry under a well-known URL, which makes it convenient for other agents to locate the RegistryAgent by communicating with the RMI registry.

Communication Communication in SoFAR is based on performatives. The message intent is expressed using one of nine performatives defined in SoFAR and its content is in the form of an ontology[6]. By exchanging messages, agents are able to communicate information about their perception of the surrounding environment with regard to the problem domain. Agent communication is carried out over a virtual link defined by a startpoint and an endpoint. The startpoint is one end of the communication link where the messages are sent from and the endpoint is the other end that extracts the messages from the communication link and forwards them onto the agent. SoFAR offers synchronous and multicast communication via RMI.

4.4.4.2 Prototype Systems

Client-server DDLS The client-server DDLS, see Figure 4.5, possesses only a single LinkServerAgent to serve all clients in the system. All personal linkbases are stored and maintained locally by the LinkServerAgent. A link service request is wrapped in an HTTP request and subsequently sent from a UserAgent to the LinkServerAgent. The LinkServerAgent queries against the linkbases specified in the request and returns the result, if any, to the UserAgent. If the request is regarding link following, it will be further directed to an HTTP proxy which, in turn, forwards the request to an HTTP server from which the requested document is fetched.

Centralised P2P DDLS The centralised P2P DDLS is shown in Figure 4.4 and the related description can be referred to in Section 4.4.3.

4.4.4.3 Component Interaction

From the perspective of the framework, agents in SoFAR are regarded as communication objects interacting with one another by means of a set of predefined communication acts, or performatives. All performatives are implemented by binary methods and these methods are wrapped in messages passed between agents when inter-agent communication occurs. The explanations for performatives that will appear in Figure 4.6 and Figure 4.7 are outlined as follows.

- *Query_ref*: a synchronous exhaustive search for all terms that match the predicate being passed as an argument.

[6] An ontology is a formal specification that helps promote interoperability between agents. It consists of a domain specific vocabulary and a set of assumptions about the intended meaning of words. Assertions, queries and requests against the specific domain are composed by the vocabulary and exchanged among agents.

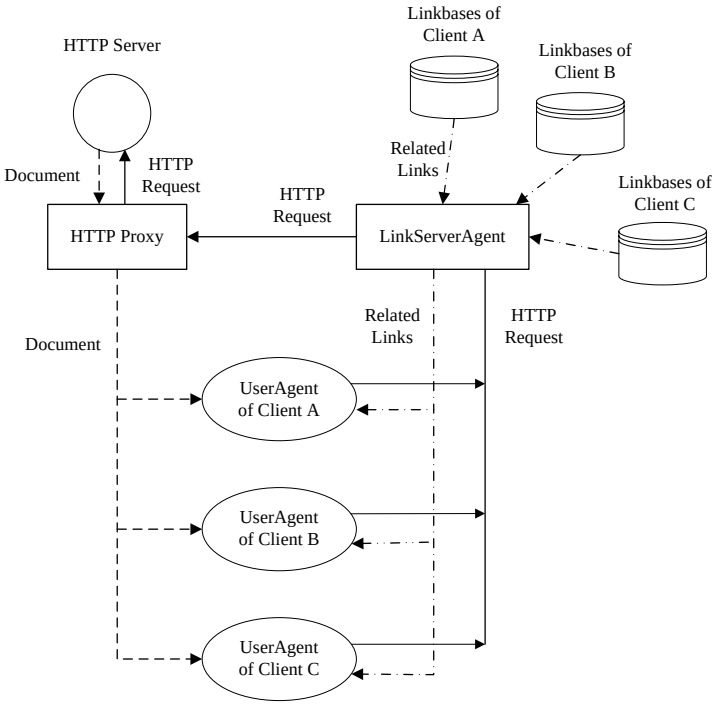

FIGURE 4.5: Client-Server Model for the DDLS

- *Request*: a performative used by an agent to request other agents to carry out some tasks on its behalf.

Below is an example of the usage of *Query_ref* and *Request*:

```
a = this.findAgentTermsForQueryRef (new privateOrPublic ());
privateOrPublic pob = new privateOrPublic ();
pob.initNameOfLinkbase (nameOfCheckedLinkbase);
results = a[0].startpoint ().query_ref (pob, null);
b = this.findAgentTermsForRequest (new createLinkbase ());
createLinkbase cl = new createLinkbase ();
cl.initNameOfLinkbase (nameOfLinkbase);
b[0].startpoint ().request (cl, null).
```

In the example above, an agent initially queries the RegistryAgent using findAgent-TermsForQueryRef (new privateOrPublic ()) to find the AgentTerms of all agents that are able to handle the predicate 'privateOrPublic'. An AgentTerm is a term representing

an agent[7]. The agent then asks one of the agents located (the first one in this case) to check whether the linkbase denoted by 'nameOfCheckedLinkbase' is private or public. Similarly, if an agent anticipates creating a new linkbase with a name indicated by 'nameOfLinkbase', it needs to query the RegistryAgent using findAgentTermsForRequest (new createLinkbase ()) to discover the AgentTerms of all agents that are able to handle the predicate 'createLinkbase'. If there is at least one agent satisfying the requirement, the agent can request any of the located agents to create the new linkbase.

Client-Server DDLS Figure 4.6 depicts the component interaction in the client-server DDLS in which each client is accompanied by a UserAgent. The single LinkServerAgent provides all the functionality that a link service should offer, for example, adding and deleting links from a specified linkbase (*Request*), searching for links according to some criteria (*Query_ref*), etc. The LinkServerAgent fulfils its tasks on its own without help from any other agent.

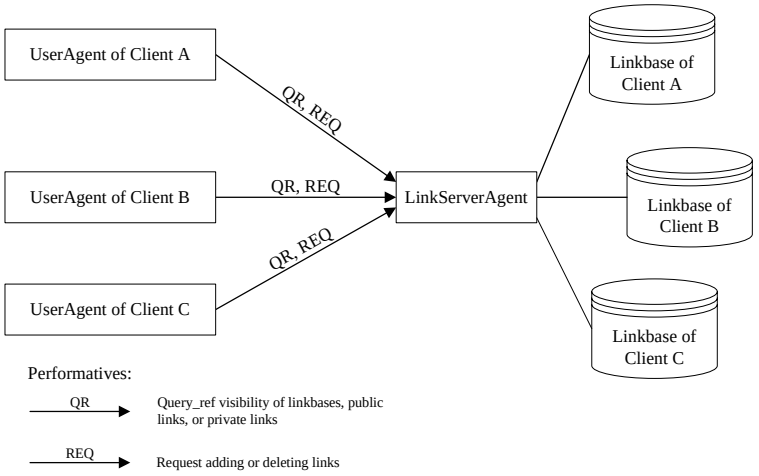

FIGURE 4.6: Component Interaction in the Client-Server DDLS

Centralised P2P DDLS The interaction between components in the centralised P2P DDLS is shown in Figure 4.7. In the centralised P2P DDLS, a pair of agents consisting of the UserAgent and the LinkServerAgent are supplied to each client. The pair reside on a single host and provide the same link service functionality as those on any other host in the system. Unlike in the client-server DDLS, a user who requests links from the linkbases of others in the centralised P2P DDLS needs to inform his/her UserAgent of the request which will in turn queries the associated LinkServerAgents. The UserAgent acquires information about all the other available LinkServerAgents

[7] An AgentTerm comprises the following fields: object ID and runtime information about the agent that further embodies fields such as the user of the agent, the host on which the agent is running, the type of the agent and the time the agent is started.

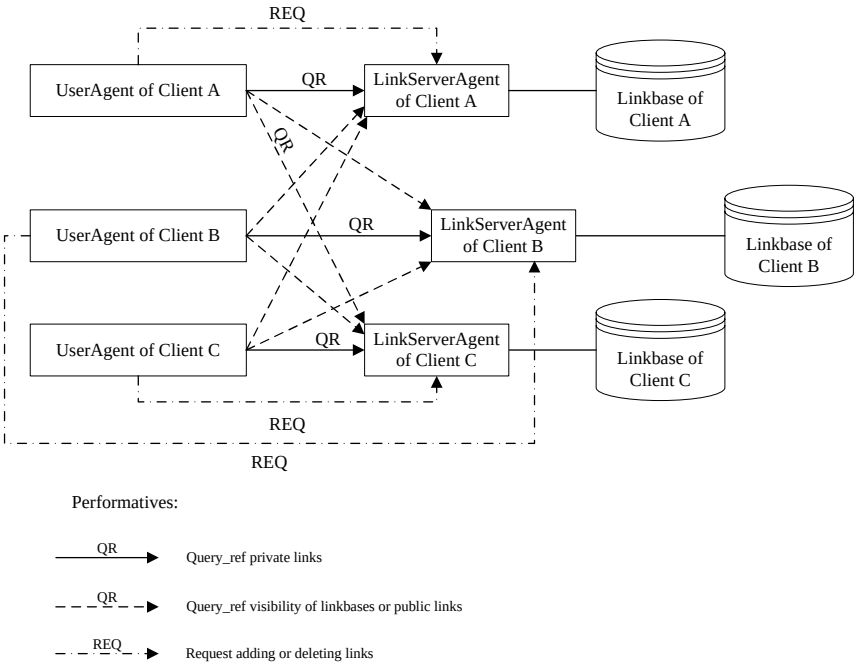

FIGURE 4.7: Component Interaction in the Centralised P2P DDLS

by submitting a query (*Query_ref*) to the RegistryAgent. The latter queries all registered capabilities, gets the information about target LinkServerAgents, and imports the description of public linkbases from those LinkServerAgents. All LinkServerAgents cooperate within this scenario.

4.5 Evaluation

It was declared in Section 4.2 that the centralised P2P DDLS aimed at supporting resource sharing-based collaboration among a community of people in a well-arranged environment, together with several requirements for the DDLS identified in Section 4.3. The design of the DDLS was therefore undertaken based on such an understanding. The prototype developed can demonstrate, to some extent, that the DDLS which adopts a centralised P2P architecture has satisfied all the requirements. This section continues to reveal what happens behind the scene that gives the centralised P2P DDLS its strengths and weaknesses. This is achieved by abstracting the operations that the client-server DDLS and the centralised P2P DDLS share in common, and conducting an analysis and comparison of the architecture of both.

4.5.1 Operational Analysis

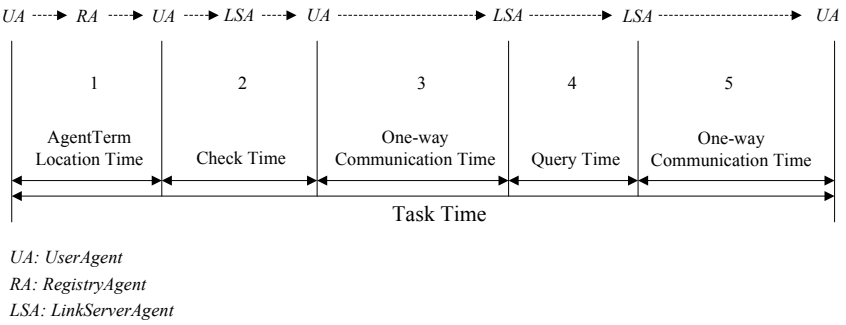

FIGURE 4.8: Composition of Task Time for a Link Retrieval Request

To facilitate understanding the typical operational procedure in the DDLS, a request for link retrieval is examined. In response to the operations that need to be conducted to satisfy the request, the task time, the AgentTerm location time, the check time, the query time and the one-way communication time are defined as follows.

- The task time is the time recorded at a link service requestor (or a UserAgent) to send the request for links in specified linkbases, and receive either one message with all the links meeting the specified requirement or another indicating no links has been found.
- The AgentTerm location time is recorded from the point at which a query regarding link service providers is sent from a UserAgent to the RegistryAgent until the time a reply is received by the UserAgent which contains either a list of available link service providers or an empty list indicating the unavailability of any link service provider.
- The time spent in checking the accessibility of specified linkbases is called the check time, and is calculated from the point at which the link service requestor sends the check request, until the time it receives the result from the related LinkServerAgent.
- For each incoming query, a LinkServerAgent searches its linkbases and retrieves links. The elapsed time, measured on the node where the LinkServerAgent resides, is the query time.

- The one-way communication time refers to the period between a link retrieval request travelling from a UserAgent to a LinkServerAgent that satisfies the request, or vice versa.

Figure 4.8 shows the composition of the task time for a link retrieval request. The agents involved in the time fragments are sequentially labelled in italics.

The client-server DDLS and the centralised P2P DDLS may incur different AgentTerm location time, see phase 1 in Figure 4.8, since they lead to a different number of agents being registered at the RegistryAgent. With a linear search mechanism, the time to locate an AgentTerm is $\Theta(n)$, given n is the number of all AgentTerms that are being searched. However, this difference in the AgentTerm location time would vanish if the search mechanism supports the location of any entity to be completed within the time that is independent of the number of the entities being searched. For instance, a search based on the hash table may lead to the AgentTerm location time scaling as $\Theta(1)$ in both DDLSs.

The check time is specifically defined for all link service requests regarding public linkbases, whereas requests involving private linkbases have no such needs. The associated phase (2) relates to a request sent from a UserAgent to a LinkServerAgent which carries out the accessibility check and returns the result back to the UserAgent. The check time can be the same in both the client-server DDLS and the centralised P2P DDLS with respect to the same link service request, unless both the UserAgent and the LinkServerAgent reside on the same node in the client-server DDLS[8].

The one-way communication time is the primary metric which demonstrates the difference between the client-server and the centralised P2P DDLSs due to their inherent architectures. The difference can be illustrated by a link service request regarding retrieval of links in private linkbases. In the client-server DDLS, the retrieval of private links involves a UserAgent interrogating a LinkServerAgent on a different node, whereas the same task occurring in the centralised P2P DDLS relates to the UserAgent querying the LinkServerAgent residing locally. The one-way communication time in the client-server DDLS is subject to the latency in inter-network communication, because agents reside on multiple nodes and their communication over the network unavoidably suffers from the latency that exists in the network.

[8]In the client-server DDLS, the LinkServerAgent can be practically hosted by any node. Therefore, a node hosting a LinkServerAgent can also be hosting a UserAgent, which may lead to the accessibility check occurring locally. However, the same check in the centralised P2P DDLS always involves a LinkServerAgent and a UserAgent on different nodes. Specifying that the UserAgent and the LinkServerAgent reside on different nodes in the client-server DDLS is to assure that the comparison between both DDLSs is carried out under the identical circumstance.

The client-server and the centralised P2P DDLSs lead to the same query time, provided in both cases the nodes on which the LinkServerAgents reside possess the same computing power.

4.5.2 Feature Comparison between the DDLSs with Different Architectures

Though prototypes were implemented based on two different underlying models, an effort was made to keep them functionally equal in order to achieve a fair comparison. Table 4.1 summarises the comparison in which the advantages of the centralised P2P DDLS over the client-server DDLS are labelled with $\surd$.

Feature	**Client-Server DDLS**	**Centralised P2P DDLS**
Linkbase organisation $\surd$	fixed	flexible
Conditions of link service delivery	lenient	stringent
Degree of information freshness $\surd$	low	high
Overhead of link service delivery	little	much
User's dependence on a single link server $\surd$	much	little
Removal of the single point of failure $\surd$	no	yes
Scalability$\surd$	low	high

TABLE 4.1: Feature Comparison between the Client-Server and the Centralised P2P DDLSs

- **Linkbase organisation** The central LinkServerAgent in the client-server DDLS requires the same organisational structure for all personal linkbases of different owners, or a schema that depicts individual structures should be available and accessible to the link server (i.e. the LinkServerAgent) otherwise. In contrast, the centralised P2P DDLS allows each link server to organise its local linkbases in whatever way it prefers because the link server is the only party that maintains and manipulates a peer's personal linkbases directly.

- **Conditions of link service delivery** The client-server DDLS does not require all registered users to be online when serving link requests, because all public linkbases are maintained on a single node and access to the linkbases does not necessarily entail the presence of their owners. This is an advantage over the centralised P2P DDLS in which the public linkbases are available provided their owners are online.

- **Information freshness for real-time search** Users may not be able to publish the latest update of linkbase information to other users in a timely manner in the client-server DDLS. This may be a consequence of the disconnection of users from the link server. However, the centralised P2P DDLS enables modification to be reflected immediately once a peer updates the information about its linkbases. For example, if a peer changes the accessibility property of its linkbases, any subsequent search associated with the linkbases will be conducted in response to the update.

- **Overhead of link service delivery** The centralised P2P DDLS brings some extra operations for the link service request compared to the client-server DDLS. For example, a search for links from public linkbases in the centralised P2P DDLS involves a UserAgent interrogating the associated LinkServerAgents to obtain information about all public linkbases and to search for links in the linkbases. The same process occurring in the client-server DDLS typically relates to a UserAgent querying against a single LinkServerAgent, which yields less operations and messages. Moreover, the centralised P2P DDLS entails more agents in operation. Let u be the number of users, $P(u)$ and $C(u)$ represent the number of agents registered at the RegistryAgent in the centralised P2P DDLS and the client-server DDLS, respectively. The following functions

  $$P(u) = 2u + 1,\ C(u) = u + 1$$

  reveal that the number of agents in the centralised P2P DDLS increases two times as fast as that in the client-server DDLS as the number of users increases. Although the difference in location time can be eliminated through certain search mechanisms, the cost and resource consumption of extra agents introduced by the centralised P2P DDLS remain.

- **Users' dependence on a single link server** Users in the client-server DDLS rely more on the link server than those in the centralised P2P DDLS. Without the link server, the client-server DDLS can not enable users to access their own private linkbases since all the functionality of link services can only be accessed through the link server. In contrast, the centralised P2P DDLS alleviates users' dependence on a single link server by allowing each user to have his/her own link service provider. Therefore, users always have access to their own linkbases as well as to the public linkbases whose owners are online.

- **Single point of failure** The centralised P2P DDLS decreases the possibility of the single point of failure through distributing linkbases and link services across the

system and directly transferring required links from the provider to the requestor, whereas the client-server DDLS does not.

- **Scalability**[9] Scalability has been partially achieved in the centralised P2P DDLS through decentralising both linkbases and link services across the system. Peers directly obtain links from providers, and a single link service request can be processed by multiple service providers at one time. The net effect of a decreased number of centralised operations and that of parallelism enable the DDLS scale to a certain degree, although the location of linkbases is still very much centralised.

4.6 Summary

This chapter discussed several design issues for the DDLS, including resource description, organisation and operations, user interface and system functionality, and architecture. In particular, the choice of the software architecture was of primary concern. To accommodate the requirements for the DDLS applicable to a well-arranged environment, a centralised P2P architecture was proposed and a prototype based on the architecture was developed. To facilitate the explanation and comparison, another DDLS prototype adopting the client-server model was also developed.

The centralised P2P DDLS has demonstrated, through an analysis and comparison, its superiority to the client-server DDLS in the sense that the former incurs less communication across multiple nodes when a link service request regarding private linkbases is involved, thus yielding a more rapid delivery of link services. Also, the decentralised aspect of the centralised P2P paradigm increases the autonomy of each peer and the degree of information freshness, reduces users' dependence on a single link server, eliminates the single point of failure and enhances the scalability of the DDLS. It should be recognised, however, that the centralised P2P DDLS achieves its advantages at a cost of potentially more overhead for delivering link services, more stringent conditions of the service delivery, etc (see Table 4.1 for details).

So far, the objective of this work is partially achieved by a centralised P2P DDLS. To be fully capable of supporting collaboration in an ad hoc environment, the DDLS will need to adopt an unstructured P2P model. The next chapter will identify the features of and requirements for the unstructured P2P DDLS and explains why the existing

[9]The word *scalability* has long been used in software development to describe the capability of a system to function well as it is changed in order to handle increasing demand or load that it would be placed on.

approaches to different P2P systems are not sufficient to achieve the eventual objective of this work.

Chapter 5

Rethinking the P2P Paradigm

5.1 Introduction

Initial effort to extend the traditional hypermedia link service adopting a client-server architecture to P2P environments was described in Chapter 4. The consequence is a centralised P2P DDLS in which resource discovery resorts to a central directory and resource fetching is inherently based on P2P. Circumstances with ad hoc properties preclude the existence of such a centralised directory and a delicately designed indexing scheme and search algorithm are therefore crucial to such systems.

This chapter discusses in more detail P2P computing, an area with major concepts and technologies adopted by the DDLS. The definition of P2P is followed by the categorisation of P2P systems based on domains to which they have been applied. This chapter also presents a taxonomy of contemporary P2P systems based on architecture, examines their solutions to indexing and searching resources, compares their strengths and weaknesses, and identifies the essential properties that an indexing scheme and search algorithm for the unstructured P2P DDLS should possess.

5.2 P2P Computing

P2P (Clark 2001) can be described as an overlay network in which a group of peers communicate using the same networking program. Each peer possesses the same capabilities and can be autonomous, i.e. taking actions independently of one another. P2P computing is not a new technology in nature. Early distributed applications such as

email systems and telephone services exhibit similarity to P2P. Email systems built on SMTP[1] rely on local servers to establish connections to other peer servers for mail delivery, and telephone services work by setting up channels between switching offices to support simultaneous telephone calls.

Nonetheless, the novelty of P2P technologies developed over the last few years is that they enable Internet-connected personal computers to play more important roles than those played by client-server or master-slave systems. Although without continuous connections to a network, decentralised personal computers can establish periodic connections to one another. Sending and receiving messages and transferring files occur directly between these computers. In addition, splitting a computational problem into small independent parts enables each computer to process a different part and send the result back to a central server for collection. The P2P network aims to provide services and resources even in environments with unstable connectivity.

5.2.1 Categories of P2P Systems

P2P systems have typically been seen in the following domains: distributed computing, file sharing, collaborative systems and P2P platforms, according to the classification by Milojicic et al. (2002).

- **Distributed computing** Distributed computing is more recently, in the late 1990s, used to refer to harnessing the collaboration of Internet-connected personal computers. By pulling together the processing power of a network of personal computers, distributed computing enables large computations. Whether a distributed computing system is also a P2P system depends on the amount of computational tasks each computer executes and the extent of autonomy that each computer bears. Although distributed computing utilises spare computing resources of decentralised computers, Internet latencies are the principal obstacles to performing critical communications and computational tasks. This leads to a limited scope of distributed computing domains.

- **File sharing** File sharing must be one of the most popular and successful P2P application areas. Aggregating storage from distributed participants, file sharing systems achieve a potentially unlimited area for the exchange and sharing of files and music clips. The file reference may be the only knowledge required for file retrieval. P2P file sharing systems enforce duplication and replication policies to

[1] Simple Mail Transfer Protocol

offer availability, reliability and faster retrieval. Moreover, by building anonymity into systems using related algorithms, information about individual peers can not be identified (Clarke et al. 2001). Efficient location and search capabilities are the focus of research on P2P file sharing systems.

- **Collaborative systems** Collaboration at the application level can be established among peers. Examples can be found in online games and instant messaging systems. Peers carry out the given task in collaboration with others within the same group. An event occurring at one peer will be notified to others for corresponding action. Collaborative systems need to provide fault tolerant schemes for event delivery. As requested by distributed computing, capabilities to circumvent real-time constraints are also necessary features required of P2P collaborative systems.

- **Platforms** P2P platforms support a wide range of P2P applications by offering basic core functionality and high level services. Objectives of P2P platforms include an infrastructure for network programming and computing (e.g. JXTA) and access to Web services on the Internet from any available devices (e.g. .NET My Services).

5.2.2 Features of P2P Systems

This section provides an overview of the features exhibited by existing P2P systems. They help explain the mechanisms utilised by P2P systems, the advantages they offer and the challenges designers of P2P systems face.

- **Decentralisation** The definition of P2P conveys two meanings: resource decentralisation and processing decentralisation. Computational resources, such as CPU[2] cycles, data and computers, could be widespread, while processing might involve distributed data and algorithms. Compared to traditional client-server systems, P2P systems enable resources to be stored locally and accessed without the intervention of a central server. Therefore, individual peers gain more control over their own resources. Furthermore, decentralisation reduces the risk of the single point of failure commonly found in client-server systems.

- **Autonomy** Peers form a virtual network in which each of them can act as a server, a client, or both on top of the physical infrastructure. They store data and perform tasks on their own behalf instead of relying on a dedicated server. The autonomy is introduced by the independence of peers from the restrictions imposed by

[2]Central Processing Unit

infrastructure protocols and services, and it gives rise to increased complexity of service (or resource) location. The increased autonomy also brings increased concerns regarding security because policies and restrictions may need to be applied to peers' behaviour.

- **Ad hoc behaviour** The temporary presence of participants results in the ad hoc nature of a P2P system. They may join and withdraw from the network at any time. This may not be a significant problem for P2P file sharing systems. However, in a collaborative P2P system, the absence of a particular peer may lead to the failure of the collaboration, although the situation could be ameliorated by supporting transparent delay of communication to the disconnected peer. P2P systems should consider the dynamics of their peer network and provide solutions to accommodate changes of peers to maintain consistency and stability, and increase robustness.

- **Anonymity** Users and machines on a network can be uniquely identified by their IP address. It is not desirable because users may be concerned with the ramifications of their actions, publishers may be worried about being deprived of rights to speak freely, and storage systems may need to resist attempts by external systems to detect and spoil data. Support for anonymity is also seen as a defense against the censorship of digital content. Research on anonymous routing protocols aims to disassociate a user's IP address from the traffic that results from message routing. This primarily involves initiator anonymity, responder anonymity and mutual anonymity between a pair of communicating parties.

- **Scalability** Decentralisation yields improved scalability in P2P systems in which operations and computation are carried out in a decentralised manner and so are data storing and exchanging. Moreover, scalability is accomplished by the parallelism seen in both the programming model and the communication model. For instance, Napster (Napster 2001) achieves good scalability by introducing a P2P communication model which allows peers to directly download files from others that have the required documents. Good scalability should not be accomplished at the cost of other desirable features. Recent P2P systems which employ distributed hash table (DHT) techniques, such as CAN (Ratnasamy et al. 2001), Chord (Stoica et al. 2001) and Pastry (Rowstron and Druschel 2001), trade flexible network topology and data placement for satisfactory scalability by modelling the identifier space for efficient lookup.

- **Self-organisation** The absence of a central authority for control in some of the P2P systems partially accounts for the dynamism they exhibit. Peers join and

leave from the peer network frequently and the system relies on its own mechanism to recover from the inconsistent state and become stable. However, as the system scales, it is very difficult to predict the scale of the future number of peers and workload. Therefore, the probability of failure is potentially on the increase. P2P systems should possess the self-organisation capability to function even in the face of an enormous increase in the number of peers without the need for re-configuration or human intervention.

- **Security** The direct interaction between peers results in more security concerns in P2P systems than their alternatives, such as client-server and centralised systems. Apart from the security requirements shared with distributed systems, P2P systems confront new challenges in dealing with security issues. For instance, direct transferring files between peers in file sharing systems may lead to false information from an unreliable peer. Even worse cases include the execution of code from a malicious peer. Therefore, trust between peers is an important property that a peer can utilise to build its evaluation of others' reputation. P2P file sharing systems attract many public concerns about security. In particular, the ease of file copying raises worries about protection of intellectual property, and this is one of the reasons that the future of Napster-like file sharing systems is always tangled with copyright law.

- **Interoperability** P2P systems are highly dependent on the network topology and supporting applications, based on which distinct protocols are developed that define the way peers communicate over the network. Interoperability facilitates shared conversation and cooperation among peers from different systems. These peers are therefore allowed to exchange requests and resources. The principal reason for pursuing interoperability between different P2P systems is to avoid redundant development of services already provided by other systems and to focus on only the services that are exclusively offered by individual P2P systems. Efforts towards the realisation of improved interoperability between P2P systems include developing standards and specifications, achieving mutual understanding between system developers, and devising an infrastructure that offers services to accommodate various needs of ported P2P systems.

5.3 A Taxonomy of P2P Systems

Unlike the classification presented in Section 5.2.1, this section utilises a taxonomy to classify the P2P systems into groups by architectures. This is because the architecture

aspect of P2P systems draws more interest of this work. Some of the contemporary P2P systems and applications in each of the groups are examined.

5.3.1 Centralised P2P

A centralised P2P system typically has a central repository to maintain the information about resources that peers in the network possess. Resource discovery in such a system involves querying against the central repository and identifying the best peer (depending on users' needs) that matches the request. The address of the best peer is returned to the query originator and the subsequent file exchange occurs directly between the two associated peers. The centralised P2P systems require that a central peer or a group of dedicated peers coordinate update to the information held in the repository, which may be difficult and expensive.

