Rewiring Peer Connections over Semantic Overlay Networks

Rewiring Peer Connections over Semantic Overlay Networks

Paraskevi Raftopoulou

Ph.D. Thesis, Department of Electronic & Computer Engineering

Technical University of Crete, December 2009

Rewiring Peer Connections over Semantic Overlay Networks

Paraskevi Raftopoulou

A thesis submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy in
Electronic & Computer Engineering

Doctoral Committee

Associate Professor Euripides G.M. Petrakis, Supervisor
Professor Stavros Christodoulakis, Member
Assistant Professor Vasilis Samoladas, Member

Technical University of Crete
2009

Acknowledgements

Writing a doctoral thesis needs assistance and supervision. Sincere thanks to my supervisor, Assoc. Prof. *Euripides G.M. Petrakis* for his unending support throughout the years of my PhD. His enthusiasm and desire for perfection was an inspiration for me. I will always consider him a friend and a valuable source of advice.

I would also like to thank the members of my doctoral committee, Prof. *Stavros Christodoulakis*, Assist. Prof. *Vasilis Samoladas*, Assist. Prof. *Michail G. Lagoudakis*, Assoc. Prof. *Manolis Koubarakis*, Prof. *Dimitris Plexousakis* and Prof. *Timos Sellis* for their comments and suggestions on improvements and extensions of this work.

Last years, I was a member of the *Intelligent Systems Research Group*. Many thanks to all my my colleagues in the group for their continuous support and to the technical staff for providing a professional research infrastructure.

Throughout this thesis I received financial support from the *Technical University of Crete* (TUC) and from the Greek Ministry of Education through the *Heraclitus* fellowship program. I would like to thank all the people in the institutions for these research grants.

During an eight-month research visit, I was allowed to work in a great research group in Max-Panck-Institut für Informatik (MPII). A special thanks goes to Prof. *Gerhard Weikum* for his insightful comments on improvements and extensions of this work. Many thanks to all members of the *Databases and Information Systems Department* for the nice and international atmosphere.

Special thanks go to my family for their endless love and encouragement throughout all my life endeavors. Their support in difficulties was an extra motivation for this effort. I hope they are proud of me.

Words cannot always express the feelings for special persons that make your life meaningful. Persons that are a stand in difficult personal moments and a relief after long working hours. Christos Tryfonopoulos, a heartfelt thanks for your endless love and unconditional support.

The good times spent with Amalia, Petros, Georgia, Costas, Nikos, Sotiris, Matoula, Stamatis, Despina, Natasha, and Christiana made this thesis an easier task. A big thanks for your indispensable friendship.

To my husband Christos,
for our long journey together.

Περίληψη

Τα τελευταία χρόνια, δεδομένα (π.χ., ερευνητικές εργασίες, βιβλία κτλ.) παράγονται με καταιγιστικούς ρυθμούς και γίνονται διαθέσιμα από διαφορετικές κι ανεξάρτητες πηγές πληροφορίας (π.χ., βιβλιοθήκες πανεπιστημίων ή απλούς χρήστες) κατανεμημένες οπουδήποτε στον κόσμο. Τα συστήματα ομότιμων κόμβων (peer-to-peer) αποτελούν μια ιδιαίτερα δημοφιλή υποδομή για την οργάνωση των κατανεμημένων πηγών (και των δεδομένων τους), αφού είναι εύκολα επεκτάσιμα για μεγάλο αριθμό πηγών, προσαρμόσιμα στις δυναμικές συμπεριφορές των πηγών και ανεκτικά σε λάθη. Στα συστήματα αυτά ένας μεγάλος αριθμός από αυτόνομους κόμβους διαθέτει τους πόρους του και οι κόμβοι επικοινωνούν μεταξύ τους για την ανταλλαγή δεδομένων και υπηρεσιών. Στη βιβλιογραφία, υπάρχουν πολλές ερευνητικές εργασίες που αφορούν στην τοπολογία των δικτύων ομότιμων κόμβων, στις δυνατότητες δρομολόγησης ερωτήσεων πάνω από τέτοιου είδους δίκτυα, στους μηχανισμούς αποθήκευσης των δεδομένων (ή των ερωτήσεων) και στους μηχανισμούς ταιριάσματος ερωτήσεων και δεδομένων. Τελικά, το ερευνητικό ενδιαφέρον έχει μετατοπιστεί από τα απλά δίκτυα (π.χ., Napster, Gnutella) σε πιο πολύπλοκα βασισμένα για παράδειγμα, σε κατανεμημένους πίνακες κατακερματισμού(distributed hash tables - DHTs), ή σε πρωτόκολλα μικρο-κόσμου (small-world protocols).

Στόχος των πρωτοκόλλων στα δίκτυα ομότιμων κόμβων είναι η οργάνωση, η διαχείριση κι η ανάκτηση των δεδομένων χωρίς ιδιαίτερη προσπάθεια από τους χρήστες του συστήματος, με το ελάχιστο κόστος από πλευράς μηνυμάτων και φόρτου του δικτύου και φυσικά με την καλύτερη δυνατή απόδοση από πλευράς απαντήσεων στον χρήστη. Τα σημασιολογικά δίκτυα επικάλυψης (semantic overlay networks) προσφέρουν έναν τρόπο οργάνωσης των δικτύων ομαδοποιώντας τις πηγές πληροφορίας. Αυτή η ομαδοποίηση επιτυγχάνεται με πρωτόκολλα που εκτελούνται ανεξάρτητα σε κάθε κόμβο του

δικτύου με σκοπό να ενώσουν τους κόμβους που έχουν κοινά ενδιαφέροντα (π.χ., παρόμοιο περιεχόμενο). Σε τέτοιου είδους οργανωμένα δίκτυα, οι ερωτήσεις στέλνονται στην ομάδα πηγών που έχει τις περισσότερες πιθανότητες να γνωρίζει την απάντηση και έτσι, η ανάκτηση των δεδομένων γίνεται πιο αποδοτική.

Σε αυτήν την εργασία, παρουσιάζουμε το i*Cluster*, μια κατανεμημένη αρχιτεκτονική ομότιμων κόμβων για ανάκτηση δεδομένων, και ορίζουμε προσεκτικά όλα τα πρωτόκολλα που διέπουν τις αλληλεπιδράσεις των κόμβων. Κάθε κόμβος περιγράφει τα ενδιαφέροντά του (peer interests) στο δίκτυο, και ακολουθεί τα συγκεκριμένα πρωτόκολλα για να συνδέεται στο δίκτυο (peer join), να βρίσκει κόμβους με κοινά ενδιαφέροντα και να ανανεώνει τις συνδέσεις του (peer rewiring), καθώς επίσης για να δρομολογεί ερωτήσεις (query routing), να βρίσκει και να επιστρέφει απαντήσεις στον ενδιαφερόμενο χρήστη (document retrieval).

Η δουλειά μας εστιάζει στην ανανέωση των συνδέσεων και στην οργάνωση των κόμβων. Μελετάμε τα λειτουργικά ζητήματα που σχετίζονται με την ανανέωση των συνδέσεων μεταξύ των κόμβων και συζητάμε μια σειρά από επιλογές που αφορούν στο σχεδιασμό της στρατηγικής οργάνωσης των κόμβων. Επίσης, προτείνουμε ένα νέο μέτρο για την αξιολόγηση της οργάνωσης του δικτύου, το οποίο ονομάζουμε ποιότητα ομαδοποίησης (clustering efficiency) κ. Χρησιμοποιούμε αυτό το μέτρο για να αξιολογήσουμε τις στρατηγικές οργάνωσης. Κατόπιν, θεωρούμε μια ρεαλιστική δυναμική συμπεριφορά του δικτύου (δηλ. ένα δίκτυο όπου συμβαίνουν εισαγωγές και διαγραφές κόμβων και περιεχομένου) και μελετάμε τον τρόπο που αυτή η δυναμική συμπεριφορά επηρεάζει την οργάνωση των κόμβων και φυσικά την ανάκτηση δεδομένων. Τέλος, προσαρμόζουμε την αρχιτεκτονική μας ώστε, παράλληλα με την ανάκτηση, να υποστηρίζει και διάχυση πληροφορίας, και την εφαρμόζουμε σε ένα περιβάλλον κατανεμημένων ψηφιακών βιβλιοθηκών.

Η πειραματική αξιολόγηση της αρχιτεκτονικής μας έγινε με δύο πραγματικές συλλογές κειμένων: (α) μια συλλογή από 556.000 κείμενα γενικού ενδιαφέροντος και (β) μια συλλογή από περίπου 32.000 ιατρικά κείμενα. Τα κείμενα αυτά μοιράστηκαν σε ένα σύνολο από κατανεμημένους και ανεξάρτητους κόμβους. Η αρχιτεκτονική i*Cluster* αποδεικνύεται πολύ πιο αποδοτική σε σύγκριση με (α) μια γνωστή από τη βιβλιογραφία μέθοδο αυτό-οργάνωσης κόμβων που υποστηρίζει μηχανισμούς ανάκτησης πληροφο-

ρίας και (β) τη μέθοδο διάχυσης ερωτήσεων στο Διαδίκτυο (flooding). Τα πρωτόκολλά μας οργανώνουν τους κόμβους του δικτύου γρήγορα κι η οργάνωση αυτή βοηθάει θεαματικά στην ανάκτηση των δεδομένων. Η αρχιτεκτονική μας αποδεικνύεται αποδοτική ακόμα και σε δυναμικά δίκτυα. Στην εφαρμογή των πρωτοκόλλων στο περιβάλλον των κατανεμημένων ψηφιακών βιβλιοθηκών, τα πειραματικά αποτελέσματα δείχνουν χαμηλό φόρτο μηνυμάτων, αποδοτική ανάκτηση δεδομένων και εύκολη προσαρμογή της αρχιτεκτονικής μας σε μεγάλα δίκτυα. Τέλος, τα πειραματικά αποτελέσματα επιβεβαιώνουν την εξάρτηση της ανάκτησης των δεδομένων από τη στρατηγική οργάνωσης των κόμβων, και μας δίνουν στοιχεία για την επιλογή της κατάλληλης στρατηγικής οργάνωσης.

Abstract

Today's content providers are naturally distributed and produce large amounts of information every day, making peer-to-peer data management a promising approach offering scalability, adaptivity to dynamics and failure resilience. The management of large volumes of data in peer-to-peer networks has generated additional interest in methods for effective network organisation based on providers' contents and consequently, in methods supporting information retrieval. Overlay architectures (or overlay networks) are being introduced as tools for the organisation and sharing of information residing in a network of peers. Moreover, semantic overlay networks partition the overlay layer by clustering the data sources. This is achieved through a rewiring protocol that is executed independently by each peer. The purpose of this protocol is to establish connections among peers being semantically, thematically or socially similar (i.e., peers sharing similar interests). By this, the problem of finding the most relevant resources is reduced to one of locating the clusters similar to the query.

In this thesis, we present i*Cluster*, a generic architecture for supporting full-fledged information retrieval in large-scale peer-to-peer networks, along with its associated organisation and query forwarding protocols. The main focus of this work is on rewiring. We study the functional issues related to peer rewiring and discuss a number of choices in designing the rewiring strategy. In addition, we introduce clustering efficiency κ, a measure that quantifies the quality of network organisation by exploiting the underlying network structure. Clustering efficiency is used for evaluating the performance of rewiring. We study the system performance and identify the rewiring protocol that proves efficient under peer churn. Finally, we apply our architecture in a digital library use case and support searching and

filtering functionality using the same infrastructure.

The experimental evaluation with real-word data and queries demonstrates that i*Cluster* achieves significant performance improvements (in terms of communication load and retrieval accuracy) over a state-of-the-art peer-to-peer clustering method. Compared to exhaustive search by flooding, i*Cluster* exchanged a small loss in retrieval accuracy for much less message flow. In addition, the proposed distributed protocols are proven efficient under peer churn, shown to achieve high retrieval accuracy and scale up well for large networks. Our experimental results confirm the dependence between rewiring strategies and retrieval performance, and give insights on the trade-offs involved in the selection of a rewiring strategy.

Contents

List of Tables

List of Figures

List of Abbreviations

APS	Access Point Structure
ASL	Adaptive Space Linearisation
BS	Biased Sampling
CAN	Context Addressable Network
DHT	Distributed Hash Table
DL	Digital Library
GUID	Globally Unique Identifier
GW	Gradient Walk
HTTP	HyperText Transfer Protocol
IF	Information Filtering
IP	Internet Protocol
IR	Information Retrieval
LSI	Latent Semantic Indexing
P2P	Peer-to-Peer
RDF	Resource Description Framework
RDF(S)	Resource Description Framework Schema
RT	Routing Table

RS	Random Sampling
RW	Random Walk
SDI	Selective Dissemination of Information
SL	Symmetric Links
SON	Semantic Overlay Network
SRT	Semantic Routing Table
SSW	Semantic Small World
TTL	Time-To-Live
URL	Uniform Resource Locators
VSM	Vector Space Model
XML	eXtensible Markup Language

Chapter 1

Introduction

Knowledge sharing aims at real-time, low overhead and dynamic information sharing and dissemination within user communities (e.g., members of an organisation) sharing common interests or goals [AP04]. The introduction of peer-to-peer (P2P) paradigm to knowledge sharing in user communities is currently seen as a promising technology to overcome many challenges. P2P systems offer the potential for low-cost sharing of information, while ensuring autonomy and privacy of the participating entities [MKL+02]. The main idea behind P2P networks is that, instead of relying on central components, functionality is provided through decentralised overlay architectures. Overlay architectures (or overlay networks) are being introduced as tools for the organisation and sharing of different aspects of knowledge residing in a network of peers. In overlay networks, information sources are organised independently of the underlying physical network: "close nodes" might exist within the same sub-network or across the Internet. Formally, a P2P overlay network is a directed graph with vertices corresponding to information sources and edges corresponding to links between nodes (or information sources). Nodes connected with each other are called *neighbours*. Queries are then, propagated within the network (searching for information qualifying query criteria) by following the edges. Gnutella[1] and Freenet[2] are two well-known systems built upon the idea of (unstructured) overlay networks. The popularity of these systems has propelled re-

[1] http://www.gnu.org/
[2] http://freenetproject.org/

search towards this direction, while the proliferation of social networking has added yet another interesting dimension to the problem of searching for content in overlay networks.

Semantic Overlay Networks (SONs) partition the overlay layer by clustering data sources. In this case, the problem of finding the most relevant resources is reduced to one of locating the cluster nearest to the query. The idea behind SONs is that peers being semantically, thematically or socially similar (i.e., peers sharing similar interests) are *organised* into groups. While being highly flexible, SONs improve query performance and guarantee high degree of peer autonomy [GMY02]. This technology has proven useful not only for information sharing in distributed environments, but also as a natural distributed alternative to Web 2.0 application domains, such as decentralised social networking in the spirit of Flickr[3] or del.icio.us[4]. Contrary to structured overlay networks that focus on providing accurate location mechanisms (e.g., [SMLN+03, RFH+01]), SONs are better suited for loose P2P architectures assuming neither a specific network structure nor total control over the location of the data. Additionally, SONs offer better support of semantics due to their ability to provide mechanisms for approximate, range, or text queries, and emphasise on peer autonomy (e.g., [KA04, TXD03, LW06, Sch05, ACMH03]).

The management of large volumes of data in P2P networks has generated additional interest in methods for effective *network organisation* based on peer contents and consequently, in methods supporting *information retrieval* (IR) over semantic overlay networks (e.g., [KP08, KJ04]). Network organisation in a SON is achieved through a *rewiring protocol* that is executed independently by each peer. The purpose of this protocol is to establish connections among *similar peers*. This is achieved by creating new connections to similar peers and by discarding connections that are outdated or pointing to dissimilar peers. By this, *clusters* of peers with similar interests emerge in the network. Queries can then be resolved by routing the query towards clusters (instead of sole peers) based on their likelihood to match the query. Once reaching a cluster, the peer receiving the query is responsible for forwarding

[3]http://www.flickr.com/
[4]http://www.del.icio.us.com/

it to other peers within the same cluster (i.e., to similar peers).

1.1 Problem Definition

Rewiring aims to establish clusters of peers (e.g., based on their content) by creating connections among similar peers and by disregarding connections that are outdated or pointing to dissimilar peers. Provided such a network structure, information retrieval is resolved by routing each query towards clusters based on their likelihood to match the query. Once reaching a cluster, the peer receiving the query is responsible for forwarding it to other peers within the same cluster. By this process, it is therefore entailed that the underlying network organisation affects both the retrieval performance and the network load.

Existing research solutions assume a fixed rewiring procedure for organising semantic overlay networks and search the network by exploiting certain architectural [KJ04, TXD03, TXKN03] or modelling [ACMH03, SL06] aspects. However, the effectiveness of information retrieval depends not only on the type of content stored by the peers and on the type of the queries allowed but also, on the rewiring procedure applied.

1.2 Solution Outline

The present work goes the existing research one step further and suggests that the retrieval performance depends on the rewiring strategy applied. To the best of our knowledge, there is no comprehensive study showing the effect of rewiring on system's performance. This is exactly the problem this thesis is dealing with.

Overall, this thesis provides a better understanding on the functional issues related to the design and performance of the rewiring protocol applied and its effects on retrieval accuracy and network load. To show proof of concept, we propose i*Cluster*, a generic P2P architecture that supports information retrieval functionality. i*Cluster* utilises widely adopted techniques from semantic overlay networks, such as characterisation of peers by content, peer clustering and link rewiring. In our SON architecture, peers are characterised by documents in the

spirit of [SMZ03, LC03, LLS04, Sch04, KJ04, MSZ05]. Each peer in the network maintains links to other peers with similar content to form clusters. This is achieved through a rewiring protocol which executes independently on each peer as in [HB07, LWSN03, SMZ03, Sch04, MSZ05]. The methods referred to above use the peer organisation, emerging from the execution of the rewiring protocol, to efficiently route queries to the clusters of peers that are more likely to answer the query. In our setting, a number of choices in designing a rewiring strategy are considered and all are implemented and evaluated (in terms of network load and retrieval efficiency) using real-world data and queries. Based on insight acquired by studying these alternatives, we propose optimisations to the rewiring protocol. The best rewiring strategy is designed based on experimental results and on the analysis of these results.

1.3 Contributions

The contributions of this thesis are both of a theoretical and a practical nature. At the theoretical level, this work proposes i*Cluster*, a decentralised architecture based upon ideas from both unstructured peer-to-peer systems and semantic overlay networks. The rewiring process is considered as a critical component of this architecture. We investigate the functional issues related to the design and the performance of the rewiring protocol, and explore the dependency of the retrieval performance on the rewiring process.

Complementary to i*Cluster* and rewiring, this work introduces clustering efficiency κ, a network clustering measure. Clustering efficiency is proposed in a attempt to describe and quantify the connectivity of the network in a formal way.

Next, we investigate the effectiveness of the rewiring under a realistic dynamic scenario where peers join, participate, leave and re-join the network. We consider that peers and data follow appropriate dynamic models that exist in the literature and study the way that network dynamics affect the system performance.

Finally, we revisit our architecture to support two-tier functionality under a single unifying framework. We introduce i*ClusterDL*, a digital library use case, as

an application scenario.

At the practical level, the experimental evaluation with real-word data and queries demonstrates that i*Cluster* achieves significant performance improvements (in terms of communication load and retrieval accuracy) over a state-of-the-art peer-to-peer clustering method. Compared to exhaustive search by flooding, i*Cluster* exchanged a small loss in retrieval accuracy for much less message flow.

In addition, we show how peer organisation is affected by the different design choices of the rewiring mechanism and how these choices affect the performance of the system overall (both in terms of communication overhead and retrieval effectiveness). Our experimental evaluation confirms the dependence between rewiring strategies and retrieval performance, and gives insights on the trade-offs involved in the selection of a rewiring strategy. Additionally, we investigate the effectiveness of the proposed clustering efficiency measure in estimating the connectivity of the underlying network when using different rewiring strategies.

The analysis of the experimental results on i*Cluster* under a realistic dynamic setting suggests the most promising rewiring protocol (i.e., the one that performs best in terms of retrieval performance and network load). Moreover, the experimental evaluation of i*ClusterDL* demonstrates that our architecture achieves high retrieval accuracy at a small communication cost and scales up well for large networks.

1.3.1 Intelligent Clustering

i*Cluster* combines ideas from a symmetric network structure [HCW05] for providing scalability, robustness and fault-tolerance, with ideas from semantic overlay networks [CGM03] for improving retrieval effectiveness and network traffic in semantic overlay networks. i*Cluster* architecture and the corresponding protocols are presented in [RP08b].

Similarly to unstructured P2P systems (e.g., [CHM+02, HCW05]), i*Cluster* is a completely decentralised architecture where no peer is more important than any other. Similarly to SON-based systems (e.g., [JM04, PGW+08]), i*Cluster* applies a periodic rewiring protocol to cluster peers with similar interests [VSI07, LLS04]

along with a fireworks-like technique [TXD03, NSC02] for routing queries within the network. In a typical SON system, peer interests may vary from document descriptions (e.g., document term vectors) [LC03], to topics in a hierarchy [KJ04], schema synopses [KTS07], or ontology concepts [Sch04]. Peers are typically specialised and identified in the network using only a single interest (e.g., the most important or representative of their document collection). In i*Cluster*, peers may be characterised by more than one interests reflecting the size and the content diversity of their document collection. By allowing multiple peer interests, i*Cluster* is the first system to overcome the specialisation assumption common in semantic overlay networks.

1.3.2 Peer Rewiring

In i*Cluster*, peer organisation is achieved through the rewiring strategy. In this thesis, we introduce the functional issues related to peer rewiring and investigate the effects of rewiring on the retrieval performance. The results of this study are presented in [RPT09].

We provide a better understanding of the functional issues related to the design and performance of the rewiring protocol applied by answering the following fundamental questions: (i) what is an appropriate forwarding strategy for rewiring messages within a semantic overlay network, (ii) what information should be used to update peer connections, (iii) should peer connections be bi-directional or just one-directional, (iv) how coherent and well-connected should a cluster of similar peers be? The different rewiring strategies emerging as a result of different combinations of the above are thoroughly discussed and evaluated using real-world data and queries. We eventually propose an optimal rewiring strategy based on the analysis of the experimental results.

1.3.3 Clustering Efficiency

Complementary to i*Cluster* and rewiring, this thesis introduces the so called henceforth *clustering efficiency* κ, a measure for representing the clustering quality of the network (i.e., how well the similar peers are clustered together). Clustering efficiency

measure is presented in [RP08a].

Clustering efficiency measure has the following advantages over two popular measures appearing in the literature: (i) gives information about the underlying network involving more than just the immediate neighbours of the peers and (ii) measures how well the network is structured from an information retrieval point of view. The experimental results demonstrate that clustering efficiency reflects retrieval effectiveness. Thus, the higher the values of the clustering efficiency measure are, the better the underlying network organisation is in terms of retrieval effectiveness.

1.3.4 Peer Churn

In real-world applications, a peer joins the network when a user starts the application. While being connected, the user contributes its resources to the network and searches for resources provided by other users. The peer leaves the system when the user exits the application. The independent arrival and departure of users creates the collective effect called *churn* [SR06]. These user-driven dynamics of peer participation are taken into account in the design and evaluation of the peer-to-peer system proposed in this thesis.

Based upon a realistic scenario where peers join, participate, leave and re-join the network, i*Cluster* is utilised to organise the network and retrieve data under such a dynamic setting. By adopting appropriate dynamic models from the related literature, we study the way peer churn affects the system performance and identify the rewiring protocol that performs well in this setting. In the light of the above, the analysis of the experimental results on i*Cluster* under such a dynamic setting suggests the optimal rewiring protocol (i.e., the one that performs best in terms of retrieval performance and network load). To the best of our knowledge, this is the first study that concerns the way peer rewiring is affected by and deals with the dynamic behaviour of the peers.

1.3.5 Digital Library Use Case

We introduce i*ClusterDL*, a digital library use case of the ideas and the techniques presented in this thesis. i*ClusterDL* is a novel P2P architecture supporting self-organising digital libraries that demonstrates (i) the feasibility of supporting rich query models without resorting to structured overlays and (ii) the benefits derived from a looser organisation of system components. i*ClusterDL* is build upon a two-tier version of the i*Cluster* architecture and is designed to support retrieval and filtering functionality under a single unifying framework. i*ClusterDL* architecture along with the corresponding protocols are presented in [RPTW08].

To the best of our knowledge, i*ClusterDL* is the first approach towards efficient organisation of digital libraries in semantic overlay networks that supports both retrieval and filtering functionality. The proposed architecture is automatic (requires no intervention and minimal administration), general (requires no previous knowledge of the DL contents and works for any type of data model or content), adaptive (adjusts to changes of DL contents), efficient (offers fast query processing) and accurate (achieves high recall).

1.4 Thesis Outline

This thesis is comprised of eight chapters, the first being the current introductory chapter. The remainder of this thesis is organised as follows. Chapter 2 gives an overview of the existing work in the areas of peer-to-peer networks, semantic overlay networks, network clustering measures, and information retrieval, information filtering and digital libraries in a P2P context. Chapter 3 introduces i*Cluster*, the architecture proposed in the thesis, along with its associated peer organisation and query forwarding protocols. i*Cluster* forms a testbed for studying the performance of alternative rewiring approaches, clustering quality, peer churn, information retrieval and filtering in digital libraries. Chapter 4 introduces clustering efficiency κ, a measure that quantifies the quality of network organisation. Chapter 5 focuses on rewiring; it presents design choices along with an extensive study of the functional issues related to peer rewiring. Chapter 6 extends the i*Cluster* architecture under

churn. While peers and data follow appropriate dynamic models, we study the system performance and identify an appropriate rewiring protocol. In Chapter 7, a digital library use case for searching and filtering in a distributed digital library environment is put forward. The proposed digital library architecture supports both retrieval and filtering functionality using the same infrastructure. This thesis is concluded in Chapter 8 by summarising the main results and pointing out some open questions and directions for future work in the area of peer-to-peer and distributed information systems in general.

Chapter 2

Related Research

In this chapter we provide an overview of previous research which is relevant to the topics of this thesis. Initially, we focus on architectural aspects and present a comprehensive survey of P2P networks, ranging from unstructured approaches to super-peers, semi-structured and structured overlays. Then, we present research related to data organisation and retrieval in semantic overlay networks followed by work focusing on network dynamics and concepts quantifying network clustering. Finally, we provide a brief survey of the related research in the context of information retrieval and information filtering for the digital library domain.

2.1 Peer-to-Peer Networks

In peer-to-peer (P2P) systems, a very large number of autonomous computing nodes (the peers) contribute their resources and rely on each other for data and services. P2P systems offer the potential for low-cost sharing of information while ensuring autonomy and privacy of the participating entities. The main idea behind P2P is that, instead of relying on central components, functionality is provided through decentralised overlay architectures. In overlay networks, peers typically connect to a small set of other peers. Queries are then propagated to the network searching for information qualifying query criteria by utilising existing overlay connections and following a predetermined query forwarding strategy. Popular systems

such as Napster[1], Gnutella[2], Freenet[3], Kazaa[4], Morpheus[5] and others have made popular this model of interaction.

Ideas from P2P computing has proven useful not only for information sharing in distributed environments but also, as a natural distributed alternative to Web 2.0 application domains such as decentralised social networking in the spirit of Flickr[6] or del.icio.us[7]. P2P networks can also be applied to other distributed applications such as Grid computation (e.g., SETI@Home[8] or DataSynapse[9]), collaboration networks (e.g., Groove[10]), IP telephony (e.g., Skype[11]) and even new ways to design Internet infrastructure that supports sophisticated patterns of communication and mobility.

P2P networks are typically distinguished according to their topology into *unstructured* networks, *structured* networks and *hierarchical* networks. In unstructured networks, all peers are equal and form an overlay network with no restrictions on topology and no centralised source of information. Gnutella is considered to be the prototype *symmetric* or *unstructured* P2P network. Since its original proposal, the inefficiencies of the basic Gnutella protocol have been studied and various proposals for more efficient search in unstructured P2P networks are suggested in the literature (see [TR03] for a recent comparison). On the other hand, structured networks have a regular topology, e.g. rings or hypercubes, and are devised as a remedy for the routing and object location inefficiencies of unstructured networks. Distributed hash tables (DHTs) (e.g., [RD01, RFH+01, SMLN+03]) are a prominent class of structured overlays that attempt to solve the object lookup problem by offering some form of distributed hash table functionality: assuming that data items can be identified using *unique numeric keys*, DHT nodes cooperate to store keys for each other. Semantic overlay networks (SONs) (e.g., [Sch04, ACMH03]) stand in-between un-

[1] http://www.napster.com/
[2] http://www.gnu.org/philosophy/gnutella.html
[3] http://freenet.sourceforge.net/
[4] http://www.kazaa.com/
[5] http://www.music-city.org
[6] http://www.flickr.com/
[7] http://www.del.icio.us.com/
[8] http://www.setiathome.ssl.berkeley.edu/
[9] http://www.datasynapse.com/
[10] http://www.groove.net/
[11] http://www.skype.com/

structured and structured P2P networks; they organise peers that are semantically, thematically or socially similar (i.e., peers sharing similar interests) into groups, but assume neither a specific network structure nor control over the location of the data. SONs are best suited for loose P2P architectures. SONs, while being highly flexible, guarantee high degree of peer autonomy, improve query performance and enhanced query flexibility due to their ability to provide mechanisms for approximate, range, or text queries. Finally, hierarchical networks (e.g., [ITKD04, LC05]) partition the peers into two sets: super-peers and clients. In a super-peer network, all super-peers are equal and have the same responsibilities. Each super-peer serves a fraction of the clients and keeps indices on the resources of those clients. Super-peers interact by following a specific protocol (e.g., a symmetric one like Gnutella, a DHT or a SON protocol). Clients, which are equal to each other, can run on user computers and resources (e.g., files in a file-sharing application) are kept at client nodes. Clients learn about resources by querying super-peers and download resources directly from other clients.

In the rest of this section, we discuss the first three systems that popularised the P2P paradigm (Napster, Gnutella and Freenet), introduce DHTs, SONs and super-peer networks.

2.1.1 Three Influential Peer-to-Peer Systems

The common goal of the three pioneer systems, Napster, Gnutella and Freenet, was to facilitate the discovery and sharing of files (e.g., images, audio and video) among a large set of *peers* (user computers) located at the "edge of the Internet" [Ora01, MKL+02]. The files to be shared are stored at the peers and, after being discovered by an interested party, they are downloaded using a protocol similar to Hypertext Transfer Protocol (HTTP). Beyond this basic goal, there are important differences among the three systems regarding the *metadata* kept at each network node, the *topology* of the P2P network, the *placement* of the shared files, the *routing algorithms* for queries and replies, the degree of *privacy* offered to its users, etc.

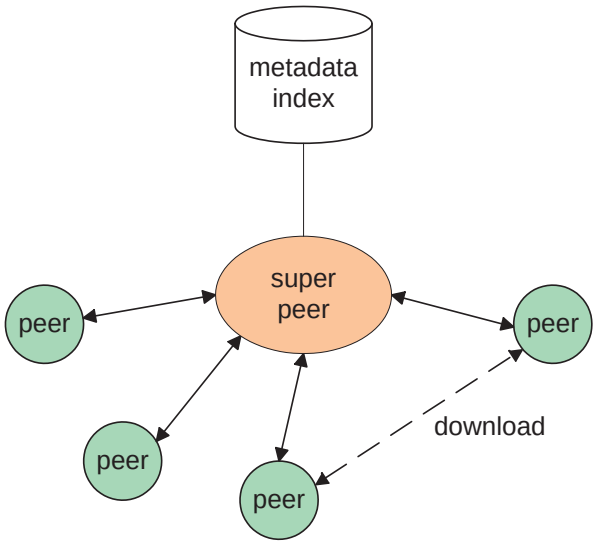


Figure 2.1: An example of the Napster P2P system

Napster

In Napster (shown in Figure 2.1), a large cluster of servers owned by the respective company maintains a *metadata index* that keeps track of active peers, descriptions of the files the peers are willing to share and certain quality of service parameters such as bandwidth and duration of the connection. Peers (clients!) connect to the system by connecting to *one* of these servers and publish a description of the files they want to share with other peers. Peer queries are sent to their selected server which returns a list of matching files, the address of the peer that owns each file and other important information pertinent to the peer (e.g., bandwidth as reported by the peer). Peers can then *directly* download selected files from the peer of their choice. Napster is a *hybrid* P2P system [YGM01] with some client-server features (register, publish, query) and a single P2P feature (download). Napster contains also a *structured* lookup mechanism: each peer has well-defined information about other peers in the system [BKK+03]. However, approaches such as Napster have inherent problems of scalability and resilience, so one has to use techniques such as indexing for scalability and replication for resilience [GM02].

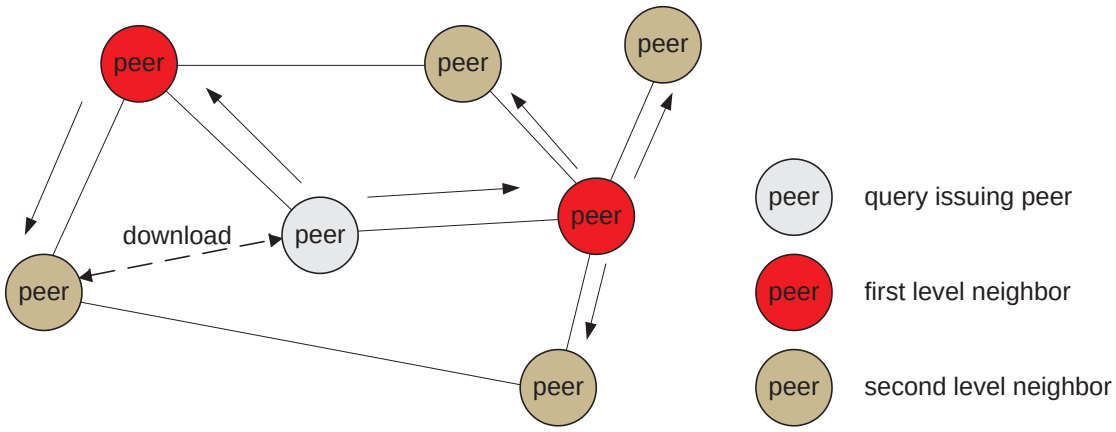


Figure 2.2: An example of the Gnutella P2P system

Gnutella

On the opposite end (in terms of decentralisation), Gnutella (shown in Figure 2.2) has no centralised servers and use symmetric lookup mechanisms [MKL+02]. Each node in a Gnutella network is a peer that can act as a client and as a server at the same time thus, no peer is more important than any other peer. Gnutella peers form an *overlay network* by setting up connections to peers of their choice. For connecting to the Gnutella network addresses can be initially found by the interested user by consulting web pages such as `gnutellahosts.com` or `router.limewire.com`. Gnutella offers primitives *ping* and *pong* for discovering parts of the network and facilitate its maintenance while peers enter and leave the system.

To discover a file, a Gnutella peer issues a query to its neighbours with whom it has open connections. The query is accompanied by a time-to-live (TTL) counter that specifies how many hops this query is allowed to travel in the Gnutella network. Each peer that receives this request processes it using its local file collection and returns Uniform Resource Locators (URLs) pointing to matching files to the source of the request. Then, this peer decrements the TTL counter of the request by one. If the value of the TTL counter is greater than 0, then this peer forwards the query to its neighbours. This process is repeated and eventually more pointers to matching files are returned to the source of the request. Thus, Gnutella uses the classical *flooding* technique [BG87] for routing queries. Replies reach the source peer by travelling along the reverse path followed by the query, since a peer processing

a request in a Gnutella network does not know the identity of the source peer issuing this request. However, privacy of information requesters' and information providers' identities are not really protected (e.g., Gnutella messages containing Internet Protocol (IP) addresses and URLs are returned to information requesters so that they can retrieve the files they desire).

Gnutella is considered to be prototypical symmetric or unstructured P2P network. Since its original proposal various studies for more efficient search in unstructured P2P networks are known to exist (see [TR03] for a recent comparison).

Freenet

Freenet is a network of peers connected to each other for the purpose of sharing information in the form of data files [CHM+02]. Like Gnutella, Freenet keeps a *completely decentralised architecture* which ensures scalability, robustness and fault-tolerance. At the same time, Freenet invests effort in ensuring the survivability of published information, the adaptability to usage patterns and the protection of the anonymity of information providers, consumers and holders. All these features are what distinguish Freenet from Napster and Gnutella (see Table 2.1).

To add a file, a user sends to the network a message (insert message) containing the file and an assigned location-independent *globally unique identifier* (GUID) which is computed using a SHA-1 [SHA95] secure hashing function. Each GUID consists of two parts: a *content-hash key*, which is obtained by hashing the contents of the file and it is used for low-level data storage, and a *signed-subspace key* intended for higher-level human use like traditional filenames. Insert messages follow the same procedure also pursued by a request message for that file, thus files are stored in the peers exactly where queries will go looking for them and routing tables are updated accordingly [CHM+02]. This procedure is described below.

To retrieve data, a user sends a request message to the network and the corresponding GUID of the file. Whenever a peer receives a request, it first checks its local resources. If the file is found, the peer returns it to the requester together with a tag identifying itself as the holder. If the file is not found, *one* of the neighbour

System	hybrid	symmetric	scalable	robust	adaptable	anonymous
Napster	✓					
Gnutella		✓	✓	✓		
Freenet		✓	✓	✓	✓	✓

Table 2.1: Comparison of the three influential P2P systems

peers contained in the routing table with the closest matching key is chosen and the request is forwarded to it. This is a basic difference with the Gnutella algorithm: Gnutella does not perform *heuristic* search and sends the query to *all* neighbour peers. When the data file is found, it is returned to the requester via the same path. Additionally, intermediate peers save an entry in their routing tables associating the requested key with the source peer. Depending on their distance from the data source, each peer might also cache a local copy of the file.

The anonymity of a file source is ensured by having intermediate peers occasionally altering the holder tags to point to themselves as data holders. This does not compromise discovery of the file later because the identity of the true data holder is kept at the peer's routing table and routing tables are never revealed. The anonymity of an initiator of a query is also ensured since a peer cannot know whether its neighbour peer is the one interested in the results of the query or is simply forwarding a message.

Each data request in Freenet is given a TTL count (like in Gnutella), which is decremented at each peer the request goes through. To prevent requests from going into an infinite loop, Freenet assigns a unique identifier to each request so that a peer will never forward a request that received for a second time.

2.1.2 Distributed Hash Tables

The success of P2P protocols and applications, such as Napster and Gnutella, motivated researchers from the distributed systems, networking and database communities to look more closely into the core mechanisms of these systems and investigate how these could be supported in a principled way. This quest gave rise to a new wave of distributed protocols, collectively called Distributed Hast Tables (DHTs), that aim primarily at the development of P2P applications [SMLN+03, RFH+01,

RD01, ACMD+03, ZHS+04, MNR02, AEABH03]. DHTs are *structured* P2P systems which attempt to solve the following *look-up problem*: Given a data item X stored at some distributed dynamic set of peers, find X.

The core idea in all DHTs is to solve this look-up problem by offering some form of distributed hash table functionality: assuming that data items can be identified using *unique numeric keys*, DHT nodes cooperate to store keys for each other (data items can be actual data or pointers). Implementations of DHTs offer a very simple interface consisting of two operations:

- `put(ID, item)`. This operation inserts item with key `ID` and value `item` in the DHT.
- `get(ID)`. This operation returns a pointer to the DHT node responsible for key `ID`.

Although the DHTs available in the literature differ in their technical details, all of them address the following issues [BKK+03]:

1. **Mapping keys to nodes in a load-balanced way.** Keys and nodes are identified using a binary number. Each key is stored at one or more nodes with identifiers "close" to the key in the identifier space.
2. **Forwarding a lookup for a key to an appropriate node.** Any node that receives a query for a key identifier k returns the data item X associated with k if it owns k, otherwise it forwards k to a node with identifier "closer" to k using only local information.
3. **Building routing tables adaptively.** DHTs adapts to node joins, leaves and failures, and updates the information maintained in the routing tables with little effort.

The way different approaches address these issues provide us with a good high-level categorisation of existing DHTs [BKK+03, AGH04]. Aberer et al. [AAG+05] provide a reference model to unify different P2P approaches and also, distinguish

six basic concepts common on all structured overlays such as routing strategy, maintenance and identifier space management. Blanas and Samoladas [BS07] introduce a novel performance analysis for search protocols on structured overlays.

2.1.3 Semantic Overlay Networks

Formally, a P2P Semantic Overlay Network (SON) is a directed graph with vertices corresponding to information sources (peers) and edges corresponding to links between information sources. Queries are propagated within the network following the edges and according to some query routing strategy used. Peers connected with each other are called *neighbours*. Each peer maintains a routing index and query messages are propagated within the network using information stored in this index.

SONs (e.g., [CGM03, TXD03, Sch04]) can be regarded as a softer alternative for decentralised indexing compared to rigid semantics imposed by hashing. SONs capture object (data or peer) relationships and let peers connect with each other based on their likelihood to contain related information, but assume neither a specific network structure nor control over the location of the data. In SONs, the overlay network is partitioned by clustering the available data sources according to their content. The problem of finding the most relevant resources is then reduced to locating the cluster of peers that is the most similar to the query. SONs, contrary to DHTs, provide semantic (and social) information about peers and their content by capturing object relationships and are able to support rich data models and expressive query languages (e.g., VSM, LSI) without resorting to architectural [TIK05] or algorithmic [BMT+05, AT05] modifications and optimisations.

Semantic overlay networks are typically based on ideas from *small-world networks* [CGM02]. In a small world network, peers are not necessarily neighbours of all other peers with relevant information, but can be reached from every other peer in a small number of routing hops [Wat99]. A small world network resembles the social phenomenon, referred to as "small world effect", implying that every member of the world can be reached by any other member through a short chain of social acquaintances [Wat99]. Merugu et al. [MSZ05] show that constructing small world-

like overlay topologies improves retrieval performance.

Peer organisation in a SON is achieved through a *rewiring protocol* that is (periodically) executed independently by each peer. The purpose of this protocol is to establish connections among *similar peers*. This is achieved by creating new connections to similar peers and by discarding connections that are outdated or pointing to dissimilar peers. The goal of a rewiring protocol is to create *clusters* of peers with similar interests. Queries can then be resolved by routing the query towards clusters based on their likelihood to match the query. Once reaching a cluster, the peer receiving the query is responsible for forwarding it to other peers within the same cluster. Concepts such as peer rewiring and peer clustering are fundamental in designing a SON.

2.1.4 Super-Peer Networks

In a super-peer system, all super-peers are equal and have the same responsibilities. Each super-peer serves a fraction of the clients and keeps indices on the resources of these clients. Super-peers interact by following a specific protocol (e.g., a symmetric one like Gnutella, a DHT or a SON protocol). Clients can run on user computers and resources (e.g., files in a file-sharing application) are kept at client peers. Clients are equal to each other since the software running at each client peer is equivalent in functionality. Clients learn about resources by querying super-peers and download resources directly from other clients.

Issues involved in the design and implementation of super-peer systems have recently been studied thoroughly in the literature. Yang and Garcia-Molina [YGM03] presented one of the first works to study the characteristics of super-peer networks and present the performance trade-offs associated with them. Super-peers are generally either volunteers that want to offer their extra resources to the community or are chosen and "promoted" by some mechanism (e.g., a gossiping mechanism as in [JMB06]). The nice properties of super-peer networks made them an attractive architectural solution adopted by a number of systems and research proposals such as Edutella [NWQ+02], P2P-DIET [ITKD04, CIKN04] and also [LC05, RC04, LC04].

2.2 Peer Organisation in SONs

Properties such as the autonomy, the flexibility and the efficiency provided by semantic overlay networks has propelled research towards this direction. Moreover, the management of large volumes of data in P2P networks has generated additional interest in methods for effective network organisation based on peer contents and consequently, in methods supporting information retrieval over semantic overlay networks (e.g., [KP08, KJ04]). In this section, we survey research related to the SON domain that forms the basis of our work. We present studies that (i) focus on architectural issues (i.e., overlay topologies along with the corresponding protocols), (ii) exploit certain modelling aspects of the P2P network (e.g., models describing peer similarity), (iii) deal with network dynamics and (iv) propose measures quantifying network clustering.

2.2.1 Architectures

Recent work include the study by Li et al. [LLS04], which introduces the notion of Semantic Small-World (SSW) for organising the overlay network by exploiting the semantics of data objects (e.g., documents) stored locally on peers. This work presents (i) a mechanism that constructs a small-world overlay network, which is scalable to large network sizes and large numbers of data objects while requiring low maintenance overheads to adapt to dynamic membership and content changes and, (ii) a dimension reduction technique (called adaptive space linearisation (ASL)) for constructing a one-dimensional SSW to address the challenges raised by high dimensionality of semantic space. Although this works aims at exploiting the properties of unstructured P2P networks, it assumes that each peer node is responsible for management of its data objects and/or the location information of data objects stored at other peers (corresponding to a semantic subspace) considering thus, some type of control imposed also by DHTs.

Hidayanto et al. [HB07] propose self-organising peers into communities based on their content (called “expert network”), the issued queries (called “interest groups”), or both the content and the queries (called “hybrid groups”). This work proposes a

SON architecture that aims at improve retrieval performance. However, the research included in this paper is still preliminary as it only discusses and evaluates an instance of this general idea.

In a similar spirit, Loser et al. [LT05] propose a three-layer organisation of peers and suggest combining information from all layers for routing queries. According to this work, peers store Resource Description Framework Schema (RDF(S)) statements. A statement can describe either data (e.g., `uri1 isAbout Semantic Web`) or conceptual information (e.g., `University subClassOf Organisation`). The authors consider either a general query language that allows to query an arbitrary RDF(S) model or queries for documents that are about a certain topic. The network is then organised based on peers' interactions and observations: each peer monitors which other peers frequently respond successfully to its requests for information, which peers ask similar questions, which peer provide many documents or which peer have asked many questions to a broad range of topics in the past. Then, a peer creates links to other peers (called shortcuts) in three different layers (called "recommender", "provider" and "bootstrapping") according to its observations. To route a query, a peer combines information from all layers. This work considers data items and queries organised in a topic hierarchy and is meant as solution to knowledge management application domain.

In [LWSN03], Loser et al. introduce the concept of semantic overlay clusters for super-peer networks. This work aims at clustering peers that store complex heterogeneous schemas by using a super-peer architecture. Each information provider peer (client) is identified in the network with a metadata Resource Description Framework (RDF) model. Each super-peer represents a cluster of domain specific information and is related to exactly one clustering policy. Every clustering policy consists of rules expressing which information provider peers are allowed to join the cluster and which services are denied to enter the cluster. For joining the network an information provider peer chooses an arbitrary super-peer to which it forwards its model. The super-peer first matches the model against its clustering policy and allows or denies the join of the peer to its cluster, then it broadcasts the model to all other super-peers in the super-peer network. Finally, a set of clients together with

their super-peer forms a cluster. This work aims at clustering peers according to their schemas and bridging the heterogeneity between different metadata schemas by enabling mappings within a particular domain through super-peers.

Along the same lines, Klampanos et al. [KJ04] propose an hierarchical architecture for IR-based clustering of peers in semi-collaborating overlay networks. Information provider peers expose their local document collections to the network and they are clustered according to their information content. Client peers issue queries to the network, view and retrieve documents. Representative peers (hubs) form the message routing layer of the network. These are the only peers that are allowed to interconnect with each other as well as with other kinds of peers. Hubs maintain the descriptors of all peer clusters in the network and are responsible for handling meta-information, routing query requests and delivering of results. The hierarchical organisation proposed in this paper may create a bottleneck problem for peers up the hierarchy since, all communication has to go through hubs.