5.3.1.1 Napster

Napster (Napster 2001) promoted the centralised P2P model. It was introduced in 1999 as a file sharing software to exchange MP3 files on the Web. A centralised directory (actually several) which describes how files are distributed in Napster is maintained. Nodes register with the directory when they join the Napster network. Users run the Napster program in order to participate in the file sharing process. If an Internet connection is detected on a user's computer, the Napster software will help establish another connection between the computer and one of the Napster's Central Servers. The Napster Central Server keeps a directory of all client computers connected to it and stores information about them. A user's file request will be placed to the Napster Central Server that the user's computer connects. The Server then looks up its directory to check any match for the request. If there is any, the Napster Central Server will return a list of all the matches and related information about them, such as the IP address and the file size, to the requestor. The user decides which file he/she wants to download and tries to establish a direct connection between his/her own computer and the computer with the desired file (also referred to as the client computer) by sending a message that contains the IP address of his/her computer to the client computer. Once the connection is established, the client computer will directly transfer the file to the file requestor. The connection is terminated by the client computer when downloading is finished.

5.3.2 Unstructured P2P

Unstructured P2P systems include Gnutella (Gnutella 2001) and Routing Indices (Crespo and Garcia-Molina 2002a). The overlay topologies of both systems have ad hoc properties and the placement of their data objects is unrelated with the overlay topology. Freenet (Clarke et al. 2001) is also unstructured since in Freenet the overlay topology is loosely controlled and the file placement is based on hints[3].

5.3.2.1 Gnutella

Gnutella (Gnutella 2001) is a P2P file sharing protocol. Applications that implement the Gnutella protocol allow users to search for and download files from other users on Internet accessible hosts. To join the system, a peer initially connects to one of the several peers already available known by out-of-band mechanisms, hence forming an application level overlay on top of the physical network. Once attached to the network, peers interact with one another by means of messages. When a user wishes to search for a file, a query is issued and broadcast to all attached nodes. The recipients may not respond with results if they do not have any desired data object. Nonetheless, they will further forward the query to their attached nodes. Gnutella uses limited flooding to distribute queries by attaching a TTL (Time-To-Live) tag which specifies the maximum number of hops a query can be relayed. Replies are routed back to the peer that originally generates the query.

The advantage of such an unstructured system is that it models the real world better than a structured P2P system (see Section 5.3.3) with the placement of data objects being not subject to any knowledge of the network topology. However, it is hard to locate the desired information without flooding queries to most parts of the overlay network. Broadcasting on every query is not scalable. As more nodes join, more queries may be generated and transmitted. In addition, flooding needs to be curtailed at some point, which leads to some peers with desired data objects missing from the overall result.

Research centered around the scalability of Gnutella makes up the majority of the work. To seek more scalable search methods, Lv et al. (2002) study alternative search methods in a Gnutella network with several overlay topologies, including a power-law random graph, a normal random graph, a Gnutella graph and a two-dimensional Grid.

[3]However, Freenet is sometimes regarded as a structured P2P system because its search algorithm resembles those used in DHTs-based structured P2P systems (see Section 5.3.3).

They propose a search mechanism based on multiple random walks to substitute the flooding of queries in Gnutella, and reveal through simulation that, with a fixed number of random walkers, the multiple random walk algorithm can help locate the desired data object almost as quickly as Gnutella's flooding while reducing the network traffic by two orders of magnitude in many cases at the expense of a slight increase in the number of hops. As demonstrated by the study, scalable searches in unstructured networks are mainly attributed to the following principles: adaptive termination of query forwarding, reduced message duplication, and small granularity of the coverage of visited nodes.

However, Ritter (2001) argues that Gnutella is never an ideal distributed and fully capable network and proves that Gnutella is mathematically and technologically unable to scale to a network of any reasonably large size. It is identified that the fundamental flaws of Gnutella lie in its major architecture and cannot be mitigated effectively without a re-design at the most basic level.

5.3.2.2 Freenet

Freenet (Clarke et al. 2001) is a cooperative distributed information storage and retrieval system designed to offer privacy protection and information availability. By pooling spare disk space across hundreds of thousands of collaborative computers into a self-organising virtual file system, Freenet allows individuals to insert, store and retrieve files anonymously without compromising any of its principle design goals.

The basic unit of storage in Freenet is a file. Nodes in Freenet query one another to store and retrieve files which are named by location independent keys. The keys are typically generated by using the hash function on the text description of files provided by users when storing files in the network. Because a key can be computed from the description of a file, the file description (rather than the key) is commonly made available to users of Freenet by mechanisms such as websites. To optimise the search, Freenet clusters files with similar keys on a single node and maintains the information of successful searches in a local routing table. Simulation performed by Clarke et al. (2001) shows that the request path length of Freenet scales approximately logarithmically with respect to the network size.

The operations supported in Freenet include insertion of and search for files, with both working in a similar way. Each request is given a pseudo-random identifier and a hops-to-live limit, the latter of which is analogous to the TTL employed by Gnutella. If a search message is received by a node with the desired file, the entire file will be returned as a successful result and the file's key will be inserted into the local routing

table. This is carried out recursively until the file is returned to the initial requestor with the file replicated at each node along the search path. Otherwise, the message is forwarded to another node with the most similar key in the local routing table. An insertion operation starts with a search operation. If a file with the specified key is found, a collision notification will be sent to the insertion originator. If no file is discovered, the new file will be placed on a node with files sharing similar keys. New nodes should contact at least one node that already exists in the network to bootstrap themselves into the system and perform a search request for announcing their presence.

5.3.2.3 Routing Indices

Crespo and Garcia-Molina (2002a) propose a distributed search in unstructured P2P systems which builds Routing Indices (RIs) to facilitate query forwarding. RIs do not maintain any dedicated node as the repository for indices of resources in the network. They utilise the topic information of documents to create related groups. Each peer has a local RI to maintain information on different topics along each path to its neighbours which, in turn, collect topic information along each path to their neighbours using the same mechanism. The way information is utilised and stored makes it expensive to create and update RIs. Therefore, updates are suggested to be processed in batches if doing so will not yield significant difference. RIs assist peers to forward queries to good neighbours. The goodness of a neighbour is based on the number of documents on specified topics. RIs deliver a good search result in a domain in which the category of documents can be easily identified. However, as the number of categories increases, RIs will consume a large amount of space for the maintenance of indices. Moreover, the close interrelationship between RIs of related peers entails the accuracy of each RI. RIs consist of three schemes: the compound, the hop-count and the exponentially aggregated RIs.

The Compound Routing Indx (CRI) is a data structure that summarises both the number of documents along each path and the number of documents on individual topics. Given the index, the goodness of a neighbour can be computed using a simplified model. The value of the goodness indicates the sequence of query forwarding and a query is first sent to the best neighbour. The CRIs do not reflect the difference in hops (or cost) that queries should travel to obtain the documents of interest, and therefore they do not produce satisfactory results.

Alternative RIs are presented which include hop-count RIs and exponentially aggregated RIs. The core part of a hop-count RI is a variant index of a CRI. The hop-

count RIs group documents based on hops. Therefore, the goodness of neighbours can be computed based on a model that takes into account the cost of document fetching because of the presence of hop information. Also, the number of messages required to get documents can be obtained by using a cost model called the regular-tree cost model. The model assumes that document results across the network are uniformly distributed and the network itself is a regular tree. The disadvantage of hop-count RIs is that they incur much higher storage and transmission cost than the CRIs. This problem can be addressed by the exponentially aggregated RIs at the cost of some potential loss in accuracy. An exponentially aggregated RI stores the result of applying the regular-tree cost formula to a hop-count RI. Simulation demonstrates that the exponentially aggregated RIs outperform the hop-count RIs in most cases. However, the assumption of the regular-tree model may not always hold in many settings.

5.3.3 Structured P2P

Structured P2P systems feature the use of DHTs which assume that the network topology is tightly controlled and the placement of files (or other data objects) is precisely determined in such systems. Examples include CAN (Ratnasamy et al. 2001), Chord (Stoica et al. 2001) and Pastry (Rowstron and Druschel 2001). Typical DHTs resolve a keyword to a location where the contents are located or from where queries about the contents can be further routed. The inherent self-organisation is attributed to the distribution of keys in a uniform space in which node and object identifiers share the same key space. Adopting DHTs requires unique hash techniques that transform the search criterion into a unique key.

5.3.3.1 Chord

Chord (Stoica et al. 2001) is a scalable distributed lookup protocol that efficiently locates the specific node that stores a particular data item. The only operation that Chord supports is to map a key onto a node. Applications of Chord, such as data location, can be implemented by associating a key with each data item and storing the key/data item pair at the node that the key is mapped to.

Chord uses consistent hashing to allocate each key and node an m-bit identifier. A key's identifier is generated by hashing the key, while a node's identifier is produced by hashing its IP address. The hash function assigns keys to nodes and enables each node to receive roughly the same number of keys. Identifiers are ordered in a circle modulo 2^m.

Key k is assigned to the first node whose identifier is equal to or follows the identifier of k in the identifier space. The node is referred to as the *successor node* of key k. In Chord, each node need only be aware of its successor node on the identifier circle. Queries for a given identifier can be passed around the circle via these successors until a node whose identifier succeeds the desired identifier is encountered. The node is the one that the query should be mapped to. Consistent hashing is also designed to achieve minimal disruption in Chord during node arrival and departure. When a node n joins the network, certain keys maintained by n's successor are assigned to n. Meanwhile, when node n leaves the network, all of its keys will be reassigned to its successor. Therefore, transferring keys due to node arrival and departure only affects the immediate neighbour of a joining or a leaving node.

The base Chord protocol may lead to all nodes being traversed during query mapping. To implement scalable and rapid key location, each Chord node maintains additional information in its routing table. A routing table, referred to as a *finger table* in Chord, has (at most) m entries. In addition to the keys that the current node stores, each entry maintains a key identifier, an interval of key identifiers starting from the key identifier, and a successor node which is responsible for the key identifier. A node n which is searching for the successor node of key k will first refer to its finger table to find an interval that contains k. The corresponding successor node is the next one that n should visit in order to locate the successor node of k. By repeating the process, n learns nodes whose identifiers are closer and closer to and precede k. The successor of the node which most closely precedes k is the successor of k.

Chord deals with node failure by allowing each node to maintain a list of multiple nearest successors on the Chord identifier circle. A node replaces its information about a failed successor with the next live one on the list. In a dynamic environment, the capability of preserving the correctness of locating every key is desirable. Chord has a stabilisation scheme to maintain correct routing information with gracefully degraded performance even in the face of concurrent joins, and lost and reordered messages. Chord can achieve the scalability that P2P systems with widespread use of broadcast lack. Given an N-node network, each Chord node maintains routing information about only $\Theta(\log N)$ other nodes. The cost of a Chord lookup scales as $\Theta(\log N)$ and, with high probability any node joining or leaving a Chord network uses no more than $\Theta(\log N^2)$ messages.

5.3.3.2 Content Addressable Network

The term *Content Addressable Network* (CAN) refers to a scalable indexing mechanism which maps file names to their locations in the system (Ratnasamy et al. 2001). CAN is able to provide not only P2P systems but also large scale storage management systems hash table-like functionality on a very large scale, such as the Internet.

As a distributed infrastructure, CAN uses a virtual d-dimensional coordinate space to store (key, value) pairs as follows. Key k in a pair (k, v) is deterministically mapped onto a point P in the coordinate space by means of a uniform hash function. Pair (k, v) is then stored in a zone which contains point P. The entire coordinate space is partitioned among all nodes in the system so that each node owns its individual, distinct zone. To obtain a zone from the overall coordinate space, a new CAN node first discovers the IP address of a bootstrap node already in the system. The bootstrap node supplies the IP addresses of several randomly chosen nodes in the system. The new node randomly chooses a point in the space and sends a JOIN request to the node whose zone the chosen point is in. It obtains a zone from the current occupant node and learns the IP addresses of its coordinate neighbours from the previous occupant. A request, such as insert, lookup or delete, for a particular key is routed in CAN by following the straight line through the Cartesian space from the source to the destination coordinate. CAN always forwards a request towards the neighbour that is the closest to the destination CAN node whose zone contains the key.

Each CAN node maintains a routing table with $\Theta(d)$ entries which refer to its neighbours in the d-dimensional space. If the coordinate space is partitioned into n equal zones, the average routing path length is $(d/4)(n^{1/d})$, which indicates that, unlike Chord, CAN's routing table does not grow with the network size. To reduce the routing path length and path latency, CAN can either increase the dimension of its coordinate space, or maintain multiple independent coordinate spaces (called 'reality' in CAN) and assign each node in the system a zone in each of the coordinate spaces. The high per-node neighbour state and maintenance traffic are offset by improved data availability and fault-tolerance. As opposed to increasing dimensions, increasing realities yields shorter path length but improved data availability and fault-tolerance. Other measures to improve data availability include using multiple hash functions. CAN achieves its robustness by using a takeover algorithm which ensures one of the failed node's neighbours to take over responsibility. The designers present some initial results on the construction of CAN topologies that are consistent with the underlying IP topology to avoid unnecessary message routing via distant nodes. Furthermore, CAN employs caching

and replication techniques to deal with 'hot spot'[4] management.

5.3.3.3 Pastry

Pastry (Rowstron and Druschel 2001) was designed to be a general substrate for the construction of a variety of P2P applications, such as data storage and sharing, group communication and naming systems. It is a scalable P2P content location and routing scheme at the application level, based on a self-organising overlay network which consists of nodes connected to the Internet.

Each node in Pastry is randomly assigned a 128-bit unique identifier (nodeId) which indicates the location of the node in a circular nodeId space ranging from 0 to $2^{128}-1$. The nodeIds are so generated, typically by computing a cryptographic hash of nodes' public keys or their IP addresses, that the resulting set is uniformly distributed in the 128-bit nodeId space. NodeIds and keys are both thought of as a sequence of digits with base 2^b. Each node maintains a *leafset* and a routing table for routing. The leafset consists of the L[5] clockwise and counter-clockwise neighbours of the current node in the circular nodeId space. A node's routing table is organised into $\lceil \log_{2^b} N \rceil$ rows with each comprising 2^b-1 entries, given N the number of Pastry nodes in the network. At row n of the routing table, the 2^b-1 entries refer to nodes whose nodeIds share the first n digits with the current node's nodeId, but whose $(n+1)$th digit has one of the 2^b-1 possible values other than the $(n+1)$th digit of the current node's nodeId. When a node joins the system, it can initialise its leafset and routing table and maintain the consistency of all system invariants by exchanging $\Theta(\log N)$ messages.

Keys can be generated in different manners, such as computing the hash of the file's name and owner, or the topic name. A (key, value) pair is inserted in the circular space by using Pastry to route the pair to the node whose nodeId is numerically closest to the given key. Routing in Pastry is carried out as follows. In each routing step, a node forwards a message to another node whose nodeId shares with the key a prefix that is at least one digit longer than the prefix that the key shares with the current node's nodeId. If there is no such a node, the message is forwarded to a node whose nodeId shares with the key a prefix as long as the current node but is numerically closer to the key than the

[4] In hypermedia terminology, hot spots are also referred to as persistent selections which represent selections within components (information managed by applications) that persist between application sessions and can be accessed later (D'Arlach and Leggett 1994). A hot spot typically takes the form of a specifically defined area that contains a hyperlink. However, in this work 'hot spot' is used to denote the phenomenon that in a query pattern a certain key is requested extremely often and the node holding that key becomes overloaded.

[5] $|L|$ is a configuration parameter with a typical value of 16 or 32.

current node's nodeId. With normal operations, the expected number of routing steps is $\Theta(\log N)$ and each node maintains a routing table with $\Theta(\log N)$ entries.

Pastry takes into account locality properties so as to enhance routing performance. An application is assumed to provide functionality helping a Pastry node to determine the 'distance' of another node to itself by means of network proximity. Network proximity is based on a scalar proximity metric, such as the number of IP routing hops and the geographic distance. For instance, Pastry can determine the k nodes whose nodeIds are numerically nearest to the specified key. Based on its estimation of the density of nodeIds exhibited by local information, Pastry adopts a heuristic to ensure that a message is likely to be forwarded to first reach a node with the numerically nearest address among those k nodes.

5.4 A Web-based P2P Open Hypermedia System - the Unstructured P2P DDLS

The essential characteristics of an unstructured P2P DDLS are initially identified in this section, together with certain associated requirements to be satisfied. This section subsequently explains why the existing approaches are not applicable to the unstructured P2P DDLS with respect to the requirements.

5.4.1 Characteristics and Requirements

As a product of both the DLS technology and P2P computing, the unstructured P2P DDLS exhibits the properties that can be seen in both fields and it faces, in the meanwhile, challenges that neither of them has ever encountered.

- **Ad hoc properties** The unstructured P2P DDLS aims to serve in an ad hoc collaborative environment in which peers are at distributed locations and resources available at any particular time is unknown and unpredictable. Both services and resources from a particular peer are more probabilistic than deterministic.
- **Without centralised control** A central authority indicates the possibility of registering available resources at one place. As a consequence, peers that require particular resources can make use of the information advertised at the same place.

It is the scheme that a centralised P2P system adopts. The challenge for the unstructured P2P DDLS is how to conveniently publish resource information and efficiently discover resources without any central authority in ad hoc settings which can be pervasively identified nowadays.

- **Resource description** The collaborative participants provide links, which are stored and manipulated in linkbases, to one another. For ease of management and link discovery, the convention requires that links referring to related documents should be grouped and stored together (Carr et al. 1995). Therefore, the primary content of documents that links in a linkbase refer to can be represented by keywords which may be used as the indices when link discovery occurs. Because participants may store the representation of linkbases using various data models and syntax, the incompatibility issue needs to be resolved. A data model and syntax, which convey the precise representation of linkbases and exclude the unambiguity when converting linkbase representations from their adopted data models, are required to facilitate interoperability.

- **Resource publishing** The difficulty arising in resource publishing is a consequence of unavailability of a centralised mechanism, such as a centralised directory in which resource providers can advertise their resources and indicate from where these resources can be obtained. A potential way to publish available resources in the system is to express the characteristic information about the resources in a local document and offer it upon other peers' request. Although it leads to the advertised information being accessible to a narrower scope of peers, the approach does not necessarily exclude the reachability of that information through indirect interaction.

- **Resource discovery** No hints are employed to guide the Gnutella search and such a 'blind' search results in a large number of unnecessary messages. To avoid this, DDLS resource discovery may take advantage of the feature of linkbases that the primary concepts related to a peer's linkbases can be represented by a vector of keywords which may act as indices for resource discovery. If the topology of the DDLS peer network takes into account the conceptual relationship between linkbases of peers, the search for linkbases may potentially be more efficient. However, this search cannot be fully supported by the standard keyword-based match and therefore a semantic search is required. Furthermore, a semantic search is required because the DDLS aims to facilitate the search of users whose information needs are usually expressed by a limited number of search terms. Related work on semantic search can be referred to in Appendix A.

- **System re-organisation** The topology of the DDLS peer network is not immutable because of the arrival and departure of peers. Changes may also result from the update of peer resources. Through re-organising the peer network, peers can select neighbours which will accommodate their future needs with high probability. Therefore, the performance of resource discovery in the DDLS can be enhanced.

5.4.2 Limitations of Existing Approaches

5.4.2.1 Centralised P2P Solution

The centralised P2P typically requires a constantly updated directory which maintains an index of available resources hosted at different locations. The resources of interest are located by querying the central directory server. Centralised systems are vulnerable to attacks and make it difficult and expensive to update indices. Also, the scalability issue remains as a tough task to achieve.

5.4.2.2 Unstructured P2P Solution

Gnutella may be the most appealing and controversial unstructured P2P system as of today. Without an index of documents, Gnutella propagates queries from nodes to nodes until either matching documents are found or the termination conditions are satisfied. Flooding the network with queries incurs an enormous amount of messages and network traffic which are not desirable and affordable in ad hoc settings in which communication may be subject to available bandwidth.

Other solutions, such as RIs, make assumptions about the grouping of documents by topics. It may be applicable to a system maintaining documents regarding a specific domain since the number of categories can be predicted or estimated. However, when applied to a system hosting documents across a large number of domains, RIs will consume a large amount of space for maintaining indices of nodes in response to the various categories the documents belong to. The accuracy of the RI hosted by individual nodes governs nearly all operations in the system. For instance, when creation and update operations are carried out along a chain of nodes, any participating node which refuses to cooperate will result in a system functioning improperly.

5.4.2.3 Structured P2P Solution

In recent years, the P2P community has extensively researched into the DHTs for unstructured P2P systems. The adoption of DHTs assumes a highly structured system in which the network topology is tightly controlled and the placement of files (or other data objects) is precisely determined. A structured P2P system indexes the search space and provides a mapping between hashed keys of file identifiers and locations for efficient query routing to the node with desired files. This is practical because peers in structured P2P systems are primarily intended to store resources, whereas the DDLS focuses on resource sharing and peers in the DDLS are owners, other than storage providers, of the resources they maintain. Therefore, it is infeasible to apply techniques such as DHTs that assign nodes, which may not necessarily be the owners, to host resources.

DHTs offer a very scalable solution to matching queries since the mapping between a file identifier and a location is deterministic. They heavily rely on the uniqueness of hashed keys and only files whose identifiers exactly match the requirement specified in the query are returned. In contrast, the DDLS aims to provide more than that. The search algorithm in the DDLS is to enable a mechanism that extends the discovered resources from those with exactly matched keywords to resources carrying related concepts. This can be potentially achieved by examining the semantics of resources peers maintain.

Typical DHTs resolve a keyword to a location where the contents are located or from where queries about the contents can be further routed. Hence, it supports the search on a single hash expression once at a time. However, the query of a typical semantic search may consist of a random combination of terms and the relationship among them. Carrying out the search by using DHTs involves searching for targets that match at least one of the terms at a time and performing conjunctive or disjunctive operations on the result with the assistance of filter structures, such as Bloom filters (Mullin 1990). The relationship among terms is thoroughly omitted during the search and it is clear that the result may not reveal all targets that should have been located.

The distribution of resources and requests across peers is one of the components that affect the reachability of resources in a P2P system (Ledlie et al. 2002). If, for example, the popularity of potential resources in the working scenario of the DDLS follows a Zipf's distribution (see Section 6.5.2), the formation of 'hot spots' will very likely occur if DHTs are employed (Ganesan et al. 2003). Suppose there are 100 peers in the DDLS and they share 100 topics representing the primary content of resources. The popularity of the topics follows a Zipf's distribution (see Figure 5.1). Topic A is the most popular topic in the system with 100 instantiations. While topic B, the 15th most

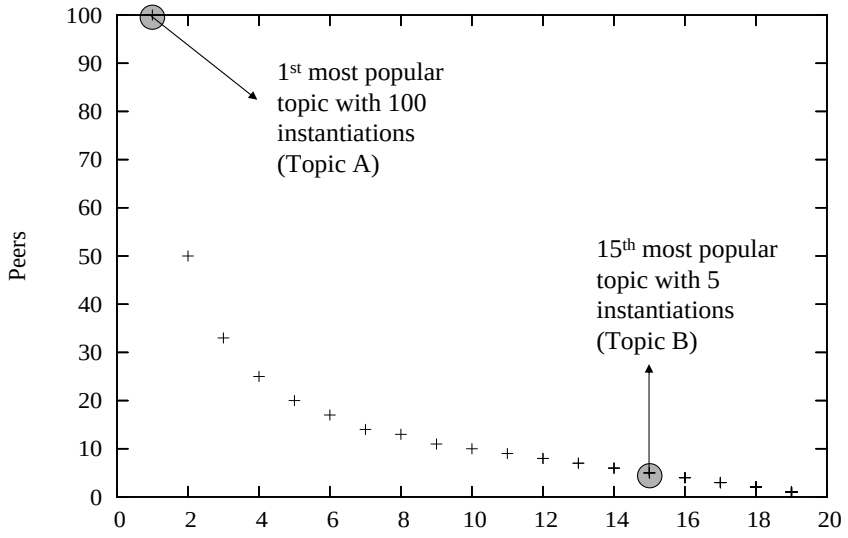

FIGURE 5.1: Topics Following Zipf's Distribution

popular topic, possesses 5 instantiations. Also, suppose that Chord is used to model the search space in the DDLS as shown in Figure 5.2. Node 0 stores all the keys of topic B and node 5 maintains those for 100 instantiations of topic A. The phenomenon of 'hot spot' occurs at node 5 because it is heavily burdened with all the instantiations of topic A. Although load balancing techniques could be utilised to ameliorate the situation, a joint query involving multiple topics would lead to a more complicated search. It is not known whether the benefit resulting from the adoption of load balancing techniques can compensate for the overhead caused by carrying out a joint query.

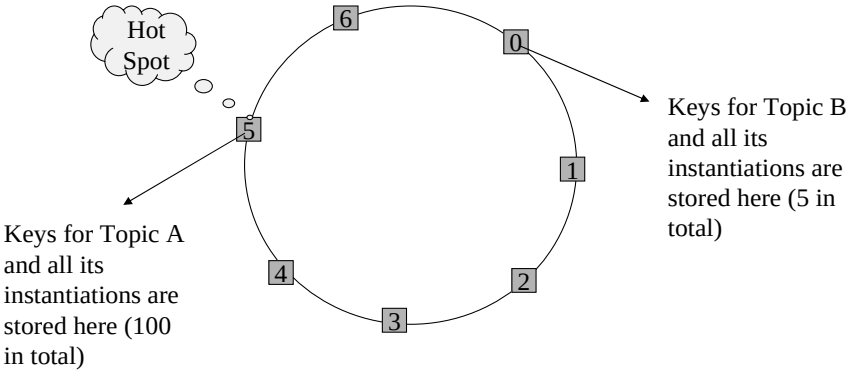

FIGURE 5.2: An Example of Using Chord to Model the DDLS Search Space

5.5 Summary

This chapter presented a more detailed background for P2P computing, examining the definition, categories and features of P2P systems. A taxonomic review of the approaches employed in various contemporary P2P systems was also given.

DHTs are promising in many application areas and are extensively researched among others. However, by analysing the characteristics of and requirements for the DDLS, one would realise that the widespread applicability of DHTs cannot help solve problems arising from the particular context of the unstructured P2P DDLS because of the inherent nature of the way that DHTs work. For instance, the semantic search required by the unstructured P2P DDLS calls for a different mechanism to discover the resources of interest, which cannot be achieved by DHTs as demonstrated by the analysis carried out in Section 5.4.2.3.

In the next chapter, the DDLS will shift to an unstructured P2P system in which the issues centering around resource discovery based on semantics are the main focus of attention.

Chapter 6

Evolution of the DDLS into an Unstructured P2P System

6.1 Introduction

This chapter presents the work of evolving the centralised P2P DDLS demonstrated in Chapter 4 into an unstructured P2P system. It begins with an introduction to the peer network in the unstructured P2P DDLS, involving the relationship between peers and how peers relate to the peer network due to their arrival and departure. The details of how resources should be described in the given context are provided and this is followed by the description of a distance-based semantic search algorithm adopted by resource discovery. Finally, this chapter presents simulation that investigates the search algorithm with varying distributions of potential resources and different query profiles involved.

6.2 DDLS Peer Network

In contrast to the peer network in the centralised P2P DDLS, the one in the unstructured P2P DDLS is characterised by decentralisation of control - with the absence of a central service directory. In the centralised P2P DDLS, peers rely heavily on the central service directory to locate link service providers and linkbases (or links), whereas the decentralised nature of the unstructured P2P DDLS entails collaboration between peers to fulfill most of the tasks. Peer arrival, update and departure entailing notification sent to related peers demonstrates such a close relationship among peers in the unstructured

P2P DDLS. The following subsections discuss the relationship between peers and that between peers and the peer network.

6.2.1 Peer Relationship

In the DDLS[1], the relationship between peers includes neighbours, contacts and disconnected peers. A peer p_i obtaining the published topic information, which is the representation of the abstract concepts a peer's linkbases are associated with (see Section 6.3.2), from p_j indicates that:

1. p_i has a neighbour p_j;
2. p_j has a contact p_i.

If p_i has never acquired information from p_k and vice versa, p_i and p_k are referred to as disconnected peers.