Emphasising on protocols, Voulgaris et al. [VSI07] propose an epidemic protocol[12] that clusters peers with similar content. Peer clustering is done in a completely implicit way, that is without requiring the user to specify any preferences or to characterise the content of files being shared. The idea is to let peer self-organise based only on local interactions. The authors anticipate that the probability of a neighbour satisfying a peer's request is proportional to the semantic proximity between the peer and its neighbour. Consequently, the number of requests satisfied by neighbouring (instead of random) peers is expected to be maximised when a peer selects the semantically closest peers out of the whole network in its semantic view (routing index). Although the proposed mechanism wants to exploit semantic similarity among peers, it also relies on randomly selected peers for the fulfillment of more than 65% of the requests.

Schmitz [Sch04] assumes that peers share concepts from a common ontology and proposes strategies for organising peers into communities with similar concepts. Each peer periodically initiates a rewiring procedure and forwards its expertise to the network. In order to build a clustered network using only locally available knowledge,

[12]Epidemic protocols exploit randomness to disseminate information across peers.

the peers use (for the forwarding of their expertise) strategies based on walks on the P2P network (i.e., gradient and/or random walk). Next, the author proposes and experiments with a bunch of query routing strategies (and their combinations) trying to maximise retrieval efficiency. The strategies provided in this paper rely only on local knowledge and interactions, are scalable and resilient to node failures, and improve the performance of query operations. However, these strategies work well if all peers share the same ontology which may not be realistic. Labelling and maintaining more general peer communities in a decentralised manner could be an interesting extension. Additionally, the framework used for clustering and routing leaves a lot of room for tuning parameters and combining strategies.

Another track of work includes [JM04], where Jelasity and Montresor observe that aggregation (i.e., global network information as average load) may reveal useful information for peers while self-organising in large-scale networks. The authors present an epidemic protocol capable of exploiting this information. The proposed approach acts proactively and does not create bottlenecks as the approximation of the aggregates is continuously present at all peers.

2.2.2 Models

Focusing on modelling issues, Schmitz and Loser [SL06] present a model, along with evaluation metrics and data sets for semantic overlay networks. Their model constitutes a generic description of the fundamental issues related to a SON, which are the peer's contents, its expertise, the similarity function, the routing index, the rewiring strategy and the query routing strategy. The focus of this work is on providing a common modeling standard for the behaviour of P2P networks and in particular for the behaviour of semantic P2P networks.

In [SMZ03], Sripanidkulchai et al. introduce the notion of peer clustering based on similar interests rather than similar content. According to this work, peers are organised on the top of the existing Gnutella network to improve retrieval performance. The authors identify that if a peer has a particular piece of content that one is interested in, then this peer is likely to have other pieces of content that one is

also interested in. These peers are considered to have the so-called "interest-based locality". The authors propose the creation of interest-based shortcuts among peers by exploiting the interest-based locality. Then, peers use these shortcuts to locate content. When shortcuts fail, peers resort to the underlying Gnutella overlay for content location. In this way, the proposed shortcuts provide a loose structure on top of Gnutella's unstructured overlay.

Aberer at al. [ACMH03] introduce a decentralised process that, relying on pairwise local interactions, incrementally develops global agreement and obtains semantic interoperability among data sources. This work considers scenarios of large collections of data, as for example biological or genomic databases where federating loosely-coupled databases exist. In such an application scenario, a database is considered as a peer that offers data, and is organised according to some schema expressed by a data model (e.g., relational, XML, or RDF). The authors observe that semantic interoperability is always based on some form of agreement and propose achieving implicit agreements by assuming the existence of local agreements[13] provided as mappings between different schemas (i.e., agreements established in a P2P manner). This model is orthogonal to research on P2P architectures and could be introduced into some of them.

In [PLM+08], Penzo et al. propose a strategy for incremental clustering of semantically related peers. Each peer is represented by a set of concepts describing its main topics of interest, which are extracted from the DBLP XML schema. According to this paper, SONs are clusters of connected peers that share similar concepts and evolve incrementally as new peers join the network. When a peer joins the system, it performs a coarse-grained selection of neighbours by accessing the Access Point Structure (APS), that is a structure which maintains information about the SONs available in the network. An accurate choice of its neighbours within the SON is then supported based either on a semantic similarity threshold (range-based selection), or on a maximum number of neighbours (k-NN selection). This research relies on a peer data management system as a reference scenario and proposing clustering together peers that share similar schemas in an effort to eliminate the inefficiency

[13]The local agreements are established in a manual or semiautomatic way.

imposed by the traversal of semantic mappings.

Along the same lines, Kantere et al. [KTSR09] use the SON paradigm to solve the query degradation problem in P2P databases. This work introduces GrouPeer, a decentralised approach that assumes the absence of global schema information. The key characteristic of GrouPeer is that peers can independently choose which rewritten version of a query to answer. Then, query initiators evaluate the received answers and send feedback to the corresponding content providers. By this process, remote peers on query propagation paths are discovered, semantically similar peers are clustered together and the overlay network is restructured. GrouPeer is shown to cluster similar peers together (by contacting only a small number of the nodes in the overlay), and by this produce accurate answers.

Parreira et al. [PMW07] introduce the notion of "peer-to-peer dating" that allows peers to decide which connections to create and which to avoid based on various usefulness estimators. Although this work makes use of the P2P paradigm, it concerns web linkage. According to this paper, each peer (i.e., web site) has two types of links to other peers: random links and semantic links. The random links are needed to keep the whole network together and are dictated by the underlying network protocol. The semantic links are established by letting peers meeting other peers that they still do not know (like "blind dates"). According to its criteria, if a peer likes another peer then it inserts this peer into its friend list. These criteria can be various measures (and their combination) like past behaviour, level of trust, collection similarity, overlap between the collections and authority scores. The friends are then annotated with statistics to form a semantic routing table (SRT) in which the peers are ordered according to their usefulness. The information stored in SRTs is then used during query routing in search engines. The authors claim that this approach can be also used in existing P2P systems since there is no need to change the existing infrastructure.

Koloniari et al. [KP08] model peer clustering as a game where peers try to maximise the recall for their local query workload by joining the appropriate clusters. This paper presents an approach for handling semantic overlay networks reliant to decisions that each peer makes based on local criteria. The problem of peer clustering

is modeled as a game in which the peers are the players that decide whether they should move to another cluster so as to minimise an individual cost function. This cost function depends on the membership cost entailed in belonging to a cluster and the cost of evaluating the peer query workload at remote clusters.

Other works in the area include the embedding of metric spaces in the SON paradigm, as in [LP07, SBDZ08], in an effort to broaden the applicability of SONs.

2.2.3 Peer Churn

Most of the above research proposals, while exploiting certain architectural [KJ04] or modelling [ACMH03, SL06] aspects of peer organisation, assume (for their experimental evaluation) an ideal scenario where peers never leave or join the network. However, studies assuming network dynamics are also known to exist [SR06, LYRL07, YLWL06]. A peer joins the network when a user starts the application. While being connected, the user can contribute resources to the network and search for resources provided by other users. The peer leaves the system when a user exits the application. Stutzbach and Rejaie [SR06] define such a join-participate-leave cycle as a *session*. The independent arrival and departure of peers creates the collective effect called *churn*. The dynamics of user-driven participation is a critical issue, since churn affects the overlay structure [SRS05], the resiliency of the overlay [LYRL07, YLWL06], the selection of key design parameters [LSK+05] and the content availability which in turn, affects retrieval effectiveness.

Node-Based Models

Gummadi et al. [GDS+03] present and analyse a 200-day trace of Kazaa[14] peer-to-peer file sharing traffic collected by passively monitoring a router at the University of Washington. This measurement study aims at characterising the population of peers that participate in P2P networks with respect to their bandwidth, latency and availability. The authors suggest using this knowledge to evaluate the newly established P2P systems.

[14] `http://www.kazaa.com/`

Stutzbach and Rejaie [SR06] suggest that passive monitoring techniques are likely to underestimate session lengths, as some peers may not generate traffic through the observation point. Modelling churn though, requires fine-grained and unbiased information about the arrival and the departure of peers in a network. This task is rather challenging due to the large size and highly dynamic nature of P2P systems. Stutzbach and Rejaie [SR06] identify the key challenges in characterising churn, determine common pitfalls in measuring churn (such as biased peer selection) and develop techniques to address these difficulties.

Leonard et al. [LYRL07] present a realistic model for peer lifetimes in P2P networks and investigate the resilience of random graphs to lifetime-based peer failure. The authors also derive a model exploiting the probability that a P2P system partitions under churn. Yao et al. [YLWL06] introduce a generic model that explicitly captures the heterogenous behaviour of peers. This work regards each peer as an alternating renewal process and considers that on-line/off-line durations are independent and unique for each peer. Mitra et al. [MGGP08] model peer churn using degree-independent and degree-dependent node failure, and develop a framework to examine and analyse the stability of generalised P2P networks against churn. The authors use a super-peer network as a case study for their stability analysis.

System-Based Models

The work presented in [LNBK02, MRT+05, KHG08] focus on system-based models. System-based churn models view churn in the distributed system as a holistic phenomenon, without modeling individual process behaviour. In general, holistic phenomenons are related to or concerned with wholes or with complete systems rather than with the analysis of, treatment of, or dissection into parts.

Liben-Nowell et al. [LNBK02] present a theoretical analysis of P2P networks by considering concurrent joins and unexpected departures. The authors focus on Chord[15] [SMLN+03] and study the process by which Chord maintains its distributed state as nodes join and leave the system. Mostefaoui et al. [MRT+05] propose a

[15]Chord is a P2P system that implements a distributed hash table abstraction.

model applicable to dynamic systems which is defined by a threshold value (that captures the liveness part of the system) and basic communication abstractions (such as query-response and persistent reliable broadcast). To show the relevance of the model, the authors adapt a protocol designed for the static model and point out that this protocol operates within the proposed dynamic model. Ko et al. [KHG08] present two new system-based churn models, called Train and Crowd, that are respectively a tractable and a realistic model. The authors prove how analysis using the tractable model can be generalised to the more realistic model for an important class of stability properties, derive bounds for these properties when using several protocols and use real churn traces to verify their models.

Evaluation Studies

Under a different perspective, Atalla et al. [AMA+08] study the potential impact of peer churn on P2P applications concerned to Web search. This work aims at providing a realistic analysis of P2P applications by quantifying the impact of both the churn and the malicious behaviour of peers on the effectiveness of P2P Web searching. Their findings reveal that both factors impose serious constraints in P2P searching and the authors stress the need for application-specific reputation and incentive mechanisms[16].

Another track of work includes the evaluation of protocols well-suited for symmetric P2P overlay networks. These protocols are simulated under an event-based setting that includes concurrent and interleaved join and leave operations. Ganesh et al. [GKM03] present the design, the theoretical analysis and the evaluation of the SCAMP self-organising protocol. In [VSI07], Voulgaris et al. present Cyclon, a proactive protocol that clusters peers with similar content. The authors investigate the behaviour of their protocol as nodes join and leave the system. Baldoni et al. [BBR+06] compare SCAMP and Cyclon in an event-based simulation setting that includes join and leave operations as well as variable message transfer delay. The

[16]Incentive mechanisms motivate each peer to behave in a system efficient way. Researchers have proposed various incentive mechanisms, based on payment, punishment, or service differentiation, to encourage cooperative behaviour among peers [SKT09, PS05, FLSC04, GLBM01].

authors point out that after a very small number of join and leave operations, the overlay is partitioned permanently (for both protocols examined) in (two or more) separate sub-networks.

2.2.4 Clustering Measures

Another track of work in the area of semantic overlay networks is related to measures *quantifying* the quality of network clustering. Over the past few years, a wide range of concepts have been proposed and investigated. Watts and Strogatz [WS98] introduce the clustering coefficient measure to determine whether a graph represent a small-world network. The clustering coefficient of a vertex in a graph quantifies how close the vertex and its neighbours are from being a clique (complete graph).

Hansen et al. [HAH07], working on protein interactions, present a generalised version of the classical clustering coefficient measure. This work aims towards a measure that can be applied to all types of networks (e.g., networks with directed links) and can provide information about the underlying structure within the networks. In [FHJS02], Fronczak et al. assert that the standard clustering coefficient measure does not provide any useful insights of complex network structure and dynamics. This work extends the standard clustering coefficient by introducing higher order clustering coefficients that describe interrelations between vertices belonging to the nearest neighbourhood of a certain vertex in the complex network. Other works in the area, as for example [BGW08, RMJ07], discuss indices that measure the quality of a graph clustering mechanism. The works referred to above, mainly aim at clustering per se. In almost all studies, the authors consider a graph or a network and propose clustering techniques that maximise some clustering measure.

In [FC06], Forstner and Charaf propose a protocol for clustering P2P networks and evaluate this protocol using a modified clustering coefficient measure. They idea of the proposed protocol is to arrange peers in a topology so as to eliminate counterproductive links and minimise network traffic, while performing successful queries. A query is considered successful when a document matching the query is retrieved. This work applies to mobile environments, so network traffic is of

major importance. Contrary, in this thesis information retrieval is treated as the important issue; we identify a clustering measure that quantifies the underlying network structure, aiming at reflecting retrieval effectiveness.

2.3 IR and IF in the Digital Library Domain

Two-tier architectures is a natural solution for addressing architectural issues in digital library (DL) domains. Lu and Callan [LC03] study the problem of content-based retrieval in distributed DLs focusing on resource selection and document retrieval. They propose a two-level hierarchical P2P network where DLs are clients that connect to super-peers which form an unstructured P2P network in the second level of the hierarchy. A recent contribution by the same authors [LC05] suggests organising super-peers into neighborhoods to devise a method for super-peer selection and ranking. OverCite [SCL+05] is proposed as a distributed alternative for the scientific DL CiteSeer[17] by utilising a DHT infrastructure to harness distributed resources (storage, computational power, etc.).

To the best of our knowledge, P2P-DIET [IKT04] and LibraRing [TIK05] are the first approaches that try to support both information retrieval (IR) and information filtering (IF) functionality within a single unifying framework. P2P-DIET utilises an expressive query language based on IR concepts and is implemented as an unstructured P2P network with routing techniques based on shortest paths and minimum weight spanning trees. LibraRing [TIK05] provides protocols to support both IR and IF functionality in DLs using distributed hash tables (DHTs). The DHT is used to make sure that queries meet the matching documents (in the IR scenario) or that published documents meet the indexed continuous queries (in the IF scenario). Contrary to LibraRing, MinervaDL [ZTW07] suggests using Chord DHT to disseminate and store *statistics* about the document providers rather than the documents themselves.

DESENT [DNV06] is the first work applying the concept of semantic overlay networks in a DL domain. In DESENT [DNV06], SONs are organised as a hierarchy

[17] http://citeseer.ist.psu.edu/

of clusters. Each cluster is represented by the so called "cluster gateway" (a single peer within the cluster). In turn, groups of clusters form super-clusters with their own gateways. Each cluster gateway maintains information about all other clusters (and super-cluster) representatives. All communication within a semantic overlay network is propagated through gateways thus, creating communication bottlenecks at these peers.

The work presented in this thesis is the first comprehensive architecture that exploits SONs to provide a framework for self-organised and self-managed DLs that can offer a rich quiver of tools to end-users. Contrary to DESENT [DNV06], our architecture maintains a flat structure of clusters with no representatives and the communication can be routed through any peer within a cluster thus, avoiding the bottleneck problem. Moreover, our architecture supports IF functionality (in addition to IR) at no extra communication cost.

2.4 Conclusions

This chapter surveys related literature on P2P systems, starting with three pioneer systems that popularised the P2P paradigm and concluding with an extensive survey of the state of the art in semantic overlay networks.

Chapter 3

An Architecture for P2P Information Sharing

In this chapter we present i*Cluster* (from the words *intelligent* and *clusters*), an architecture for supporting full-fledged information retrieval in large-scale P2P networks. Section 3.1 provides an overview of i*Cluster* architecture. In Section 3.2 we discuss the proposed architecture, while Section 3.3 presents the protocols that orchestrate the peer interactions. Section 3.4 summarises our approach.

3.1 Introduction

i*Cluster* combines the symmetric network structure (also used by known systems such as Freenet and Gnutella) ensuring scalability, robustness and fault-tolerance, together with ideas from semantic overlay networks for providing enhanced retrieval support and good performance in terms of increased recall and low network traffic. i*Cluster* architecture along with the protocols for peer organisation and query routing are presented in [RP08b].

Similarly to unstructured P2P systems, i*Cluster* is a completely decentralised architecture where no peer is more important than any other. Similarly to SON-based systems, i*Cluster* applies a periodic rewiring protocol (e.g., [VSI07, LLS04]) for clustering peers with similar interests and a fireworks-like technique (e.g., [TXD03, NSC02]) for routing queries within the network. In a typical SON system, peer

interests may vary from descriptions of document collections [LC03], to topics in a hierarchy [KJ04], schema synopses [KTS07], or ontology concepts [Sch04]. In i*Cluster* the interests of a peer are identified by using its local document collection. Any peer has a tunable and dynamic number of interests depending on its capabilities, collection size and content diversity. To the best of our knowledge, i*Cluster* is the first system to overcome the specialisation assumption[1] common in SONs [NWQ+02].

3.2 i*Cluster* Architecture

We consider a network where a large number of peers connect to each other and share information in the form of data files. The peers are responsible for serving both users *searching* for information and users *contributing* resources to the network. In this architecture, no peer is more important than any other. Similarly to the prototypical symmetric P2P networks, i*Cluster* keeps a completely decentralised architecture ensuring scalability, robustness and fault-tolerance. Moreover, i*Cluster* peers run (independently to each other) a rewiring protocol and form clusters based on their likelihood to contain similar content. In this way, peers form an overlay network by setting up connections to peers of their choice. The role of the overlay network is two-fold: it acts as the glue between information producers and information consumers through a distributed self-organising repository that can be queried efficiently and serves as a fault-tolerant and scalable routing infrastructure. A high-level view of the i*Cluster* architecture is shown in Figure 3.1.

Each i*Cluster* peer is characterised by its information content (i.e., its document collection), which may be either automatically (by text analysis) or manually assigned to each document (e.g., tags or index terms). To identify its interests, a peer categorises its documents by using an external reference system (i.e., an ontology as in [Sch04] or a taxonomy such as the ACM categorisation system) or by clustering [HK01]. Thereupon, a peer may be assigned more than one interests. Interests are created and deleted dynamically to reflect a peer's variety in the documents it contributes to the network. Each peer maintains a *routing index* (RI) holding

[1]Each peer has only one interest.

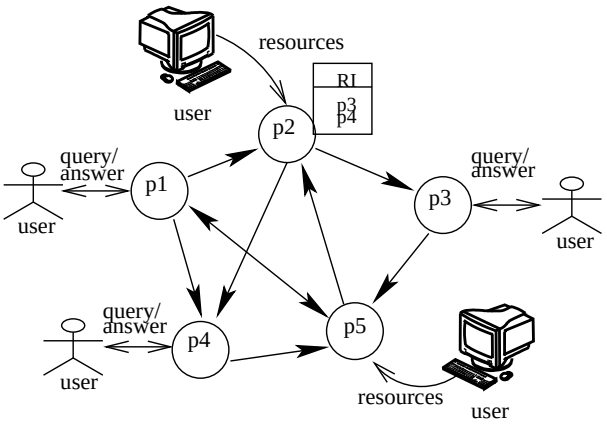


Figure 3.1: A high-level view of the i*Cluster* architecture

information for short- and long-range links to other peers:

short-range links correspond to *intra-cluster* information (i.e., links to peers with similar interests)

long-range links correspond to *inter-cluster* information (i.e., links to peers having different interests and thus belonging to different clusters)

Entries in the routing index contain the IP addresses of the peers the links point to and the corresponding interests of these peers.

A peer may participate in more than one network clusters, which depict its interests. To account for this information, each peer maintains a separate routing index for each one of its interests. Thus, the number of routing indexes maintained by a peer corresponds to the number of peer's interests and is in turn, related to the clusters in which this peer belongs. For example, if a peer is interested in topics A and B, then it maintains a routing index with short-range links to other peers interested in topic A and another routing index with short-range links to peers interested in topic B. Long-range links connecting clusters A and B with peers having different interests are maintained as well. Typically, the interests of a peer are expected to restrict to a relatively small number and the peer will not maintain a large number of links to other peers. Additionally, peers may merge or split their interests by merging or splitting the corresponding routing indexes to keep control on the number of outgoing links that have to be maintained.

Our architecture is general enough to cover fundamental design aspects of existing semantic overlays: as typically assumed in the literature peers are categorised by their content [MSZ05, KJ04, Sch04, LLS04, LC03, SMZ03] and links to other peers involve not only similar peers, but also dissimilar ones to support the connectivity between different clusters as in [MSZ05, Sch04, SMZ03, TXKN03]. The rewiring procedure is executed locally by each peer and aims at clustering peers with similar content, while queries are forwarded to peer clusters that are similar to the issued query. The corresponding protocols for peer rewiring and query routing are presented in the next section.

3.3 i*Cluster* Protocols

The main idea behind semantic overlay networks is to let peers self-organise into clusters with semantically similar content. Then, query execution is performed by identifying the cluster of peers with similar content and by addressing a peer within this cluster. Then, this peer is responsible for forwarding the query to more peers within its cluster. In this section, we present the basic protocols that specify the way peers join the overlay network, connect to and disconnect from the network, self-organise into clusters, and the way queries are processed.

3.3.1 Joining Protocol

When a peer p_i connects to the i*Cluster* network for the first time, it has to follow the join protocol. Initially, p_i categorises its documents into one or more categories. Consequently, p_i may have more than one interests $I_{i\kappa}$, with κ denoting the number of p_i's interests, stored in its *interest list* $I(p_i)$. A peer interest is represented by the corresponding index terms (if an external reference system has been used) or by the centroid vector (if documents have been categorised by clustering).

For each distinct interest I_{ik} peer p_i maintains a separate routing index RI_{ik}, which contains short-range and long-range links. Entries in the routing index are of the form $(ip(p_j), I_{jk})$, where $ip(p_j)$ is the IP address of peer p_j and I_{jk} is the k-th interest of p_j. The number of routing indexes maintained by a peer equals the

number of its interests. Peers may merge or split their routing indexes by merging or splitting their interests reliant to changes in their content.

A routing index is *initialised* (when a peer is joining the network, or upon the bootstrapping) as follows: peer p_i collects in RI_{ik} the IP addresses of $s+l$ randomly selected peers using out-of-band means, where s is the number of short-range links and l is the number of long-range links. These links will be refined according to the interest I_{ik} of p_i using the rewiring protocol described in the next section.

In the following, for simplicity of the presentation, we assume that each peer p_i has only one interest I_i. The same discussion applies for multiple interests, since peers maintain one routing index per interest and the rewiring strategy is applied independently for each peer interest.

3.3.2 Connecting/Disconnecting Protocol

When a peer p_i connects to the network, it uses its interest list $I(p_i)$ and the corresponding routing indexes p_i used before logging-off. In this way, peers do not have the overhead to categorise their documents and recompute their interests every time they connect to the network. Naturally, peers may recompute their interests reliant to changes in their content. Additionally, by using the information stored in the routing indexes before logging off, peers do not start from a random position in the network. While being connected to the network, p_i refines if needed the information stored in its routing index using the rewiring protocol described in the next section.

3.3.3 Rewiring Protocol

Peer organisation proceeds by establishing new connections to similar peers and by discarding old ones. Each peer p_i periodically (e.g., when joining the network or when its interests have changed) initiates a rewiring procedure (independently for each interest) by computing the intra-cluster similarity (or *neighborhood similarity*)

$$NS_i = \frac{1}{s} \cdot \sum_{\forall p_j \in RI_i} sim(I_i, I_j), \tag{3.1}$$

where s is the number of short-range links of p_i according to interest I_i, p_j is a peer contained in the RI_i that is on-line, I_j is the interest of p_j and $sim()$ can be any appropriate similarity function between peer interests (e.g., the cosine similarity between the term vector representations [Sal89]). The neighborhood similarity NS_i is used here as a measure of *cluster cohesion*. If NS_i is greater than a threshold θ, then p_i does not need to take any further action since it is surrounded by peers with similar interests. Otherwise, p_i issues a FindPeers$(ip(p_i), I_i, L, \tau_R)$ message, where L is a list and τ_R is the time-to-live (TTL) of the message. List L is initially empty and will be used to store tuples of the form $\langle ip(p_j), I_j \rangle$, containing the IP address and interest of peers discovered while the message traverses the network. System parameters θ and τ_R need to be known upon bootstrapping.

A peer p_j receiving the FindPeers() message appends its IP address $ip(p_j)$ and its interest I_j to L (or the interest most similar to I_i if p_j has multiple interests), reduces τ_R by one and forwards the message to the m neighbouring peers ($m \leq s$) with interests most similar to I_i. This message forwarding technique is referred to in the literature as *gradient walk* (GW) [SDCM06, Sch04]. When $\tau_R = 0$, the FindPeers() message is sent back to the message initiator p_i. Figure 3.2 illustrates the above rewiring procedure in algorithmic steps.

When the message initiator p_i receives the FindPeers() message back, it utilises the information contained in L to update its routing index RI_i by replacing old short-range links corresponding to off-line peers or to peers with less similar interests with new links corresponding to peers with more similar interests.

Peers also store long-range links in their routing indexes which stand as short paths to dissimilar clusters. For the update of the long-range links, peer p_i uses a random walk in the network with TTL t_R to discover peers with dissimilar interests.

Procedure Rewiring($p_i, I_i, \tau_R, \theta, m$)
Initiated by p_i when neighborhood similarity NS_i drops below θ.

input: peer p_i with interest I_i and routing index RI_i
output: updated routing index RI_i

1: compute $NS_i = \frac{1}{s} \cdot \sum_{\forall p_j \in RI_i} sim(I_i, I_j)$
2: **if** $NS_i < \theta$ **then**
3: $L \leftarrow \{\ \}$
4: create FindPeers()
5: $p_k \leftarrow p_i$
6: **repeat**
7: send FindPeers() to
m neighbours of p_k with interests most similar to I_i
8: let p_j be a neighbour of p_k receiving FindPeers()
9: $L \leftarrow L :: \langle ip(p_j), I_j \rangle$
10: $p_k \leftarrow$ every p_j receiving FindPeers()
11: $\tau_R \leftarrow \tau_R - 1$
12: **until** $\tau_R = 0$
13: return list L to p_i
14: update RI_i with information from L

Figure 3.2: The rewiring protocol

3.3.4 Query Processing Protocol

Let us assume that user issues a query q through peer p_i, where q is a term vector. Initially, p_i compares q against its interest I_i[2]. If $sim(q, I_i) \geq \theta$, where $sim()$ can be any appropriate similarity function between a query vector and a peer interest (e.g., the cosine similarity between their term vector representations [Sal89]) and θ is the similarity threshold, then p_i creates a message of the form QUERY$(ip(p_i), q, \tau_b)$, where τ_b is the query TTL, and forwards it to *all* its neighbours using the short-range links in RI_i. This forwarding technique is referred to as *query broadcasting* (or *query explosion*) [Sch04]: the query is broadcasted to the neighbours of p_i, which (due to clustering) are similar to p_i and will (most likely) be able to answer the query.

If $sim(q, I_i) < \theta$, peer p_i forwards a QUERY$(ip(p_i), q, \tau_f)$ message with query TTL τ_f to m peers connected to p_i (using the short-range and the long-range links in RI_i) with interests most similar to q. The query message is thus, forwarded through distinct paths from peer to peer until a peer p_j similar to the query is reached (i.e., a peer p_j with interest I_j such that $sim(q, I_j) \geq \theta$). This query forwarding technique is referred to as *fixed forwarding* [Sch04], since forwarding proceeds until q reaches a cluster of similar peers. Then, q is broadcasted in the neighborhood of this peer as described in the previous paragraph. All forwarding peers execute the aforementioned protocol and reduce τ_f by one at each step of the forwarding procedure. The combination of the two parts of query routing (i.e., fixed forwarding and broadcasting) referred to in the literature as *fireworks technique* [NSC02, TXD03].

Notice that the value of the broadcasting TTL τ_b is different from the value of the fixed forwarding TTL τ_f. Typically, τ_b is smaller than τ_f since in broadcasting the QUERY() message needs to reach peers only a few hops away (i.e., in the same cluster of the message recipient). In the case of fixed forwarding the message needs to explore regions of the network that are possibly far away from the query initiator. Figure 3.3 presents an the query processing protocol.

[2]In the case that p_i has multiple interests then it compares q independently against each one of its interests.

Procedure Query_Processing($q, p_i, \tau_f, \tau_b, \theta, m$)
Compares query q against the document collection of p_j, retrieves matching documents and forwards q to the network.

input: query q issued by peer p_i and threshold θ
output: list R of documents similar to q

1: **if** $sim(q, I_i) \geq \theta$ **then**
2: compare q against p_i's local document collection
3: **if** $sim(q, d) \geq \theta$ **then**
4: $R \leftarrow R :: \langle p(d), m(d), Sim(q, d) \rangle$
5: send message RETRES($ip(p_j), R$) to p_i
6: $\tau_b \leftarrow \tau_b - 1$
7: forward QUERY() to all short-range links in RI_i
8: **else**
9: forward QUERY() to m neighbours of p_i with interests most similar to q
10: $\tau_f \leftarrow \tau_f - 1$
11: **repeat** the above procedure for p_i's neighbours
12: **until** $\tau_f = 0$ or $\tau_b = 0$

Figure 3.3: The query processing protocol

3.3.5 Information Retrieval Protocol

Assume that a peer p_j similar to the query q is reached. Apart from executing the forwarding protocol, p_j also applies the following procedure for retrieving information similar to q. Query q is matched against p_j's local document collection and all documents d with $sim(q, d) \geq \theta$, where $sim()$ can be any appropriate similarity function between a query and a document vector (e.g., the cosine similarity between their vector representations [Sal89]) and θ is the similarity threshold, are retrieved and ordered by similarity to the query. Subsequently, p_j creates a result list R containing tuples of the form $\langle p(d), m(d), sim(q, d)\rangle$ for each relevant document d, where $p(d)$ is a pointer to d and $m(d)$ are metadata describing d (e.g., document title, author and an excerpt of the document's text in the style of search engine result presentation). The resulting list is placed in a message of the form RetRes$= (ip(p_j), R)$ and is returned to the peer that initiated the query using the contact information contained in the Query() message. In this way, query initiator p_i accumulates the results obtained by different peers, merges the different lists in a single list that contains unique entries sorted by descending similarity and presents the results to the user.

3.4 Conclusions

We presented i*Cluster*, a generic architecture designed for information retrieval that uses a semantic overlay network as its routing infrastructure. In the context of this architecture, we presented a suite of protocols that define the behaviour of the peers that populate the system. i*Cluster* is the first system to overcome the specialisation assumption, provides scalability, robustness and fault-tolerance, and enables the organisation of the participating peers according to the available resources for improving retrieval effectiveness and network traffic.

Chapter 4

A Measure for Cluster Cohesion

Measures for quantifying clustering quality (i.e., how well similar peers are clustered together in the network) are presented and discussed in this chapter. Section 4.1 surveys existing work in this area. Section 4.2 focuses on two popular measures namely (i) clustering coefficient [WS98] and (ii) generalised clustering coefficient [HAH07]. Section 4.3 introduces a new measure, coined *clustering efficiency* measure κ, while Section 4.4 concludes the chapter.

4.1 Introduction

Several works on peer organisation using semantic overlay networks (e.g., [DNV06, Sch04, LLS04]) are based on the idea of small-world networks [Wat99]. A small-world network is a type of graph in which nodes are not neighbours of one another, but can be reached from every other node by a small number of routing hops or steps. Small-world networks are typically characterised by small path length (for reaching a cluster) and high values of clustering coefficient [WS98]. Because of the small path length most pairs of peers will be connected by at least one short path. Therefore, queries can be routed quickly towards a relevant peer. On the other hand, clustering coefficient is geared towards the formation of cliques. It follows that high values of clustering coefficient are characterised by densely connected clusters of peers. More recent work on network cohesion modify [FHJS02, FC06] or generalise [HAH07] the above clustering coefficient measure.

In this chapter, we introduce a measure that quantifies the underlying (dynamic) P2P structure (for directed and undirected networks) by focusing on the retrieval effectiveness of the network, so that the higher the value of this measure is the better the performance of retrievals will be. This measure, denoted as *clustering efficiency* measure κ, assesses on the quality of clustering in semantic overlay networks. Unlike clustering coefficient that takes into account only the immediate neighbours of a peer, the proposed measure considers information about how well all peers containing similar information are clustered together. This work is presented in [RP08a].

4.2 Two Popular Clustering Measures

In this section, we present two popular measures appearing in the literature: (i) clustering coefficient [WS98], which is the measure typically used to evaluate clustering cohesion in a network and (ii) generalised clustering coefficient [HAH07], which extends the previous measure by taking into account the whole neighborhood around a peer.

4.2.1 Clustering Coefficient

The clustering coefficient measure introduced by Watts and Strogatz [WS98] is used to determine whether a graph is a small-world network. The clustering coefficient measure is nowadays widely used in semantic overlay networks to describe the effect of peer clustering [Sch04, HLY06]. The clustering coefficient c_i for peer p_i is formally defined as the ratio of links between the peers within p_i's neighborhood (i.e., between the peers contained in p_i's routing index) with the number of links that could possibly exist between these peers. In other words, clustering coefficient is a measure that quantifies how close a peer and its neighbours are from being a clique (i.e., complete graph).

If s is the number of peers in the routing index RI_i of p_i, then each peer $p_j \in RI_i$ could connect to $s-1$ other peers in this neighborhood. Thus, all possible connections between the peers contained in RI_i are $s(s-1)$. The clustering coefficient of

peer p_i is defined as:

$$c_i = \frac{|\{\langle p_j, p_k \rangle\}|}{s(s-1)}, \quad p_j, p_k \in RI_i, \; p_k \in RI_j, \tag{4.1}$$

where $\{\langle p_j, p_k \rangle\}$ is the set of existent peer connections and $||$ denotes the number of items in the set. The clustering coefficient measure is equal to 1 if every neighbour connected to p_i is also connected to every other peer within the neighborhood and equal to 0 if no peer connected to p_i connects to any other peer in p_i's neighborhood.

The clustering coefficient for the whole network is defined as the average of the clustering coefficient of all peers in the network:

$$\overline{c} = \frac{1}{N} \sum_{i=1}^{N} c_i, \tag{4.2}$$

where N is the number of all peers.

The clustering coefficient measure takes into account only the immediate nearest neighbours of a peer and the links between these peers. However, this may not always correspond to the underlying network structure. For example Figure 4.1 presents three (small) networks. All of them have the same number of peers and the same number of links between these peers, but different structure. The clustering coefficient of the central peer is the same for all cases (equals 1/3), although the network structure is different, for example the peers in the rightmost figure are less interconnected than the peers in the first two figures.

4.2.2 Generalised Clustering Coefficient

The so called generalised clustering coefficient introduced by Hansen et al. [HAH07] tries to broaden the clustering concept by taking into account a local sub-network characterised by a radius r around a peer. The generalised clustering coefficient gc_i of a peer p_i is formally defined as the number of paths of length n to peers that can also be reached by a path of length m from peer p_i, divided with the number of all

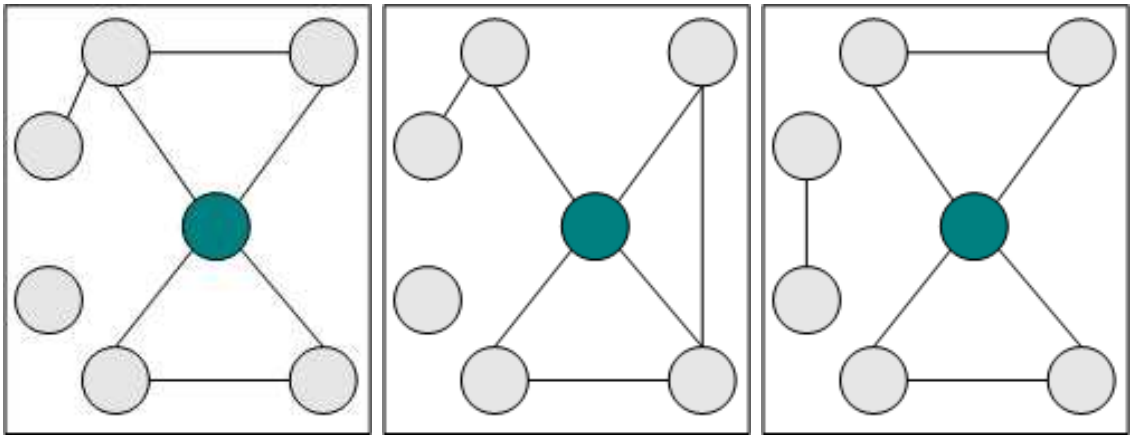


Figure 4.1: Example networks with the same number of links and peers, but different structure

possible paths that could exist in p_i's sub-network:

$$gc_i = \frac{p_i(m,n)}{\prod_{j=0}^{n-i}(N_i^r - j)}, \tag{4.3}$$

where N_i^r is the number of peers within radius r around p_i. The generalised clustering coefficient takes values in the interval $[0,1]$ and is reproduced to classical clustering coefficient when $r = 1$.

The generalised clustering coefficient for the whole network is defined as the average of the generalised clustering coefficient of all peers in the network:

$$\overline{gc} = \frac{1}{N}\sum_{i=1}^{N} gc_i, \tag{4.4}$$

where N is the number of all peers.

The generalised clustering coefficient stems from work in protein interaction networks and broadens the notion of clustering (as defined by the previous measure) by considering more than just the immediate neighbours of a peer. However, the emphasis of this measure is on how the network is structured as a whole.

4.3 Clustering Efficiency

In the following, we introduce a new network clustering measure, henceforth denoted as *clustering efficiency* measure κ. Clustering efficiency measure is characterised by radius τ_b around a peer. For the definition of the clustering efficiency measure, query routing protocol is also taking into account: when a query q reaches a similar peer p_i ($sim(q, I_i) \geq \theta$), then q is forwarded with TTL τ_b to all p_i's neighbours using the short-range links of p_i (implying that p_i's neighborhood consists of all peers by radius τ_b around p_i). Since the network is organised into clusters of similar peers and queries address these clusters, it is important that all peers are clustered together.

Formally, the clustering efficiency κ_i for a peer p_i is defined as the number of peers p_j similar to p_i ($sim(I_i, I_j) \geq \theta$) that can be reached from p_i within τ_b hops following short-range links, divided by the total number of peers in the network similar to p_i:

$$\kappa_i = \frac{\sum_{j=1}^{N} p_j : \{d_G(p_i, p_j) \leq t_b, sim(I_i, I_j) \geq \theta\}}{\sum_{k=1}^{N} p_k : \{sim(I_i, I_k) \geq \theta\}}, \tag{4.5}$$

where $d_G()$ is the distance (measured in number of hops) of the peers in the graph. The clustering efficiency measure equals 1 if the neighborhood of p_i contains all peers similar to p_i. Conversely, the clustering efficiency equals 0 if no peer similar to p_i is contained in the neighborhood of p_i.

The clustering efficiency for the network as a whole is defined as the average (over all peers in the network) of the clustering efficiency of each peer:

$$\overline{\kappa} = \frac{1}{N} \sum_{i=1}^{N} \kappa_i \tag{4.6}$$

Clustering efficiency is defined to reflect retrieval effectiveness. As it will be shown in the experiments (Section 5.3.4), clustering efficiency is directly associated with the performance of retrievals: the higher the value of clustering efficiency

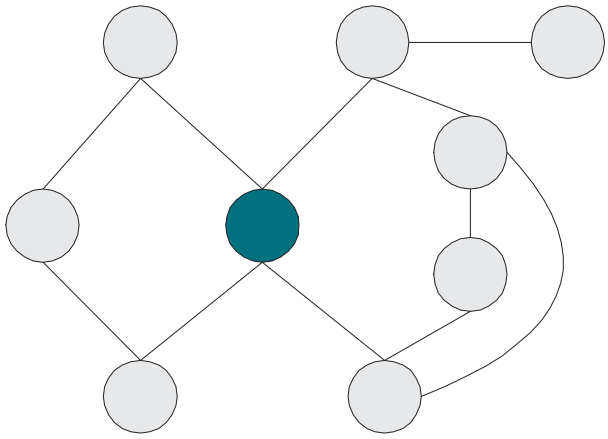


Figure 4.2: Example network structure representing a neighborhood

$\overline{\kappa}$ is, the better the performance of retrievals will be. Clustering efficiency gives information about the underlying network involving more than just the immediate neighbours of the peers and measures how well the network is structured from an information retrieval point of view. To show proof of concept, in Figure 4.1 the clustering efficiency of the central peer is 5/6 for the first two cases and 4/6 for the rightmost structure, where peers are less interconnected. In addition, consider the example neighborhood shown in Figure 4.2. The clustering coefficient of the central peer equals 0, the generalised clustering coefficient equals 1/6 and the corresponding clustering efficiency equals 1.

4.4 Conclusions

In this chapter, we introduced a new concept for network clustering, denoted as clustering efficiency measure $\overline{\kappa}$. By definition, clustering efficiency aims to reflect retrieval effectiveness: the higher the values of the clustering efficiency measure are, the better the underlying network organisation is in terms of retrieval effectiveness. Clustering efficiency measure has the following advantages over two popular measures appearing in the literature: (i) gives information about the underlying network involving more than just the immediate neighbours of the peers and (ii) measures how well the network is structured from an information retrieval point of view.

Chapter 5

Rewiring Peer Connections

This chapter considers rewiring as a critical component of i*Cluster* architecture, investigates the functional issues related to the design and the performance of the rewiring protocol, and explores the dependency of the retrieval performance on the rewiring process. The functional issues related to rewiring are introduced in Section 5.1. We consider and discuss a number of choices in designing a rewiring strategy in Section 5.2, while the experimental evaluation of the proposed strategies using real-world data and queries is presented in Section 5.3. Conclusions are presented in Section 5.4.

5.1 Introduction

Peer organisation in a semantic overlay network is achieved by applying a rewiring strategy which is periodically executed by each peer. This procedure establishes new connections to similar peers and disregards connections to peers that are dissimilar. Retrieval effectiveness is then improved by exploiting this information at query time (as queries may address clusters of similar peers). Although all systems based on semantic overlay networks apply some rewiring technique, there is no comprehensive study showing the effect of rewiring on system's performance.

In Chapter 3 we introduced i*Cluster*, a P2P network where peers are responsible for serving both users searching for information and users contributing information to the network. Each peer is characterised by its information content (i.e., its docu-

ment collection) which may be either automatically (by text analysis) or manually assigned to each document (e.g., tags or index terms). i*Cluster*, based on ideas from traditional distributed IR and recent work on P2P networks, aims at better retrieval effectiveness in terms of recall and network traffic. i*Cluster* applies a periodic rewiring protocol to cluster peers with similar interests and a fireworks-like technique to route queries within the network.

In this chapter, a framework for studying the attribution of rewiring strategies in semantic overlay networks is proposed. Initially, we provide a better understanding on the functional issues related to the design and the performance of rewiring protocols by answering the following fundamental questions: (i) what is an appropriate forwarding strategy for rewiring messages within a SON, (ii) what information should be used to update peer connections, (iii) should peer connections be bi-directional or just one-directional, (iv) how coherent and well-connected should a cluster of similar peers be? Based on insight acquired by answering the above questions, several variants of a generic rewiring approach are reviewed and evaluated. We show how peer organisation is affected by the different design choices of the rewiring mechanism and how these choices affect the performance of the system overall (both in terms of communication overhead and retrieval effectiveness). Our experimental evaluation with real-word data and queries confirms the dependence between rewiring strategies and retrieval performance, and gives insights on the trade-offs involved in the selection of a rewiring strategy. All the work included in this chapter is presented in [RPT09].

5.2 Rewiring Strategies

In the following, the different components of rewiring are identified and several variants to the basic rewiring protocol[1] are proposed corresponding to modifications in (i) the routing technique used by peers to forward rewiring messages, (ii) the amount of information available to peers for updating their short-range links, (iii) the technique for establishing links, and (iv) the strategy used by peers to update their long-range links. In each of the following sections, we propose design choices

[1]The basic rewiring protocol is presented in Chapter 3, Section 3.3.3.

for each component of rewiring, identify their differences from the basic protocol and discuss their performance.

5.2.1 Forwarding of Rewiring Messages

A very important component of the rewiring protocol is the routing technique used by the peers to forward the rewiring messages. We introduce three different message forwarding techniques for the FindPeers() message and elaborate on the peer clustering performance.

The Gradient Walk (GW) Strategy

This strategy is used in the basic rewiring protocol. A message recipient p_j forwards the FindPeers$(ip(p_i), I_i, L, \tau_R)$ message to the set of m peers stored in its routing index RI_j with interests most similar to I_i (the interest of message initiator p_i).

The idea behind the Gradient Walk (GW) strategy is to forward the rewiring message to peers which are likely to be similar to I_i, collect information about these peers in the FindPeers() message, use this information to update p_i's short-range links and by this, connect p_i to similar peers.

A drawback of this strategy is encountered for the case where clusters of similar peers cannot be reached or have not yet been formed. This case is typical encountered at system bootstrapping and also at periods of high churn, where peers initiating a rewiring process have low probability to discover other similar peers. The GW strategy is then expected to lead to poor peer clustering.

The Random Walk (RW) Strategy

Under the Random Walk (RW) strategy, a message recipient p_j forwards the FindPeers() message to a set of m randomly chosen peers stored in its routing index RI_j. This is implemented by modifying the line 7 of Figure 3.2 as follows:

`send FindPeers() to` m `random neighbours of` p_k

The idea behind the RW strategy is to explore the network for discovering peers

with interests similar to I_i (the interest of the message initiator p_i). At periods where the network is not well-clustered, this random exploration will increase the probability of finding peers with interests similar to I_i. Compared to the GW strategy, RW is expected to converge to a clustered network faster. Since peer clustering is used to support retrieval functionality, the RW strategy is also expected to exhibit better retrieval performance compared to the GW strategy, at least for un-organized networks.

The Combined (GW+RW) Strategy

In the combined (GW+RW) strategy, a message recipient p_j forwards the FindPeers() message with equal probability either to (i) a set of m randomly chosen peers, or (ii) the set of m peers with interests most similar to the interest I_i of the initiator peer p_i. The GW+RW strategy presented here resembles the strategy used for the forwarding of the rewiring messages in [Sch04]. This is implemented by modifying the line 7 of Figure 3.2 as follows:

```
generate a random number x
if x ≥ 0.5 then
  send FindPeers() to
  m neighbours of p_k with interests most similar to I_i
else
  send FindPeers() to m random neighbours of p_k
```

This strategy aims at combining the benefits from the RW and the GW strategies, as the RW strategy can be applied in unclustered networks for discovering similar peers, while the GW strategy can be applied for exploring neighborhoods of similar peers.

The rationale of applying the GW+RW strategy is not only to connect p_i to similar peers discovered by forwarding the message to similar peer clusters, but also by enabling propagation of the forwarding message to other similar peers through other non-similar ones. However, the GW+RW strategy combines the two components in a naive way by invoking one of the two components based on a random decision.

The Informed (inf-GW+RW) Strategy

Under the informed (inf-GW+RW) strategy, a message recipient p_j forwards the FindPeers() message either to (i) a set of m randomly chosen peers, or (ii) the set of m peers with interests most similar to the interest I_i of the initiator peer p_i, taking into account information on how well the network is clustered. This strategy goes one step further from the GW+RW strategy, since it invokes each component based on an informed (rather than a random) decision. In inf-GW+RW, p_j takes into account (its personal view of) how well the network is organised and decides whether to use gradient or random walk for message forwarding. This is implemented by modifying line 7 of Figure 3.2 as follows:

```
if NS_k ≥ θ then
  send FindPeers() to
  m neighbours of p_k with interests most similar to I_i
else
  send FindPeers() to m random neighbours of p_k
```

Contrary to the GW+RW strategy, inf-GW+RW puts emphasis on peer perception of network organisation.