6.2.2 Supporting the 'Published Topic List' Data Structure

peerID	α^{dist}	topics
j	0	
k	2	D, E
m	5	G, H, I, J, K

TABLE 6.1: A Published Topic List in the Cache of Peer p_i

It will be mentioned in Section 6.2.3 that upon its arrival at the DDLS, each peer randomly selects a set of neighbours. The peer caches a list of published topics which reflect the semantic relationship between its resources and those of all its neighbours. The size of the cache is determined by a specified value or 128, whichever is smaller. In the list, each entry comprises three fields. The first field indicates all the neighbours which share α^{dist} (shown by the second field) related topics with the current peer. The contents of these topics are listed in the third field. Table 6.1 gives an example published topic list in the cache of peer p_i. It indicates that p_i shares no relevant topics with p_j. The neighbours that have semantically related topics with p_i are p_k and p_m, which have 2 and 5 relevant topics, respectively.

[1]The term 'the DDLS' refers to the unstructured P2P DDLS in the rest of this chapter unless specified otherwise.

6.2.3 Construction of Peer Network

The peer network can be modelled by a graph G in which the vertices represent peers and the edges denote the semantic relationship between the resources possessed by peers. Each of the peers publishes a list of topics which represent the primary content of the documents links in its linkbases refer to. The topic lists can be asynchronously updated by individual peers. The topology of the peer network takes shape in accordance with the semantic properties of peers' resources[2]. Therefore, the DDLS peer network is also known as the semantic overlay. The construction of the semantic overlay is explained by giving a description of peer arrival as follows.

Initially, when a peer p_{new} joins network G, it contacts a set of randomly selected peers[3] represented by p_{new}^{random}. p_{new} informs each of the randomly selected peers in p_{new}^{random} of its topic list T_{new}. It is assumed that the environment provides each peer with a capability to identify the semantic relationship between entities (see Section 6.4.2). All neighbours return related topics to p_{new}. p_{new} will subsequently take advantage of this information to construct its published topic list as well as form part of the semantic overlay.

Figure 6.1 describes the algorithm for individual processing of the published topic lists from neighbours at peers which accompanies the construction of the semantic overlay. The processing procedure is carried out in parallel at each of the peers in p_i^{random} and leads to the creation of an overlay with clustered information. Peers in the randomly connected network represent the semantic relationship between their resources and those of others via α^{dist}. However, not every peer may have an overlap in the semantic description of resources with others. In such a case, the associated information is stored in the published topic list with $\alpha^{dist} = 0$.

Figure 6.2 demonstrates what occurs when a peer p_{new} joins the DDLS peer network consisting of p_j, p_k, p_l, p_m and p_n. p_{new} randomly selects a set of peers that include p_m and p_n and notifies both of its topic list. Through the same process, another peer p_l obtains information from p_{new}. The arrows in the figure show the direction that collected information flows.

[2] In essence, the peer network is randomly established at the construction stage and the formation of its topology in response to the semantic relationship between resources of pairs of peers only occurs during re-organisation. The reason for this will be given in Section 7.8.

[3] These peers are randomly chosen from the identifier space or obtained via multicast.

Notations:

Peer network:	G
Set of peer identifiers:	P
Set of topics:	T
A peer instance in G including a peer identifier and its topics T_i (defined later):	$(p_i, T_i),\ with\ p_i \in P$
Topics of p_i :	$T_i = \{t \mid t \in T, t \times t \subseteq T\}$
Set of identifiers of the randomly chosen neighbours of p_i:	p_i^{random}
Capacity of the cached topic list E_i (defined later) of p_i:	e_i^{max}
Cardinality of intersection between topics of p_i and those of its neighbour:	$\alpha_{i,k}^{dist}$
Set of semantically related topics between p_i and the same neighbour mentioned above:	$T_{i,k}^{common} = \{t_m \mid t_m \in T, t_m \times t_m \subseteq T\}$
Set of identifiers of neighbours sharing $T_{i,k}^{common}$ with p_i:	$id_{i,k}^{peer} = \{p_m \mid p_m \in P\}$
Entry of the cached topic list E_i of p_i:	$E_i^k = \{(\alpha_{i,k}^{dist}, T_{i,k}^{common}, id_{i,k}^{peer}) \mid 0 < k \leq e_i^{max}\}$
Cached topic list of p_i:	$E_i = \{E_i^k \mid 0 < k \leq e_i^{max}\}$
Calculating the intersection between topics of p_i and those of p_j:	$\varrho(p_i, p_j)$
Adding the identifier of the new neighbour p_k to $id_{i,v}^{peer}$:	$id_{i,v}^{peer}.addElement(p_k)$
Inserting an entry E_i^v into E_i:	$E_i.insertElement(E_i^v)$

Initial settings:

$Online = true$;
$Allow\_queries = false$.

Algorithm for processing of the published topic lists from neighhours at p_i:

For $u = 1$ *to* $|p_i^{random}|$
  $\alpha_{i,v}^{dist} = |\varrho(p_i, p_i^{random}[u])|,\ with\ 0 < v \leq |p_i^{random}|$;
  $T_{i,v}^{common} = \varrho(p_i, p_i^{random}[u])$;
  $id_{i,v}^{peer}.addElement(p_i^{random}[u])$;
  $E_i.insertElement(E_i^v)$.

FIGURE 6.1: *Construction of the Semantic Overlay*

6.2.4 Peer Departure

6.2.4.1 Notifying Contacts of Leaving Peers

The departure of a peer results in a notification sent to its contacts. Each contact then updates its cached published topic list by searching for the entries that involve the leav-

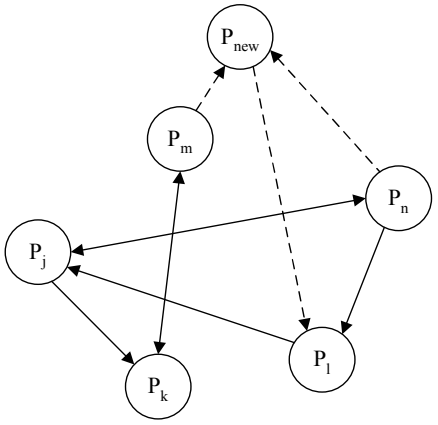

FIGURE 6.2: A Peer p_{new} Joins the Semantic Overlay

ing peer and removing them.

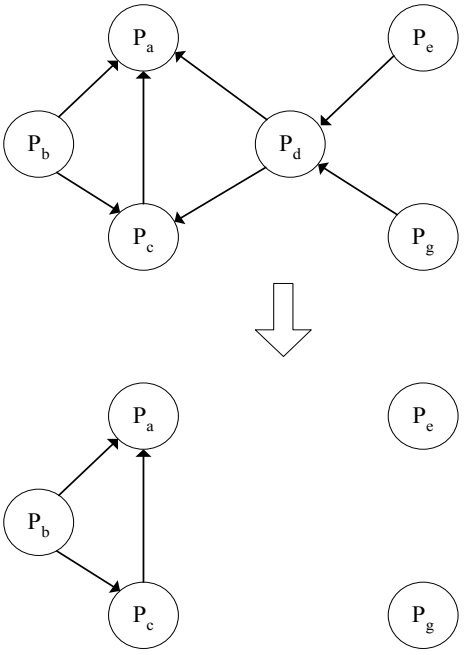

FIGURE 6.3: Contact with Peers Lost due to a Leaving Peer

A peer departure can potentially lead to network partition. The issue is more critical when the partition involves a large number of peers, which entails efficient recovery of information about the lost community. For instance, the consequence of the departure of p_d, see Figure 6.3, is that the peers on the left side of p_d lose contact with those on the right side. To have a robust system, measures are introduced to overcome such

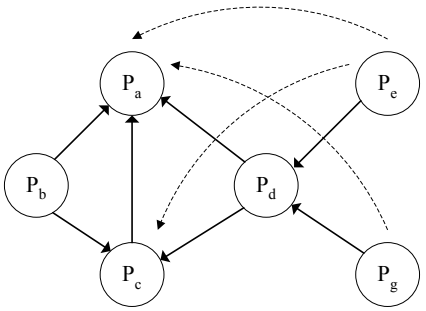

FIGURE 6.4: Leaving Peer p_d Notifies Contacts of its Neighbours

a loss. In addition to informing the contacts of its departure, a peer should also include information about its neighbours. Therefore, its contacts gain the knowledge of others which are two hops away. Upon the receipt of a departure notification, the contacts of a leaving peer decide which neighbours of the leaving one can be chosen as their new neighbours, thus preventing those neighbours from being isolated. The neighbours of a leaving peer become isolated if they are not qualified as the neighbours of any contact of the leaving peer. Figure 6.4 illustrates that a leaving peer p_d notifies its contacts p_a and p_c that it has two neighbours p_e and p_g. By analysing the usefulness (see Section 7.4) of p_e and p_g, contact p_a decides to incorporate both into its published topic list, whereas p_c selects only p_e as its new neighbour.

Figure 6.5 presents the detailed algorithm. A leaving peer p_i takes the initiative to inform each of the contacts of its neighbours before its departure from the peer network. Each contact p_u selects the neighbours of p_i to be their new neighbours on the basis of their usefulness. If a neighbour of p_i is more useful than at least one of p_u's current neighbours, p_w for example, it will be incorporated into the cache of p_u through $p_u.addNeighbour(p_w, T_w)$.

6.2.4.2 Merging Published Topic Lists from Leaving Peers

A departure notification is also dispatched to the neighbours of a leaving peer if the peer is prepared to share its entire published topic list with neighbours. Sharing this information allows neighbours of the leaving peer to choose others as their new neighbours based on a usefulness analysis.

Suppose peer p_j, in Figure 6.6, is going to disconnect from the peer network. It has two neighbour p_i and p_k. Based on a usefulness analysis of p_i, p_k discovers that p_i

Notations:

Same as in Figure 6.1. In addition,

Neighbours of p_i*:*	$B_{p_i} = \{(p_j, T_j) \mid i \neq j, p_j \in id_{i,h}^{peer}, 0 < h \leq e_i^{max}\}$
Contacts of p_i*:*	$C_{p_i} = \{(p_j, T_j) \mid i \neq j, p_i \in id_{j,h}^{peer}, 0 < h \leq e_j^{max}\}$
p_i *notifying contact* p_u *of its neighbour* p_w*:*	$p_i.notify(p_u, (p_w, T_w))$
Computing the usefulness of p_j *with respect to* p_i*:*	$\iota_{i,j}$
Adding a new neighbour p_j *to* p_i*:*	$p_i.addNeighbour(p_j, T_j)$

Algorithm for a leaving peer p_i to notify contacts of its neighbours:

For $(p_u, T_u) \in C_{p_i}$
  For $(p_w, T_w) \in B_{p_i}$
    $p_i.notify(p_u, (p_w, T_w))$;
    For $k = 1$ *to* e_u^{max}
      If $p_v \in id_{u,k}^{peer}$ *AND* $\iota_{u,w} > \iota_{u,v}$*, then*
        $p_u.addNeighbour(p_w, T_w)$.

FIGURE 6.5: Algorithm for Leaving Peer p_i Notifies Contacts of its Neighbours

possesses much more relevant resources than some of its current neighbours. Hence, it contacts p_i and requests its published topic information. It should be stressed that the usefulness analysis carried out by p_k can only take advantage of the topic information about p_i held by p_j which is however partial due to the way the published topic list is constructed (see Section 6.2.2).

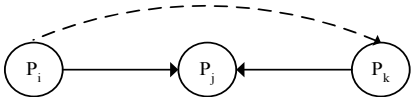

FIGURE 6.6: Peer Departure

6.3 Resource Description

Linkbases are one of the most essential resources in the DDLS. The accurate description of such resources facilitates satisfying the link service request, for instance a request for link retrieval. Section 4.4.1 proposed the description of linkbases in a centralised P2P system through encoding the metadata of links in the XML model and syntax, and re-

source discovery is carried out based on the metadata of links. This section advocates that, not only the links in a linkbase, but also the linkbase itself could be depicted in a certain manner for ease of meeting link service requests. Resource discovery is therefore performed based on the metadata of linkbases instead of that of links. This approach leads to coarser granularity of resource representation in the DDLS but improved scalability. This is feasible because the requirement for a semantic search mechanism (see Section 5.4.1) does not impose fine granularity upon the unstructured P2P DDLS.

The XML model and syntax are demonstrated, see Section 6.3.2, less suitable for describing linkbases in an unstructured P2P DDLS. Resource description will instead utilise RDF which appears very promising for accomplishing resource description in a link service with decentralisation of control. RDF is presented and its use in describing the DDLS linkbases is explored in the following subsections.

6.3.1 Resource Description Framework (RDF)

The Semantic Web (see Section 1.3) is an extension to the current Web and based on RDF standards and other standards yet to be defined. As a foundation for modelling and interchanging metadata about the resources on the Web, RDF aims to provide interoperability between applications that exchange machine understandable information. The characteristic accords with the design objective of the DDLS to describe linkbases as resources and facilitate resource sharing among various linkbase owners dispersed globally.

The basic RDF model is designed to represent named properties and property values. It is a syntax neutral way of representing RDF expressions and currently RDF relies on the support of XML. There are three kinds of objects in the basic data model: resources, properties and statements. Resources are identified by a URI reference plus optional fragment identifiers. The description of resources is partly represented by properties which are thought of as attributes of resources and convey the relationship between resources. Statements are assertions about resources with named properties and values. Resources, properties and values in a statement are called subject, predicate and object, respectively.

6.3.2 DDLS Resource Description

The XML model and syntax were chosen in Section 4.4.1 to present metadata of links in the centralised P2P DDLS. Elements, such as *description*, *endpoint* and *url*, were employed to convey the primary content of the document a link in the linkbase connects, anchors of a link, etc. However, the XML model is less advanced than the RDF model in the following aspects (in decreasing order of importance to resource description in the DDLS):

- **No associated semantics:** XML centers on being a data format standard, whereas RDF attempts to bring meaning, or semantics, to the data represented and can also represent relationship.
- **Complex query:** The representation of a fact in XML can be done in a large number of ways, which makes querying the XML logical tree difficult since all sets of possible representations of the fact should be converted into one statement. However, RDF does not bear this drawback since it specifies a standard way of representing facts by statements. Without any unambiguity, many representations of a fact in XML always lead to the same RDF tree, thereby querying RDF is much easier.
- **Dependability on schema:** The modification made to the schema, for instance, adding or removing an element, may invalidate a query that is based on the structure of the document.
- **Inability to enable computers to infer or deduce:** XML alone has no facilities to describe a vocabulary. However, when using a RDF model to represent data, one can either use existing vocabularies or creating his/her own ontologies. The combination of a RDF model and associated ontologies enables computers to discover the semantics of data and to infer or deduce facts.
- **Reliance on a common syntax for two applications to communicate:** XML alone requires that two parties to agree on a common syntax for communication. In contrast, using the RDF model allows two parties to communicate with different syntax through the concept of equivalence.
- **Less meaningful element name:** A meaningful element name is a crucial hint for human readers. Without a reference to the schema, nothing except the document structure can be deduced from an XML document, whereas in RDF, the element *description* and its attribute *about* point out where the identification of the resources being described can be located.

The DDLS is an *open link service* which serves hyperlinks by retrieving them from linkbases on demand. Each linkbase maintains a list of hyperlinks related to abstract concepts. This allows characterising a peer's linkbases on the basis of concepts in terms of a topic vector. The RDF model is employed to represent the DDLS linkbase. The RDF description of a linkbase can be augmented with related information, such as location and type, and this information is encoded in sets of triples.

```
<?xml version="1.0" encoding="UTF-8"?>
<rdf:RDF xmlns:rdf=http://www.w3.org/1999/02/22-rdf-syntax-ns#
    xmlns:lb="http://www.semanticweb.com/rdf/linkbase-ns#">
    <rdf:Description about="http://www.semanticweb.com/linkbase/research/linkbase.xml">
        <rdf:type resource="http://www.semanticweb.com/rdf/linkbase-ns#Linkbase" />
        <lb:topic>theory</lb:topic>
    </rdf:Description>
</rdf:RDF>
```

FIGURE 6.7: An Example of Using the RDF Model to Represent the DDLS Linkbase

An example DDLS linkbase represented by the RDF model is demonstrated in Figure 6.7. It indicates that the resource being described is *http://www.semantic web.com/linkbase/research/linkbase.xml*. The type of the resource is defined as another resource *http://www.semanticweb.com/rdf/linkbase-ns#Linkbase*. The primary content of this resource is *theory*.

Each peer in the DDLS holds an incomplete view of all the resource information available in the system. To enable resource sharing among peers, the DDLS needs a resource discovery mechanism which takes advantage of resource description that individual peers publish. In the following section, this issue will be investigated.

6.4 Resource Discovery

Resource discovery in the DDLS revolves around the location of desired linkbases. The centralised P2P DDLS in Section 4.4.4 employs a service directory to effect resource discovery. However, such a centralised mechanism is not applicable to an unstructured P2P DDLS which is characterised by decentralisation of control and aims to discover resources with matching semantics. In this section, the concept of the DDLS semantic search is introduced, which is followed by a distance-based semantic search algorithm devised for resource discovery in the unstructured P2P DDLS (Zhou et al. 2003).

6.4.1 DDLS Semantic Search

Resource discovery in the DDLS context locates linkbases that satisfy queries posed by users. The simplest case for the resource discovery mechanism performs a keyword-based search. The keyword-based search is carried out by comparing the query expression against the description of available linkbases. The search returns all linkbases whose descriptions match the query expression on a syntactic basis. As already mentioned in Section 6.3.2, each linkbase is associated with abstract concepts and a keyword-based search will restrict the search result to linkbases with matching topics rather than those with matching concepts. This limitation can be overcome by a semantic search. The semantic search discovers not only linkbases which are explicitly specified by the topics in a query, but also those which are conceptually related. The enabling techniques, such as ontologies and inference logic, provide a way to identify the match on which the semantic search relies. Related work on semantic search can be found in Appendix A.

6.4.2 Major Assumptions of the Semantic Search Algorithm

The semantic search algorithm to be presented is based on several assumptions as follows.

1. *Moderate number of participants*
   The typical scenario that the DDLS applies to involves a small number (hundreds) of associated participants (peers).

2. *Capability to identify semantic relationships*
   The environment provides each peer with a capability to identify the semantic relationship between topics representing the primary content of resources, such as 'being semantically related'. The semantic similarity can be described in different ways and the existence of such a mechanism is assumed. It may exist in the form of a controlled vocabulary or may be based on inference logic, or otherwise.

3. *Statically defined relationship among topics*
   The algorithm assumes that the semantic relationship among topics is statically defined and does not cater for environments in which the semantic relationship can be constantly redefined.

6.4.3 Query Mechanism: Topic Query and Associated Operations

The DDLS supports queries with conjunctive and disjunctive operations on query predicates – in this work topics are used to define query predicates. A query expression is represented by conjunctive/disjunctive operations on groupings of topics. Figure 6.8 presents an example RDF topic query. The extensibility of RDF is utilised to define a tag <rdfsq: equal> that represents the state of being semantically related. Tags, such as <rdf:Integer> and <rdf:String>, can be employed to identify primitive data types. The query result needs to return all the linkbases which are semantically related to both 'Topic A' and 'Topic B' or 'Topic C'.

```
1: <rdfq:rdfquery>
2:   <rdfq:From eachResource="http://www.ddls.com/
3:    linkbases/collaborative_environment_x/peer_linkbase">
4:    <rdfq:SELECT>
5:      <rdfq:Condition>
6:        <rdfq:and>
7:          <rdfsq:equal>
8:            <rdfq:Property name="lb:topic"/>
9:            <rdf:String>Topic A</rdf:String>
10:         </rdfsq:equal>
11:         <rdfsq:equal>
12:           <rdfq:Property name="lb:topic"/>
13:           <rdf:String>Topic B</rdf:String>
14:         </rdfsq:equal>
15:       </rdfq:and>
16:       <rdfq:or>
17:         <rdfsq:equal>
18:           <rdfq:Property name="lb:topic"/>
19:           <rdf:String>Topic C</rdf:String>
20:         </rdfsq:equal>
21:       </rdfq:or>
22:     </rdfq:Condition>
23:   </rdfq:SELECT>
24: </rdfq:From>
25:</rdfq:rdfquery>
```

FIGURE 6.8: The Typical Specification of DDLS Topic Queries

All topics in a conjunctive predicate need to be satisfied by the description of linkbases simultaneously, whereas those in a disjunctive predicate can be evaluated against the description of linkbases respectively. The result of a disjunctive query in the DDLS is typically generated by merging the results of conjunctive sub-queries. For instance, in Figure 6.8, the query is initially split into two sub-queries, each of which contains a conjunct, wrapped in two separate messages. One sub-query is constructed

by statements from line 7 to 14 with italics typeface and the other from line 17 to 20. Both the query and its sub-queries are assigned with Universally Unique Identifiers (UUIDs) (The Open Group 1997). Suppose that the query identifier is *2fac1234-31f8-11b4-a222-002035b29092*, which is inherited by both of its sub-queries. The first sub-query with sub-query identifier *58f202ac-22cf-11d1-b12d-002035b29092* returns all the linkbases having related topics with both 'Topic A' and 'Topic B'. The second one with sub-query identifier *5a389ad2-22dd-11d1-aa77-002035b29092* fetches all the linkbases possessing related topics with 'Topic C'. Results of sub-queries returned to the query originator will be merged given the same query identifier *2fac1234-31f8-11b4-a222-002035b29092*.

6.4.4 Distance-based Semantic Search Algorithm

The search algorithm uses the distance α^{dist} as shown in Table 6.1 that represents the proximity of resources of a pair of neighbours. Based on the proximity, a peer further propagates queries to either some or all of its neighbours. The details of the algorithm are explained as follows. Any participating peer can initiate a semantic search query. The query is evaluated against the initiator's cached information to determine the distance between the query expression and the cached information about the neighbours. If the query evaluator finds a match, it routes the query to the associated peers. A match means there is an overlap between the query topics and the topics in an entry of the published topic list of the query evaluator. In case no match has been found, the query is propagated to all neighbours of the current query evaluator. Subsequently, the query will be successively evaluated by each of the recipient peers. The number of hops for query propagation is limited by the life time of the query, expressed by a TTL tag as used in Gnutella. Query matches are directly routed back to the query initiator. The algorithm for processing queries at p_i is presented in Figure 6.9.

6.5 Simulation

The aim of this section is to present a series of experiments which simulate resource discovery in the DDLS to investigate the behaviour and measure the performance of the search algorithm that resource discovery relies on. After giving a brief introduction to the simulator which was employed by all experiments in this work, this section presents a factor that is closely associated with simulation: topic distribution. This is followed by the description of the metrics in performance measurement and all major issues

Notations:

Same as in Figure 6.1. In addition,

Set of query identifiers:	Q
An incoming topic query instance of p_i *including a query identifier and its topics (defined later):*	$(q_i^{in},\ T_i^{query}),\ with\ q_i^{in} \in Q$
Topics of a topic query instance of p_i*:*	$T_i^{query} = \{t \mid t \in T, t \times t \subseteq T\}$
Forwarding an incoming query (q_i^{in}, T_i^{query}) *from* p_i *to* p_j*:*	$p_i.forwardQuery(p_j, (q_i^{in}, T_i^{query}))$

Initial settings:

$Online = true;$
$Allow\_queries = true;$
$Overlap = false.$

Algorithm for query processing at p_i:

For $u = |E_i|$ *downto* 1
  If $\alpha_{i,u}^{dist} \geq |T_i^{query}|$*, then*
    If $T_i^{query} \subseteq T_{i,u}^{common}$*, then*
      $Overlap = true;$
      For each $p_w \in id_{i,u}^{peer}$
        $p_i.forwardQuery(p_w, (q_i^{in}, T_i^{query})).$
If $Overlap \equiv false$*, then*
  For $u = |E_i|$ *downto* 1
    For each $p_w \in id_{i,u}^{peer}$
      $p_i.forwardQuery(p_w, (q_i^{in}, T_i^{query})).$

FIGURE 6.9: Algorithm for Query Processing at p_i

regarding the search algorithm that needs to be explored through simulation. Finally, all the experiments are presented.

6.5.1 Overview of the Simulator

A simulator, which was designed to provide the operational conditions of the DDLS for testing and evaluation purposes, is provided and utilised in all the simulation described in this work. One of the many basic requirements for the simulator is to set up the entire experimental environment with each node/peer bearing equal capability in terms of computing power.

The simulator is implemented in Java and deployed on a single machine. Each peer is effected as an object and communicates with one another through message passing. A single message queue is provided which hosts the incoming and outgoing messages of all peers to be processed. Realising the single message queue through a thread for all the peers instead of allowing each peer to have a message queue, alleviates the heavy use of the CPU caused by multiple executions of all threads. In a time-sliced system, as used in the DDLS simulation, the CPU is divided into time slots and each of the equal-and-highest priority threads is iteratively given a time slot in which to run. The Java scheduler chooses the following thread to execute in a round robin fashion. This does not guarantee that the requirement for equal capability of peers is enforced. The simulator is therefore so implemented that peers process the same amount of messages in each allocated time slot in turn. This is one of the methods that enforce equal capability shared by all peers.

6.5.2 Topic Distribution

The DDLS search mechanism locates semantically related resources. Therefore, the distribution of individual topics which represent the primary content of resources is not of interest. Instead, topics are grouped by semantics and the distribution of such topic groups are what should be utilised to study the DDLS search. From now on, topic popularity (or probability) defined later refers to the popularity (or probability) of topic groups each of which has distinct semantics unless indicated otherwise.

First, the Zipf's distribution of topics is investigated. Zipf's law (Zipf 1949) is named after the Harvard linguistic professor George Kingsley Zipf (1902-1950). It states that the frequency of occurrence of some event (P), as a function of the rank (i) that is determined by the frequency of occurrence, is a power-law function $p_i \propto \frac{1}{i^\alpha}$ with α close to unity. It has been shown that Zipf's distribution characterises the use of words in a natural language, for instance English.

The term 'topic popularity' used in simulation represents how popular a topic/-topics is/are in terms of the number of peers holding it/them. Let t_i be the topic popularity of the i'th topic in a Zipf's distribution.

$$t_i \propto \frac{1}{i^\alpha}$$

where $\alpha = 1$. The Zipf's distribution of topics in a system of 100 peers that share 100 various topics is demonstrated in Figure 5.1.

To compare against the Zipf's distribution, the uniform distribution of topics is chosen in which each peer is shared by the same number of peers. The term 'topic probability' used in simulation denotes the percentage of peers that possess the topic(s) compared to all peers in the system. Let t_i be the topic probability of any topic in a uniform distribution.

$$t_i = C$$

where C is a constant.

6.5.3 Metrics and Issues

Algorithm performance is a complex aspect that could be measured by various indices, for instance search speed and accuracy, the number of messages sent, system load and resource consumption. In the context of the DDLS, performance issues are primarily measured by the following metrics:

- Hops: delay in finding all answers as measured in the number of hops, also known as path length;
- Recall: the percentage of matches that can be found;
- Broadcast rate[4]: the time of broadcast carried out by all peers to propagate queries over a period of time.

Elements which may have an effect on hops, recall and broadcast rate will be investigated. It is conjectured that exploring answers to the following questions would be helpful in understanding the search algorithm.

1. What is the behaviour of the semantic search when single or multiple topics are involved?
2. What is the relationship between the amount of information a peer should cache about its neighbours and the search performance?
3. Does the resource (or topic) distribution have an impact on the search performance?

[4]This metric can be used to estimate the consumption of network resources during resource discovery, for instance, the number of potential messages generated with each query.

4. Should the peer network remain unchanged or be re-organised to improve the search performance? If so, what techniques should be utilised to guide the re-organisation process?

6.5.4 Single Topic Search

The simulation on single topic search among topics following a Zipf's distribution and uniform distributions was conducted, respectively. The experimental settings and discussion are presented in this section.

Experiment 1

Experimental settings The first experiment examined the relationship between the cache rate and the number of hops required to achieve the maximum recall. The cache rate represents the percentage of peers whose topic information is in the cache compared to all peers in the system. This experiment was performed with 100 peers in a controlled environment, where the distribution of topics was kept constant throughout the experiment and the number of peers in the system was restricted. Each peer could cache the topic information of a specified percentage of all peers and randomly choose a list of topics from a global list of 100 entities, ensuring that the topics of all peers followed a Zipf's distribution and uniform distributions, respectively. The cache rate was varied from 1%, 2% to 90%. The fifteenth most popular topic, shared by 5 peers (out of 100) in the system, was chosen as the query topic for the experiment with a Zipf's distribution. A topic with the topic probability of 5%, i.e. shared by 5 peers out of 100, was randomly chosen from the global list to formulate a query for the experiment with a uniform distribution. In both experiments, a topic shared by the same number of peers (5) was chosen to formulate a single topic search. This was meant to contrast the behaviour and performance of the search involving one distribution with those involving the other.