Figure 5.2 illustrates how the different forwarding strategies affect peer clustering. We used the same unclustered (random) initial network on Figure 5.1 as input for all the strategies and visualised the network when it converged to a stable organisation. Dark-coloured nodes represent peers with similar content (i.e., belonging to the same class) whereas, light-coloured nodes represent peers from other categories that the former are connected to. When using the GW strategy for rewiring, dark-coloured peers store many connections to dissimilar peers and are not well-clustered (Figure 5.2(a)). The RW and the inf-GW+RW strategies result in better network organisation by linking peers with the same interest (Figure 5.2(b) and (d)). In the case of the GW+RW strategy, notice the existence of some isolated peers that are peers non-reachable by other similar peers (e.g., in the lower right part of Figure 5.2(c)), a situation that is expected to undermine retrieval effectiveness.

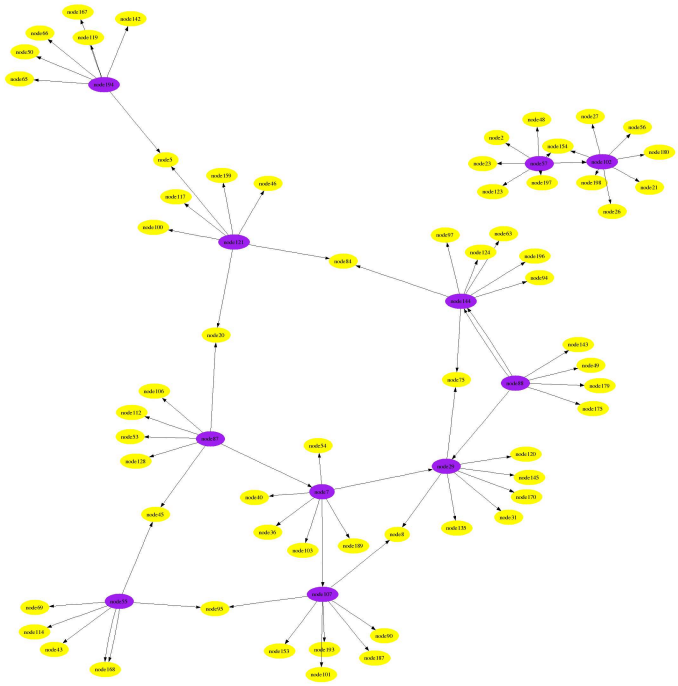
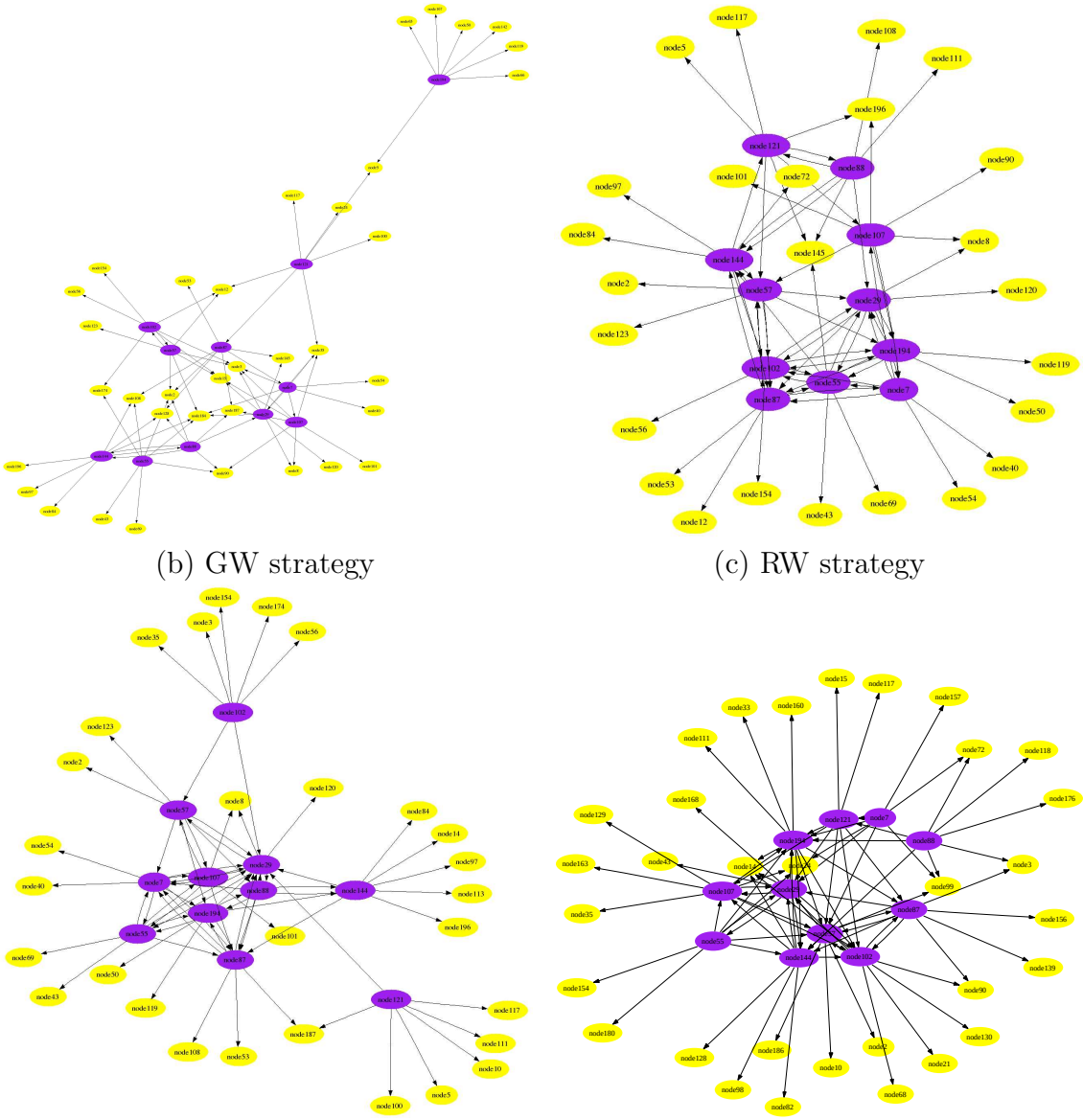


Figure 5.1: Initial network topology

5.2.2 Updating the Short-Range Links

In the basic organisation protocol, only the initiator of a FindPeers() message may utilise the message contents to collect information about peers with similar interests and update its routing index. In what follows, we examine the idea of letting also peers along the forwarding path to utilise the information collected by the message. In this way, other peers, aside from the initiator peer, may exploit this information to update their routing indexes and improve clustering without incurring extra message traffic. This strategy is expected to increase the speed of clustering.

In the following, we utilise a new system-wide parameter denoted as *refinement probability* ϱ, which is related to a peer's decision to exploit (or not) the information contained in the FindPeers() message it receives. This decision is taken by each peer in a fully decentralised way. Parameter ϱ takes values in the interval $[0, 1]$ and is used as follows: each peer receiving a FindPeers() message may utilise the information contained in it (i.e., the interests of previous message recipients) with probability ϱ. When $\varrho = 0$, *no peer* apart from the message initiator use the contents of FindPeers() message, while when $\varrho = 1$, *all peers* participating in the

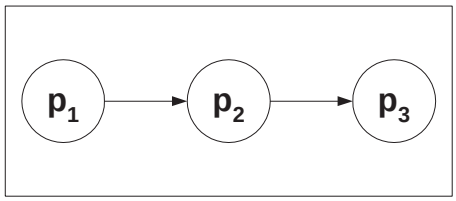


(b) GW strategy (c) RW strategy

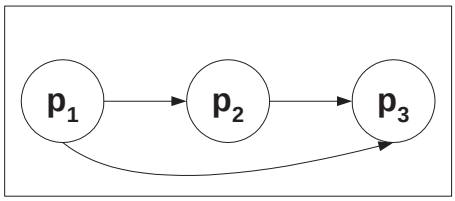
(d) GW+RW strategy (e) inf-GW+RW strategy

Figure 5.2: Network organisation for different message forwarding strategies

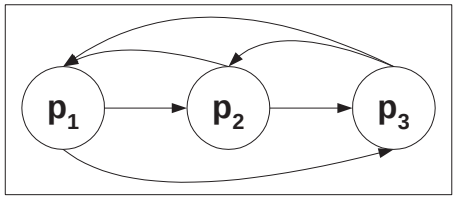
forwarding of the FindPeers() message exploit this information to update their routing indexes. When $0 < \varrho < 1$, a peer p_j receiving a FindPeers() message exploits this information with probability ϱ. Notice that, when $\varrho = 0$ this protocol is reduced to the basic protocol. System parameter ϱ is user-defined and needs to be known upon bootstrapping. The use of ϱ is implemented by modifying the line 6 of Figure 3.2 as follows:



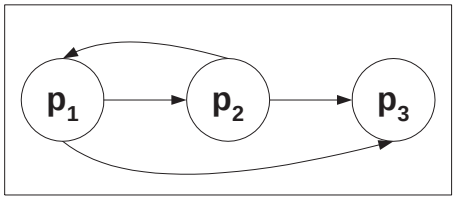
(a) Initial connections between peers p_1, p_2 and p_3



(b) Rewired peer connections for $\varrho = 0$



(c) Rewired peer connections for $\varrho = 1$



(d) A possible rewiring of connections for $0 < \varrho < 1$

Figure 5.3: An example illustrating the usage of the refinement probability ϱ

`let` p_j `be a neighbour of` p_k `receiving` FindPeers()

p_j `generates a random number` x

`if` $x \geq \varrho$ `then update` RI_j `with information from` L

If peer p_j decides to utilise the information contained in a FindPeers() message, then p_j updates its routing index RI_j by replacing short-range links that are outdated or pointing to peers with dissimilar interests with links found in the message. It follows that, the initiator peer p_i of FindPeers() message *always* explores the information contained in the message to update its short-range links, since this peer initiated the rewiring process.

Example 1 *Let us assume a network where each peer can store only 2 links to other similar peers ($s = 2$), rewiring TTL is 2 ($\tau_R = 2$) and that some arbitrary peers p_1, p_2 and p_3 that belong to the network are all interested in the same topic. Assume also, that these peers are initially connected to each other as shown in Figure 5.3(a). In the following we demonstrate how peer organisation is affected by different values of refinement probability ϱ.*

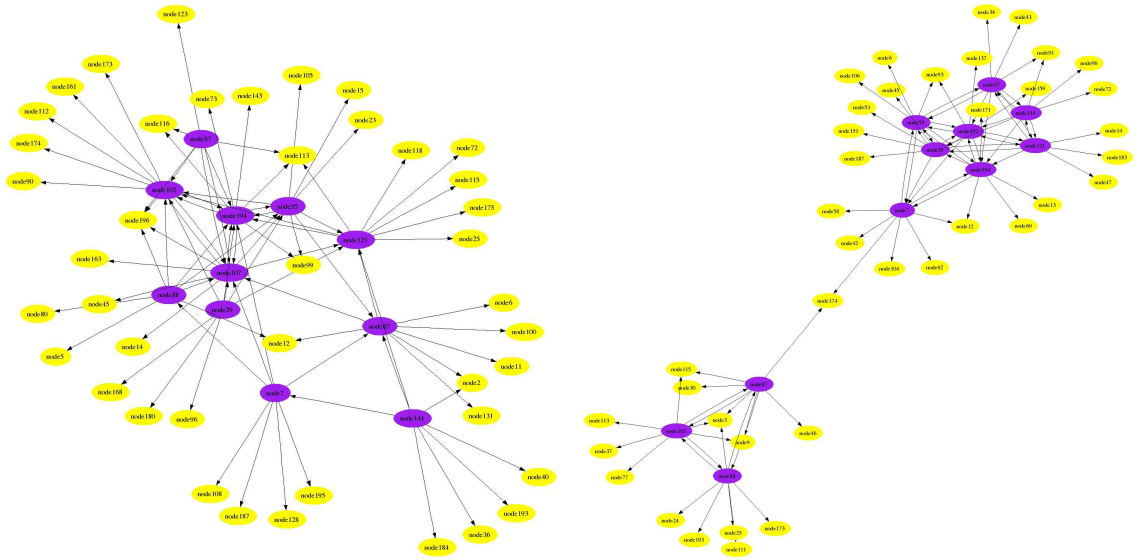
- $\varrho = 0$*: When peer p_1 decides to initialise a rewiring procedure, it sends a* FindPeers() *message with its interest to peer p_2. By definition, when $\varrho = 0$ no peer apart from the message initiator uses the contents of the message.*

Thus, p_2 does not utilise the information in the message, appends its own interest to it and forwards it to peer p_3. Similarly, p_3 appends its interest to the message and sends it back to the initiator peer p_1 (since τ_R has reached 0). Subsequently, p_1 uses the information in the message and connects to p_3 since they have similar interests. The resulting (rewired) links are shown in Figure 5.3(b).

- $\varrho = 1$*: In this case, all forwarding peers exploit the information contained in the* FindPeers() *message to refine their short-range links. When peer p_1 decides to initialise a rewiring procedure it contacts peer p_2. Since $\varrho = 1$, p_2 uses the information contained in the message and creates a link to p_1 as they have similar interests (remember that the message contains also the interest of the message initiator). Next, p_2 appends its interest to the message and forwards it to peer p_3, which also uses the information contained in the message and connects to both p_1 and p_2. Similarly, p_3 appends its interest to the message and sends it back to p_1 that in turn, updates its routing index. The resulting links are shown in Figure 5.3(c).*

  Notice that all, p_1, p_2 and p_3, have their short-range links occupied by peers that have similar interests simply by using the rewiring message that p_1 initiated. To achieve this when $\varrho = 0$ would require more rewiring messages, since p_2 and p_3 would need to initiate their own rewiring procedure to collect information about similar peers and refine their links. In our example setting, $\varrho = 1$ seems to achieve a good result by organising p_1, p_2 and p_3 in a fully-connected cluster. However, this cluster (Figure 5.3(c)) is isolated from the rest of the similar peers that may exist in the network, since there is no free short-range link to connect to other peers with similar interests. Creating isolated peer clusters has a negative effect in recall, as a query might not reach all relevant clusters in the network.

  Notice that allowing peers to store more short-range links in their routing indexes could eliminate the isolated peer clustering problem. However, there is a trade-off related to the size of the routing index. Increasing the number of the short-range links eliminates the probability to create isolated peer clusters,



(a) $\varrho = 0$ (b) $\varrho = 1$

Figure 5.4: Network organisation when using different values of ϱ

but might result in more ambiguous peer clusters. However, the idea of peer rewiring is to cluster similar peers together, rather than let each peer has a vast knowledge of the network. In addition, increasing the number of short-range links stored by each peer results in increased message traffic when processing a query, since queries are broadcasted in neighborhoods of similar peers (according to the query processing protocol). Consequently, the number of short-range links stored by a peer must be restricted to peers satisfying the peer similarity criterion as defined in Section 3.3.3.

- $0 < \varrho < 1$*: To avoid generating excess message traffic during rewiring (as with* $\varrho = 0$*) and also creating isolated peer clusters (as with* $\varrho = 1$*),* ϱ *should be appropriately tuned to a value between 0 and 1. Section 5.3 shows that when* $0 < \varrho < 1$ *the rewiring procedure performs better both in terms of reduced network load and retrieval effectiveness. Figure 5.3(d) shows a possible network organisation when* $0 < \varrho < 1$ *(in this case, peer* p_2 *utilised the information in the rewiring message initiated by peer* p_1*).*

Figure 5.4 illustrates how ϱ affects peer clustering. The initial peer connections are shown in Figure 5.1. Dark-coloured nodes represent peers of the same class, whereas light-coloured nodes represent peers from other categories that the former

are connected to. Notice that, when $\varrho = 0$ most of the similar peers are clustered together, but there are also peers that are hard to reach (dark-coloured peers in the lower right part in Figure 5.4(a)). When $\varrho = 1$ peers of the same category have formed two well-connected but disjoint neighborhoods (Figure 5.4(b)). This reveals an inherent weakness of rewiring: similar peers are not always clustered together, since they may form more than one separate clusters. Notice that this will also affect retrieval performance since queries are not always capable of locating all disjoint clusters in the network.

To recap, refinement probability ϱ denotes the number of peers that update their links by exploiting the rewiring messages initiated by other peers. In general, the lower the value of ϱ is, the more messages are needed for achieving a clustered organisation. Reversely, the higher the value of ϱ is, the faster the network will converge to a clustered organisation. However, setting ϱ to a high value might result in isolated clusters. The objective is to achieve a network organisation by adjusting ϱ to an intermediate value so that, similar (but disjoint) clusters are connected by a path while maintaining low message traffic. This (optimal) value of ϱ is determined by experimentation. Parameter ϱ does not depend neither on the document content nor on the total number of peers in the network.

5.2.3 Introducing Symmetric Links

In the basic organisation protocol, when a peer p_i discovers that its interest is similar to that of a peer p_j, it is not entailed that p_j is aware of this similarity. We modify the basic protocol to facilitate symmetric links (SL) among peers. Under this strategy, if a peer p_i discovers another similar peer p_j through a FindPeers() message, p_i updates its routing index RI_i and sends a SimPeer$(ip(p_i), I_i)$ message to p_j to inform it that they share similar interests. This is implemented by modifying the line 14 of Figure 3.2 as follows:

`update` RI_i `with information from` L

`send` SimPeer$(ip(p_i), I_i)$ to $p_j : p_j \in L, sim(I_i, I_j) \geq \theta$

When p_j receives a SimPeer() message, it uses this information to decide whether to refine or not its short-range links with a link to p_i. Notice that symmetry

in link creation is not enforced, since p_j may decide not to update its routing index RI_j if it is already connected to similar peers (i.e., $sim(I_j, I_k) \geq sim(I_j, I_i)$ for all interests I_k in RI_j). Notice also, that the SimPeer() message is sent only when p_i updates its routing index. The only cost associated with this modification is one extra message for each RI_i update.

The rationale behind the symmetric link approach is achieving faster network convergence by exploiting all information available during rewiring. However, creating clusters of similar peers with the SL strategy might yield highly coherent peer neighborhoods that in turn, increase the probability of creating isolated cliques of similar peers (similarly to the case of $\varrho = 1$). Section 5.3.3 explores the effect of the SL strategy and compares it against protocol variations with non-symmetric links (denoted as non-SL strategy).

5.2.4 Updating the Long-Range Links

Long-range links allow peers to connect with other peers irrespective of content. Long-range links in a semantic overlay network are exploited to facilitate long jumps in the network and allow reaching distant peer neighborhoods with small communication overhead. Two strategies for maintaining long-range links in a semantic overlay network are discussed below.

The Random Sampling (RS) Strategy

According to the Random Sampling (RS) strategy, a peer p_i creates a FindDisPeers$(ip(p_i), I_i, \tau_R)$ message to search for peers with dissimilar interests. This strategy is adopted by the basic rewiring protocol of Section 3.3. The message is forwarded in the semantic overlay network using a random walk strategy until message TTL τ_R reaches zero. Then, peer p_j that received the message, compares its interest I_j against I_i contained in the message and if $sim(I_j, I_i) < \theta$ it appends its IP address $ip(p_j)$ and its interest I_j to the message and sends it back to p_i. Otherwise, it sends an appropriate message to notify p_i that they share similar interests. A peer p_i that receives a FindDisPeers() message back, checks whether it already

has a long-range link to a peer p_k, for which $sim(I_j, I_k) \geq \theta$. If so, p_i disregards the message and re-initiates the procedure, since it already stores a short path towards the cluster that both p_k and p_j belong. Otherwise, p_i updates the long-range links of RI_i with a pointer to p_j.

The Biased Sampling (BS) Strategy

The idea behind the Biased Sampling (BS) strategy is to use the rewiring protocol and the corresponding rewiring messages to collect information about peers with non-similar interests. With this strategy, a peer p_i receiving a FindPeers() message updates its long-range links as follows: It randomly selects a peer p_j contained in the message, for which $sim(I_i, I_j) < \theta$ and $sim(I_j, I_k) < \theta$ holds, for all peers p_k stored as its long-range links. In this way, the long-range links of p_i contain (i) peers with interests dissimilar to I_i and (ii) peers with dissimilar interests between each other (i.e., links to different peer clusters). Subsequently, p_j forwards the FindPeers() message according to the rewiring protocol used.

Notice that peers do not have to explicitly initiate a rewiring procedure to update their long-range links. However, peers that belong in the same neighborhood will tend to have high overlap in their long-range links, since they use the same messages to acquire these links. This *biased sampling* of the network may reduce retrieval effectiveness, since queries will follow similar routes and thus, leave parts of the network unexplored.

5.2.5 Combining all Together

All the modifications to the basic rewiring protocol discussed above produce a wide variety of design choices that can be combined to increase the retrieval effectiveness and reduce the message overhead of any SON-based system. Summarising, the different rewiring strategies presented earlier are the following: (i) GW, RW, GW+RW or inf-GW+RW, (ii) ϱ taking values in the interval $[0, 1]$, (iii) SL or non-SL and (iv) RS or BS. Since the proposed strategies can be utilised independently of each other, this creates a large space of possible combinations. Notice that different (combina-

Procedure Rewiring2($p_i, I_i, \tau_R, \theta, m, \varrho$)
Initiated by p_i when neighborhood similarity NS_i drops below θ.

input: peer p_i with interest I_i and routing index RI_i
output: updated routing index RI_i
updated routing indexes RI_j

1: compute $NS_i = \frac{1}{|s|} \cdot \sum_{\forall p_j \in RI_i} Sim(I_i, I_j)$
2: **if** $NS_i < \theta$ **then**
3: $L \leftarrow \{\ \}$
4: create FindPeers()
5: $p_k \leftarrow p_i$
6: **repeat**
//forward FindPeers() *message using*
//strategy $S = \{GW, RW, GW+RW,$ *inf-*$GW+RW\}$
7: send(FindPeers(), S)
8: let p_j be a neighbour of p_k receiving FindPeers()
//update short-range links with probability ϱ
9: p_j generates a random number x
10: **if** $x \leq \varrho$ **then**
11: update RI_j with information from L
//if symmetric links are used
12: **if** $SL = 1$ **then**
13: send SimPeer($ip(p_j), I_j$) to $p_\lambda : p_\lambda \in L, sim(I_j, I_\lambda) \geq \theta$
14: $L \leftarrow L :: \langle ip(p_j), I_j \rangle$
15: $p_k \leftarrow$ every p_j receiving FindPeers()
16: $\tau_R \leftarrow \tau_R - 1$
17: **until** $\tau_R = 0$
18: return FindPeers() message to p_i
19: update RI_i with information from L
//if symmetric links are used
20: **if** $SL = 1$ **then**
21: send SimPeer($ip(p_i), I_i$) to $p_\lambda : p_\lambda \in L, sim(I_i, I_\lambda) \geq \theta$

Figure 5.5: The modified rewiring protocol

tions of) rewiring strategies may emphasise recall, while others may be efficient in terms of network traffic. In the next section, we show how each individual strategy affects the performance of the basic protocol and also, identify interesting strategy combinations that can further improve system performance. Figure 5.5 presents the algorithm implementing the modified rewiring protocol. To demonstrate how these strategies can be combined together, we have parameterised each module used in the initial protocol of Figure 3.2 taking into account the alternative strategies.

5.3 Experimental Evaluation

In this section, we present the evaluation of the proposed rewiring protocols using two real-world datasets with web and medical documents.

5.3.1 Performance Measures

As it is typical in the evaluation of P2P information retrieval systems, performance is measured in terms of network traffic and retrieval effectiveness. The *network traffic* is measured by recording the number of rewiring (respectively search) messages sent over the network during rewiring (respectively querying). The retrieval effectiveness is evaluated using *recall* (i.e., the number of relevant documents retrieved over the total number of relevant documents in the network). Additionally, *recall/(search) message* is used to quantify a benefit/cost metric. Notice that in our setting precision is always 100% since only relevant documents are retrieved. In this section, our evaluation is mostly goal-oriented: we are interested in measuring the system performance directly, without resorting to clustering quality measures (e.g., [HLY06]). One strategy is better than another if it presents high recall for less network traffic.

5.3.2 Experimental Testbed

Datasets. The first dataset contains over 556,000 web documents from the TREC-6[2] collection belonging in 100 categories and has been previously used to evaluate

[2]http://boston.lti.cs.cmu.edu/callan/Data/

Parameter	Symbol	Value
peers	N	2,000
short-range links	s	8
long-range links	l	4
similarity threshold	θ	0.9
rewiring TTL	τ_R	4
fixed forwarding TTL	τ_f	6
broadcast TTL	τ_b	2
message fanout	m	2

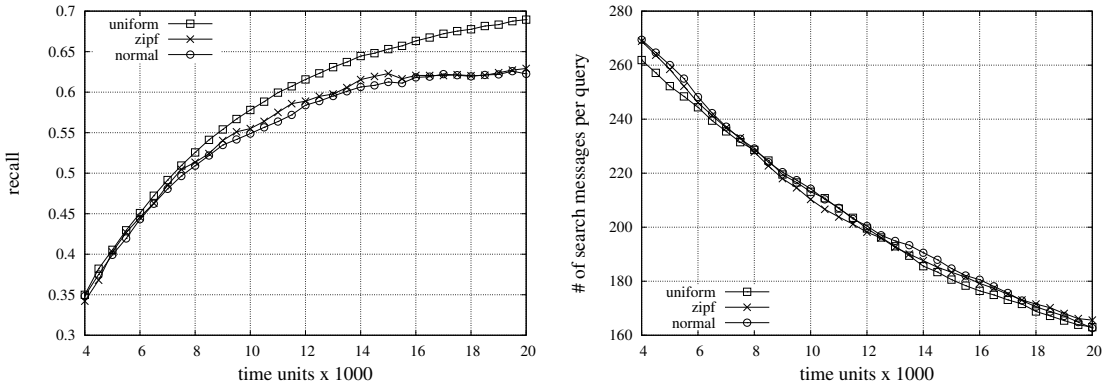
Table 5.1: Baseline parameter values

information retrieval algorithms over distributed document collections (e.g., [XC99]). The second dataset is a subset of the OHSUMED TREC[3] document collection that contains over 30,000 medical articles from 10 different categories as suggested in [SKK00]. The queries employed in the evaluation of both corpora are strong representatives of document categories (i.e., the topics of the categories).

Setup. We consider N loosely-connected peers, each of which contributes documents in the network from a single category. At the bootstrapping, peers join the network and are connected as described in Section 3.3.1. The base unit for time used in the experiments is the period t. The start of the rewiring procedure for each peer is randomly chosen from the interval $[0, 4K \cdot t]$ and its periodicity is randomly selected from a normal distribution of $2K \cdot t$, in the spirit of [Sch04]. Therefore, each peer starts (and goes over again) independently the rewiring process. We start recording the network activity at time $4K \cdot t$, when all peers have initiated the rewiring procedure at least once.

We used a network size of 2,000 peers. The average number of peers per class for the TREC-6 (respectively OHSUMED) corpus was 20 (respectively 200), with standard deviation 4.42 (respectively 68.8). We experimented with different values of similarity threshold θ, message forwarding TTL τ_R and query forwarding TTLs τ_b, τ_f. We consider that a given parameter value is better than another if it results in better clustering and retrieval for less communication load. The baseline parameter values used for the experiments are summarised in Table 5.1.

[3]http://trec.nist.gov/data/t9_filtering.html



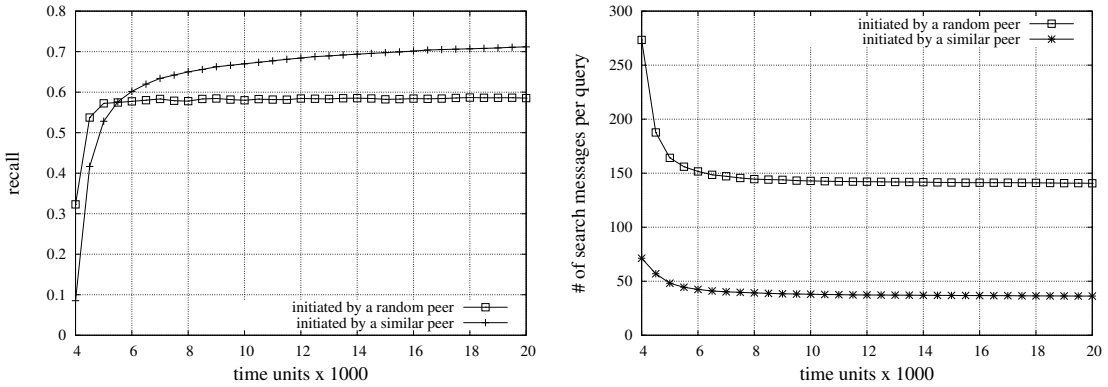
(a) Recall (b) Search messages per query

Figure 5.6: Performance under different document distributions for the RW strategy (TREC-6)

The simulator used to evaluate the rewiring protocols and their modifications was implemented in C/C++ and all experiments were run on a Linux machine. Our results were averaged over 25 runs (5 random initial network topologies and 5 runs for each topology). Peers use a specified rewiring protocol to update their links. Query processing is carried out as described in Section 3.3.4.

Document Distribution. We examined the dependence of the system performance on the distribution of the documents over the peers. Thus for a given rewiring strategy, we measured both retrieval performance and network traffic for three different real-life document distributions. Figure 5.6 illustrates retrieval performance, in terms of recall and search messages, as a function of time for the RW strategy when the documents are distributed to the peers (i) uniformly, (ii) using a heavy-tailed distribution (i.e., zipf) and (iii) using the normal distribution. As shown in Figure 5.6(a), recall is marginally affected by the document distribution. In terms of network traffic, Figure 5.6(b) illustrates that the number of search messages is not affected by the document distribution. In the rest of the evaluation, we use the uniform distribution to assign documents to peers.

Query Distribution. We also examined the dependence of the system performance on the query distribution. For a given rewiring strategy we measured both retrieval performance and network traffic for two different query distributions. Figure 5.7 illustrates retrieval performance, in terms of recall and communication load, as a function of time for the RW strategy (i) when the queries are initiated by similar



(a) Recall (b) Number of search messages

Figure 5.7: Performance under different query distributions for the RW strategy (TREC-6)

peers and (ii) when the queries are initiated by random peers. As shown in Figure 5.7(a), assuming a random query distribution over the peers results in worse retrieval performance. In terms of network traffic, Figure 5.7(b) illustrates that the number of search messages are 70% more in the case that queries are initiated by random peers. Although a setting where the queries follow peers' interests is a more realistic scenario for a content-sharing network, we assume that queries are issued from random peers in the network in the rest of the section since we want to evaluate the rewiring strategies under a stress test for a semantic overlay network.

5.3.3 Evaluation Results

Using Different Forwarding Strategies

Figure 5.8 illustrates the retrieval effectiveness of the network as a function of time for different forwarding strategies. The data set used in this set of experiments is TREC-6. At the beginning of the rewiring procedure ($t = 4K$) peers are still randomly connected, the network is not yet organised into clusters of peers with similar interests and the values for recall are low (around 35%). After some time, when all peers have executed the rewiring protocol more than once ($t = 8K$), we can observe the effect of the different forwarding strategies to peer organisation and consequently, to retrieval effectiveness. The GW strategy improves recall only by 3%, which can be attributed to poor network organisation. The RW strategy

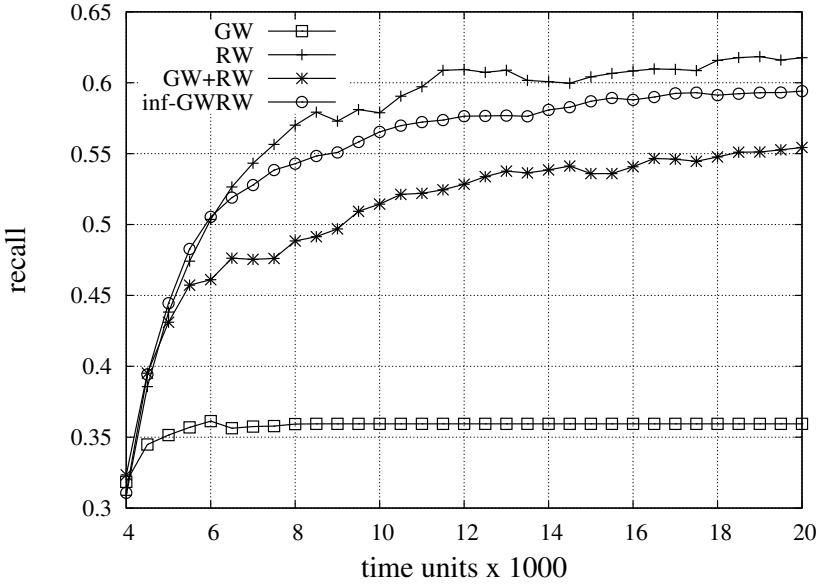


Figure 5.8: Retrieval effectiveness for different forwarding strategies (TREC-6)

demonstrates the best retrieval performance overall achieving up to 12% better recall than GW+RW and up to 8% better recall than inf-GW+RW. In turn, the inf-GW+RW strategy outperforms GW+RW achieving up to 5% better recall.

The fact that the RW strategy results in the best retrieval performance and also converges faster towards a clustered organisation of peers is attributed to the way it organises the network. The RW strategy explores the network in a random fashion increasing the probability to discover peers with similar interests, since peers are initially randomly connected. Even when similar peers start to get organised into clusters, using the RW strategy proves efficient in discovering isolated peers. The GW strategy explores peers' neighborhoods to discover peers with similar interests and yields poor results, since in an unclustered (random) network peers are not always connected to other similar peers. The GW+RW strategy performs a random selection of a forwarding strategy at each invocation and is thus, limited by the bad performance of the GW strategy. The inf-GW+RW strategy initially performs at least as good as the RW strategy. However, as the network converges towards a stable network, peers tend to choose the gradient walk for the forwarding of the rewiring messages reducing the probability to discover isolated peers.

Figure 5.9 shows the recall for both TREC-6 and OHSUMED corpora when the network is organised using different forwarding strategies. The values of recall

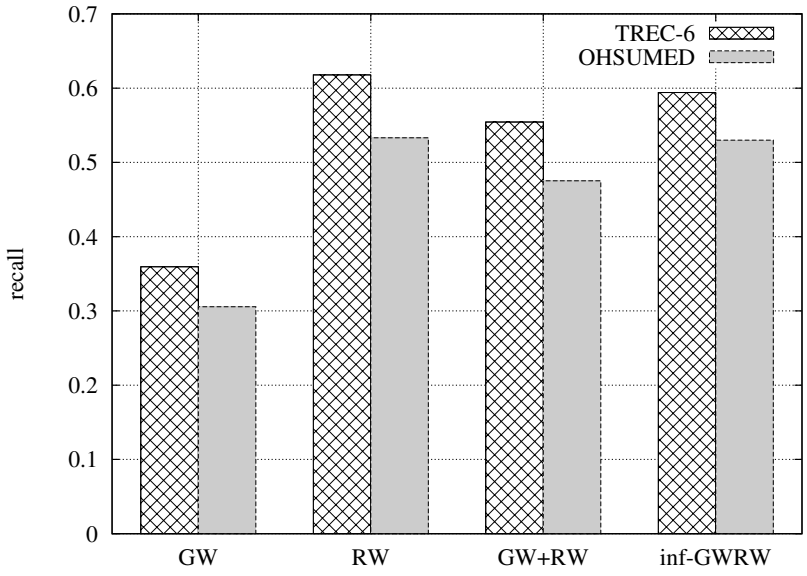


Figure 5.9: Retrieval effectiveness for different forwarding strategies

presented in the figure correspond to an organised network ($t = 20K$). The RW strategy proves to be the best for both corpora in terms of recall.

The lower values of recall in the case of OHSUMED corpus are due to the different number of peers per class. Specifically, there are 200 peers per class when using the OHSUMED corpus, whereas there are 20 peers per class in the case of TREC-6 corpus. This difference in the number of peers per class affects retrieval performance in two ways: (i) The rewiring process tries to collect all similar peers in the same neighborhood. Naturally, the smaller the number of peers per class is, the easier and the faster all similar peers are clustered together. When using the TREC-6 corpus the rewiring task is easier compared to when the OHSUMED corpus is used. (ii) Recall depends (among others) on the broadcasting TTL τ_b and on the number of short-range links s stored by each peer. Even though all similar peers are gathered together in the same neighborhood (no matter their number), there is the case that a query may not reach all peers within the cluster if the diameter of the corresponding peer neighborhood is bigger than s^{τ_b}. We choose to use small values for s and τ_b, since peer clusters are more cohesive and network traffic is kept low.

Figures 5.10 and 5.11 present network traffic (rewiring and search messages) for the four strategies over time. In terms of rewiring messages, Figure 5.10 shows that the network initially presents a high message overhead, which is greatly reduced

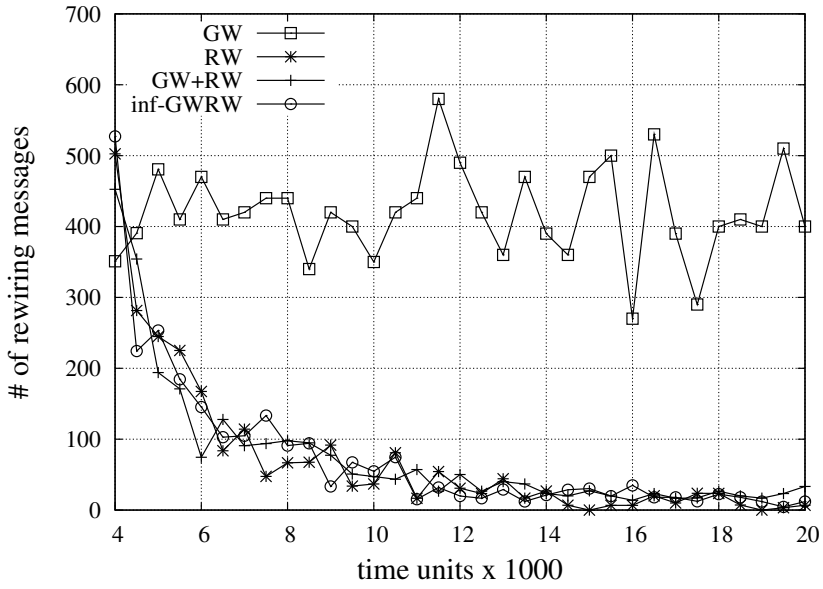


Figure 5.10: Rewiring messages for different forwarding strategies (TREC-6)

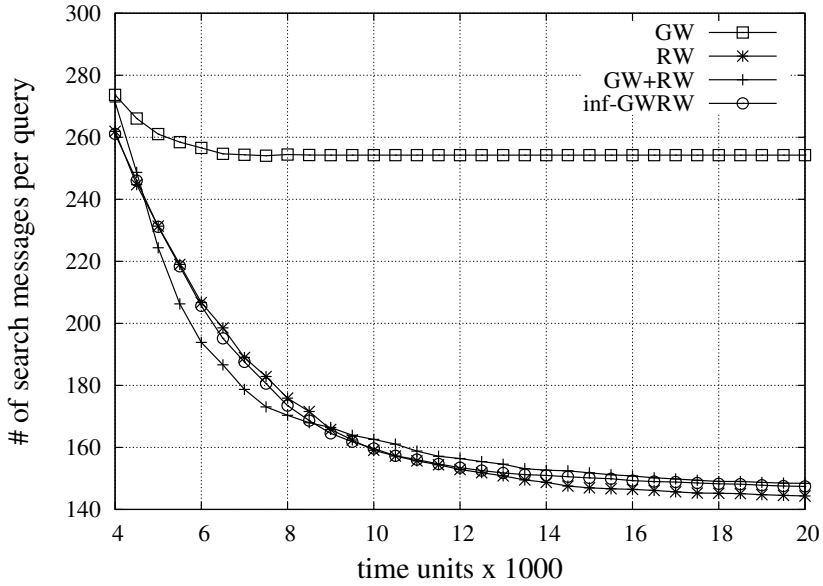


Figure 5.11: Search messages per query for different forwarding strategies (TREC-6)

(over 200%) for the RW, the GW+RW and the inf-GW+RW strategies when the network starts to get organised into clusters ($t > 8K$). Apparently, the GW strategy does not manage to reach an effective peer organisation and peers continue executing the rewiring protocol to discover peers with similar interests, which leads to high message traffic. In reverse, the other three strategies manage to efficiently and quickly organise the network and maintain an effective peer organisation at a small communication cost.

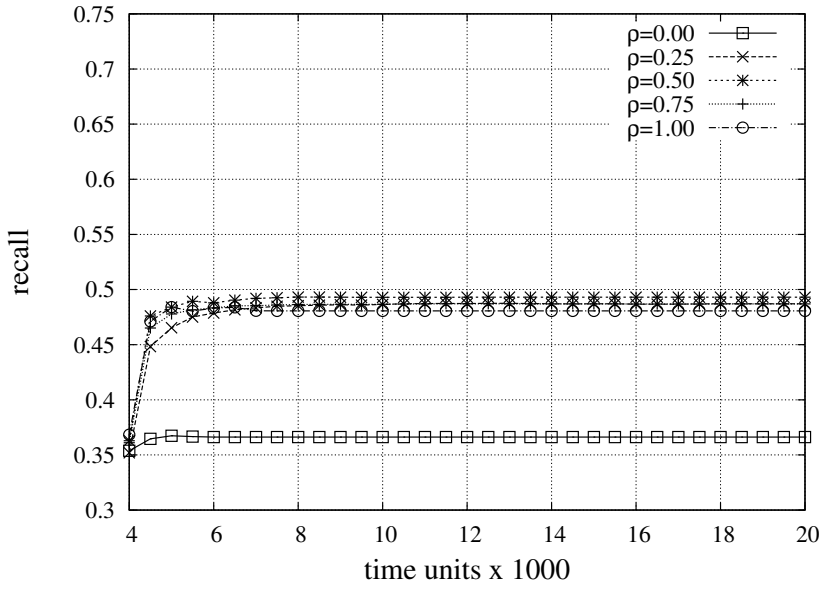


Figure 5.12: Retrieval effectiveness for different values of ϱ when using the GW strategy (TREC-6)

All strategies need high number of search messages (as shown in Figure 5.11) when the network is not yet organised into coherent neighborhoods (left-most points in the x-axis). However, this message overhead is decreased (65% decrease) for the RW, the GW+RW and the inf-GW+RW strategies as peers get organised into clusters with similar interests (right-most points in the x-axis). The GW strategy does not improve network traffic, as it does not manage to efficiently organise the network.

Notice (by looking at both Figures 5.10 and 5.11) that the inf-GW+RW and the GW+RW strategies performs slightly better in terms of overall network traffic compared to the RW strategy. We are driven to the same conclusions concerning the network traffic when using the OHSUMED corpus.

Varying the Refinement Probability

In this section, we investigate the retrieval effectiveness for various values of ϱ (i.e., the system parameter denoting the probability that a peer updates its short-range links based on the content of a forwarded rewiring message) as a function of time for two forwarding strategies: (i) GW used in our basic protocol and (ii) RW exhibiting better retrieval performance than its competitors.

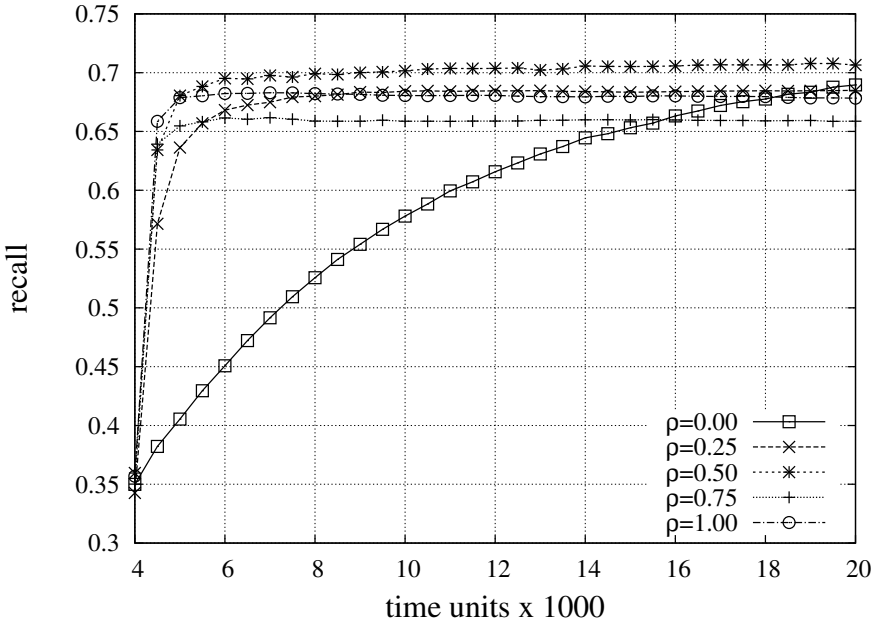


Figure 5.13: Retrieval effectiveness for different values of ϱ when using the RW strategy (TREC-6)

Figure 5.12 illustrates the way ϱ affects retrieval performance when the GW strategy is used. When the network is unorganised ($t = 4K$) the queries cannot be routed efficiently resulting in low recall (approximately 35%). When the network starts to organise ($t = 6K$), higher values of recall are achieved. The retrieval performance of the GW strategy improves 50% for $\varrho > 0$, which is attributed to the fact that more peers update their short-range links by exploiting the rewiring messages.

Figure 5.13 presents recall over time when using the RW strategy. When ϱ is low (i.e., < 0.5), the network is clustered at a slow rate since more peers need to initiate the rewiring process. As ϱ increases, the network converges faster to organised clusters of peers. When ϱ is high (i.e., > 0.5), network organisation and recall have converged to a stable state fast and remain unchanged. In some cases (e.g., ϱ=1), the relatively low values of recall imply the existence of isolated peers. The higher value of recall is achieved for $\varrho = 0.5$ (i.e., when half of the forwarding peers use FindPeers() message to update their short-range links).

In general, when $\varrho > 0$ the network converges faster to a clustered peer organisation by using less messages, and when $\varrho < 1$ peers are discouraged from creating

	OHSUMED			TREC-6		
ϱ	recall	search msgs	recall/msg (x10^{-3})	recall	search msgs	recall/msg (x10^{-3})
0.00	0.53	282	1.88	0.69	172	4.01
0.25	**0.68**	282	**2.41**	0.68	142	4.79
0.50	0.55	**232**	2.37	**0.72**	**140**	**5.14**
0.75	0.54	238	2.31	0.66	141	4.68
1.00	0.58	261	2.22	0.67	140	4.79

Table 5.2: Performance for RW when the network is organised ($t = 20K$) for both corpora

isolated clusters. The ideal value for ϱ is the one resulting in connected clusters (i.e., there is a path connecting all similar peers), while using minimum number of rewiring messages. Table 5.2 presents recall, search cost and recall/message for the RW strategy, for varying ϱ and over both corpora in an organised network ($t = 20K$). When $\varrho = 0$, RW presents the lowest recall/message for both corpora. When $\varrho = 0.50$, RW presents a good performance for both corpora (lowest message traffic, highest and second highest recall/message and a high recall value). The best value of ϱ in terms of recall is 0.25 when using OHSUMED corpus and 0.5 when using TREC-6 corpus.

The different effect of ϱ on system performance across different corpora is attributed to the different number of peers per class. Remember that increasing the value of ϱ, the probability of creating fully-connected peer clusters is also increased. Given that the number of short-range links is much smaller than the number of peers per class, in the case of OHSUMED corpus increasing the value of ϱ turns similar peers to create many isolated clusters thus, resulting in bad retrieval performance. Conclusively, the higher the number of peers per class is, the lower the value of ϱ should be. In this way, more rewiring messages may be generated, getting though an efficient peer clustering.

Figure 5.14 shows the number of messages per query for $\varrho = 0$ (i.e., the value used in the basic rewiring protocol) and 0.5 (i.e., the value that resulted in the best retrieval performance) for the GW and the RW strategies. When the network is not yet organised a higher number of search messages is needed. This message overhead is decreased (65% decrease for RW) when $\varrho = 0.5$ as peers get organised.