Discussion The results in Table 6.2 and Table 6.3 show that, regardless of the distribution the topic in a query is associated with, the cache rate is inversely proportional to the average number of hops that are needed to achieve the maximum recall. It is observed that except the cases in which the cache rate equals 1% or 2%, i.e. each peer only caches the topic information from one or two of its neighbours, the resource discovery mechanism in the DDLS can lead to a satisfactory recall (at least 98%) within the experimental settings. The cache rate being 1% and greater only guarantees that each peer is aware of at least another peer (a neighbour), whereas it is not assured that each peer is known by at least another peer (a contact). Therefore, the maximum recall

cannot always reach 100%. The data collected in Table 6.2 and Table 6.3 is the average result from multiple executions (20) of the experiment in which different topologies of the peer network were generated.

Cache rate	1%	2%	3%	4%	5%
Average number of hops	0.01	7.36	5.85	5.09	4.41
Maximum recall	1%	80%	100%	100%	100%
Cache rate	10%	20%	30%	40%	50%
Average number of hops	2.83	2.05	2.00	2.01	1.98
Maximum recall	100%	99%	98%	99%	100%
Cache rate	60%	70%	80%	90%	
Average number of hops	1.92	1.84	1.63	1.40	
Maximum recall	100%	100%	100%	100%	

TABLE 6.2: Relationship between the Cache Rate and the Average Number of Hops (Zipf's Distribution)

Cache rate	1%	2%	3%	4%	5%
Average number of hops	2.47	8.86	6.14	4.91	4.13
Maximum recall	9%	80%	100%	100%	100%
Cache rate	10%	20%	30%	40%	50%
Average number of hops	3.09	2.06	1.98	2.00	1.96
Maximum recall	100%	100%	98%	99%	100%
Cache rate	60%	70%	80%	90%	
Average number of hops	1.98	1.87	1.66	1.46	
Maximum recall	100%	100%	100%	100%	

TABLE 6.3: Relationship between the Cache Rate and the Average Number of Hops (Uniform Distribution)

Experiment 2

Experimental settings The second experiment aimed to explore the properties of the semantic search algorithm of the DDLS in which topics follow a Zipf's distribution. It was carried out over the first nineteen most popular topics. In a peer network consisting of 100 peers, a Zipf's distribution of 100 topics yields 19 bands, see Figure 5.1, each of which is occupied by topics that are shared by the same number of peers. The cache rate was kept at 5% throughout the experiment and all the other experimental settings were retained as in Experiment 1. To ensure each peer has at least one neighbour, the cache rate should be 1% or greater. However, it is shown that a very low level of the cache rate results in unacceptable recall, for instance 1% in a Zipf's distribution and 9% in uniform distributions, which does not demonstrate the typical behaviour and performance of the semantic search but represents the extreme case. Therefore, this experiment and all those presented later use 5% as the cache rate because such a relatively low cache rate is more realistic for a peer network which allows for a wide range of the number of peers.

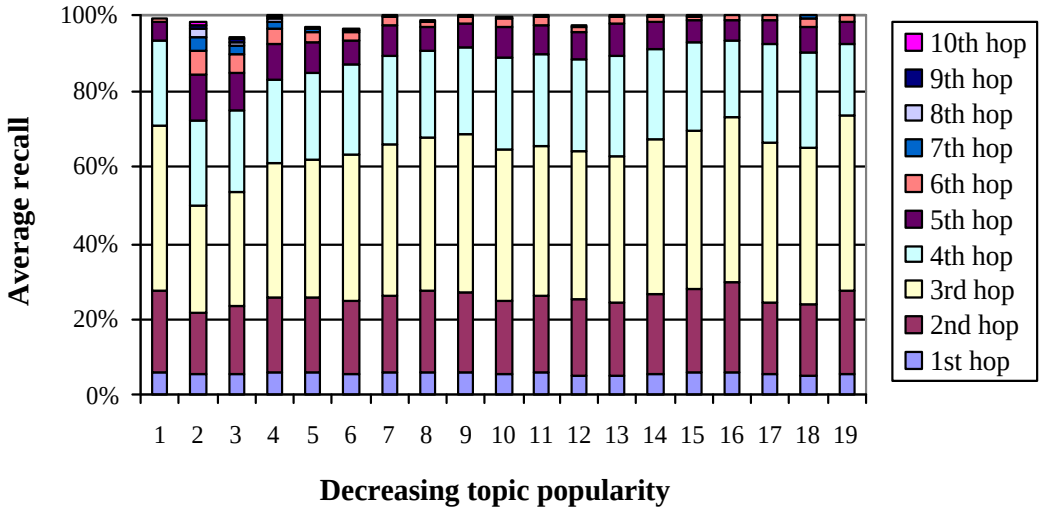

FIGURE 6.10: Average Recall Level at Progressive Hop Counts in Single Topic Search (Zipf's Distribution)

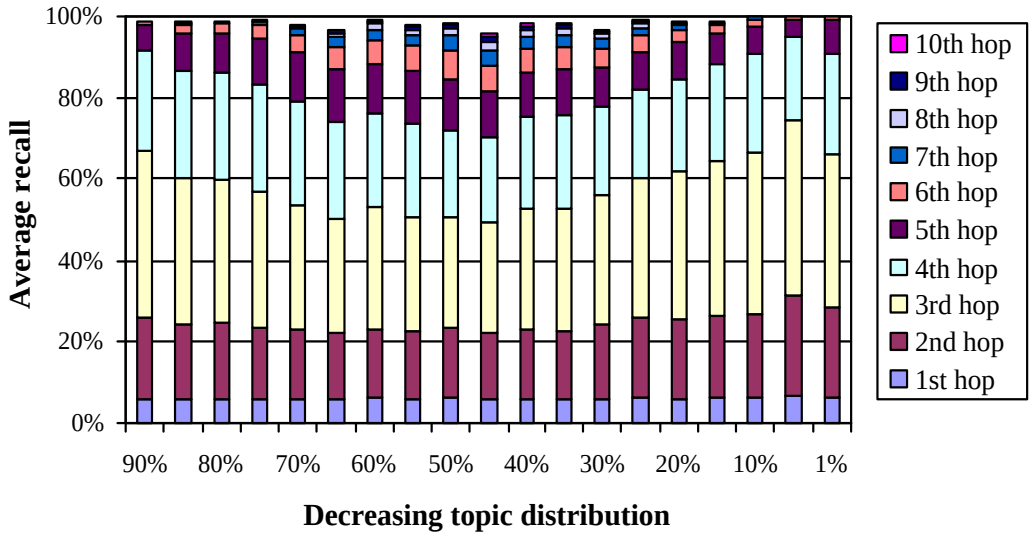

FIGURE 6.11: Average Recall Level at Progressive Hop Counts in Single Topic Search (Uniform Distribution)

Discussion The recall level gained as the hop count increases in search of topics with different popularities in a Zipf's distribution is plotted in Figure 6.10. It is shown that the second most popular topic which is shared by 50% of all peers in the system is accompanied by the lowest recall level at almost every hop. The discovery indicates that a search for that topic will lead to the greater average number of hops to achieve

a certain level of recall, compared to a search for any other topic. Table 6.4 further demonstrates this with the maximum average number of hops produced by a search for the second most popular topic. Thereafter, the average number of hops that are needed to reach the maximum recall is in proportion to topic popularity, with the least popular topic resulting in the least average number of hops to achieve the maximum recall. This phenomenon can be explained as follows. The number of hops needed to achieve the maximum recall is subject to the probability of the query topic. On the one hand, the discovery of a certain number of instantiations of a topic with little probability results in more hops. On the other hand, according to the search algorithm (see Section 6.4.4), little probability of the query topic yields little probability of overlap between peers, thus triggering a higher broadcast rate which implies less hops to locate the same number of instantiations. Let x be the probability of a query topic and $f(x)$ be the number of hops to achieve the maximum recall: $f(x) = d(x) * b(x)$. $d(x)$ and $b(x)$ are associated with the two aspects as analysed above with $d(x)$ being a decreasing function and $b(x)$ being an increasing function of x over [0, 1]. Because $f(x)$ is continuous over [0, 1] and $f(0)$ = $f(1)$ = 0, there must be at least a single point x at which $f(x)$ has its maximum.

Experiment 3

Experimental settings The third experiment was set up to investigate the properties of the semantic search algorithm of the DDLS in which topics follow uniform distributions. It was performed with the topic probability ranging from 1%, 5%, 10% to 90%. Again, the cache rate was kept at 5% throughout the experiment and all the other experimental settings were kept as in Experiment 1.

Discussion Figure 6.11 shows that the recall level increases at progressive hop counts in search of topics with different probabilities in uniform distributions. Within the first 10 hops, a search for topics with a probability of 45% yields the lowest recall level at almost every hop. This phenomenon is analogous to the discovery in Figure 6.10 that the search for the second most popular topic (shared by 50 peers out of 100) results in the lowest recall level at approximately every hop. Table 6.5 reveals that the topic probability of 55% is related to the maximum average number of hops to obtain the maximum recall. The average number of hops increases before the topic probability reaches 55% and decreases thereafter. In contrast to the experiment with topics from a Zipf's distribution (see Experiment 2) in which 50% is the turning point for all observations, this experiment (with topics from uniform distributions) shows that the turning point exists in the range of [45%, 55%]. It is speculated that both results should be consistent with each other and the inconsistency in the results is due to the limited experimental conditions.

Topic popularity	1	2	3	4	5
Average number of hops	5.47	8.10	6.33	5.84	5.20
Maximum recall	99%	99%	94%	100%	97%
Topic popularity	6	7	8	9	10
Average number of hops	5.01	5.00	4.74	4.71	4.85
Maximum recall	96%	100%	99%	100%	99%
Topic popularity	11	12	13	14	15
Average number of hops	4.75	4.60	4.70	4.44	4.22
Maximum recall	100%	98%	100%	100%	100%
Topic popularity	16	17	18	19	
Average number of hops	4.01	3.96	3.77	3.06	
Maximum recall	100%	100%	100%	100%	

TABLE 6.4: Average Number of Hops to Achieve the Maximum Recall (Zipf's Distribution)

Topic probability	90%	85%	80%	75%	70%
Average number of hops	5.51	6.04	6.04	6.32	6.81
Maximum recall	99%	99%	99%	99%	98%
Topic probability	65%	60%	55%	50%	45%
Average number of hops	7.59	7.66	8.28	7.72	7.83
Maximum recall	97%	99%	98%	99%	97%
Topic probability	40%	35%	30%	25%	20%
Average number of hops	6.98	6.60	6.16	5.79	5.44
Maximum recall	99%	99%	97%	99%	99%
Topic probability	15%	10%	5%	1%	
Average number of hops	5.00	4.88	4.03	3.08	
Maximum recall	99%	100%	100%	100%	

TABLE 6.5: Average Number of Hops to Achieve the Maximum Recall (Uniform Distribution)

6.5.5 Multiple Topic Search

The simulation on multiple topic search was performed to explore the search performance under the circumstances in which topics with distinct popularities are involved in a single semantic search.

Experiment 1

Experimental settings The settings were kept as in Experiment 2 of single topic search. However, a multiple topic query involves two topics with different popularities in a Zipf's distribution. Each multiple topic search is based on the second most popular topic and another less popular one[5]. For instance, the first multiple topic search relates

[5]The most popular topic is not used because it is shared by all peers in the network and using it with another less popular topic in a multiple topic search resembles a single topic search for the less popular topic.

Topic Popularity	Average Maximum Recall	Average Number of Hops	Average Matches	Average Broadcast Rate
2	99%	8.10	50.00	
3	94%	6.30	33.00	
(2, 3)	99%	5.30	18.40	87.30
2	99%	8.10	50.00	
4	100%	5.84	25.00	
(2, 4)	100%	4.88	11.40	93.01
2	99%	8.10	50.00	
5	97%	5.20	20.00	
(2, 5)	100%	4.88	10.6	95.60
2	99%	8.10	50.00	
6	96%	5.01	17.00	
(2, 6)	100%	4.62	9.00	96.61
2	99%	8.10	50.00	
7	100%	5.00	14.00	
(2, 7)	100%	4.37	6.40	97.60
2	99%	8.10	50.00	
8	99%	4.74	13.00	
(2, 8)	100%	4.46	5.40	97.81
2	99%	8.10	50.00	
9	100%	4.71	11.00	
(2, 9)	100%	4.23	6.40	98.21
2	99%	8.10	50.00	
10	99%	4.85	10.00	
(2, 10)	100%	4.00	5.00	98.61
2	99%	8.10	50.00	
11	100%	4.75	9.00	
(2, 11)	100%	4.25	5.40	98.00
2	99%	8.10	50.00	
12	98%	4.60	8.00	
(2, 12)	100%	3.89	3.60	99.20
2	99%	8.10	50.00	
13	100%	4.70	7.00	
(2, 13)	100%	3.88	3.00	99.01
2	99%	8.10	50.00	
14	100%	4.44	6.00	
(2, 14)	100%	3.68	2.40	99.21
2	99%	8.10	50.00	
15	100%	4.22	5.00	
(2, 15)	80%	2.97	1.80	99.41

TABLE 6.6: Multiple Topic Search based on Two Topics with Distinct Popularities in Zipf's Distribution

Topic Popularity	Average Maximum Recall	Average Number of Hops	Average Matches	Average Broadcast Rate
2	99%	8.10	50.00	
16	100%	4.01	4.00	
(2, 16)	80%	2.80	2.00	99.80
2	99%	8.10	50.00	
17	100%	3.96	3.00	
(2, 17)	100%	3.26	1.20	99.41
2	99%	8.10	50.00	
18	100%	3.77	2.00	
(2, 18)	20%	0.56	0.20	99.60
2	99%	8.10	50.00	
19	100%	3.06	1.00	
(2, 19)	60%	1.74	0.60	99.41

TABLE 6.7: Multiple Topic Search based on Two Topics with Distinct Popularities in Zipf's Distribution (Continued)

to the second and the third most popular topics, and the last multiple topic search involves the second and the nineteenth most popular topics. The cache rate was set as 5%. The network topologies generated in single topic search for individual topics (with the same experimental settings) were employed.

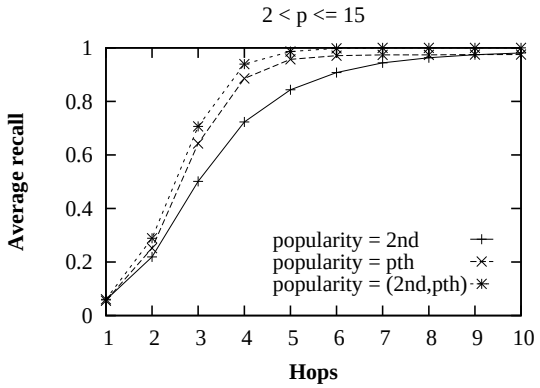

FIGURE 6.12: Average Recall Level at Progressive Hop Counts in Multiple Topic Search (Zipf's Distribution)

Discussion Table 6.6 and Table 6.7 present the result of the experiment with multiple topic search involving topics from a Zipf's distribution. It is observed that the average number of hops to achieve the average maximum recall decrease as the probability (or popularity) of the component topics (see the column titled 'Average Matches') reduces. This is analogous to the pattern that exists in the simulation on single topic search, see Table 6.4. However, the probability of the component topics chosen for

each multiple topic search is less than 50%[6] in this experiment. Therefore, only the situation in which the average number of hops that are needed to achieve the maximum recall is in proportion to the probability of the component topics, can be observed. The average broadcast rate is inversely proportional to the probability (or popularity) of the component topics, because less popular query topics yield less probability of overlap between peers, thus triggering a higher broadcast rate to locate all the instantiations. The recall level obtained at progressive hop counts is plotted in Figure 6.12[7]. The result of all multiple topic searches carried out shows that the average recall level at each hop is inversely proportional to the topic probability (or popularity). Table 6.6 and Table 6.7 reveal that the behaviour of multiple topic search exhibits similarity to that of single topic search, see Figure 6.10.

Experiment 2

Experimental settings The settings were kept as in Experiment 3 of single topic search, except that each multiple topic query involves two topics from a uniform distribution. Again, the network topologies generated in single topic search for individual topics (with the same experimental settings) were employed.

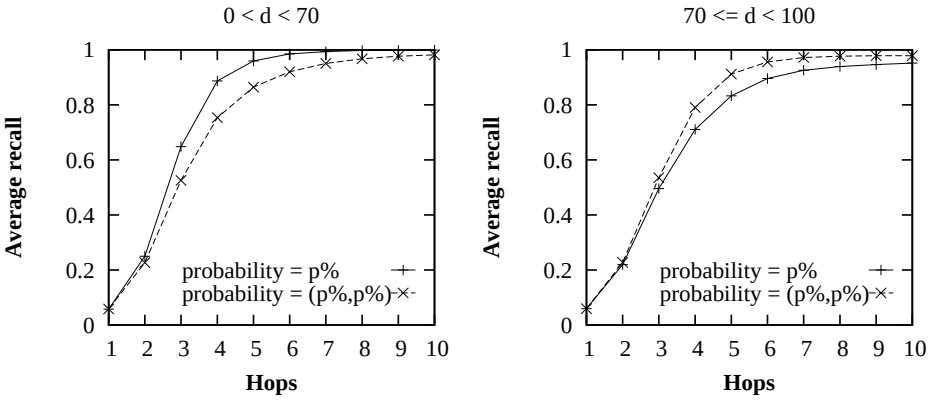

FIGURE 6.13: Average Recall Level at Progressive Hop Counts in Multiple Topic Search (Uniform Distribution)

Discussion Table 6.8 demonstrates the result of the experiment with multiple topic search involving topics from uniform distributions. It can be observed that the average number of hops to achieve the average maximum recall increases before the probabil-

[6]The maximum probability of the component topics is 18.40%.

[7]This figure is a schematic of a series of figures sharing the similar trend of each curve and the same relative relationship among the curves. Simulation results with $p > 15$ are not shown because they are not typical due to the little probability of the component topics, e.g. 2.00%, 1.20%, 0.20% and 0.60% in Table 6.7.

Topic Probability	Average Maximum Recall	Average Number of Hops	Average Matches	Average Broadcast Rate
90%	99%	5.51	90.00	
(90%, 90%)	98%	5.95	80.40	0.72
80%	99%	6.04	80.00	
(80%, 80%)	98%	7.05	63.80	12.73
70%	98%	6.81	70.00	
(70%, 70%)	96%	7.61	50.60	35.28
60%	99%	7.66	60.00	
(60%, 60%)	99%	6.90	34.20	65.84
50%	99%	7.72	50.00	
(50%, 50%)	99%	5.53	24.20	81.95
40%	99%	6.98	40.00	
(40%, 40%)	100%	5.24	16.20	89.40
30%	97%	6.16	30.00	
(30%, 30%)	99%	4.76	10.20	94.43
20%	99%	5.44	20.00	
(20%, 20%)	100%	3.75	4.00	99.00
10%	100%	4.88	10.00	
(10%, 10%)	98%	2.68	1.00	99.40
1%	100%	3.08	1.00	
(1%, 1%)	0%	0.00	0.00	99.21

TABLE 6.8: Multiple Topic Search based on Two Topics with Distinct Probabilities from Uniform Distributions

ity of the component topics drops below 50.60% and decreases thereafter, as the topic probability of the component topics (see the column titled 'Average Matches') reduces. The pattern is the same as can be seen in the simulation on single topic search, see Table 6.5. Again, the average broadcast rate is inversely proportional to the probability of the component topics. The explanation, which was given in the previous experiment for the relationship between the average broadcast rate and the popularity of the component topics in a multiple topic search for topics from a Zipf's distribution, is also applicable herein.

The recall level at progressive hop counts is plotted in Figure 6.13[8]. It is shown that the average recall level at each hop is proportional to the topic probability when the probability of each component topic is greater than 70%[9] and is inversely proportional to the topic probability when the probability of each component topic is less than 70%. This phenomenon is in accordance with the one observed in single topic search, see

[8]This figure is a schematic of a series of figures possessing the similar trend of each curve and the same relative relationship among the curves.

[9]Table 6.8 shows that the row with the topic probability in single topic search equal to 70% is a turning point.

Figure 6.11.

6.6 Understanding the Semantic Search through Simulation

Simulation, which covers 5 distinct but interrelated experiments, has been conducted on single topic search and multiple topic search so as to explore the behaviour and performance of the distance-based semantic search algorithm. The experiments investigated the Zipf's distribution and the uniform distribution of topics - the two most potential distributions if topics are sorted by popularity. The main metrics in performance evaluation include hops, recall and broadcast rate which are employed to describe the principal findings outlined as follows.

1. In a single topic search, the cache rate (except the extreme cases, such as 1% in the simulation) is inversely proportional to the average number of hops that are needed to achieve the maximum recall.

2. The search for a topic with a probability less than 50% yields the average number of hops needed to achieve the maximum recall inversely proportional to topic probability (or popularity). In contrast, the search for a topic with a probability more than 50% results in the average number of hops needed to achieve the maximum recall proportional to topic probability (or popularity). This phenomenon was observed in simulation with both Zipf's distribution and uniform distribution of topics.

3. If the single topic in a search is replaced with multiple topics, 2 topics for example, the observations mentioned above remain unchanged.

4. The average broadcast rate is inversely proportional to topic probability (or popularity).

While simulation on single topic search revealed the essential behaviour and performance of the semantic search algorithm, extending the simulation to multiple topic search was intended to confirm the universality of such properties. Examining both finding 2 and 3 described above, one would discover that:

1. Because topics with distinct semantics are considered independent of one another, the joint popularity (or the joint probability) of component topics should be used

in multiple topic search as the topic popularity (or probability) used in single topic search, when predicating the behaviour and search performance of the semantic search.

2. Topic popularity (or probability) of topics is what should be taken into account, other than the distribution of all topics, when discussing the behaviour and performance of the semantic search.

6.7 Summary

According to the requirements for an unstructured P2P DDLS as identified in Chapter 5, this chapter revised the resource description mechanism by adopting the RDF model and presented the formal representation of a distance-based semantic search algorithm. The meaning conveyed by *semantic search* in the context of the DDLS was highlighted. It is clear that the approaches of related work documented in Appendix A can be complement, instead of alternatives, to the DDLS semantic search. This is because the DDLS assumes the ability of identifying semantically related terms that all related work possesses, and focuses on an underlying mechanism that supports the semantic search in a P2P context that none of them is able to tackle. Simulation which involved different topic distributions and query profiles has been carried out to investigate the behaviour and performance of the semantic search algorithm. Many similarities have been observed from the simulation with a Zipf's distribution and uniform distributions of topics. The simulation result has given an answer to the first three questions raised in Section 6.5.3 that of the anticipated behaviour of the semantic search, the impact of the cache rate and resource (or topic) distribution on the search performance. The last question which relates to re-organisation of the peer network is left to be explored by the next chapter.

Chapter 7

Re-organising the DDLS Peer Network

7.1 Introduction

The DDLS peer network was constructed with a goal to facilitate resource sharing-based collaboration. In Chapter 6, essential understanding of the extent to which this goal can be accomplished with the help of devised mechanisms, has been obtained through simulation. This chapter investigates the way of further boosting the performance of resource discovery through re-organisation, an action that may occur throughout the lifetime of the peer network.

The aim of this chapter is to present the concept of re-organisation in the context of the DDLS peer network and describe the techniques that can be utilised for implementing re-organisation. This chapter introduces a data structure referred to as *query history* upon which re-organisation heavily relies. The exponential decay function and the naive estimator are proposed as re-organisation techniques and simulation that verifies the effectiveness of both techniques is described. This chapter discusses a further move, using the virtual overlap which helps reinforce the principle of the semantic search algorithm, to facilitate re-organisation. A utility equation is proposed to evaluate the gains in the search performance resulting from re-organisation. Finally, a review of re-organisation is presented which looks back at the origin, features and techniques of re-organisation, summarises the principle findings of simulation on re-organisation, and gives a complementary explanation for some phenomena observed in simulation.

7.2 Concept and Forms

Re-organisation of the peer network in the context of the DDLS is defined as *an act of altering the virtual neighbourhood of any peer in the network to optimise resource discovery*. It is typically triggered by the state change of any peer in the network which includes the (dis)appearance of a peer, resource update to a peer and update of neighbours of a peer.

Re-organisation (not including that results from peer arrival and departure) is not a subject that has been researched extensively by the P2P community. This is because the inherent organisational structure of P2P systems differs from one another. For instance, centralised P2P systems would not benefit from re-organisation because resource discovery in such systems relies heavily on a centralised directory and altering the network topology does not facilitate the location and retrieval of resources of interest. For structured P2P systems that adopt DHTs to index the search space for efficient lookup, re-organisation bears no significance either. Unstructured P2P systems are perhaps the most promising systems that would benefit from employing re-organisation, because they do not depend on a central directory and neither do they have a tight control over the network topology and the placement of resources. Little work which relates to re-organisation in unstructured P2P systems has been carried out and reported. This work aims to explore the way of improving the performance of resource discovery by proposing re-organisation in the DDLS peer network, and evaluate the approach through simulation (see Section 7.5.2 and Section 7.5.4).

The major forms of re-organisation in the DDLS peer network comprise the following classes, mainly differing from one another in terms of origin.

1. Passive re-organisation: Re-organisation results from either the arrival, departure or update of peers is referred to as passive re-organisation. In this form of re-organisation, peers take an inactive part in altering the network topology, because they are committed to the peer network which regulates that the topology of the peer network should reflect the up-to-date relationship between peers. Peer arrival and departure related re-organisation has been described in Section 6.2.3 and Section 6.2.4. Update related passive re-organisation is the focus of the work presented later in this chapter.

2. Proactive re-organisation: Peers may take the initiative to perceive and predict the potential information needs through some mechanism. They can therefore re-organise the network topology to enable those needs to be satisfied in the fu-

ture with high probability. This type of re-organisation is named proactive re-organisation and can benefit from the techniques for update related passive re-organisation.

3. Autonomous re-organisation: The fullest extent of autonomy is observed in this kind of re-organisation. It typically involves peers which adjust their resources in response to information needs. Thus, the relationship between the updating peers and their neighbours may no longer hold, which gives rise to a variation of the network topology.

When re-organisation is triggered, regardless of the form, involved peers will utilise the techniques proposed later in this chapter (see Section 7.5.1 and Section 7.5.3) to discover a set of neighbours which are most qualified to help achieve the objectives of re-organisation, and replace the current neighbours with the new ones.

Though the origin of re-organisation can be as diverse as shown above, the objectives of such re-organisation are always straightforward, i.e. to reflect the up-to-date relationship between peers and to deliver an improved performance in resource discovery through the alteration of the network topology.

In the DDLS, the knowledge of resources owned by each peer is restricted to its neighbours. Peers view a limited scope of the global information about resources in the system. There are two ways for information (about resources) sharing in such a network - sharing the topic information with neighbours as in Section 6.2.3, and utilising the query routing information. The routing information of queries that have been propagated from other peers, is maintained in a data structure called *query history*. The next section introduces the data structure and related operations.

7.3 Supporting the 'Query History' Data Structure

Query history is a collection of all queries a peer has encountered over a period of time. Peers may be able to predict the future information needs of others by analysing query history, which is subsequently accompanied by rational re-organisation.

Query history is realised as a FIFO (First In First Out) queue, see Figure 7.1. History information is discarded when the queue overflows. Let T denote the set of all the topics. Each entry of query history includes three fields: the query identifer, query topics and the arrival time of the query. The set of query identifiers is Q, the capacity of

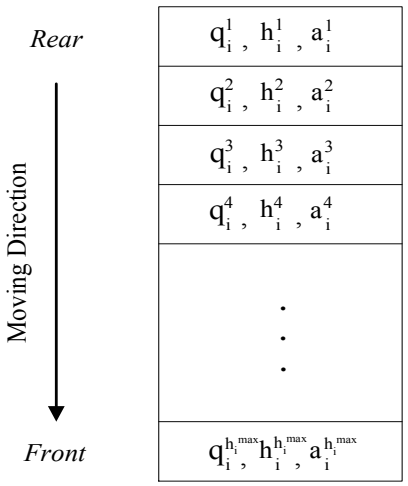

FIGURE 7.1: Query History of p_i at an Instant of Time

query history of peer p_i is h_i^{max} and the set of arrival time of queries is A. Query history of p_i can be represented by $H_i = \{(q_i^m, h_i^m, a_i^m) | q_i^m \in Q, h_i^m \in T, h_i^m \times h_i^m \subseteq T, a_i^m \in A, 0 < m \leq h_i^{max}\}$.

7.4 Criteria and Metric

One of the objectives of re-organisation is to optimise resource discovery. This entails the knowledge of resource supply and demand in the DDLS peer network. An analysis of the resource supply and demand yields two criteria which can be used to guide rational re-organisation.

1. Peers that share related topics should be incorporated into the same *cluster*[1].
2. Peers should be situated in the vicinity of those that would accommodate potential information needs with high probability.

The reason is explained as follows. Having neighbours with related topics allows a peer to identify others that accommodate similar information needs conveyed by queries. The peer can therefore propagate subsequent related queries and avoid costly broadcast.

[1]The term *cluster* is derived from the unsupervised clustering method which groups entities into clusters by the similarity of their features without any prior knowledge about the number of the clusters, which fits in with the DDLS clustering problem. However, because the resources of each peer are represented by a set of characteristic topics, a peer being incorporated into more than one cluster would occur frequently.

Key to the reduction of local broadcast is the local knowledge of *supply* available at neighbours. Meanwhile, a peer can identify the potential information needs of others from its query history and make peers with desired topics its neighbours through re-organisation. In the proposed techniques (see Section 7.5.1 and Section 7.5.3), a peer analyses the potential *demand* of others based on their previous information needs exhibited in query history and chooses appropriate peers that fulfill the requirement as neighbours.

The term *usefulness* is employed to represent the relative extent to which a peer should be considered as a neighbour of another peer during re-organisation. Assume that a candidate neighbour p_j (with respect to p_i) publishes a topic list T_j. Let $\varepsilon_{i,j}$ be a metric which represents the information needs exhibited in query history of p_i. Different techniques will be utilised to estimate the value of $\varepsilon_{i,j}$ in the following sections. Also, let $\eta_{i,j}$ denote the extent to which p_j would match the queries that p_i can satisfy:

$$\eta_{i,j} = \frac{|T_i \bigcap T_j|}{|T_i|}.$$

Let $\iota_{i,j}$ represent the usefulness of p_j with respect to p_i

$$\iota_{i,j} = \sqrt{(\kappa_1 \varepsilon_{i,j})^2 + (\kappa_2 \eta_{i,j})^2} \tag{7.1}$$

where κ_1 and κ_2 are constant coefficients associated with query history and the cached topic information, respectively. The quantitative relationship of the significance between $\varepsilon_{i,j}$ and $\eta_{i,j}$ can be adjusted by assigning specific values to κ_1 and κ_2.

Within the capacity of its cache, p_i only keeps peers with the greatest value of ι as its new neighbours during re-organisation, and discards the rest.

For ease of comprehension, it is necessary to interpret the physical significance of Equation 7.1. Figure 7.2 geometrically demonstrates the usefulness of candidate neighbours p_j and p_k with respect to p_i. The usefulness of candidates is quantitatively represented by the radius of a series of coaxial circles. The longer the radius is, the more useful a candidate is. Equation 7.1 can reflect such a relationship even without the coefficients κ_1 and κ_2. However, in some cases, a unit of change of ε may yield a different alteration in the usefulness metric ι compared to that caused by a unit of change of η. A unit of change of a variable refers to the difference between a pair of its consecutive permissible values. For instance, the (non-)existence of a topic in the overlap or in query history yields a unit of change of η or ε. It is implied that in usefulness decision the significance of a topic in the overlap may be rated differently relative to that of a topic in query history. To discard such a constraint, κ_1 and κ_2 are used to

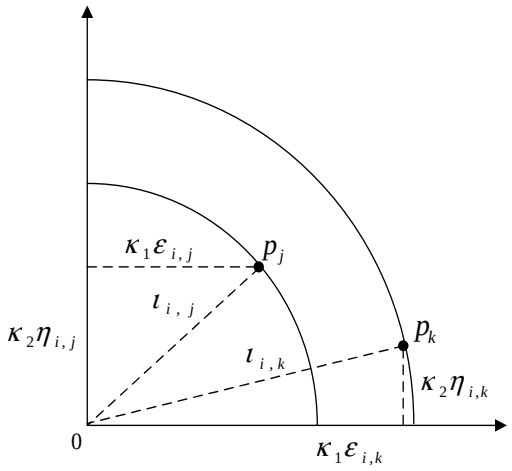

FIGURE 7.2: Usefulness of Candidates for Neighbours of p_i

leverage the contribution from both aspects. The performance of resource discovery affected by various relationships between κ_1 and κ_2 will be explored in Section 7.5.2 and Section 7.5.4.

7.5 Enabling Techniques

The standard LRU/LFU (least recently/frequently used) algorithm (Silberschatz and Galvin 1994) has gained wide acceptance in such domains as the caching strategies in Web proxies and the page replacement policy in operating systems. By keeping track of the time/number that each object or page is referenced, the LRU/LFU decides which object or page to be replaced with a new one, when the cache reaches its capacity or a page not resident in the main memory is needed by an active process. Such algorithms are of particular interest to re-organisation in the DDLS. The issue in re-organisation is to decide which queries should be taken into account to compute the usefulness of neighbours while ignoring the others, which is similar to that in both caching strategies and the page replacement policy, i.e. using some technique to measure the superiority (or suitability) of some entities over the others.

The fundamental assumption behind the LRU/LFU algorithm is that the recent past will approximate the immediate future. Therefore, the caching strategies for example, can deduce the likelihood of future references to the cached entities based on the observations of either their size, recency or frequency. The re-organisation techniques

presented in the following subsections make the same assumption and employ different strategies to support the usefulness decision, however.

The exponential decay function is used to vary the significance of queries in query history based on both the time (recency) and the number of occurrence (frequency), whereas the naive estimator diversifies the significance of queries in query history in terms of the probability of query topics (analogous to frequency). The former strategy subsumes both the LRU and the LFU while the latter is a variant of the LFU.

7.5.1 Exponential Decay Function-based Usefulness Decision

A straightforward approach to distinguish the instances of query topics with various times of occurrence is to allocate various weights to these instances. An exponential decay function $W(S(q))$ can satisfy the requirement. $W(S(q))$ is a weight function of $S(q)$ which is in turn a sequence function of an incoming query q. The following phenomena can be observed in query history to which an exponential decay function is applied:

- More recent query topic instances are always awarded higher weights for their occurrences;
- More significant difference between two query topic instances occurring more recently can be observed than that between two topic instances, with the same distance in sequence, occurring less recently.

Let h_i^{max} be the capacity of query history H_i of p_i. The exponential decay function takes the following form:

$$W(S(q)) = e^{-S(q)}.$$

For sequence function $S(q)$, the following equations hold.

$$S(q_i^1) = 1, S(q_i^2) = 2, \ldots, S(q_i^{h_i^{max}}) = h_i^{max}$$

where $q_i^1, q_i^2, \ldots, q_i^{h_i^{max}}$ are a sequence of incoming queries ordered by the arrival time with q_i^1 being the most recent incoming query.

Suppose p_i is going to decide how useful a candidate neighbour p_j is. Given an interval I, the metric $\varepsilon_{i,j}$ which represents the information needs exhibited in query

history of p_j takes into account all query instances in query history of p_i whose topics are semantically subsumed by p_j's topics. Therefore,

$$\varepsilon_{i,j} = \sum W(m) = \sum e^{-m}$$

where $0 < m \leq h_i^{max}$ and $a_i^1 - I \leq a_i^m \leq a_i^1$.

Again, let $\eta_{i,j}$ be the extent to which that p_j would match the queries that p_i can satisfy and $\iota_{i,j}$ represent the usefulness of p_j with regard to p_i.

$$\iota_{i,j} = \sqrt{(\kappa_1 \sum e^{-m})^2 + (\kappa_2 \eta_{i,j})^2} \tag{7.2}$$

7.5.2 Simulation on Exponential Decay Function Supported Re-organisation (EDFSR)

This section presents the simulation that investigates the behaviour of re-organisation of the peer network based on the exponential decay function. In addition, the simulation explores the relationship between query history and the overlap information in usefulness decision, and examines their individual significance for re-organisation.

Experimental settings The simulator introduced in Section 6.5.1 was employed to simulate a peer network consisting of 100 peers. Each peer randomly chose a list of topics from a global list of 100 entities, ensuring that the topics followed a Zipf's distribution. The cache rate was kept at 5%. A set of query topics which followed a Zipf's distribution[2] was constructed and each query (482 instantiations in total derived from the global list of 100 entities) chose a single topic from this set. The capacity of query history was h. The experimental procedure is described as follows.

1. Over a time interval *I*, q ($q \geq h$) queries are issued. A snapshot of query history of each peer and the topology of the peer network is maintained.

2. Another q queries are issued over *I* in the peer network with the same topology as that maintained in Step 1. The snapshot of the queries and the query initiators is kept.

3. Based upon the query history and the topology maintained in step 1, the peer network is re-organised since a certain percentage, known as the updating rate

[2]Studies show the presence of Zipf's law in Gnutella and Web queries (Breslau et al. 1999, Sripanidkulchai).

$u.r.$, of all peers launch an update to their resources (which also results in an update to the topic information about the resources). For the sake of simplicity, these peers do not practically update topics in simulation but only choose their new neighbours in terms of usefulness ($\kappa_1 = 0$ and $\kappa_2 = 1$). The query initiators kept in step 2 issue the same queries as in step 2 over *I*.

4. Repeat step 3 with pairs of values for κ_1 and κ_2: $(0.01, 1), (0.1, 1), (1, 1), (10, 1)$,-$(100, 1)$ and $(1, 0)$ respectively.

Based on the query history generated in step 1, step 2, 3 and 4 were set up to examine the average reduction in hops to achieve the maximum recall in various environments, including that without re-organisation (step 2) and those with usefulness-based re-organisation (step 3 and 4).

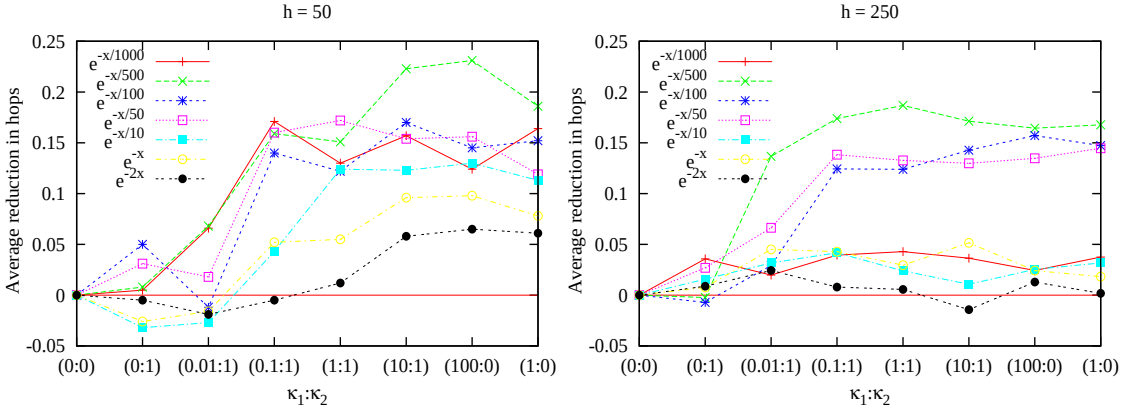

FIGURE 7.3: Average Reduction in Hops to Achieve the Maximum Recall with EDFSR, $u.r.$ = 5%

Discussion Figure 7.3 shows the average reduction in hops to achieve the maximum recall when the exponent in the exponential decay function is associated with various values. The greater the absolute value of the exponent is, the steeper the slope of the curve that represents the function would be. This feature of the exponential decay function indicates that, among others used in the experiment, $f(m) = e^{-2m}$ would result in the greatest difference between the weights allocated to a pair of query history entries. The simulation result reveals that $f(m) = e^{-\frac{m}{500}}$ yields the greatest average reduction in hops to achieve the maximum recall in most cases. However, the curve associated with $f(m) = e^{-\frac{m}{1000}}$ in the figure indicates that merely increasing the exponent does not necessarily lead to a greater average reduction in hops. This is also supported by the result of the simulation conducted with the capacity of the query history equal to 250. The objective of conducting the simulation with a different history capacity is to explore the frequency of which queries should be captured to facilitate

re-organisation. Figure 7.3 demonstrates that, with the specified experimental settings and increased query history capacity (250), the greatest average reduction in hops to achieve the maximum recall is accomplished by EDFSR associated with $f(m) = e^{-\frac{m}{500}}$ in most cases.

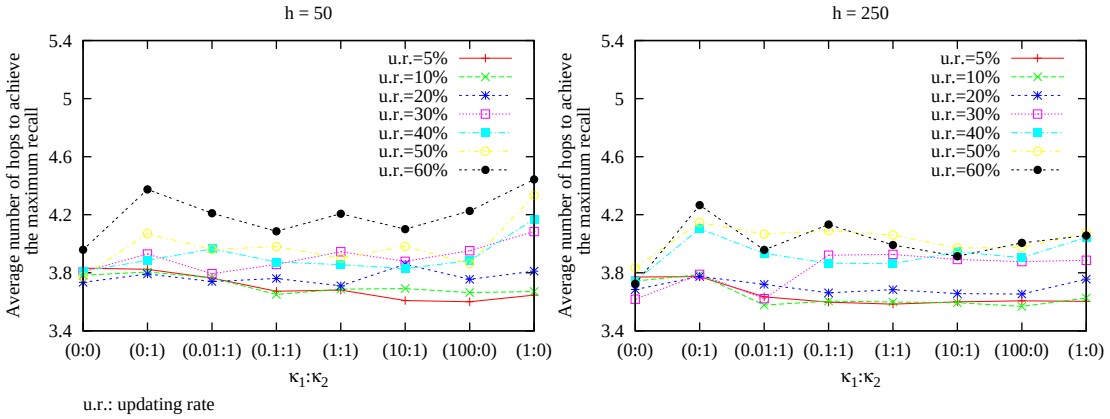

FIGURE 7.4: Average Number of Hops to Achieve the Maximum Recall with EDFSR, $f(m) = e^{-\frac{m}{500}}$

Figure 7.4 captures the impact that the updating rate has on the average number of hops to achieve the maximum recall. It can be seen that peers updating resources deteriorates the search performance in terms of the average number of hops to achieve the maximum recall. The more peers that carry out an update, the greater average number of hops are need to discover all the targets. With a greater capacity of query history, a similar pattern can be observed as in the right sub-figure of Figure 7.4.

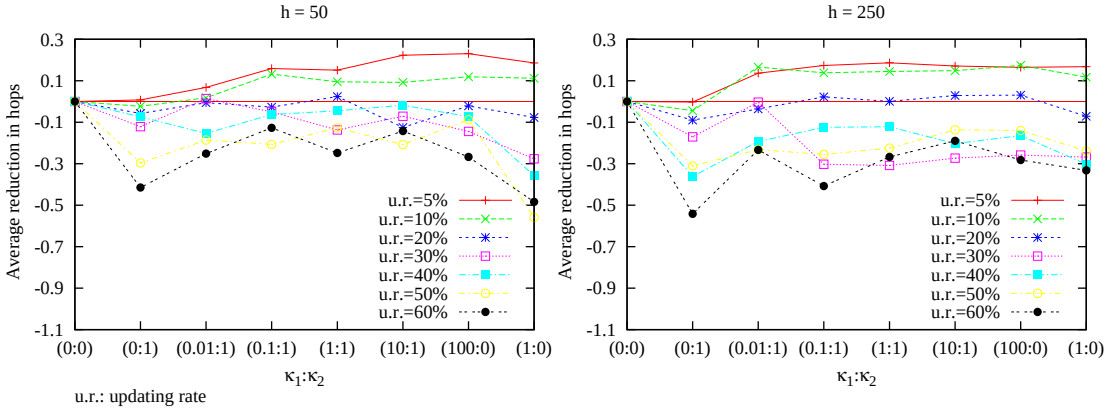

FIGURE 7.5: Average Reduction in Hops to Achieve the Maximum Recall with EDFSR, $f(m) = e^{-\frac{m}{500}}$

The average reduction in hops to achieve the maximum recall with EDFSR is demonstrated in Figure 7.5. Only a relatively low updating rate, 5% and 10% for example, leads to a positive average reduction in hops to achieve the maximum recall.

If the updating rate exceeds 20%, EDFSR does not necessarily deliver a better search performance in terms of the average reduction in hops to achieve the maximum recall.

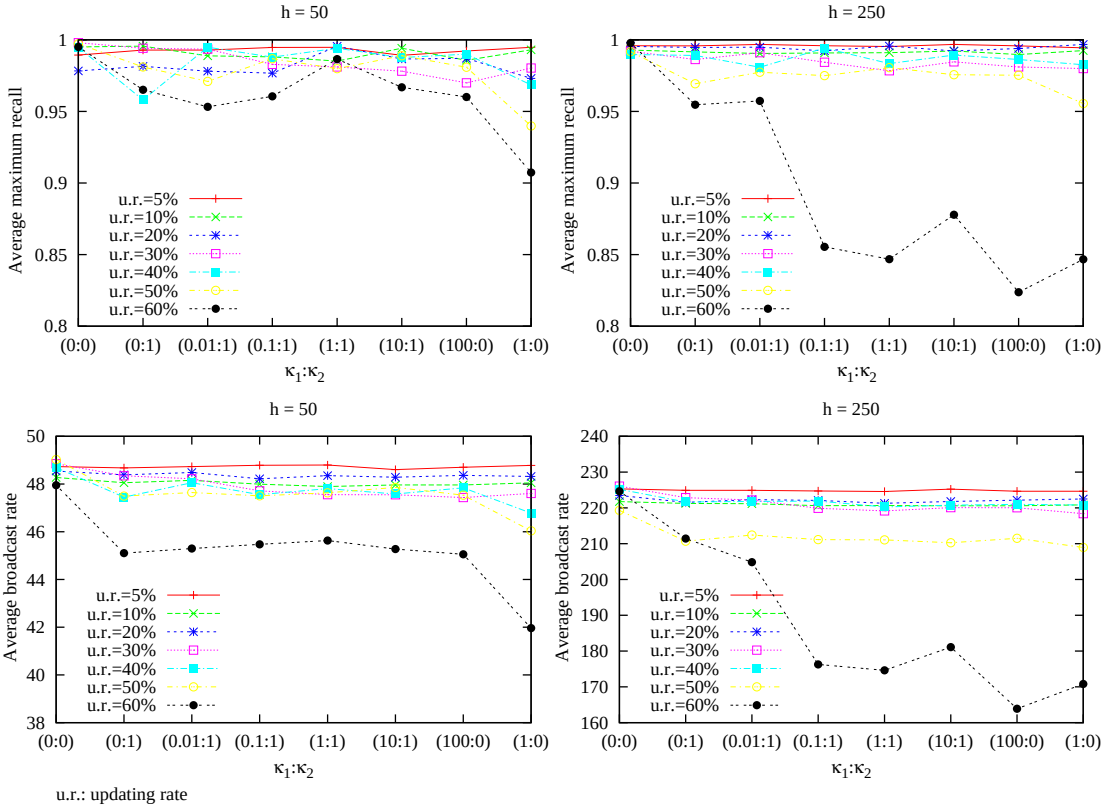

FIGURE 7.6: Average Maximum Recall and Average Broadcast Rate with EDFSR, $f(m) = e^{-\frac{m}{500}}$

The average maximum recall and the average broadcast rate with EDFSR can be referred to in Figure 7.6. The curve associated with the updating rate of 60% is under all the rest in both sub-figures, which indicates that one peer network with a higher updating rate (such as 60%) incurs a lower average maximum recall level as well as a lower average broadcast rate than another with a lower updating rate (such as 5%, 10% and 30%). Moreover, an obvious phenomenon in the figure is that the average maximum recall is primarily proportional to the average broadcast rate. In combination with Figure 7.4, one would discover that the relationship among the average maximum recall (r), the average number of hops to achieve the maximum recall (h), the average broadcast rate (b) and the updating rate (u) across different combinations of κ_1 and κ_2, can be simply depicted by

$$r * u = C_1,\ b * u = C_2,\ h/u = C_3 \tag{7.3}$$

where C_1, C_2 and C_3 are constants related to ($\kappa_1 : \kappa_2$).

7.5.3 Naive Estimator-based Usefulness Decision

The foundation of the naive estimator (Rosenblatt 1956) is that for any given h and n independent observations X_1, X_2, ..., X_n from the random variable X, the probability $P(x-h < X < x+h)$ can be approximated by the proportion of the samples falling in the interval $(x-h, x+h)$. Thus the naive estimator $\hat{f}_h(x)$ for the estimation of density value $f(x)$ at point x is defined as

$$\hat{f}_h(x) = \frac{1}{2nh}[\text{no. of } X_i \text{ falling in } (x-h, x+h)]$$

A more transparent form of the naive estimator can be procured by defining a weight function

$$w(x) \begin{cases} \frac{1}{2} & \text{if } |x| < 1 \\ 0 & \text{otherwise} \end{cases}$$

The naive estimator is therefore also written as

$$\hat{f}_h(x) = \frac{1}{n}\sum_{i=1}^{n}\frac{1}{h}w(\frac{x-X_i}{h})$$

with h being a small number.

The naive estimator is a nonparametric approach (Silverman 1986) in that less rigid assumptions, for example the density function underlying the data, are made about the distribution of the observations. It is the observed data that is crucial in deciding the estimate of $f(\bullet)$. This feature indicates that the distribution of query topics in the DDLS can be estimated based on the overall views of peers, while the probability of query topics at a single peer can be approximated by the peer's local view (Zhou et al. 2004).

The purpose of introducing query history is to perform an informal investigation into the properties of queries, and guide re-organisation of the peer network using the properties of queries revealed. In the DDLS, the probability of query topics is an important property that needs to be explored for predicting the future information needs. The naive estimator can take advantage of query history maintained by individual peers to approximate the future information needs they would encounter.

Suppose the probability of topics in query history of peer p_i can be depicted by function $f_{i,h}(t)$ of a discrete random variable t, where t denotes the least index of the same set of related topics in history entries. For example, if T_a and T_b denote the topics of the 1st and 5th entries in query history and both share the same set of related topics.

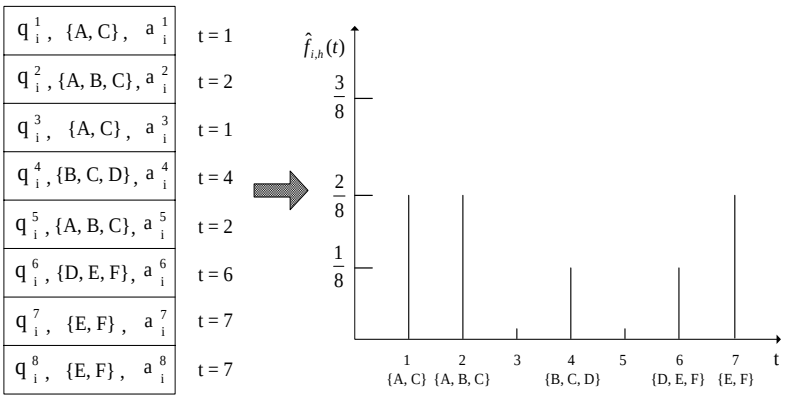

FIGURE 7.7: Computation of $\hat{f}_{i,h}(t)$ based on Query History of p_i

The observations of t for both entries will be 1 instead of 5. Using the naive estimator, the estimate of probability of topics at t is

$$\hat{f}_{i,h}(t) = \frac{1}{n}\sum_{k=1}^{n}\frac{1}{h}w(\frac{t-T_k}{h})$$

with h = 0.5. It should be stressed that $\hat{f}_{i,h}(t)$ takes into account all query history entries of p_i no matter when they arrived. This is contrary to what occurs in exponential decay function-based usefulness decision.

Assume that a candidate neighbour p_j of p_i publishes its topic list T_j. Let $\varepsilon_{i,j}$ be the estimate of the probability of topics in T_j in future queries encountered by p_i.

$$\varepsilon_{i,j} = \sum \hat{f}_{i,h}(t)$$

$\varepsilon_{i,j}$ considers the estimate at all t where the topics of history entries are semantically subsumed by topics in T_j. Figure 7.7 explains how to compute $\hat{f}_{i,h}(t)$ based on query history of p_i. If the topics in T_j comprise {A, C, E, F}, $\varepsilon_{i,j}$ should take into account both $t = 1$ and $t = 7$ at which topics of history entries are semantically subsumed by those in T_j. Therefore,

$$\varepsilon_{i,j} = \frac{2}{8} + \frac{2}{8} = \frac{1}{2}.$$

As in Section 7.5.1, let $\iota_{i,j}$ represent the usefulness of p_j with regard to p_i and $\eta_{i,j}$ be the extent to which that p_j would match the queries that p_i can satisfy.

$$\iota_{i,j} = \sqrt{(\kappa_1 \sum \hat{f}_{i,h}(t))^2 + (\kappa_2 \eta_{i,j})^2} \tag{7.4}$$

7.5.4 Simulation on Naive Estimator Supported Re-organisation (NESR)

The simulation in this section aims to explore the properties of re-organisation of the peer network based on the naive estimator. It also helps understand the quantitative relationship between query history and the overlap information in the usefulness decision during re-organisation.

Experimental settings The experimental settings remain the same as those described in Section 7.5.2 except that the usefulness decision is enabled by the naive estimator.

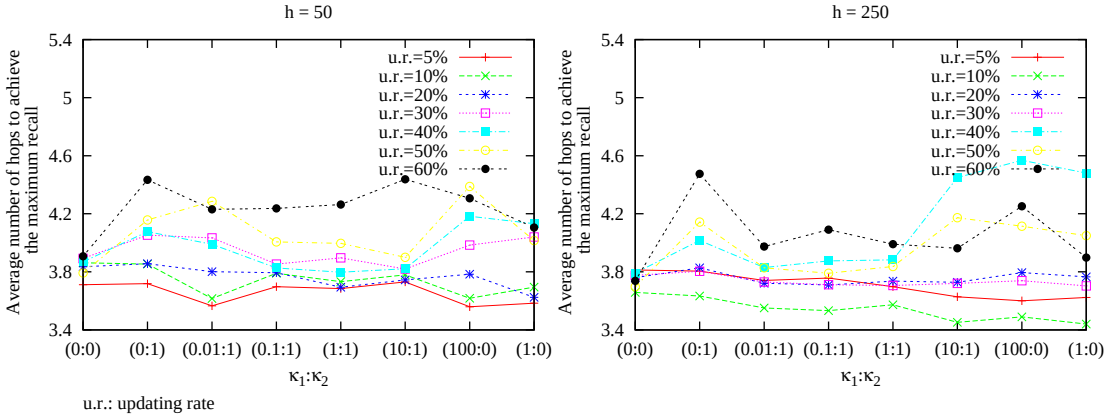

FIGURE 7.8: Average Number of Hops to Achieve the Maximum Recall with NESR

Discussion With NESR, the average number of hops to achieve the maximum recall varies with the updating rate, see Figure 7.8. Similar to the pattern observed in Figure 7.4, the more peers that conduct an update, the more average number of hops are needed in search of all targets.

It is observed in Figure 7.9 that the impact from query history is predominant in reducing the average number of hops to achieve the maximum recall when the updating rate is relatively low, for instance 5% and 10% in the experiment. However, as the updating rate increases, see the curves associated with the updating rate equal to 40%, the overlap information becomes more influential on the the average reduction in hops than query history.

This experiment also reveals that, compared to EDFSR, NESR is applicable to a more dynamic peer network, i.e. a peer network with a higher updating rate, in terms of the average reduction in hops to achieve the maximum recall. It is shown in Figure 7.5

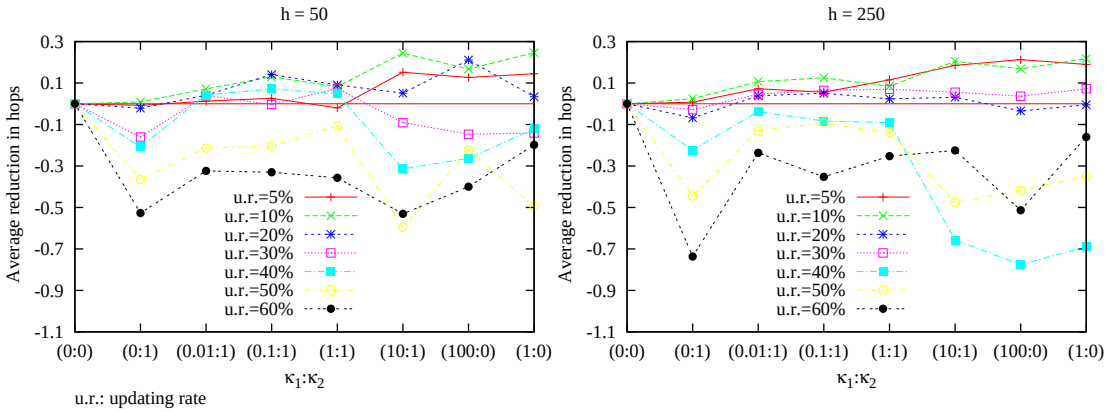

FIGURE 7.9: Average Reduction in Hops to Achieve the Maximum Recall with NESR

that a peer network with an updating rate greater than 20% suffers from a negative average reduction in hops to achieve the maximum recall with EDFSR. However, Figure 7.9 demonstrates that NESR can boost the threshold up to 40%[3].