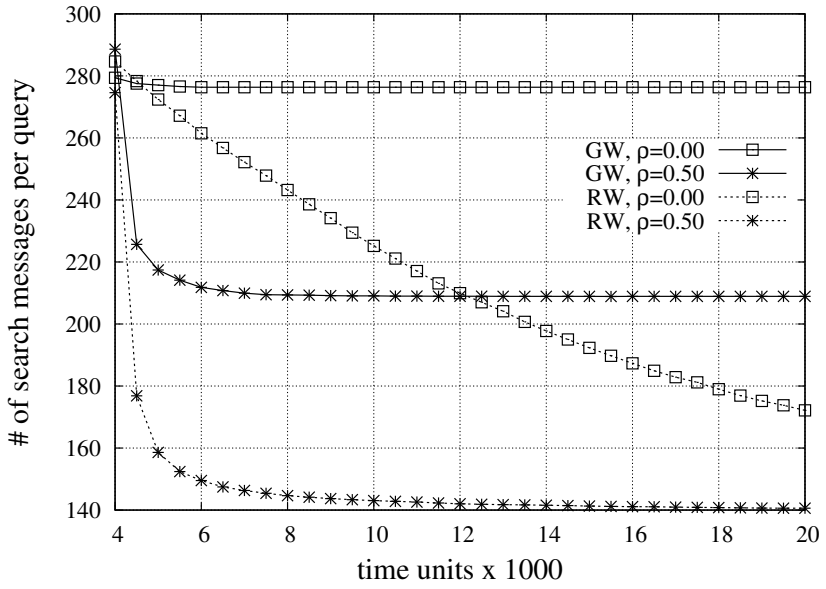


Figure 5.14: Number of search messages per query when using the GW or the RW strategy, for $\varrho = 0$ and 0.5 (TREC-6)

When $\varrho = 0$, the network converges slowly to organised peer neighborhoods and the message overhead decreases with a slower rate.

In terms of rewiring messages, the network initially presents a high message overhead, which is greatly reduced (by a factor of 15) when the network organises into coherent clusters (around moment $6K$) for $\varrho > 0$. When $\varrho = 0$, the network does not manage to quickly reach an effective peer organisation and peers keep executing the rewiring protocol presenting higher message overhead and slower decrease rate when compared to the case where $\varrho > 0$.

Symmetric Links

Figures 5.15 and 5.16 show the effect of symmetric versus non-symmetric links in the retrieval performance over time, for two different forwarding strategies and for two different values of ϱ. The data set used in this set of experiments is TREC-6. Figure 5.15 shows that the SL strategy, when used with the GW strategy, achieves better retrieval performance (about 7%) compared to the basic rewiring protocol. The effect of the SL strategy on the retrieval performance is small in the case of $\varrho = 0.50$, since some of the links introduced by SL are also created due to ϱ (forwarding peers have already updated their links to point to previous message recipients and

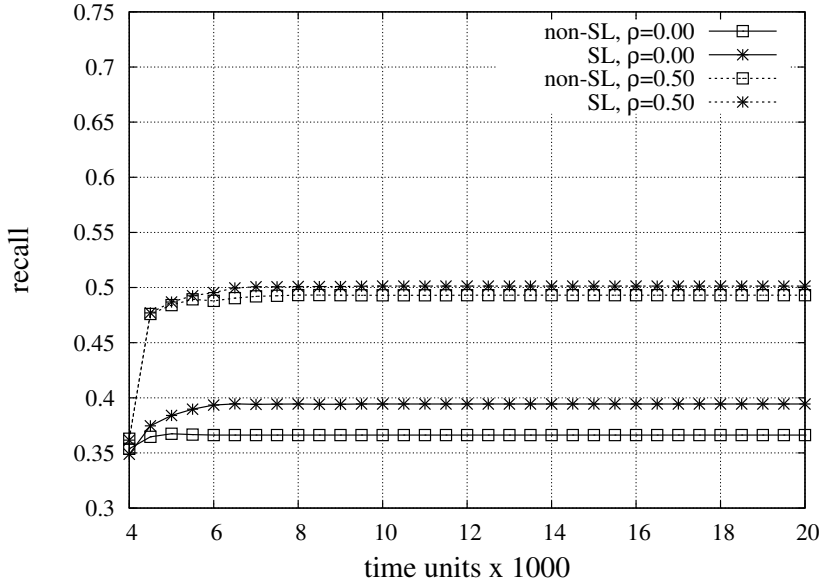


Figure 5.15: Retrieval effectiveness for SL and non-SL corresponding to $\varrho = 0$ and 0.5 when using the GW strategy (TREC-6)

	OHSUMED		TREC-6	
ϱ	non-SL	SL	non-SL	SL
0.00	0.53	0.64	0.69	0.73
0.25	0.68	0.57	0.68	0.75
0.50	0.55	0.56	0.72	0.75
0.75	0.54	0.54	0.66	0.76
1.00	0.58	0.54	0.67	0.74

Table 5.3: Recall for RW when the network is organised ($t = 20K$) for both corpora

thus, SL has no additional effect on clustering).

Figure 5.16 illustrates that the effect of creating symmetric links is higher in the case of the RW strategy, since more similar peers per rewiring process are discovered. The SL strategy improves retrieval performance by 10% for both values of ϱ and facilitates network converge towards organised peer clusters. Notice that when using the SL strategy along with $\varrho = 0.5$, the network organises faster than when using the SL strategy or $\varrho = 0.5$ alone. Consequently, creating symmetric links has a positive effect in retrieval performance.

Table 5.3 presents recall in an organised network ($t = 20K$) for both corpora and for different values of ϱ. Notice that when $\varrho > 0$, SL does not always improve recall on the OHSUMED corpus. When the OHSUMED corpus is used, there are

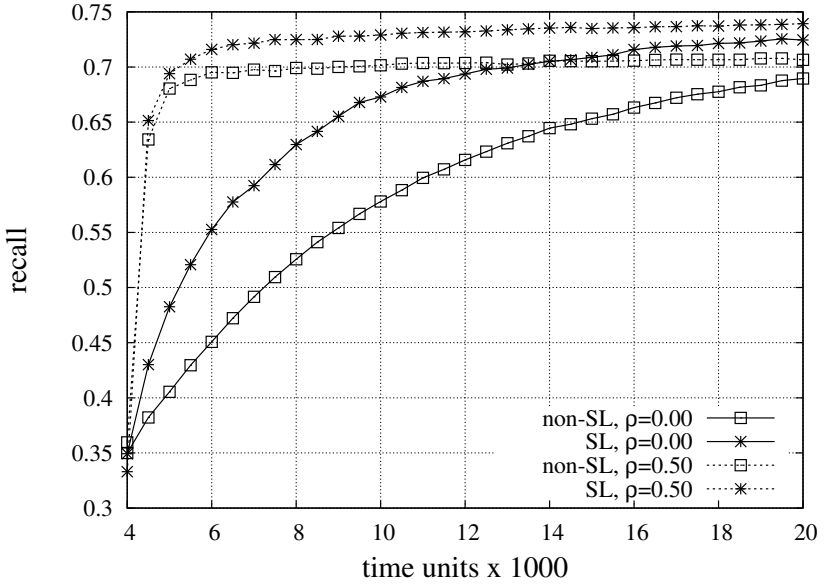


Figure 5.16: Retrieval effectiveness for SL and non-SL corresponding to $\varrho = 0$ and 0.5 when using the RW strategy (TREC-6)

many peers per category and, even though similar peers are all clustered together, the retrieval performance is also affected by other parameters, as the number of short-range links s and the broadcasting TTL τ_b.

Figure 5.17 illustrates on the number of messages per query for the RW strategy and for two values of ϱ as a function of time. When $\varrho = 0$, the SL strategy decreases network traffic by 18% when compared to the case of non-symmetric links between peers. The effect of the SL strategy in network load is smaller when $\varrho = 0.5$. By cross examining the results presented in Figures 5.16 and 5.17, we conclude that the SL strategy manages to efficiently organise peers and thus, improve retrieval effectiveness and search costs.

The SL strategy achieves better network organisation, and thus better retrieval performance, with low communication cost in terms of rewiring messages. Notice that this strategy was expected to impose extra message traffic due to the additional messages sent for supporting the symmetric links. In fact, the SL strategy decreases message traffic as it exploits existing traffic to rewire the links of the peers. In terms of rewiring, our experiments show that the SL strategy requires in general 10% less messages to organise the network and also, results in a faster clustering when compared to the non-SL strategy.

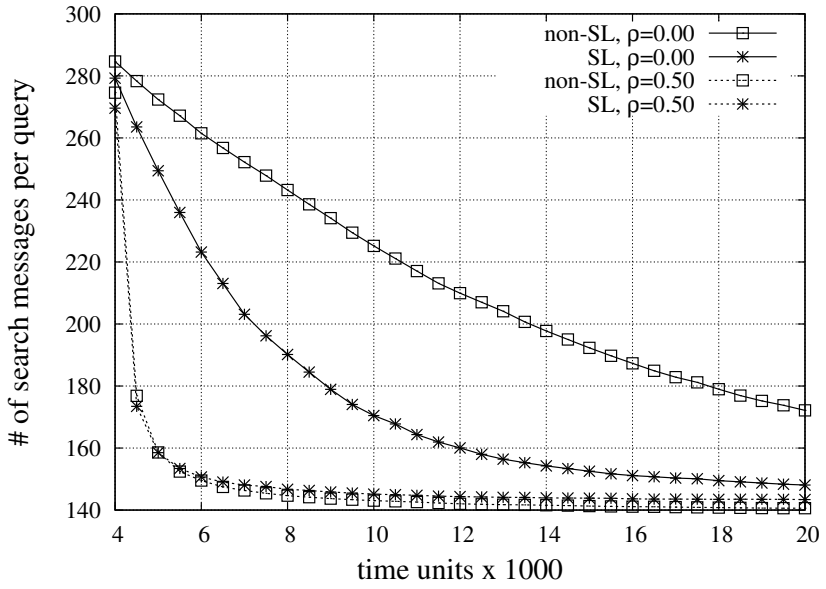


Figure 5.17: Number of search messages per query when using the RW strategy, for SL and non-SL, for $\varrho = 0$ and 0.5 (TREC-6)

Updating the Long-Range Links

The RS and the BS strategies are used for updating the long-range links. Notice that, we do not explore ϱ or SL at the same time, since they do not affect the rewiring of long-range links. The retrieval performance of RS shows an average improvement of about 5% over BS. In terms of message overhead, the RS strategy imposes around 15% more rewiring messages as it uses extra message traffic to discover peers with dissimilar interests and presents almost the same number of search messages per query compared to the BS strategy. From this set of experiments, we conclude that the update strategy of the long-range links has small impact on the performance, since both strategies perform similarly, with RS presenting slightly better retrieval but also higher message traffic during rewiring.

Composite Rewiring

In the following, we combine the best strategies identified above. The corresponding strategy is called *composite rewiring* and its performance is compared against (i) the basic rewiring protocol presented in Section 3.3.3, (ii) the clustering approach of Schmitz [Sch04] modified for a retrieval task and using the parameter values of

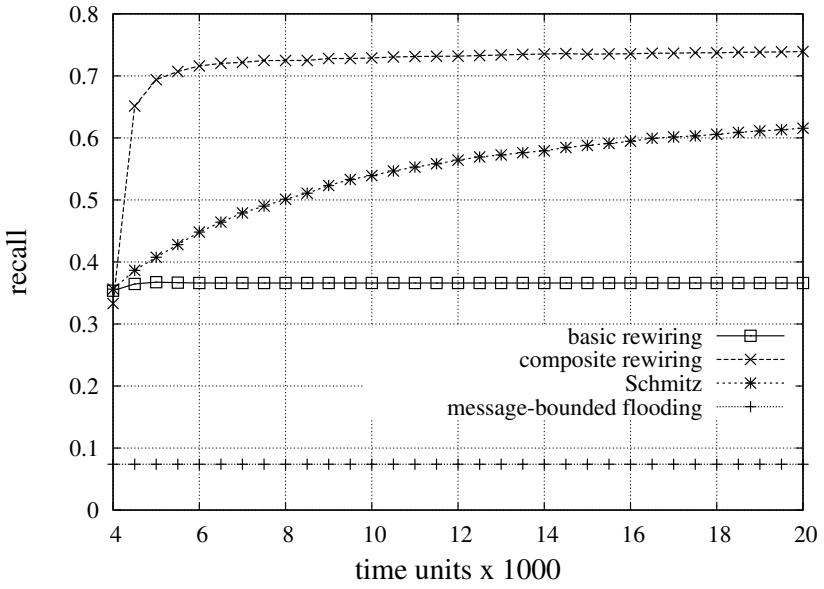


Figure 5.18: Comparison of the retrieval performance of different rewiring protocols (TREC-6)

Table 5.1 to obtain results comparable to our setting, and (iii) a message-bounded flooding algorithm, i.e. a modified flooding strategy that terminates when reaching a predetermined number of messages (in our case the number of messages of the most efficient strategy). Message-bounded flooding was implemented to serve as a baseline for the rest of the strategies.

Composite rewiring strategy uses the best strategies as identified in the previous sections: (i) comparing the different forwarding strategies, the RW strategy presented the best performance, (ii) experimenting with different values for ϱ, $\varrho = 0.5$ resulted in the best retrieval performance, (iii) showing the effect of symmetric links, the SL strategy achieved better retrieval performance with low communication cost, and (iv) experimenting on strategies for updating the long-range links, the RS strategy performed slightly better than its competitor. Therefore, composite rewiring consists of utilising the RW forwarding strategy, $\varrho = 0.5$, the SL strategy for creating short-range links and the RS strategy for updating long-range links. This combination is expected, according to the results presented earlier, to perform well in terms of recall, while imposing lower message costs than competitors.

Figures 5.18 and 5.19 illustrate the retrieval performance and communication load as a function of time for the different rewiring protocols. The data set used

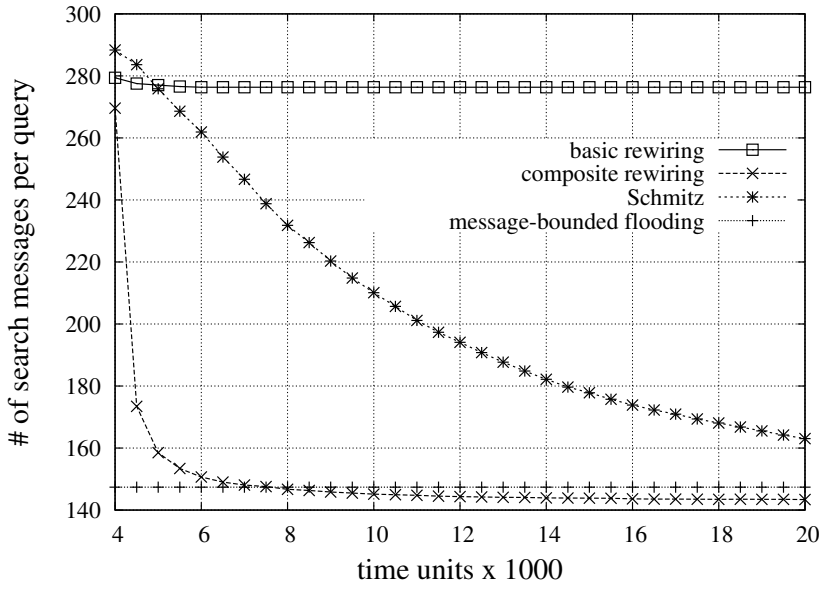


Figure 5.19: Comparison of the communication overhead (in search messages) of different rewiring protocols (TREC-6)

in this set of experiments is TREC-6. In Figure 5.18, we observe that composite rewiring achieves an 97% increase in recall when compared to the basic rewiring, and about 21% increase when compared to the approach of [Sch04]. In Figure 5.19 we observe that the composite strategy is the most efficient in terms of message traffic (remember that flooding is message-bounded). Similar results have been also obtained for the OHSUMED corpus.

Summing Up

The different rewiring components (as identified in Section 5.2) are: (i) GW, RW, GW+RW, or inf-GW+RW strategy, (ii) ϱ taking values in the interval $[0, 1]$, (iii) SL or non-SL strategy, and (iv) RS or BS strategy. There are many choices to combine making up a range of possibilities worth exploring. In this section, we explored the dependency of the retrievals on rewiring. Notice that different combinations of rewiring strategies may emphasise recall and behave slightly worse in terms of message costs, and vice versa.

When message traffic is in question, the inf-GW+RW strategy performs better, while achieving reasonably high recall values. The RW strategy emphasises recall for a small increase in message traffic (Figures 5.9 and 5.11). Setting $\varrho = 0$ leads to in-

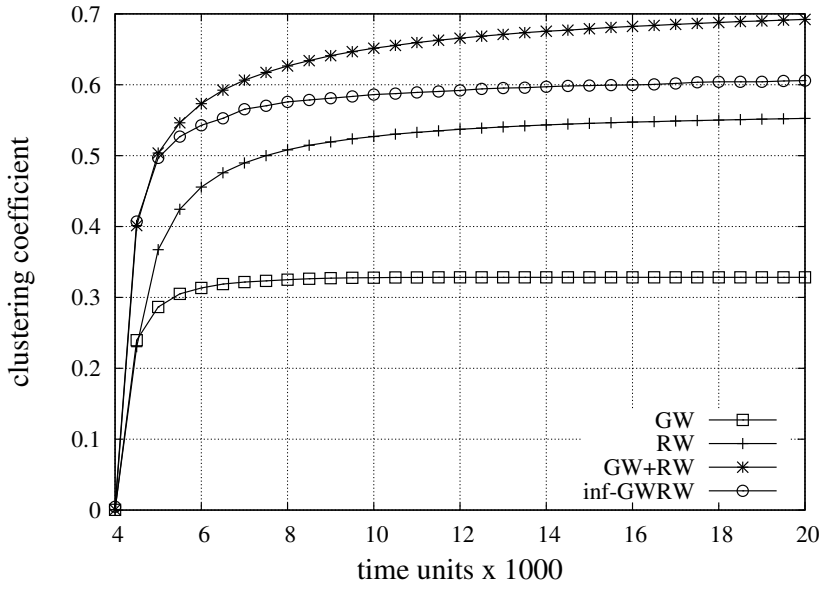


Figure 5.20: Clustering coefficient $\bar{c}$ as a function of time for different forwarding strategies

creased traffic and moderate retrieval performance, while a value of $0.25 \leq \varrho \leq 0.75$, achieves a good trade-off between traffic and retrieval effectiveness (Figures 5.12, 5.13 and 5.14). The SL strategy improves the overall system performance (Figures 5.15, 5.16 and 5.17), contrary to the updating strategy of long-range links that shows no significant performance differences.

5.3.4 Turning to Clustering Quality Measures

The purpose of this set of experiments is to provide an alternative for measuring the system performance: clustering quality measures that quantify and directly evaluate the quality of network organisation. We evaluate the effectiveness of the clustering coefficient and the clustering efficiency measures (presented in Chapter 4) under the following settings: (i) when using different forwarding strategies and (ii) when varying the refinement probability.

Using Different Forwarding Strategies

Figures 5.20 and 5.21 illustrate peer organisation as a function of time for the different forwarding strategies discussed in Section 5.2. Figure 5.20 presents the clustering

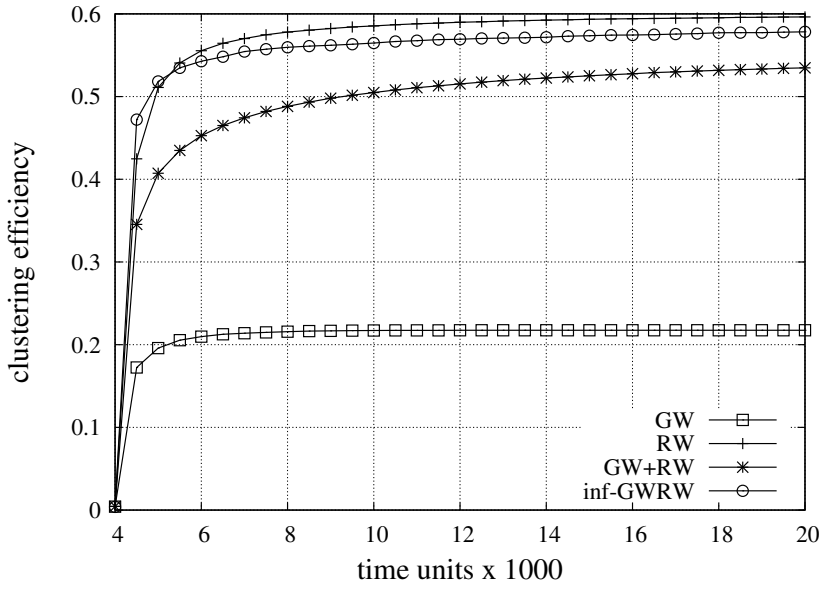


Figure 5.21: Clustering efficiency $\overline{\kappa}$ as a function of time for different forwarding strategies

coefficient measure $\overline{c}$, while Figure 5.21 presents results for the clustering efficiency $\overline{\kappa}$. At the beginning of the rewiring procedure ($t = 4K$) the values for both measures are almost 0 (for all rewiring strategies), which is attributed to the poor network organisation. When all peers have executed the rewiring protocol more than once ($t = 8K$), we observe the curves of the clustering measures to designate the effect of the different forwarding strategies on peer organisation. When the GW strategy is used, the clustering measures reach low values ($\overline{c} = 0.33$ and $\overline{\kappa} = 0.21$) and by this we can apparently allege that the network does not manage to reach an effective peer organisation. On the other hand, when one of the RW, GW+RW or inf-GW+RW strategies is used, both clustering measures achieve higher values after a small number of iterations. We can thus, turn out that these strategies manage to efficiently and quickly organise the network and maintain an effective peer organisation. Figure 5.20 shows that the clustering coefficient reaches its highest value ($\overline{c} = 0.69$) when the GW+RW strategy is used, driving us to the conclusion that the GW+RW strategy is the best choice for organising our network. However, the results presented in Figure 5.21 points out that the RW strategy is the best strategy for organising the peers into clusters, since by using the RW strategy clustering efficiency reaches its highest value ($\overline{\kappa} = 0.6$).

	$\overline{c}$	$\overline{gc}$	$\overline{\kappa}$
GW	0.33	0.48	0.22
RW	0.58	0.49	**0.60**
GW+RW	**0.69**	**0.60**	0.52
inf-GW+RW	0.61	0.58	0.57

Table 5.4: The values of clustering coefficient $\overline{c}$, generalised clustering coefficient $\overline{gc}$ and clustering efficiency $\overline{\kappa}$ in an organised network when using different forwarding strategies

We have also used the generalised clustering coefficient $\overline{gc}$ to evaluate the different forwarding strategies. Table 5.4 presents the highest values achieved ($t = 20K$) for each one of the clustering measures when using different forwarding strategies. Since the different clustering measures give contradictory results, we associate the performance of retrievals with the quality of clustering, in order to recommend the clustering measure that better expresses the quality of network organisation. We expect that the higher the clustering quality of the network is, the better the retrieval performance should be.

The retrieval effectiveness of the network as a function of time for the different forwarding strategies was illustrated in Figure 5.9. Remember that the GW strategy improves recall only by 12%. The three other strategies present much higher values of recall as the network gets organised: the RW strategy improves retrieval by 92% and achieves recall 0.63, the GW+RW strategy improves retrieval by 75% and achieves recall 0.56, and the inf-GW+RW improves retrieval by 85% and achieves recall 0.59.

Only the results obtained when using the clustering efficiency $\overline{\kappa}$ measure (Table 5.4) ranked the strategies in an order that corresponds to their retrieval performance and pointed out the RW strategy as the best strategy for organising the network. We can thus, make out that clustering efficiency is a good measure to evaluate network organisation, providing us with straightforward results concerning the underlying network organisation: the highest the value of the clustering efficiency is, the better the underlying network organisation is (in terms of retrieval effectiveness).