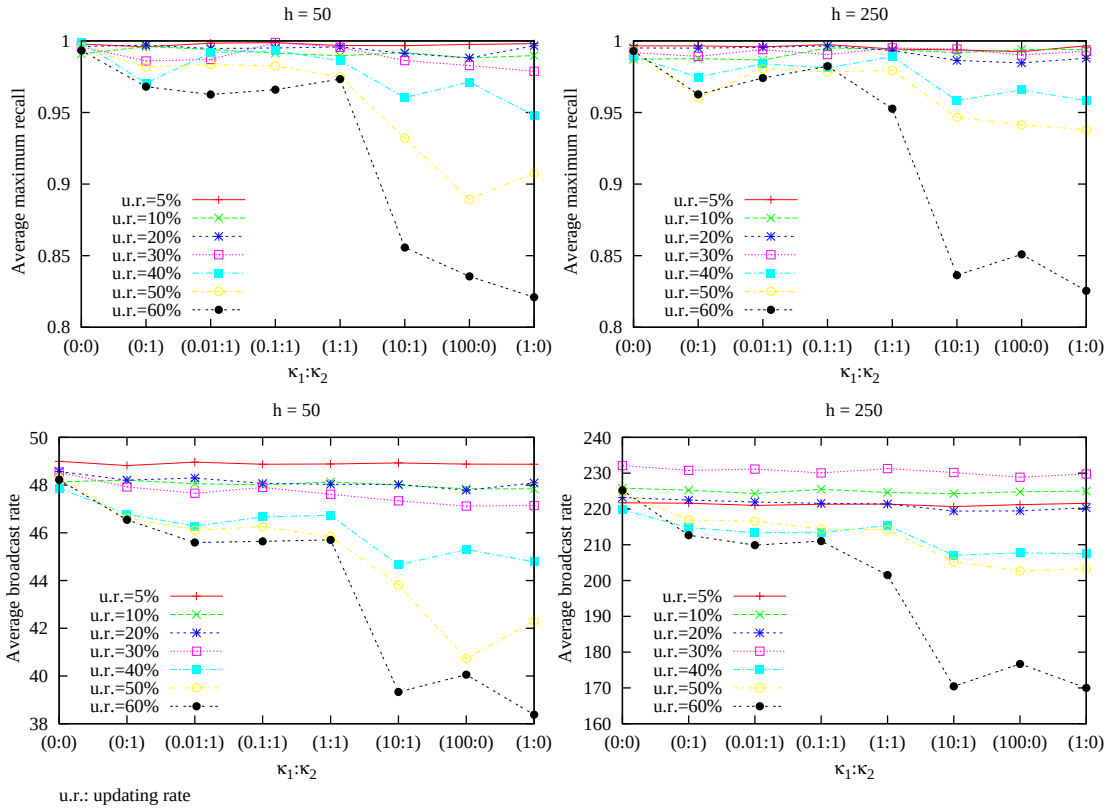

FIGURE 7.10: Average Maximum Recall and Average Broadcast Rate with NESR

Figure 7.10 illustrates the average maximum recall and the average broadcast rate

[3]The average change of all metrics introduced by different re-organisation techniques is of particular concern. Hence, using different network topologies in simulation on EDFSR and NESR is allowed.

with NESR. A greater level of the average maximum recall is always accompanied by a higher average broadcast rate, which indicates that the ratio of the average maximum recall to the average broadcast rate is nearly constant. As with EDFSR, Equation 7.3 holds across different combinations of κ_1 and κ_2 with NESR.

7.5.5 Re-organisation with Virtual Overlap

It was identified in Section 7.4 that peers sharing related topics should be incorporated into the same cluster and peers should also be situated in the vicinity of those that would accommodate their information needs with high probability. Therefore, the re-organisation techniques utilise both the overlap information between peers and query history maintained by individual peers to satisfy such requirements. Recalling the way a published topic list is constructed and the semantic search algorithm works, one might be able to realise the need for altering the data structure of the published topic list. The reason is that, although query history is taken into account in usefulness decision of re-organisation, the published topic list upon which the search algorithm primarily relies does not efficiently support and enforce the utilisation of query history. As depicted in Table 6.1, the last field in the published topic list which embodies all the shared topics between a pair of peers, guides query routing when a search is carried out. However, if, for example, peer p_j is chosen as a neighbour of p_i only because the former covers topics in the history entries of the latter but these two do not share any related topics, no measures can guarantee that the following queries which are related to p_j will be routed to p_j without using a local broadcast as expected.

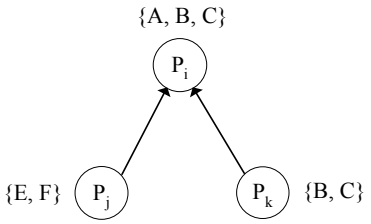

FIGURE 7.11: Semantic Search without Virtual Overlap

Figure 7.11 shows that p_i and p_j have no related topics in common, and p_j is recently chosen as p_i's neighbour because p_j satisfied queries from p_i. p_k is another neighbour of p_i. A query involving topic E will be forwarded by p_i to both p_k and p_j through a local broadcast instead of p_j only according to the search algorithm (see Sec-

tion 6.4.4). This makes the introduction of p_j as a new neighbour of p_i lack significance in terms of the reduction in broadcast rate.

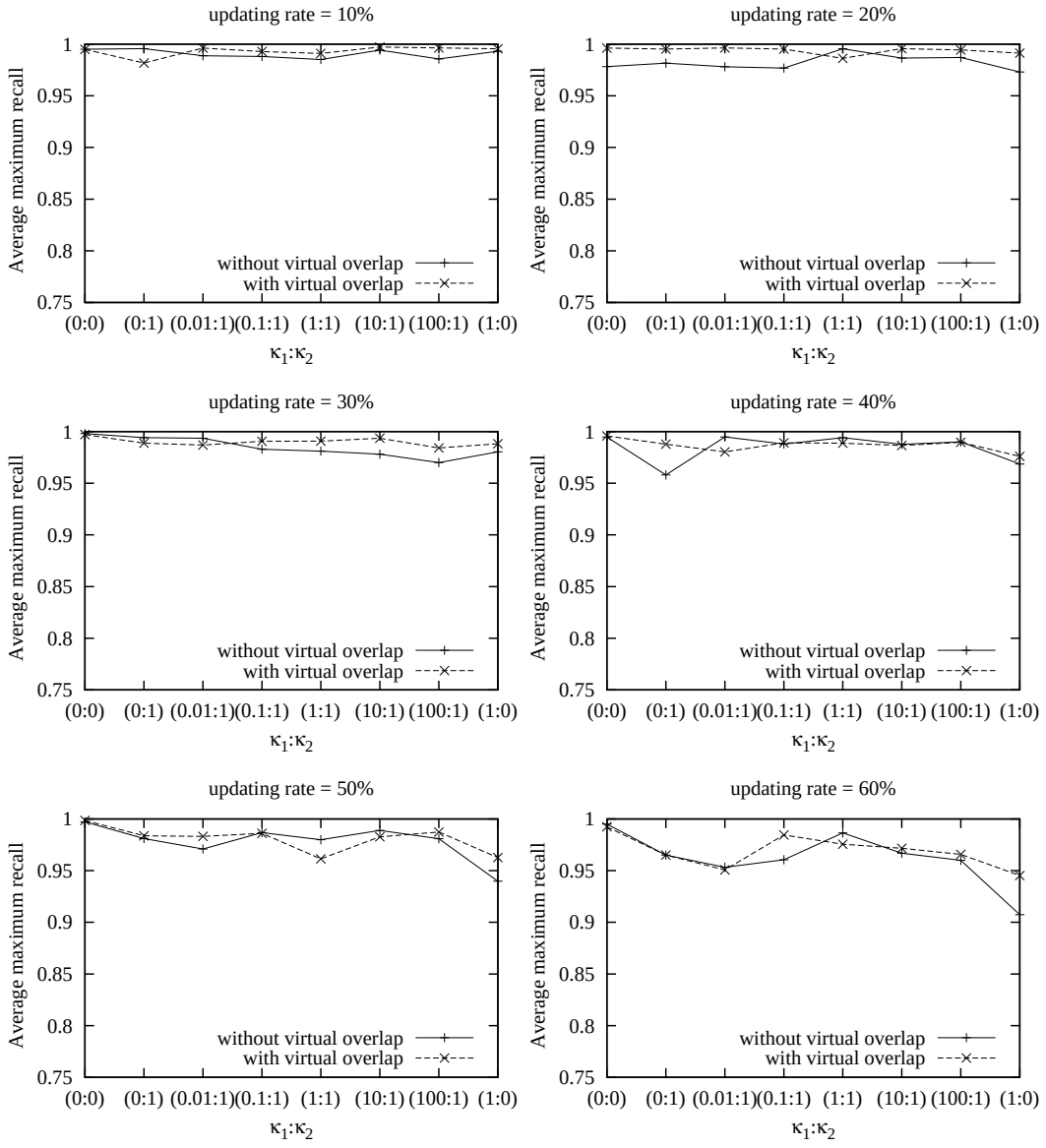

FIGURE 7.12: Average Maximum Recall with EDFSR, $h = 50$, $f(m) = e^{-\frac{m}{500}}$

One of the approaches to enforce the utilisation of query history in re-organisation is to re-define the data structure of the published topic list. The field exposing the shared topics between a pair of peers is currently responsible for hosting a *virtual overlap*. The virtual overlap consists of both the shared topics between a pair of peers and the topics in query history that a neighbour can successfully answer. The addition of history related information to the publish topic list would enable the following queries to be routed to p_j which is chosen as a neighbour of p_i because of its coverage of topics in the history entries of p_i, without using broadcast.

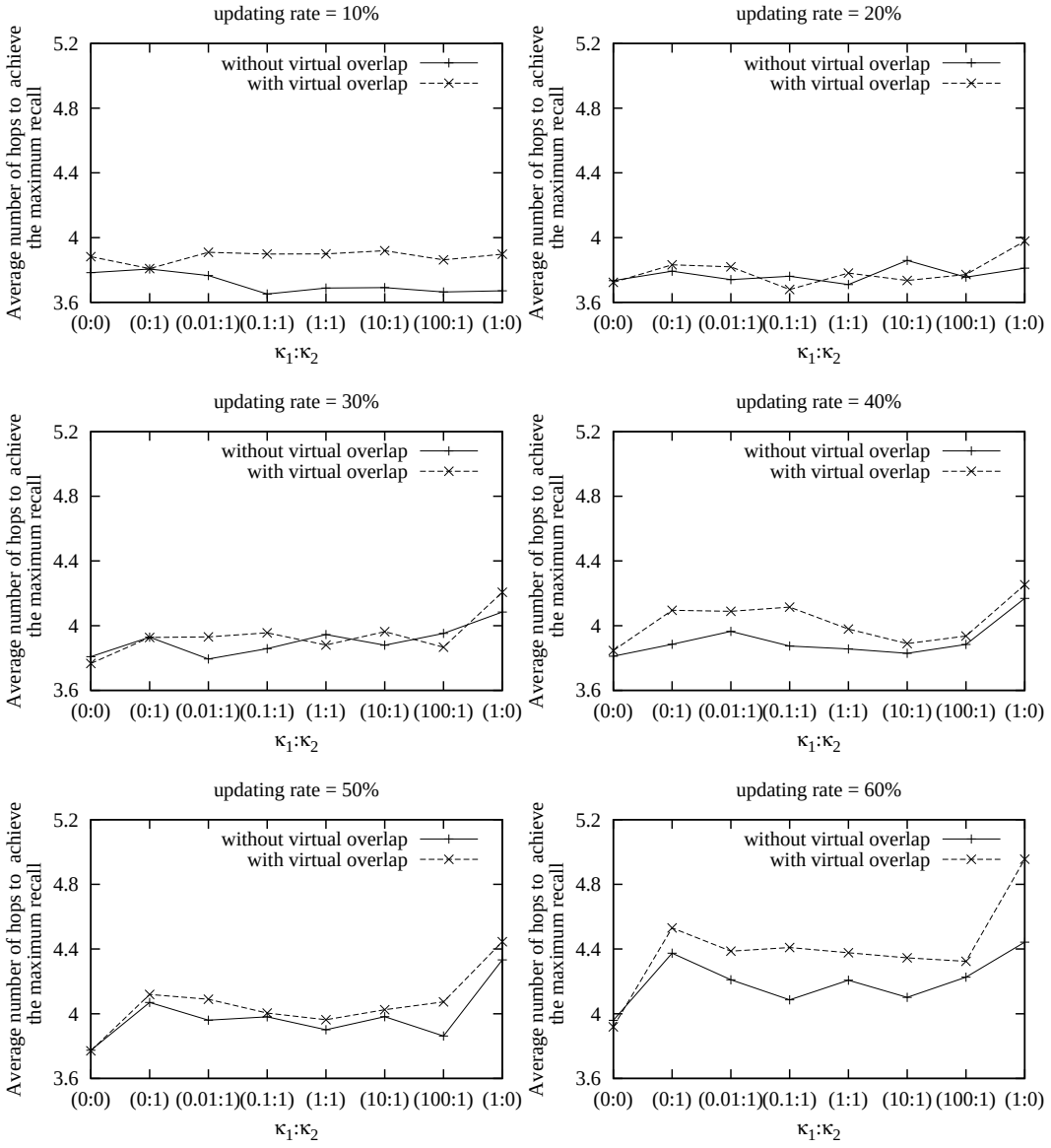

FIGURE 7.13: Average Number of Hops to Achieve the Maximum Recall with EDFSR, $h = 50$, $f(m) = e^{-\frac{m}{500}}$

A series of simulation was carried out to investigate the gains in performance resulting from applying the virtual overlap to both EDFSR and NESR. Section 6.5.3 presents three metrics: hops, recall and broadcast rate, from which the essential components in the utility function defined below to estimate the gains are derived.

$$utility = \Delta h - \Delta r + \Delta b \tag{7.5}$$

where r represents the average maximum recall, h the average number of hops to achieve the maximum recall, b the average broadcast rate and Δ denotes the decrease

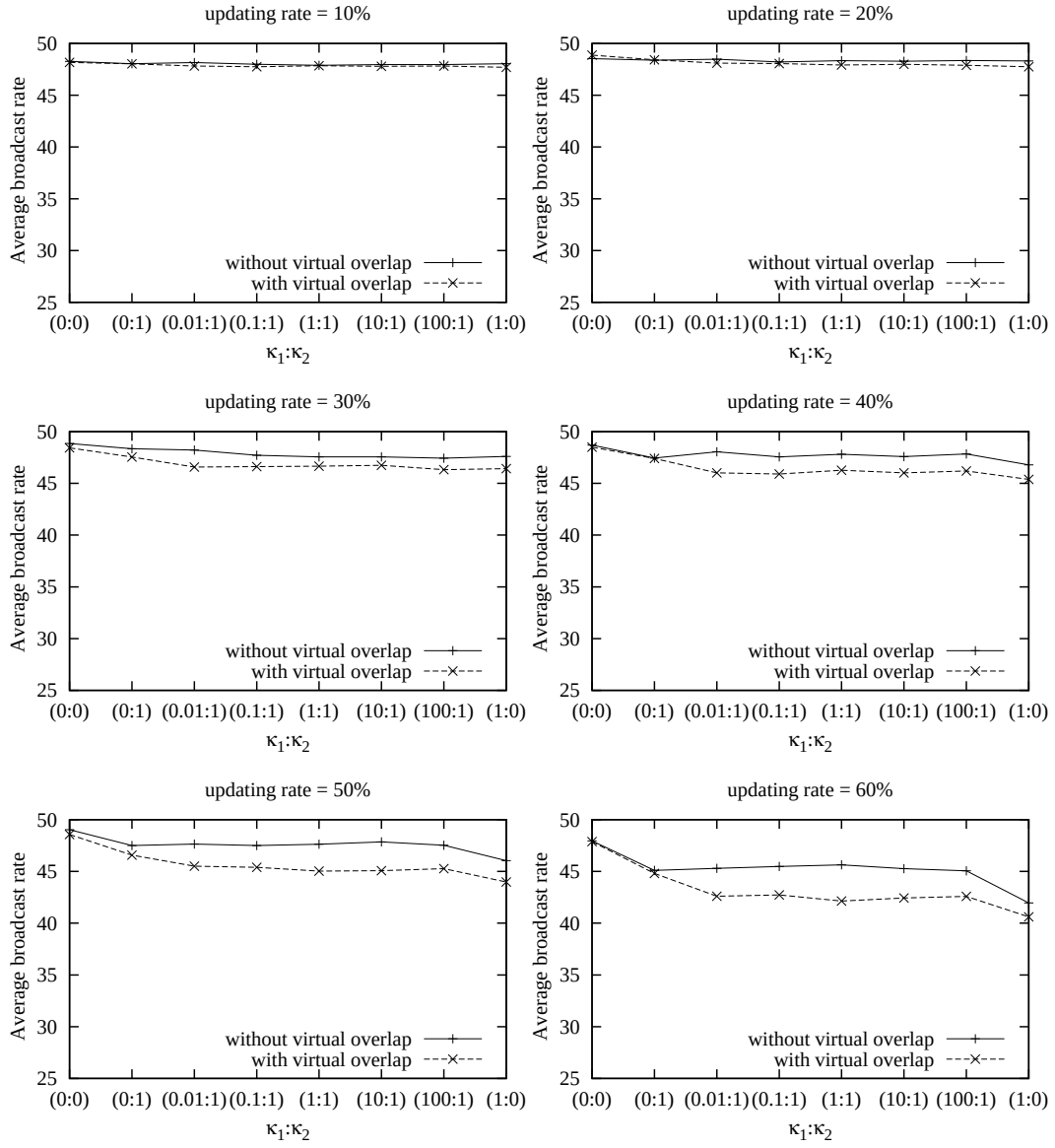

FIGURE 7.14: Average Broadcast Rate with EDFSR, $h = 50$, $f(m) = e^{-\frac{m}{500}}$

of any variable. Because the metrics are different, a normalisation process should be applied to the variables which maps the value of the variables onto an interval, such as $[0, 1]$.

Figure 7.12 shows the average maximum recall achieved by EDFSR which considers the virtual overlap, and by EDFSR which does not. It is not significant that re-organisation with the virtual overlap outperforms that without virtual overlap in most cases. Nonetheless, with $\kappa_1 : \kappa_2 = 1 : 0$, i.e. considering only the impact of query history on re-organisation, an increase of the average maximum recall is observed with various updating rates when EDFSR takes into account the virtual overlap. In addition,

a significant increase can be seen in the average number of hops to achieve the maximum recall by EDFSR that uses the virtual overlap (see Figure 7.13). Though this increase is not desirable in terms of utility, the latter can be compensated to some extent by the decrease in the average broadcast rate, see Figure 7.14. This phenomenon is explained as follows.

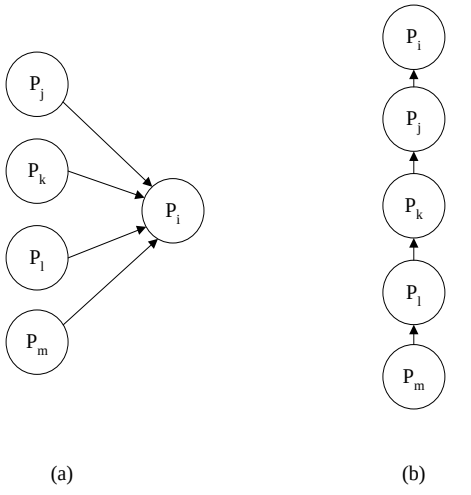

FIGURE 7.15: Re-organisation Leads to the Same Clustering but Distinct Topologies

Through clustering peers that share related topics may participate in the same cluster[4], i.e. they are more similar to each other than they are to others outside the cluster. However, it is difficult to maintain the shortest distance (in terms of hops) between peers. For instance, suppose p_i has four neighbours: p_j, p_k, p_l and p_m. Sub-figure (a) in Figure 7.15 shows the best case in which the minimum average number of hops ((1 + 1 + 1 + 1) / 4 = 1) is achieved when p_i discovers targets from all of its neighbours. Sub-figure (b) in the same figure demonstrates the worst case, after re-organisation, in which it costs a query from p_i the maximum average number of hops ((1 + 2 + 3 + 4) / 4 = 2.5) to locate all targets from p_i's neighbours, p_i's neighbours' neighbours, etc. This example indicates that, although re-organisation is able to group peers into appropriate clusters, it does not guarantee the minimum average number of hops to achieve a certain level of recall. Recalling that Equation 7.5 shows, with a certain level of the variation of recall, the gains from re-organisation can also be achieved through maximising Δb[5]. This has been accomplished and demonstrated by EDFSR using the virtual overlap, see Figure 7.14.

[4]Due to the cardinality of the topics that peers possess, each of them may belong to more than one cluster at a time.

[5]Δb denotes the decrease of the average broadcast rate.

The result of simulation on NESR that adopts the virtual overlap can be referred to in Figure 7.16, Figure 7.17 and Figure 7.18.

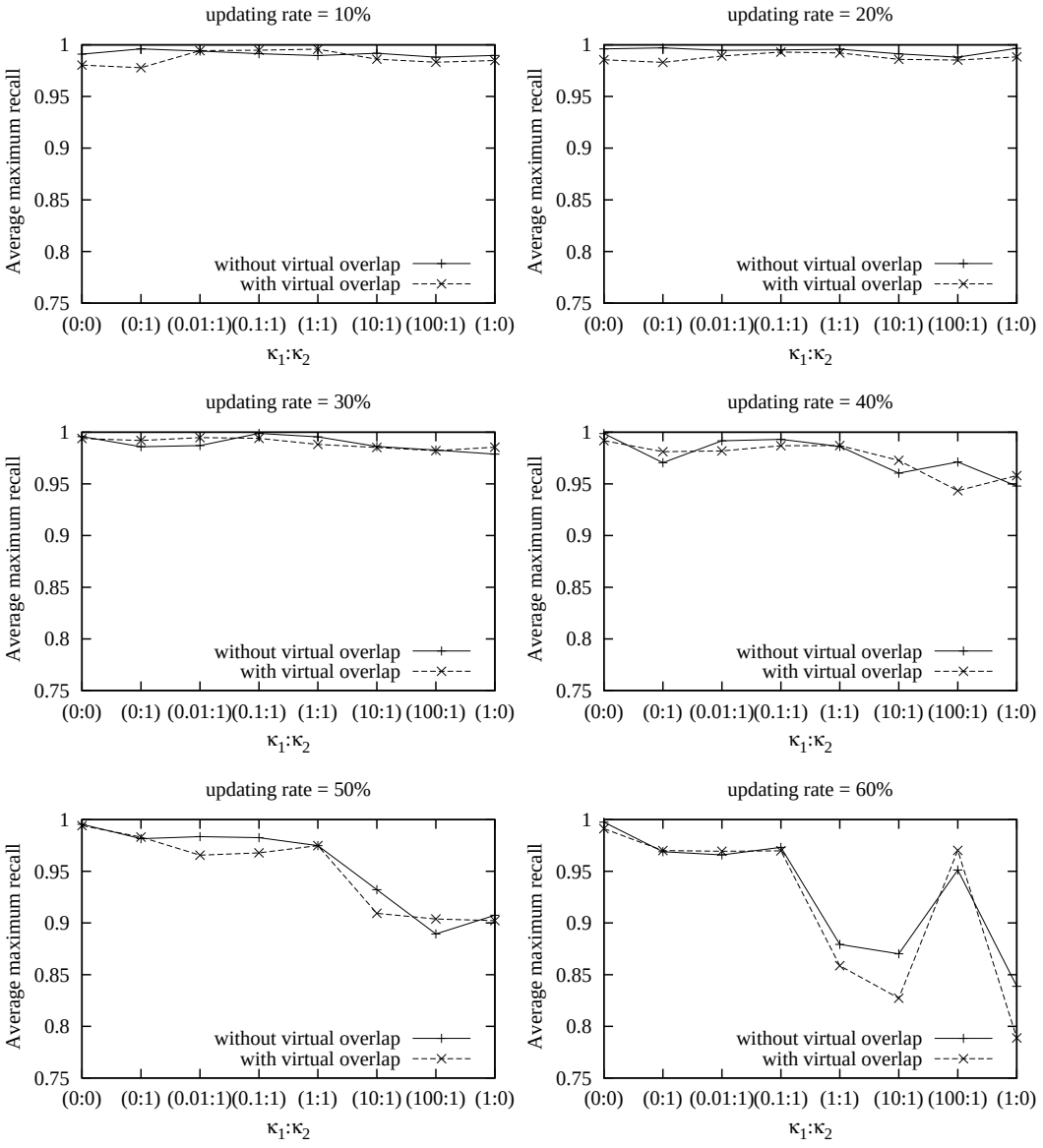

FIGURE 7.16: Average Maximum Recall with NESR, $h = 50$

7.5.6 Comparison between EDFSR and NESR

EDFSR and NESR share the same assumption that the recent past will approximate the immediate future. They both rely on the observation of queries in the past to estimate the future information needs. If comparing each figure of simulation on EDFSR with its counterpart of simulation on NESR, one would discover principally similar patterns from both.

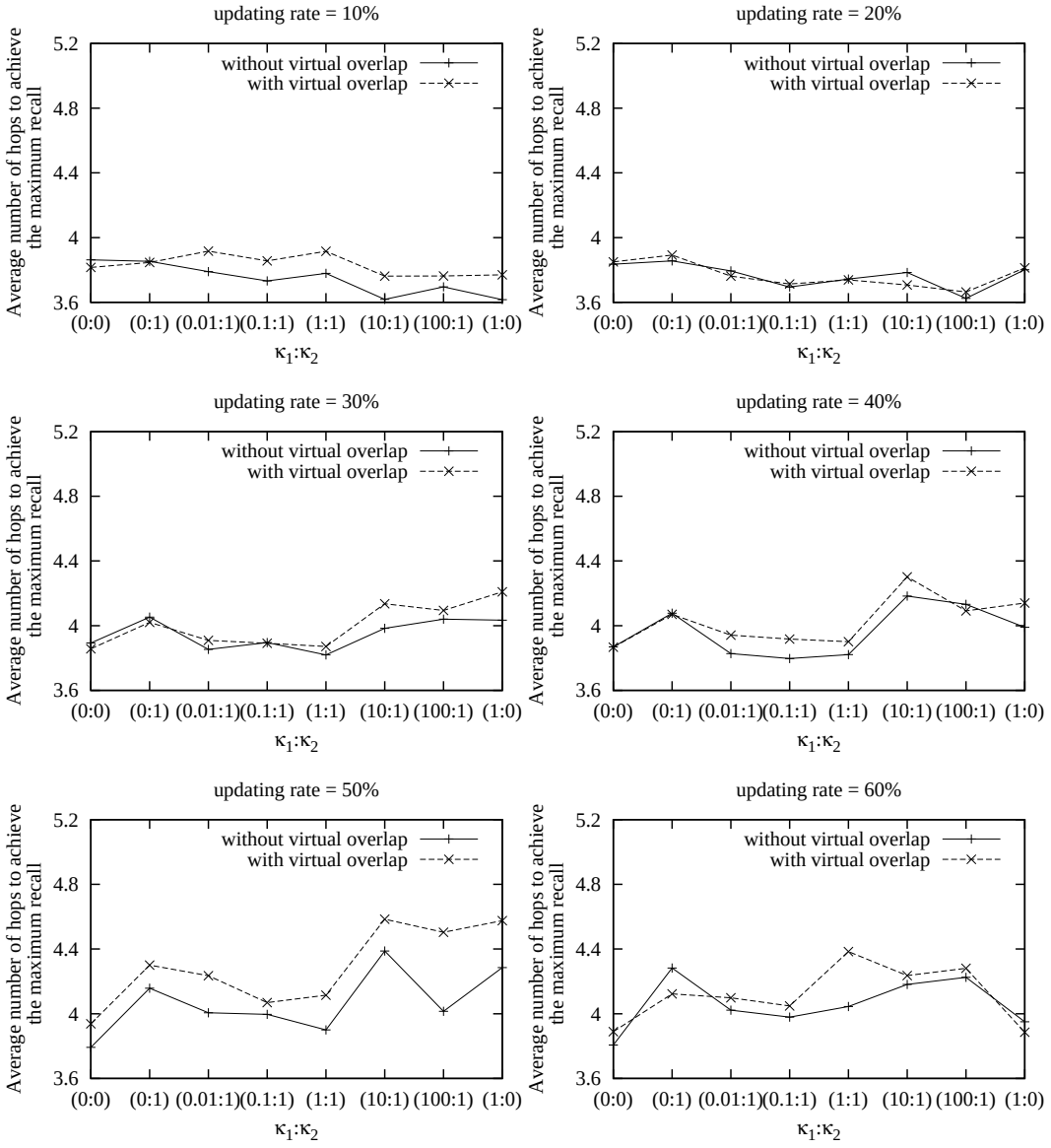

FIGURE 7.17: Average Number of Hops to Achieve the Maximum Recall with NESR, $h = 50$

By changing the updating rate in the simulation with various conditions, it was found, typically, that the greater the updating rate, the more the performance of resource discovery (except the average broadcast rate) deteriorates. This situation can be partly ameliorated by reducing the capacity of query history. Essentially, this translates to decreasing the time interval during which queries are captured. As a consequence, the updating rate will potentially be lower.

To reinforce the principle of the semantic search algorithm to achieve a better performance (a lower average broadcast rate in particular), the virtual overlap was introduced in both EDFSR and NESR. When taking into account the virtual overlap, both

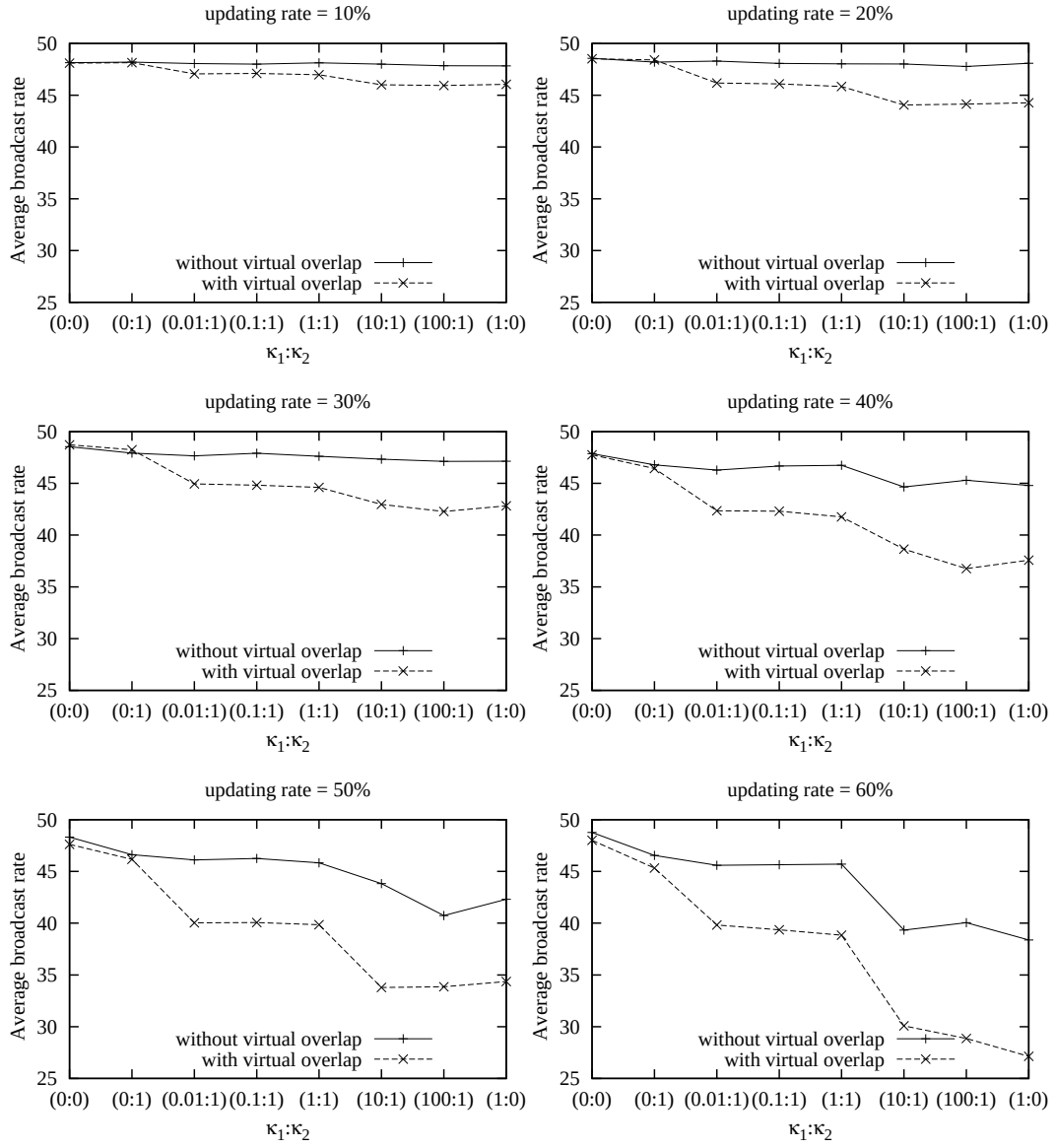

FIGURE 7.18: Average Broadcast Rate with NESR, $h = 50$

re-organisations see a decrease in the average broadcast rate with almost every combination of κ_1 and κ_2 at the cost of an increase in the average number of hops to achieve the maximum recall.

Measure	**EDFSR**	**NESR**
Foundations	recency and frequency	a variant of frequency
Operational means	Queries with the same frequency may be allocated different significance.	The equal frequency of a pair of queries in query history dictates their equal significance.
Applicability	updating rate up to 20%	updating rate up to 40%

TABLE 7.1: Comparison between EDFSR and NESR

EDFSR and NESR also greatly differ from each other in terms of foundations, operational means and application domains, see Table 7.1.

7.6 Understanding the Utility of Re-organisation

The utility of re-organisation using the virtual overlap was defined in Equation 7.5 as a combination of the average maximum recall, the average number of hops to achieve the maximum recall and the average broadcast rate which derive from the three primary metrics established for evaluating the re-organisation performance at the very start (see Section 6.5.3). One fact which needs to be emphasised is that, introducing the virtual overlap is not a deviation from the original principle of the semantic search algorithm, but a further move that reinforces the principle to achieve a better search performance (primarily represented by the reduced average broadcast rate) in re-organisation. Equation 7.5 is not exclusively utilised by evaluating the search performance with the virtual overlap supported re-organisation. It is essentially universal in measuring any effort to enhance the performance of resource discovery in the DDLS, because it expresses the primary objective of re-organisation. This section employs the utility equation to explain to what extent re-organisation has achieved the objective it aims at.

To understand the relationship between the terms in the equation, a review of the available information in the DDLS peer network is necessary. The essential information sources include the published topic list and query history. The information embodied by the published topic list reflects available resources at neighbours and the semantic relationship between resources of a pair of peers. Therefore, it acts as an indicator of how semantically related (the resources of) a pair of peers are. Meanwhile, query history captures queries in the past, and thus is utilised to estimate information needs in the immediate future.

The idea of coupling the two sources to guide re-organisation of the peer network, originates from the analysis of resource supply and demand in the network, see Section 7.4. A semantic search, the main activity the performance of which re-organisation is intended to boost, can be simply viewed as formulating a query with respect to the information needs (demand) and retrieving targets that match the query from a collection of available resources (supply). In the DDLS, information exposed by the published topic list and query history underpins such a process. Because the information is local to individual peers, the knowledge of resource supply and demand based on it is also local.

Applying the local knowledge to re-organisation, a peer acquires new neighbours that are most capable of sharing related resources with it and potentially satisfying future queries it would encounter. Re-organisation correlates a peer with those useful neighbours, and its primary *objective* of producing the higher level of the maximum recall, the less number of hops to achieve the maximum recall and the lower level of broadcast rate is expressed by the utility equation[6]. Meanwhile, re-organisation is able to yield, revealed by simulation, the less number of hops to achieve the maximum recall (only when the updating rate in the peer network is relatively low[7]) through increasing the overlap between peers and including potential resource providers to satisfy future queries.

Examining the search performance associated with the relatively low updating rates in simulation on both EDFSR and NESR (without using the virtual overlap), one would discover that the variation of the average maximum recall[8] is roughly proportional to that of the average broadcast rate across different combinations of κ_1 and κ_2. Assume that the contributions of both variations to the utility of re-organisation neutralise each other. The average reduction in hops to achieve the maximum recall can be greater than 0. Thus, taking (Δb - Δr) in Equation 7.5 as 0, the utility of both EDFSR and NESR is as follows.

$$utility = \Delta h - \Delta r + \Delta b = \Delta h > 0 \tag{7.6}$$

However, the utility of proposed re-organisation (without using the virtual overlap) is only achieved when the updating rate remains relatively low.

The following is the evaluation of the utility of re-organisation that uses virtual overlap[9]. For both EDFSR and NESR, the increase in the average maximum recall is only achieved when the updating rate is not more than 20%, and in the rest cases, the average maximum recall always decreases with re-organisation[10], see Figure 7.12 and Figure 7.16. While EDFSR hardly achieves any decrease in the average number

[6]The desirable utility value should be greater than 0.

[7]With EDFSR, the updating rate should be not more than 20%, whereas with NESR the threshold rises to 40%.

[8]The variation of the average maximum recall resulting from re-organisation can be observed by comparing the value of the average maximum recall at $\kappa_1 : \kappa_2 = 0 : 0$ with that at another combination of κ_1 and κ_2 which is used in the re-organisation.

[9]It is stressed out that Equation 7.5 is initially introduced to evaluate the gains brought by using the *virtual overlap* in re-organisation (compared to without using the virtual overlap in re-organisation). Whereas, the analysis of the utility of re-organisation here is concerned about the gains brought by conducting *re-organisation* with virtual overlap (compared to without using re-organisation at all).

[10]This can be observed from the curve which is associated with the virtual overlap by comparing the average maximum recall at $\kappa_1 : \kappa_2 = 0{:}0$ with the average maximum recall at other combinations of κ_1 and κ_2.

of hops to reach the maximum recall (see Figure 7.13), NESR can deduce the average number of hops to achieve the maximum recall with an updating rate up to 20% (see Figure 7.17). Meanwhile, a desired decrease in the average broadcast rate can always be found in both EDFSR and NESR, see Figure 7.14 and Figure 7.18. The utility of re-organisation using the virtual overlap can therefore be obtained by replacing the terms in Equation 7.5 with corresponding data from simulation. Because of different available computational conditions and requirements for re-organisation, the utility can also be computed with different weights attached to each term in the equation.

7.7 Consistency Maintenance of Associated Data Structure

Before proceeding to the conclusion, this section discusses the operations that are crucial to avoid the potential inconsistency of the associated data structure resulting from re-organisation, and thus to warrant the consistency of the peer network.

The consistency of the associated data structure, such as that of the published topic list, suffers from re-organisation unless necessary measures are undertaken to maintain it. The published topic list reflects the relationship between resources of a peer and its neighbours and it is the major source based on which the knowledge of the local peer network can be obtained. Activities such as resource discovery have heavy dependencies on local knowledge. Therefore, the inconsistency of such a data structure may deteriorate the functioning of resource discovery in the peer network. To maintain the peer network in proper functioning conditions, the published topic list needs timely update once re-organisation occurs.

With all the operations presented in Section 6.2 that maintain data consistence in passive re-organisation triggered by peer arrival and departure, this section only describes those associated with update related passive re-organisation.

The update events that trigger re-organisation consist of addition of new topics and subtraction and modification of existing topics, with modification being viewed as a combination of addition and subtraction operations.

If an update involves only the addition of new topics, the published topic list may be inconsistent since the previous overlap of topics between the updating peer and its neighbours may not reflect the up-to-date state of their relationship. The inconsistency

Notations:

Same as in Figure 6.1 and Figure 6.5. In addition,

Updated topics of p_i:	$T_i^{'} = \{t \mid t \in T, t \times t \subseteq T, T_i^{'} \cap T_i \supset \emptyset\}$
Set of identifiers of the affected neighbours because of update of p_i:	Λ_i
Adding a set of identifiers $id_{i,u}^{peer}$ to Λ_i:	$\Lambda_i.addElement(id_{i,u}^{peer})$
Removing the $u'th$ entry from the published topic list of p_i:	$p_i.removeElementAt(u)$

Algorithm for p_i to maintain its published topic list up-to-date:

$\Lambda_i = \emptyset$
If $T_i^{'} \subset T_i$ *, then* */* removal of topics */*
  For $u = 1$ *to* $|E_i|$
    If $T_i^{'} \subseteq T_{i,u}^{common}$ *, then*
      $p_i.removeElementAt(u)$;
      $\Lambda_i.addElement(id_{i,u}^{peer})$.
  For each $p_k \in \Lambda_i$
    $p_i.addNeighbour(p_k, T_k)$.
Else */* addition and modification of topics */*
  For $u = 1$ *to* $|E_i|$
    $p_i.removeElementAt(u)$;
    $\Lambda_i.addElement(id_{i,u}^{peer})$.
  For each $p_k \in \Lambda_i$
    $p_i.addNeighbour(p_k, T_k)$.

FIGURE 7.19: p_i Maintains the Published Topic List Up-to-date

could potentially occur to the entire published topic list. As a consequence, the list needs to be re-constructed.

In most cases, a subtraction operation may not yield an update affecting all entries in the published topic list. For instance, if subtraction involves topics possessed by the updating peer but not by its neighbour, the published topic list can remain intact. This is because the subtraction does not invalidate the overlap information between the pair of peers. However, if such subtraction relates to shared topics, the published topic list may become inconsistent. Under such a circumstance, only the affected entries need to be re-constructed.

Modification of topics exhibits a combined behaviour of both addition and subtraction of topics. An updating peer needs to re-create its published topic list in case of modification.

In addition to refreshing its published topic list, an updating peer is responsible for informing each of its contacts of the update, so that the latter is able to modify its cached topic information accordingly.

Figure 7.19 demonstrates how to bring the published topic list up to date by discarding information that does not reflect the newly-formed relationship between resources of a pair of peers. A peer p_i launches the refreshing process when its topic information is updated. For subtraction operation, if the updated topics are those shared by neighbours appearing in an entry with $\alpha_{i,u}^{dist}$ for example, the entry is discarded from the published topic list of p_i by $p_i.removeElementAt(u)$. p_i removes all affected neighbours and then adds them as new neighbours in order to maintain its published topic list up-to-date. For addition and modification operations, re-construction of the published topic list is compulsory.

7.8 Review of Re-organisation

An intuition of improving resource discovery in the DDLS is to re-organise the peer network wherever necessary after it is established. The reason why re-organisation does not apply to the construction of the peer network will be given later. The criteria for re-organisation state that peers possessing semantically related resources should cluster together, and if peers cannot be located in the vicinity of others with related resources, they should stay closer to peers which can potentially satisfy their future information needs. The criteria are derived from an analysis of supply and demand in the peer network (see Section 7.4). Knowledge of supply from neighbours (obtained from the published topic list) and previous demand (known from query history), enables a peer to measure to what extent a candidate neighbour would have satisfied its queries in the past. Supported by a very important assumption that the recent past will predicate the immediate future, a peer selects those with the high potential (indicated by usefulness) to fulfil its future information needs as neighbours once re-organisation is triggered.

Measuring to what extent a candidate neighbour would have satisfied the previous demand based on query history is key for a peer to identify useful neighbours. This work has explored the use of two techniques - the exponential decay function and the naive estimator - in determining the usefulness of neighbours. The exponential decay function varies the significance of queries in query history based on both the time (recency) and the number of occurrence (frequency), whereas the naive estimator diversifies the significance of queries in query history in terms of the probability of query topics (analogous to frequency).

The efficiency of both proposed re-organisation techniques (EDFSR and NESR) has been evaluated and confirmed through a series of simulations in which hops, recall and broadcast rate were utilised as metrics. The main findings of simulation on both techniques are similar in pattern, and are summarised together as follows.

1. The more peers that conduct an update, the more hops are needed in search of all targets.

2. The impact from query history is predominant in reducing the number of hops to achieve the maximum recall when the updating rate is relatively low (not more than 20% with EDFSR and 40% with NESR). However, as the updating rate increases (more than 20% with EDFSR and 40% with NESR), the overlap information becomes more influential on the reduction in the number of hops than query history. If excessive peers (more than 20% with EDFSR and 40% with NESR) carry out updates over a time interval during which queries used by EDFSR or NESR are captured, re-organisation may not necessarily lead to a better performance (except the reduced broadcast rate).

3. One peer network with a higher updating rate (such as 60%) incurs the lower level of the maximum recall, the greater number of hops to achieve the maximum recall and the lower level of broadcast rate than another with a lower updating rate (such as 5%, 10% and 30%).

Simulation has demonstrated, see finding 2, that the exponential decay function, compared to the naive estimator, is applicable to a relatively less dynamic peer network. Nonetheless, it should be pointed out that the simulation has adopted a Zipf's distribution for query topics, a pattern widely observed in large scale distributed systems such as the Web and Gnutella. This is typically a generalisation of query distribution over the course of days or even months. However, it does not capture any time related feature of queries, such as during which period a certain query has been the most popular one. This explains why the Zipf's distribution of query topics favours NESR. The real potential of EDFSR will only be fully exploited when work has been carried out to examine the typical pattern of query topics in OHSs in terms of a combination of both recency and frequency. Moreover, the same finding shows that, due to the unpredictable dynamics of the network at the construction stage, re-organisation techniques should not be applied. Therefore, it is reasonable for peers to randomly choose neighbours when the peer network is initially established.

This work also advocates measuring the increase of the resource discovery performance resulting from re-organisation through a combination of the metrics defined in

Section 6.5.3, thus producing a utility equation (see Section 7.5.5). This ensures that the evaluation of re-organisation is conducted in an overall manner. However, this work has only presented simulation results captured by single metrics, respectively, and left the evaluation to be accomplished in response to different available computational conditions and requirements for re-organisation by applying tunable weights to each term in the utility equation. The utility equation can be extended by incorporating other metrics wherever necessary as more concerns are involved.

7.9 Summary

This chapter introduced the concept of re-organisation which aims to enhance the performance of resource discovery in the DDLS by altering virtual neighbourhood of peers. The exponential decay function and naive estimator were proposed to support re-organisation. Both techniques demonstrate, through simulation, their different capabilities to improve performance with regard to different metrics.

The use of the virtual overlap, which signifies a further move to reinforce the principle of the semantic search algorithm and aims at a better search performance brought by re-organisation, was introduced. To measure the gains in performance, this chapter provided a utility equation which can be employed with different weights attached to its terms to take into account potential computation conditions and requirements for re-organisation.

The following final chapter concludes with a summary of this work and presents the potential future directions for both extending the DDLS and researching into P2P OHSs.

Chapter 8

Conclusions and Future Work

8.1 Conclusions

The scenario of collaboration based on resource sharing among a community of people with similar knowledge background inspired this work. Its requirement for equal capability and autonomy of individuals, and a cooperative effort of all people to enable collaboration makes most of the existing approaches less applicable. Meanwhile, a review of the literature reporting on related research clarifies that the realisation of such a scenario is only possible when associated technologies from multiple disciplines are involved. Essentially, this work draws on research from open hypermedia, P2P computing, the Semantic Web and information retrieval, see Table 8.1[1].

Issues	**Enabling Technologies**
resource maintenance	open hypermedia
resource management	open hypermedia
resource presentation	open hypermedia
information (about resources) encoding	the Semantic Web
resource clustering	the Semantic Web, information retrieval
resource discovery	the Semantic Web, information retrieval, P2P
architecture, topology	P2P
re-organisation	P2P

TABLE 8.1: Technologies from Multiple Disciplines Supporting the DDLS

As a complementary hypermedia link service with which clients can make en-

[1]The support of resource clustering and discovery from the Semantic Web can be enabled by its key technologies, for instance OWL (see Section 2.5) which is intended to be used for defining structured and Web-based ontologies that describe and represent an area of knowledge. However, this work assumed the existence of such a capability, and therefore did not focus on it.

quiries against distributed sets of linkbases, the DLS has demonstrated its support for supplying links that refer to concrete resources, rather than directly transferring the resources. This fundamental feature qualifies the DLS for the paradigm of resource sharing in collaboration, but mechanisms that enable efficient resource publishing and discovery are still missing to make cooperation in an ad hoc environment a reality. The main objective of this work was to explore how the DLS approach could be augmented to continuously function in an environment, as depicted by the scenario, in which distributed resources were available for sharing. To achieve this goal, a number of issues were identified in Section 1.3.

One of these issues relates to addressing resource description and maintenance so as to benefit resource discovery in a distributed environment characterising different degrees of decentralisation of control. Section 4.4.1 and Section 6.3.2 respectively discussed the use of the XML model and that of the RDF model to describe resources and to encode the information about resources at link and linkbase levels. Describing resources at the link level supports a finer grained search, whereas using associated concepts to describe resources at the linkbase level has been recognised more feasible because of the need for a semantic search to locate conceptually related resources[2]. Through an examination of the requirements for resource discovery in the DDLS (see Section 5.4.1), the semantic search was identified as an essential and indispensable mechanism, and a comparison between the XML model and the RDF model further revealed the possibility of realising the semantic search that the RDF model would enable (see Section 6.3.2).

Unlike others in the P2P community, this work urged an approach to resource discovery in an unstructured P2P system that should take into account the semantic relationship between resources and form an overlay on top of the relationship to facilitate discovery. The way the semantic overlay in the DDLS is constructed enables peers to obtain information about related resources hosted by neighbours, thus facilitating query forwarding. This work on the semantic search in an unstructured P2P system differs from others in the sense that its applicability is not restricted to a specific domain as in (Crespo and Garcia-Molina 2002b). The enabling techniques, such as ontologies and inference logic, can empower the DDLS semantic search to satisfy users by extending potential targets from resources syntactically the same to those conceptually related.

This work was further driven in pursuit of techniques that can enhance the performance of resource discovery. The potential approach, re-organisation of the peer network, is considered feasible . This is because the network topology and the location of resources should not be precisely determined and correlated with each other in

[2] Although resource description and resource discovery based on the semantic search can be performed at the link level, doing so will inevitably deteriorate the scalability of the DDLS.

the DDLS as in DHTs-based P2P solutions, which makes altering neighbourhood of peers possible. An analysis of the network supply and demand in Section 7.4 identified the available information in the peer network that could be utilised to support re-organisation.

The proposed re-organisation techniques, the exponential decay function and the naive estimator, both take advantage of the overlap information between resources of peers and the history of queries that peers have encountered, but differ in the way that query history is utilised in support of re-organisation. Their effectiveness in optimising resource discovery was demonstrated through simulation, and the enhanced performance was proposed to be measured by a combination of recall, hops and broadcast rate. Considering the different computational conditions and requirements for re-organisation in reality, this work urges a reasonable weighting scheme in evaluating the enhanced performance resulting from re-organisation.

In conclusion, the DDLS represents the research on extending open hypermedia systems into P2P environments. In particular, it revolves around issues that arise from implementing a hypermedia link service with a P2P nature. By using the open hypermedia paradigm to maintain and present resources, RDF to encode information about resources, the clustering technique to group resources and form the information space, a semantic search mechanism to discover resources, this work has demonstrated the individual functionality of technologies from multiple disciplines in conducting and promoting open hypermedia research, and more importantly, its practice and methodology can be of benefit to other research in both open hypermedia and P2P communities.

8.2 Future Work

8.2.1 System Enhancements

There are, so far, some issues in the DDLS which require further investigation. The accomplishment of these tasks will not only enhance the DDLS but also fundamentally benefit and promote the P2P open hypermedia research. The potential work is outlined as follows.

Multiple attribute-based search The way that a linkbase typically contains links that resolve to documents with semantically related content, enables associating linkbases with abstract concepts and searching for linkbases of interest based on concepts. In the DDLS, such concepts are referred to as *topics*. Supporting search based on such a single

attribute as topic is apparently, however, not sufficient for OHSs other than the DDLS, because some of the OHSs may need to search for other hypermedia structures than links, such as anchors, nodes, etc. Whether anchors and nodes can also be represented using *topic-like* attributes based on which a semantic search can be conducted, remains unexplored. Moreover, if multiple attributes are utilised to describe various hypermedia structures in the DDLS, the way that the semantic overlay is constructed is no longer appropriate, because it only takes into account the overlap information focusing on topics. For the same reason, the semantic search algorithm and re-organisation of the peer network also require a revision.

Using ontologies to identify semantic relationship An ontology is an explicit specification of how to represent objects, concepts and other entities in some area of interest and the relationship among them. The importance of ontologies in identifying the semantic relationship between resources in the DDLS, is clear. This work will need to either utilise, merge or extend existing ontologies, and establish an approach that enables the use of ontologies in a more rigorous and explicit manner. Carr et al. (2001) proposed the use of thesaurus-supported ontologies in a Conceptual Open Hypermedia Service to empower the retrieval of pertinent information, and a Description Logic model to guarantee an explicit, rigorous and declarative specification of concepts. Moreover, to support the DDLS semantic search for concepts across domains, ontology-mapping tools will also need to be developed.

Geography-based efficiency This work conducted efficiency measurement of semantic search and re-organisation in terms of hops at the application level. However, Ratnasamy et al. (2002) argued that the real efficiency of P2P routing algorithms should be measured in the end-to-end latency of the path, because the application level hops might involve a path that spanned a continent or merely a LAN, and ignoring the latency of individual hops would result in a path with high latency. A basic approach in various algorithms is to associate a cost in latency with each hop and take this cost into account when choosing either the next hop node or neighbours. This method can apparently be experimented on the DDLS for implementing geography-based efficiency in both semantic search (choosing the next hop node) and re-organisation (deciding the usefulness of neighbours).

Knowledge of resource and query distribution The query distribution of Gnutella and Web has been investigated and the typical query pattern has been discovered (Sripanidkulchai 2001, Breslau et al. 1999), which is crucial to the settings of simulation which evaluates the search performance in these and similar systems. The same work in OHSs remains unexplored. Without the knowledge of typical resource and query distri-

bution, simulation designers of P2P OHSs, such as that of this work, can only speculate the potential distribution based on results observed in other large scale distributed systems.

Scalability Decentralisation enables improved scalability in P2P systems in the sense that data storage, operations, computation and communication can be carried out in a decentralised manner. Most structured P2P systems achieve satisfactory scalability through the use of DHTs which resolve a keyword to a location where the contents are located or from where queries about the contents can be further routed. However, the same level of scalability that structured P2P systems have accomplished is a difficult target for unstructured P2P solutions (including the DDLS) to fulfill, because the latter does not, and cannot, model the information space for efficient lookup as structured P2P systems do. This work has taken into account the scalability issue in the design of the DDLS. For instance, according to the semantic search algorithm, queries are always intended to reach peers with related resources, thus avoiding a local broadcast. However, no evaluation of the DDLS scalability has been conducted yet, and therefore the understanding of the DDLS scalability is not clear and requires an analysis and simulation.

Security P2P systems, in contrast to centralised and client-server systems, incur more security threats. For the DDLS which serves as a distributed link service, security measures should be implemented to protect peer machines from crashing or sensitive data being leaked, and to prevent malicious peers from providing unreliable information. The former issue regarding the protection of peer machines can be addressed by techniques described in (Milojicic et al. 2002) typically involving enforcing safety properties and security properties. Kamvar et al. (2003) addressed the latter issue by taking into account trust between peers and allowing peers to use a global trust value to choose peers from which they download files. Hence, the system is enabled to efficiently identify malicious peers and isolate them from the network. An approach that helps identify reputable interacting parties by trust should be developed for the DDLS, and therefore peers are able to identify trustworthy peers, obtain reliable information and allow access only from trusted or authenticated counterparts.

8.2.2 Research Directions

The idea of applying the P2P paradigm to open hypermedia has yet to be well researched and realised. This work provided the methodology of implementing such an idea through constructing a distributed link server system, the DDLS, on top of a P2P

model which empowers resource sharing collaboration among a community of people with similar knowledge background. Unlike any other work in the P2P community, this research identified the features of the DDLS from the point of view of open hypermedia, and pointed out that the unstructured P2P model was more suitable for the DDLS than the centralised P2P model in an ad hoc environment. Meanwhile, unlike any other work in the open hypermedia community, this research developed, in response to an unstructured P2P paradigm, a series of mechanisms to facilitate resource discovery.

This research has been undertaken in the particular context of P2P link server systems, and the author believes that the following two aspects will warrant long term investigation in pursuit of enhancing different kinds of OHSs with the full potential of the P2P paradigm. The first involves the attitude towards selecting (reusing) or developing the proper P2P technologies for the OHS in question, while the second advocates the use of other state-of-the-art technologies from associated disciplines.

Recently, research in the P2P community has advanced technologies to satisfy the requirements for P2P systems of distinct architectures. Some technologies can be directly applied to P2P open hypermedia. For instance, Lukka and Fallenstein (2002) utilised Freenet-like GUIDs for the location and retrieval of blocks of media content for implementing the Xanadu model. Their approach was successful because DHTs offer an efficient and scalable solution to keyword-based search queries. Whereas, designers of other systems, the DDLS for example, prefer semantic search queries which are intended to discover semantically related resources. This makes any existing P2P approach less applicable. Therefore, this work concentrated on developing new mechanisms to accommodate the requirement. Little other work in this area includes the hierarchy of resemblance (HR) search (Larsen and Bouvin 2004). The HR search relied on a hierarchy data structure in which peers were ranked according to the previous search results. The well-known random walk technique was adopted to facilitate searching the hierarchy for distributed hypermedia structures. These examples demonstrate that, although advanced P2P technologies are available for direct use, analysing the features of and requirements for the target OHS to discover the potential of developing more applicable, relevant and efficient alternatives, is always of paramount importance. The author also envisions the combination of distinct P2P technologies in an OHS wherever necessary.

This research has demonstrated the feasibility and potential of utilising technologies from multiple disciplines in implementing a P2P OHS. However, it assumed the existence of some non-trivial mechanisms, such as the one responsible for identifying the semantic relationship between resources and another that conducts service discov-

ery. The former can be accomplished by using ontologies as described in Section 8.2.1, while the latter may utilise technologies from Grid research in which service discovery is among the most active research topics. The author believes that successfully addressing such issues in the context of OHSs will also enlighten and facilitate P2P research.

Appendix A

Related Work on Semantic Search

Related work on semantic search presented in this appendix shares the same understanding of the DDLS semantic search (see Section 6.4.1) that semantic search is not carried out on the basis of the terms in a query but the concepts that carry the same meaning. They offer a domain specific approach to satisfactory semantic search on a static repository. Some rely on the information collected by a Web crawler and others utilise the result from traditional Web search, which makes it unable to provide up-to-date information. The target platform of the semantic search, in all these search mechanisms, is limited to systems with either a centralised or a client-server software architecture.

A.1 Latent Semantic Indexing

Latent Semantic Indexing (LSI) (Deerwester et al. 1990) is an information retrieval technique designed to overcome a fundamental problem of matching terms of queries with those of documents on the basis of concepts instead of terms. LSI uses a statistical technique called singular-value decomposition to explore the underlying semantic structure in term-document association data and create a concept space to reflect the major associative patterns observed in data.

A matrix of terms by documents is generated initially. Each row represents a term and each column corresponds to a document. Any entry in the matrix represents the frequency of a term in a document the corresponding row and column denote. The closely associated terms and documents are positioned near one another. The matrix is decomposed into the product of three other matrices: a *Term* $\times$ *m* matrix (A), an *m* $\times$ *m*

matrix (B) and an *m* $\times$ *document* matrix (C), where m[1] is the rank of the original matrix and the *m* $\times$ *m* matrix can be thought of as the concept space. The idea is that one can use matrix A to search for concepts. Given a concept, related concepts can be retrieved via matrix B. Subsequently, given all the concepts retrieved, desirable documents can be retrieved using matrix C.

LSI addresses two classical problems in the information retrieval domain: that of synonym (different terms with the same meaning) and polysemy (one term having more than one meaning). In addition, it has been reported to outperform more conventional vector-based methods with regard to recall and precision. A manifest problem with LSI is performance. The singular-value decomposition complexity is $\Theta(N^2m^3)$ where N is the number of terms and documents and m is the number of dimensions in the concept space. Also, determining the value for m is inherently another problem encountered by LSI. The optimal value for m has been observed in addressing various domain specific problems, which implies that the decision relies on the specific collection of documents of interest.

A.2 Simple HTML Ontology Extensions

The Simple HTML Ontology Extensions (SHOE) language allows users to define controlled, shareable and extensible vocabularies and associate machine understandable meaning with them (Heflin and Hendler 2000). The vocabularies are ontologies that consist of the definition of concepts and categories, and relationship between concepts. Web pages can embed semantic markup in SHOE to describe their content, with the relationship between the concepts on Web pages being indicated by SHOE ontologies. A Web crawler is developed to search for Web pages with SHOE markup, identify category and relationship claims on the page, and store them in a knowledge base (KB).

SHOE search provides a general-purpose query tool which requires that users specify the context of queries by choosing an ontology and associated properties that satisfy their needs. Complex queries can be constructed automatically in response to the type of arguments in the queries. SHOE search bears a Web search feature. For Web pages without SHOE markup, this feature translates a SHOE search query into one that can be accepted and processed by a number of popular search engines, which enhances the Web search with SHOE specific functionality.

[1]The value of m is relatively small with the range between 50 and 350.

A.3 ASCS Semantic Search

The DARPA Agent Markup Language (DAML) enables the creation of ontologies for any domain that support the unambiguous description of Web content (Lassila et al. 2000), and the Agent Semantic Communication Service (ASCS) allows users to make precise queries for information encoded in DAML/OWL on static repositories (Li et al. 2002). The ASCS consists of two main components: a Semantic Search Agent (SSA) that helps other agents to find entities on the basis of the ontologies they share, and a Semantic Translation Service (STS) that is responsible for supporting communication between agents using different ontologies.

A DAML crawler is used to parse DAML-encoded pages and construct indices for the content. A SSA accepts a DAML/RDF query and converts it into a Prolog query. DAML statements are stored as Prolog assertions in a server which, upon the receipt of a Prolog query from the SSA, examines its repository and returns the result to the SSA. In order to find more matches in other repositories, the SSA may also send the query to an available STS. The STS reformulates the Prolog query according to the destination ontology and further converts it into DAML/RDF format to be processed by another SSA known to perform the search on the destination ontology. When the result is returned by the second SSA, the STS translates it back into the starting ontology. All responses from different paths of search are merged and presented to the user.

A.4 W3C Semantic Search

The W3C semantic search[2] focuses on a search mechanism based on the denotation of the search query and utilises relevant information aggregated from a web of distributed machine understandable data created by the Semantic Web and Web services to augment traditional search results. Semantic search is viewed by Guha et al. (2003) as an application of the Semantic Web to search problems. It divides the entire search process into two parts: using a common search engine to provide the traditional text search result and obtaining relevant data extracted from the Semantic Web to augment the traditional search result.

W3C semantic search applications are built on top of an infrastructure named TAP which provides a set of mechanisms for websites to expose data onto the Semantic

[2]It is so named as to avoid being misunderstood with the general term of semantic search. In reality, W3C semantic search is one of the applications of the semantic search referred to in the current section.

Web and for applications to consume this data via a query interface called *GetData*. For websites without the corresponding machine understandable form of data, HTML scrapers are written to locate and covert the relevant pages into machine readable data and make them available via the *GetData* interface. The TAP knowledge base offers applications an ontology that defines a number of basic terms across a broad range of domains. Thus data sources in the form of a number of triples compose a Semantic Web.

The W3C semantic search employs a simple registry to keep track of which URL has values for which properties about which classes of resources. Therefore, a query to the registry can be redirected to the website that contains the answer. The search terms are mapped to the nodes in the Semantic Web by identifying their denotations. On the basis of all returned nodes corresponding to search terms, a breadth first search is conducted in the Semantic Web graph starting from the queried terms to collect the first N triples, where N is a predefined limit. The initial effort of W3C semantic search revolves around search queries denoting people.

Appendix B

Definitions of Terms and Variables Used in Simulation

broadcast rate	the time of broadcast carried out by all peers to propagate queries over a period of time.
cache rate	the percentage of peers whose topic information is in the cache compared to all peers in the system.
hops	delay in finding all answers as measured in the number of hops, also known as path length.
recall	the percentage of matches that can be found.
topic popularity	how popular a topic/topics is/are in terms of the number of peers holding it/them.
topic probability	the percentage of peers that possess the topic(s) compared to all peers in the system.
updating rate	the percentage of all peers launching an update to their resources (which also results in an update to the topic information about the resources) over a period of time.
usefulness	the relative extent to which a peer should become a neighbour of another peer during re-organisation.

Bibliography

Anderson, Kenneth M. (1997). Integrating open hypermedia systems with the World Wide Web. In *Proceedings of the 8th ACM conference on Hypertext*, Southampton, UK, pp. 157–166. ACM Press.

Anderson, Kenneth M., Richard N. Taylor, and E. James Whitehead, Jr. (1994). Chimera: hypertext for heterogeneous software environments. In *Proceedings of the 1994 ACM European conference on Hypermedia technology*, Edinburgh, Scotland, pp. 94–107. ACM Press.

Andrews, Keith, Frank Kappe, and Hermann Maurer (1995). Serving information to the Web with Hyper-G. In *Proceedings of the 3rd International World-Wide Web Conference on Technology, Tools and Applications*, Darmstadt, Germany, pp. 919–926. Elsevier North-Holland, Inc.

Apple Computer Inc. (1989). *HyperCard stack design guidelines*. Boston, MA, USA: Addison-Wesley Longman Publishing Co.

Bass, Len, Paul Clements, and Rick Kazman (2003). *Software Architecture in Practice* (2nd ed.). Addison Wesley Professional.

Belkin, Nicholas J. and W. Bruce Croft (1992). Information filtering and information retrieval: Two sides of the same coin? *Communications of the ACM 35*(12), 29–38.

Berners-Lee, Tim (1996). The World Wide Web - Past, Present and Future. *Journal of Digital information 1*(1).

Berners-Lee, Tim, Robert Cailliau, Jean-Francois Groff, and Bernd Pollermann (1992). World-Wide Web: The Information Universe. *Electronic Networking: Research, Applications and Policy 1*(2), 74–82.

Berners-Lee, Tim, James Hendler, and Ora Lassila (2001). The Semantic Web. *Scientific American 284*(5), 34–43.

Breslau, Lee, Pei Cao, Li Fan, Graham Phillips, and Scott Shenker (1999). Web Caching and Zipf-like Distributions: Evidence and Implications. In *Proceedings*

of IEEE INFOCOM'99, New York, NY, USA, pp. 126–134.

Bush, Vannevar (1945). As We May Think. *The Atlantic Monthly*.

Carmody, Steven, Walter Gross, Theodor H. Nelson, David Rice, and Andries van Dam (1969). A Hypertext Editing System for the /360. In M. Faiman and J. Nievergelt (Eds.), *Pertinent Concepts in Computer Graphics*, Urbana, IL, USA, pp. 291–330. University of Illinois Press.

Carr, Leslie, Sean Bechhofer, Carole Goble, and Wendy Hall (2001). Conceptual Linking: Ontology-based Open Hypermedia. In *Proceedings of the 10th international conference on World Wide Web*, Hong Kong, pp. 334–342. ACM Press.

Carr, Leslie, David De Roure, Wendy Hall, and Gary Hill (1995). The Distributed Link Service: A Tool for Publishers, Authors and Readers. In *Proceedings of the 4th International World Wide Web Conference: The Web Revolution*, Boston, Massachusetts, USA, pp. 647–656.

Carr, Leslie, David De Roure, Wendy Hall, and Gary Hill (1998a). Implementing an Open Link Service for the World Wide Web. *World Wide Web Journal 1*(2), 61–71.

Carr, Leslie, Wendy Hall, and Steve Hitchcock (1998b). Link Services or Agent Services? In *Proceedings of the 9th ACM conference on Hypertext and hypermedia*, Pittsburgh, Pennsylvania, USA, pp. 113–122. ACM Press.

Clark, David (2001). Face-to-Face with Peer-to-Peer Networking. *Computer 34*(1), 18–21.

Clarke, Ian, Oskar Sandberg, Brandon Wiley, and Theodore W. Hong (2001). Freenet: A Distributed Anonymous Information Storage and Retrieval System. In *Proceedings of the ICSI Workshop on Design Issues in Anonymity and Unobservability*, Berkeley, CA, USA, pp. 311–320.

Crespo, Arturo and Hector Garcia-Molina (2002a). Routing Indices for Peer-to-Peer Systems. In *Proceedings of the 22nd International Conference on Distributed Computing Systems*, Vienna, Austria, pp. 23–34. IEEE Computer Society.

Crespo, Arturo and Hector Garcia-Molina (2002b). Semantic Overlay Networks for P2P Systems. Technical report, Computer Science Department, Stanford University.

D'Arlach, Carmen Ximena and John J. Leggett (1994). A Spatial Hypertext Editor. Technical Report TAMU-HRL-94-005, Hypermedia Research Laboratory, Department of Computer Science, Texas A&M University, College Station, Texas.

Davis, Hugh C. (1998). Referential Integrity of Links in Open Hypermedia Systems. In *Proceedings of the 9th ACM conference on Hypertext and hypermedia*, Pittsburgh, Pennsylvania, USA, pp. 207–216. ACM Press.

De Roure, David, Leslie Carr, Wendy Hall, and Gary Hill (1996). A Distributed Hypermedia Link Service. In *Proceedings of the 3rd International Workshop on Services in Distributed and Networked Environments (SDNE'96)*, pp. 156–161.

De Roure, David C., Nigel G. Walker, and Leslie A. Carr (2000). Investigating Link Service Infrastructures. In *Proceedings of the 11th ACM on Hypertext and hypermedia*, San Antonio, Texas, USA, pp. 67–76. ACM Press.

Deerwester, Scott, Susan T. Dumais, George W. Furnas, Thomas K. Landauer, and Richard Harshman (1990). Indexing by Latent Semantic Analysis. *Journal of the American Society of Information Science 41*(6), 391–407.

Englebart, Doug (1986). The augmented knowledge workshop. In *Proceedings of the ACM Conference on the history of personal workstations*, Palo Alto, CA, USA, pp. 73–83. ACM Press.

Fountain, Andrew M., Wendy Hall, Ian Heath, and Hugh C. Davis (1990). MICROCOSM: An Open Model for Hypermedia with Dynamic Linking. In A. Rizk, N. Streitz, and J. Andre (Eds.), *Proceedings of the European Conference on Hypertext (ECHT'90),*, Paris, France, pp. 298–311. Cambridge University Press.

Ganesan, Prasanna, Qixiang Sun, and Hector Garcia-Molina (2003). YAPPERS: A Peer-to-Peer Lookup Service Over Arbitrary Topology. In *Proceedings of IEEE INFOCOM*, San Francisco, California, USA, pp. 1250– 1260.

Gnutella (2001). The gnutella home page. http://www.gnutella.com/.

Goose, Stuart (1997). *A Framework for Distributed Open Hypermedia*. Ph. D. thesis, University of Southampton.

Grønbæk, Kaj, Niels Olof Bouvin, and Lennert Sloth (1997). Designing Dexter-based hypermedia services for the World Wide Web. In *Proceedings of the 8th ACM conference on Hypertext*, Southampton, United Kingdom, pp. 146–156. ACM Press.

Grønbæk, Kaj, Jens A. Hem, Ole L. Madsen, and Lennert Sloth (1993). Designing Dexter-based cooperative hypermedia systems. In *Proceedings of the 5th ACM conference on Hypertext*, Seattle, Washington, USA, pp. 25–38. ACM Press.

Grønbæk, Kaj, Lennert Sloth, and Peter Ørbæk (1999). Webvise: Browser and Proxy Support for Open Hypermedia Structuring Mechanisms on the WWW.

In *Proceedings of the 8th International Conference on World Wide Web*, Toronto, Canada, pp. 1331–1345. Elsevier North-Holland, Inc.

Grønbæk, Kaj and Randall H. Trigg (1994). Design Issues for a Dexter-Based Hypermedia System. *Communications of the ACM 37*(2), 40–49.

Guha, R., Rob McCool, and Eric Miller (2003). Semantic Search. In *Proceedings of the 12th international conference on World Wide Web*, Budapest, Hungary, pp. 700–709. ACM Press.

Haan, Bernard J., Paul Kahn, Victor A. Riley, James H. Coombs, and Norman K. Meyrowitz (1992). IRIS hypermedia services. *Communications of the ACM 35*(1), 36–51.

Halasz, Frank and Mayer Schwartz (1994). The Dexter hypertext reference model. *Communications of the ACM 37*(2), 30–39.

Halasz, Frank G. (1988). Reflections on Notecards: Seven Issues for the Next Generation of Hypermedia Systems. *Communications of the ACM 31*(7), 836–852.

Halasz, Frank G., Thomas P. Moran, and Randall H. Trigg (1986). Notecards in a nutshell. In *Proceedings of the SIGCHI/GI conference on Human factors in computing systems and graphics interface*, Toronto, Ontario, Canada, pp. 45–52. ACM Press.

Hall, Wendy, Hugh Davis, and Gerard Hutchings (1996). *RETHINKING HYPERMEDIA: The Microcosm Approach*. Norwell, MA, USA: Kulwer Academic Publishers.

Heflin, Jeff and James Hendler (2000). Searching the Web with SHOE. In *AAAI Workshop on Artificial Intelligence for Web Search*, Menlo Park, CA, USA, pp. 35–40. AAAI Press.

Jennings, Nicholas R., Katia Sycara, and Michael Wooldridge (1998). A Roadmap of Agent Research and Development. *Journal of Autonomous Agents and Multi-Agent Systems 1*(1), 7–38.

Kahan, José, Marja-Ritta Koivunen, Eric Prud'Hommeaux, and Ralph R. Swick (2001). Annotea: An Open RDF Infrastructure for Shared Web Annotations. In *Proceedings of the 10th international conference on World Wide Web*, Hong Kong, pp. 623–632. ACM Press.

Kamvar, Sepandar D., Mario T. Schlosser, and Hector Garcia-Molina (2003). The Eigentrust Algorithm for Reputation Management in P2P Networks. In *Proceedings of the 12th International World Wide Web Conference*, Budapest, Hungary, pp. 640–651. ACM Press.

Larsen, René Dalsgaard and Niels Olof Bouvin (2004). HyperPeer: Searching for Resemblance in a P2P Network. In *Proceedings of the 15th ACM conference on Hypertext and hypermedia*, Santa Cruz, California, USA, pp. 268–269.

Lassila, Ora and Ralph R. Swick (1999). Resource Description Framework (RDF) Model and Syntax Specification. W3C Recommendation, World Wide Web Consortium.

Lassila, Ora, Frank van Harmelen, Ian Horrocks, James Hendler, and Deborah L. McGuinness (2000). The semantic Web and its languages. *IEEE Intelligent Systems 15*(6), 67–73.

Ledlie, Jonathan, Jacob M. Taylor, Laura Serban, and Margo Seltzer (2002). Self-Organization in Peer-to-Peer Systems. In *Proceedings of the 10th ACM SIGOPS European Workshop*.

Li, John, Adam Pease, and Christopher Barbee (2002). Experimenting with ASCS Semantic Search. Project report, Teknowlege Corporation, Palo Alto, CA, USA.

Lowe, David and Wendy Hall (1999). *Hypermedia & the Web*. John Wiley & Sons, Inc.

Lukka, Tuomas J. and Benja Fallenstein (2002). Freenet-like GUIDs for Implementing Xanalogical Hypertext. In *Proceedings of the 13th ACM conference on Hypertext and hypermedia*, College Park, Maryland, USA, pp. 194–195. ACM Press.

Lv, Qin, Pei Cao, Edith Cohen, Kai Li, and Scott Shenker (2002). Search and Replication in Unstructured Peer-to-Peer Systems. In *Proceedings of the 16th international conference on Supercomputing*, New York, New York, USA, pp. 84–95. ACM Press.

Maurer, H. (1996). *Hyper-G is now HyperWave: The Next Generation Web Solution*. Addison-Wesley Publishing Company.

Meyrowitz, Norman (1986). Intermedia: The Architecture and Construction of an Object-Oriented Hypemedia System and Applications Framework. In *Proceedings of Conference on Object Oriented Programming Systems Languages and Applications*, Portland, Oregon, USA, pp. 186–201. ACM Press.

Milojicic, Dejan S., Vana Kalogeraki, Rajan Lukose, Kiran Nagaraja, Jim Pruyne, Bruno Richard, Sami Rollins, and Zhichen Xu (2002). Peer-to-Peer Computing. Technical Report HPL-2002-57, HP Laboratories, Palo Alto, CA, USA.

Moreau, Luc, Nick Gibbins, David De Roure, Samhaa El-Beltagy, Wendy Hall, Gareth Hughes, Dan Joyce, Sanghee Kim, Danius Michaelides, Dave Millard,

Sigi Reich, Robert Tansley, and Mark Weal (2000). SoFAR with DIM Agents: An Agent Framework for Distributed Information Management. In Jeffrey Bradshaw and Geoff Arnold (Eds.), *Proceedings of the 5th International Conference and Exhibition on The Practical Application of Intelligent Agents and Multi-Agents*, Manchester, UK, pp. 369–388. The Practical Application Company Ltd.

Mullin, James (1990). Optimal Semijoins for Distributed Database Systems. *IEEE Transactions on Software Engineering 16*(5), 558–560.

Napster (2001). The napster home page. http://www.napster.com.

Nelson, Theodore H. (1987). *Computer Lib/Dream Machines, rev. ed.* Redmond, WA, USA: Microsoft Press.

Nelson, Theodor Holm (1995). The Heart of Connection: Hypermedia Unified by Transclusion. *Communications of the ACM 38*(8), 31–33.

Nielsen, Jakob (1990). *Hypertext & Hypermedia*. San Diego, CA, USA: Academic Press Professional, Inc.

Nürnberg, Peter J., John J. Leggett, and Erich R. Schneider (1997). As We Should Have Thought. In *Proceedings of the 8th ACM conference on Hypertext*, Southampton, United Kingdom, pp. 96–101. ACM Press.

Nürnberg, Peter J., John J. Leggett, Erich R. Schneider, and John L. Schnase (1996). Hypermedia Operating Systems: A New Paradigm for Computing. In *Proceedings of the Hypertext '96 Conference*, Bethesda, Maryland, USA, pp. 194–202. ACM Press.

Østerbye, Kasper and Uffe Kock Wiil (1996). The Flag Taxonomy of Open Hypermedia Systems. In *Proceeding of the 7th ACM Conference on Hypertext*, Bethesda, Maryland, USA, pp. 129–139. ACM Press.

Pearl, Amy (1989). Sun's Link Service: A Protocol for Open Linking. In *Proceedings of the 2nd annual ACM conference on Hypertext*, Pittsburgh, Pennsylvania, USA, pp. 137–146. ACM Press.

Ratnasamy, Sylvia, Paul Francis, Mark Handley, Richard Karp, and Scott Shenker (2001). A scalable content-addressable network. In *Proceedings of the 2001 ACM SIGCOMM Conference*, San Diego, California, USA, pp. 161–172. ACM Press.

Ratnasamy, Sylvia, Scott Shenker, and Ion. Stoica (2002). Routing Algorithms for DHTs: Some Open Questions. In *Peer-to-Peer Systems: 1st International-Workshop, IPTPS 2002*, Volume 2429/2002, Cambridge, MA, USA, pp. 45–52. Springer-Verlag Heidelberg.

Reich, Sigi, Uffe K. Wiil, Peter J. Nürnberg, Hugh C. Davis, Kaj Grønbæk, Kenneth M. Anderson, David E. Millard, and Jörg M. Haake (1999). Addressing Interoperability in Open Hypermedia: The Design of the Open Hypermedia Protocol. *The New Review of Hypermedia and Multimedia (NRHM) 5*, 207–248.

Ritter, Jordan (2001). Why Gnutella Can't Scale. No, Really. http://www.darkridge.com/jpr5/doc/gnutella.html.

Rosenblatt, M. (1956). Remarks on some nonparametric estimates of a density function. *Annals Mathematical Statistics 27*, 832–837.

Rowstron, Antony and Peter Druschel (2001). Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems. In *Proceedings of the 18th IFIP/ACM International Conference on Distributed Systems Platforms (Middleware 2001)*, Heidelberg, Germany.

Silberschatz, Abraham and Peter Baer Galvin (1994). *Operating Systems Concepts* (4 ed.). Reading, MA: Addison-Wesley.

Silverman, B. W. (1986). Density estimation for statistics and data analysis. *Monographs on Statistics and Applied Probability*.

Smith, Michael K., Chris Welty, and Deborah L. McGuinness (2004). OWL Web Ontology Language Guide. W3C Recommendation, World Wide Web Consortium.

Sripanidkulchai, Kunwadee (2001). The popularity of Gnutella queries and its implications on scalability. Featured on O'Reilly's www.openp2p.com website.

Stoica, Ion, Robert Morris, David Karger, M. Frans Kaashoek, and Hari Balakrishnan (2001). Chord: A Scalable Peer-to-peer Lookup Service for Internet Applications. In *Proceedings of the 2001 ACM SIGCOMM Conference*, San Diego, California, USA, pp. 149–160. ACM Press.

The Open Group (1997). DCE 1.1: Remote Procedure Call. Technical Standard C706, The Open Group.

Theodoridis, Sergios and Konstantinos Koutroumbas (1999). *Pattern Recognition* (1 ed.). Academic Press.

Tzagarakis, Manolis, Dimitris Avramidis, Maria Kyriakopouliu, monica schraefel, Michalis Vaitis, and Dimitris Christodoulakis (2002). Structuring primitives in the callimachus component-based open hypermedia system. *Journal of Network and Computer Applications 26*(1), 139–162.

Tzagarakis, Manolis, Michalis Vaitis, Athanasios Papadopoulos, and Dimitris Christodoulakis (1999). The Callimachus Approach to Distributed Hyperme-

dia. In *Proceedings of the 10th ACM Conference on Hypertext and hypermedia*, Darmstadt, Germany, pp. 47–48. ACM Press, New York, NY, USA.

van Ossenbruggen, Jacco, Lynda Hardman, and Lloyd Rutledge (2002). Hypermedia and the Semantic Web: A Research Agenda. *Journal of Digital information 3*(1).

Wiil, Uffe Kock (1997). Open Hypermedia: Systems, Interoperability and Standards. *Journal of Digital Information 1*(2).

Wiil, Uffe Kock and John J. Leggett (1996). The HyperDisco Approach to Open Hypermedia Systems. In *Proceedings of the 7th ACM conference on Hypertext*, Bethesda, Maryland, USA, pp. 140–148. ACM Press.

Wiil, Uffe Kock and John J. Leggett (1997). Workspaces: The HyperDisco approach to Internet distribution. In *Proceedings of the 8th ACM Conference on Hypertext*, Southampton, UK, pp. 13–23. ACM Press.

Wiil, Uffe Kock and Peter J. Nürnberg (1999). Evolving hypermedia middleware services: lessons and observations. In *Proceedings of the 1999 ACM Symposium on Applied Computing*, San Antonio, Texas, USA, pp. 427–436. ACM Press.

Wiil, Uffe K., Samir Tata, and David L. Hicks (2003). Cooperation Services in the Construct Structural Computing Environment. *Journal of Network and Computer Applications 26*(1), 115–137.

Yankelovich, Nicole, Bernard J. Haan, Norman K. Meyrowitz, and Steven M. Drucker (1988). Intermedia: The Concept and the Construction of a Seamless Information Environment. *IEEE Computer 21*(1), 81–96.

Zhou, Jing, Vijay Dialani, David De Roure, and Wendy Hall (2003). A Distance Based Semantic Search Algorithm for Peer-to-Peer Open Hypermedia Systems. In Pingzhi Fan and Hong Shen (Eds.), *Proceedings of the 4th International Conference on Parallel and Distributed Computing, Applications and Technologies*, Chengdu, China, pp. 7–11. IEEE Press.

Zhou, Jing, Wendy Hall, and David De Roure (2004). When Open Hypermedia Meets Peer-to-Peer Computing. In *Proceedings of the 15th ACM conference on Hypertext and hypermedia*, Santa Cruz, California, USA, pp. 266–267. ACM Press.

Zipf, George Kingsley (1949). *Human Behavior and the Principle of Least Effort*. Cambridge, MA: Addison-Wesley.