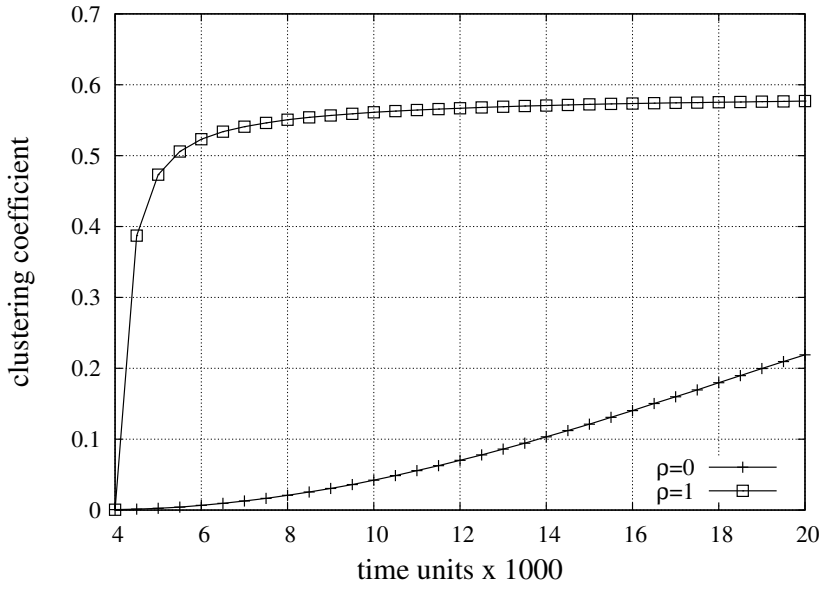


Figure 5.22: Clustering coefficient $\overline{c}$ as a function of time for different values of ϱ

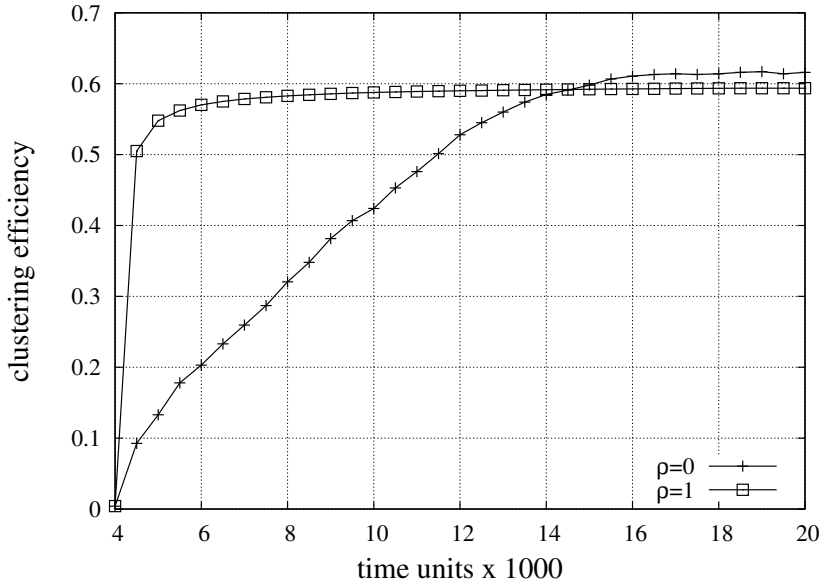


Figure 5.23: Clustering efficiency $\overline{\kappa}$ as a function of time for different values of ϱ

Varying the Refinement Probability

Figures 5.22 and 5.23 illustrate an evaluation of network organisation over time when varying the refinement probability ϱ. In this set of experiments, we used the RW strategy for the message forwarding, varied the values of ϱ and observed the way ϱ affects network organisation. Figure 5.22 presents clustering coefficient $\overline{c}$ for $\varrho = 0$ (i.e., none of the forwarding peers uses the FindPeers() message

ϱ	$\overline{c}$	$\overline{gc}$	$\overline{\kappa}$
0	0.22	**0.53**	**0.62**
1	**0.58**	0.49	0.59

Table 5.5: The values of clustering coefficient $\overline{c}$, generalised clustering coefficient $\overline{gc}$ and clustering efficiency $\overline{\kappa}$ in an organised network when using different values for ϱ

to update its short-range links) and $\varrho = 1$ (i.e., all of the forwarding peers use the FindPeers() message to update their short-range links). Initially, clustering coefficient is almost 0 for both values of ϱ, which indicates that the network is not yet organised into clusters of similar peers. Clustering coefficient increases as peers get organised ($t > 6K$). The results indicate that when $\varrho = 1$ the clustering coefficient is much higher ($\overline{c} = 0.58$) than in the case that $\varrho = 0$ ($\overline{c} = 0.22$), implying that a better network organisation is achieved in the case that $\varrho = 1$.

Figure 5.23 shows clustering efficiency $\overline{\kappa}$ as a function of time for $\varrho = 0$ and $\varrho = 1$ when using the RW strategy. Initially, clustering efficiency is almost 0 for both values of ϱ, which is attributed to poor network organisation. Clustering efficiency increases as peers get organised ($t > 8K$). The results presented in the figure illustrate that clustering efficiency is higher in the case that $\varrho = 0$ ($\overline{\kappa} = 0.62$), even though the rate of increase is slower compared to the case that $\varrho = 1$ ($\overline{\kappa} = 0.59$). Notice that in an organised network the value of $\overline{\kappa}$ achieved when $\varrho = 0$ has not great difference compared to the value of $\overline{\kappa}$ when $\varrho = 1$.

Table 5.5 presents the highest values achieved ($t = 20K$) for each one of the clustering measures when using different values of ϱ. The different clustering measures indicate contradictory results concerning the values of the rewiring probability parameter ϱ. Clustering coefficient indicates that the network is much better organised when $\varrho = 1$, while clustering efficiency and generalised clustering coefficient indicate $\varrho = 0$ as the best value and similar network organisations for both values of ϱ. We again use the performance of retrievals to opine on the quality of network organisation and decide which clustering measure is more precise in expressing the network organisation quality.

Retrieval effectiveness for both values of ϱ as a function of time was illustrated in Figures 5.12 and 5.13. Remember that when the network is unorganised ($t = 4K$)

the queries cannot be routed efficiently resulting in low recall (around 35%). When the network starts to organise into cohesive clusters ($t = 6K$), higher values of recall are achieved for both values of ϱ: when $\varrho = 0$ the recall achieved (69%) is slightly better than in the case that $\varrho = 1$ (65%).

The results of retrieval effectiveness object to the results obtained when using the clustering coefficient measure to evaluate the network organisation (Figure 5.22), coincide though to the results obtained when using the clustering efficiency measure (Figure 5.23). The retrieval performance points out the same value of ϱ for organising the network highlighted also by the clustering efficiency and generalised clustering coefficient measures, and indicates that both network organisations (either with $\varrho = 0$ or with $\varrho = 1$) have similar retrieval effectiveness. We conclude that clustering efficiency measure is an effectual measure for evaluating network organisation. Clustering efficiency provides insights on the underlying network organisation and reflects better the retrieval effectiveness of the network. For the rest of the thesis recall and clustering efficiency will be both used to evaluate a rewiring strategy.

5.4 Conclusions

In this chapter, we presented a comprehensive study of both existing and innovative strategies for rewiring. We showed how peer organisation is affected by the different design choices of the rewiring mechanism and how these choices affect the performance of the system overall (both in terms of communication overhead and retrieval effectiveness). The different rewiring components are: (i) GW, RW, GW+RW, or inf-GW+RW strategy, (ii) ϱ taking values in the interval $[0, 1]$, (iii) SL or non-SL strategy, and (iv) RS or BS strategy. The different choices for each rewiring component provide us with many choices to combine making up a range of possibilities worth exploring when designing peer rewiring.

We evaluated the proposed rewiring protocols using two real-world datasets with web and medical documents. Our experimental evaluation with real-word data and queries confirms the dependence between rewiring strategies and retrieval performance, and gives insights on the trade-offs involved in the selection of a rewiring

strategy. Different combinations of rewiring strategies may emphasise recall and behave slightly worse in terms of message costs. Subsequently, we identified a combination of components that outperforms existing rewiring protocols. We also, used clustering measures as an alternative for evaluating the system performance. Our results indicate that the clustering efficiency measure expresses network clustering quality and reflects retrieval effectiveness.

Chapter 6

Peer Churn

In this chapter, we evaluate i*Cluster* protocols under churn. Section 6.1 introduces peer churn, presents our contributions and discusses related work in the area. Section 6.2 presents a model of user behaviour and assumptions on peer arrivals and departures. A critical evaluation of the performance of i*Cluster* in such a dynamic setting is illustrated in Section 6.3, while Section 6.4 concludes the chapter.

6.1 Introduction

In the previous chapters we presented i*Cluster*, an architecture that is able to support full-fledged information retrieval in large-scale P2P networks. We provided a better understanding of the functional issues related to the design and the performance of the rewiring protocols by answering fundamental questions, presented a comprehensive study of both existing and innovative strategies for rewiring protocols and showed how performance (in terms of retrieval effectiveness and communication overhead) depends on each component of the rewiring protocol. Subsequently, we identified a combination of components that outperforms existing rewiring protocols. We evaluated the proposed rewiring protocols using two real-world datasets with web and medical documents.

Similarly to most research proposals dealing with information retrieval over semantic overlay networks, i*Cluster* assumed an ideal scenario where peers never leave or join the network. However, in real-world applications a peer joins the network

when a user starts the application. While being connected, the user can contribute its resources to the network or search for resources provided by other users. The peer leaves the system when the user exits the application. Stutzbach and Rejaie [SR06] define such a join-participate-leave cycle as a *session*. The independent arrival and departure of peers creates the collective effect called *churn*[1]. These user driven dynamics of peer participation must be taken into account in both the design and the evaluation of P2P applications, since the distribution of session length can affect the overlay structure [SRS05], the resiliency of the overlay [LYRL07] and the selection of key design parameters [LSK+05].

Modelling churn requires fine-grained and unbiased information about the arrival and departure of peers in a network. This task is rather challenging due to the large size and highly dynamic nature of P2P systems. Several measurement studies (e.g., [GDS+03]) present a high level view of churn by analysing its characteristics (e.g., median session length) in large scale P2P systems. All these studies result in a distribution modelling session lengths. However, findings vary from Poisson to heavy-tailed distributions. Stutzbach and Rejaie [SR06] identify the key challenges in characterising churn, determine common pitfalls in measuring churn, such as biased peer selection, which they believe are the main factors for the conflicting results, and develop techniques to address these difficulties. Leonard et al. [LYRL07] present a realistic model for peer lifetimes in P2P networks and investigate the resilience of random graphs to lifetime-based peer failure. Yao et al. [YLWL06] introduce a generic model that captures the heterogenous behaviour of peers. They consider each peer as an alternating renewal process and consider that on-line/off-line durations are independent and unique for each peer.

In this chapter, we consider a realistic dynamic scenario where peers join, participate, leave and re-join the network. i*Cluster* is used to organise the network, route queries and retrieve data to the interested users under such a dynamic setting. We study the way churn affects system performance and identify the rewiring protocol that performs efficiently (in terms of retrieval accuracy) under churn.

[1]The naming is based on the English idiom *to churn up*, meaning to agitate or produce motion.

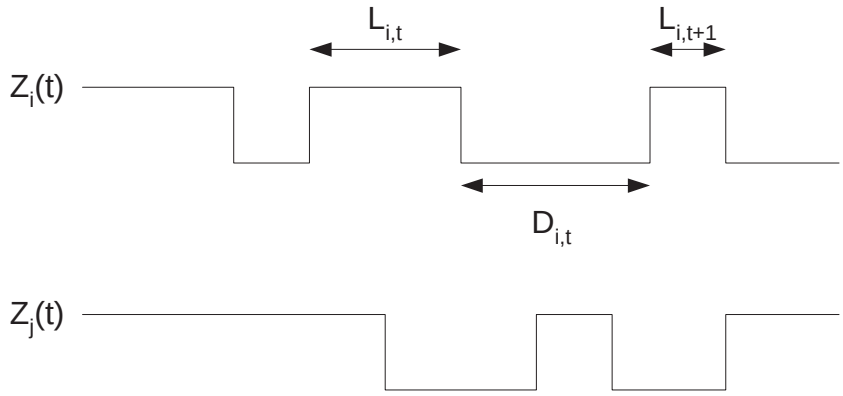


Figure 6.1: On-line and off-line behaviour for peers p_i and p_j

6.2 Modeling Peer Dynamics

We present the model of peer behaviour and specify assumptions on peer arrivals and departures based on the work in [YLWL06, LYRL07, SR06]. The model takes into account heterogeneous browsing habits, formalises recurring user participation in P2P systems and explains the relationship between the various lifetime distributions observable in P2P networks.

Churn Model. We consider a P2P network with N participating peers. Each peer p_i is either *alive* (i.e., present in the system) at a specific moment or *dead* (i.e., logged-off). This behaviour is modelled by a renewal process $\{Z_i(t)\}$ for each peer p_i as in [YLWL06]:

$$Z_i(t) = \begin{cases} 1, & p_i \text{ is alive at time } t; \\ 0, & \text{otherwise.} \end{cases}, \quad 0 \leq i \leq N \tag{6.1}$$

This framework is illustrated in Figure 6.1 that presents the renewal processes $\{Z_i(t)\}$ and $\{Z_j(t)\}$ for peers p_i and p_j respectively. In the figure, t stands for the time period and the random variables $L_{i,t} > 0$ and $D_{i,t} > 0$ represent respectively the on-line and off-line durations for peer p_i.

We next discuss the assumptions concerning this churn model:

1. To capture the independent nature of peers in a peer-to-peer system, we assume that peers behave independently of each other and processes $\{Z_i(t)\}$ and

$\{Z_j(t)\}$ for any $i \neq j$ are independent. This means that peers do not synchronise their arrivals or departures and generally exhibit uncorrelated lifetime characteristics.

2. Although the model is generic enough to allow dependencies between cycle lengths, without loss of generality we treat all lifetime[2] and off-time processes as independent and use identical distributed (iid) sets of variables. Thus, for each process $\{Z_i(t)\}$ its on-line durations $\{L_{i,t}\}$ are described by some joint distribution $F_i(x)$ and its off-line durations $\{D_{i,t}\}$ are described by another joint distribution $G_i(x)$. This means that for each peer its on-line and off-line durations are independent.

Lifetime Distribution. The on-line distribution commonly considered in the literature [YLWL06, LYRL07] is the heavy-tail (i.e., Pareto) distribution. The same consideration holds for the off-line distribution. Pareto distribution was originally used to describe the allocation of wealth among individuals since it seemed to show rather well the way that a larger portion of the wealth of any society is owned by a smaller percentage of the people in that society. This idea is sometimes expressed more simply as the Pareto principle or the "80-20 rule" which says that 20% of the population controls the 80% of the wealth [Ree01]. This idea is consistent with real life peer-to-peer scenarios, in which the majority of the users have very small lifetimes, while a handful of users stay connected for weeks contributing to the long tails of the distribution. To allow for arbitrarily small lifetimes, a Pareto distribution [YLWL06] is used to represent the on-line durations:

$$F(x) = 1 - (x/x_m)^{-k}, \;\; x_m > 0, \;\; k > 1 \tag{6.2}$$

with mean equal to $(kx_m)/(k-1)$. Off-line durations are respectively represented by an alike Pareto distribution $G(x)$. Pareto distribution is parameterised by the quantities x_m and k, which respectively stand for the *scale* and the *shape* of the distribution. The scale parameter sets the position of the left edge of the probability density. The shape parameter determines the skewness of the distribution.

[2]Notice that the on-line process of a peer is also called peer lifetime.

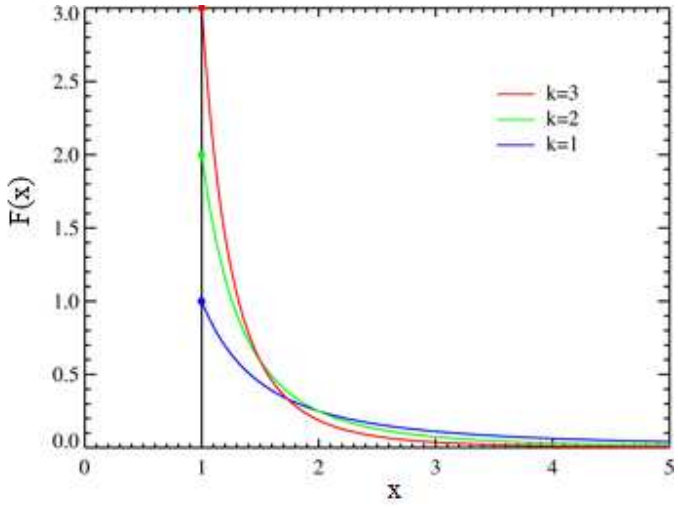


Figure 6.2: Pareto probability density functions for $x_m = 1$ and various values of k

According to the above definition, if x is Pareto distributed the probability density function (pdf) is:

$$f(x) = \begin{cases} (kx_m^k)/(x^{k+1}) & x > x_m \\ 0 & x < x_m \end{cases} \tag{6.3}$$

Figure 6.2 presents the probability density functions for $x_m = 1$ and various values of k.

Peer Availability. For convenience of notation, Yao et al. [YLWL06] define the average on-line duration (or lifetime) of a peer p_i as $l_i = E[L_i]$ and its average off-line duration as $d_i = E[D_i]$. The *availability* a_i of peer p_i, i.e. the probability that p_i is in the system at a random moment, is calculated [SGG02] as:

$$a_i = \lim_{t \to \infty} P(Z_i(t) = 1) = \frac{l_i}{l_i + d_i}. \tag{6.4}$$

According to the above presented model, the only parameters that control a peer's availability are the on-line l_i and the off-line d_i durations. Note that these parameters are *independent* and *unique* for each peer. Parameters l_i and d_i are drawn independently from two Pareto distributions. Once pair (l_i, d_i) is generated for each peer p_i, it remains constant for the entire evolution of the system.

Generating Pareto Numbers. Gray et al. indicate the way for generating numbers that follow the Pareto distribution in [GSE+00]. Based on this work, we use the following function to generate on-line l_i and off-line d_i durations for all N peers:

```
double pareto(long N, double h) {
  n1 = log(h)/log(1.0 − h)
  n2 = pow(randf(), n1)
  return ((1 + N ∗ n2)/N)
}
```

Parameter h takes values in the range $[0, 1)$ and determines the *skewness* of the distribution. This function generates numbers in the interval $(0, 1]$, which stand for the on-line or off-line session lengths. The idea is that the first $h \times N$ peers get $1-h$ of the distribution. For example, if $N = 25$ and $h = 0.20$, then $(1 - 0.20) \times 100\% = 80\%$ of the distribution goes to the first $0.20 \times 25 = 5$ peers and $(1 - 0.20) \times 80\% = 64\%$ of the distribution goes to the first peer (i.e., 0.20×5). It follows that the lower the value of the parameter h is, the more steep the distribution will be contributing to the long-tail of the distribution. When generating on-line durations, a skew distribution is interpreted to small peer lifetimes.

6.3 Experimental Evaluation

In the following, we evaluate i*Cluster* under realistic peer churn scenario using a real-life corpus with web data. In a static network (i.e., with peers never leaving the network), the rewiring task finally converges to a stable network organisation. In the case of a dynamic network, as peers leave and join the network eventually, the network becomes de-organised. The independent arrival and departure of peers affect the number of on-line peers, which in turn is reflected to the overlay network structure. Yet retrieval effectiveness is shown to be bad in a randomly organised network. Thus, the requirement for a rewiring mechanism is to be resilient to peer churn (i.e., capable of maintaining the network organised at a low cost).

In what follows, we consider two different on-line session length distributions (by giving different values to parameter h) that correspond to an easier (i.e., on average 85% of the peers are on-line at every moment) and a more difficult (i.e., on average

75% of the peers are on-line at every moment) scenario. The experiments that follow aim at evaluate the resiliency of i*Cluster* under peer churn.

6.3.1 Performance Measures

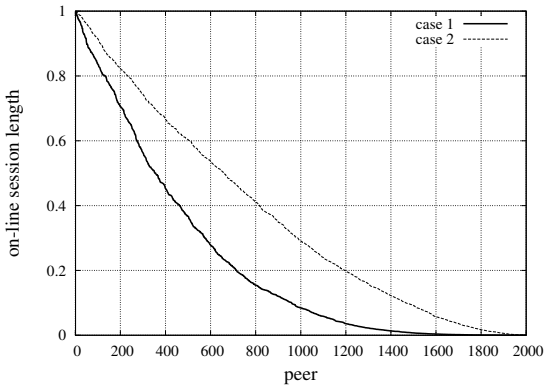
To describe the effect of peer clustering, we use *clustering efficiency* $\overline{\kappa}$ as presented in Chapter 4. Clustering efficiency takes values in the interval $[0, 1]$. The higher the value of the clustering efficiency is the better the organisation of the network is considered. The *network traffic* is measured by recording the number of messages exchanged by the peers during rewiring or querying. The retrieval effectiveness is evaluated using *recall* as the percentage of qualifying answers retrieved with respect to the total number of qualifying answers available this time in the network. Notice that precision is 100% in our approach, since only relevant documents are retrieved.

6.3.2 Experimental Testbed

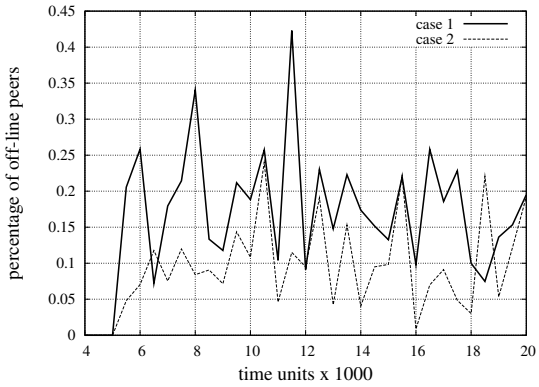
Dataset. The dataset we use contains over 556,000 documents from the TREC-6[3] collection clustered in 100 categories. This is the dataset also used to evaluate i*Cluster* and all the rewiring protocols in the previous chapters. The issued queries are strong representatives of document categories (i.e., the topics of the categories).

Setup. We consider a P2P network with N participating peers. Each peer is specialised in one category. At the bootstrapping, peers join the network and are connected as described in Section 3.3.1. The base unit for time used in the experiments is the period t. The start of the rewiring procedure for each peer is randomly chosen from the interval $[0, 4K \cdot t]$ and its periodicity is randomly selected from a normal distribution of $2K \cdot t$, in the spirit of [Sch04]. For each peer p_i $(1 \leq i \leq N)$, a pair of parameters l_i and d_i is drawn independently from two Pareto distributions. We generate these parameters according to the function presented in Section 6.2. We consider a small convergence period where all peers participating in the network are on-line $(t \leq 5K)$, then peers independently leave and join the network and we explore the way peer churn affects the system performance.

[3]http://boston.lti.cs.cmu.edu/callan/Data/



(a) Different distributions of on-line session lengths



(b) Percentage of off-line peers as a function of time

Figure 6.3: Lifetime distributions

Each peer periodically (e.g., when connecting to the network, or when its interests have changed) tries to find better neighbours by initiating the rewiring procedure. Since composite rewiring has proven efficient in terms of recall and message traffic when evaluated in a static network (Section 5.3.3), we use this strategy as the underlying protocol for organising the peers under this dynamic setting. Remember that the components of composite rewiring are the RW forwarding strategy, $\varrho = 0.5$, the SL strategy for creating short-range links and the RS strategy for updating long-range links. Query processing is carried out as described in Section 3.3.4.

The size of the network is 2,000 peers. The baseline parameter values used for this set of experiments are $\theta = 0.9$, $\tau_R = 4$, $\tau_f = 6$ and $\tau_b = 2$. Each peer stores 12 links to other peers, from which 33% are long-range links.

The simulator used to evaluate i*Cluster* under churn was implemented in C/C++ and all experiments were run on a Linux machine. Our results were averaged over 25 runs (5 random initial network topologies and 5 runs for each topology).

Lifetime Distributions. We experimented with different values of the parameter h that designates the skewness of the distribution. Figure 6.3(a) presents two different on-line session length distributions. By definition, the most skewed the distribution is, the smaller the lifetimes of the most peers are. The first case in Figure 6.3(a) corresponds to a difficult scenario compared to the second case, since peers are on-line for shorter time periods and leave the network more often.

Figure 6.3(b) illustrates the percentage of the peers that are off-line at any in-

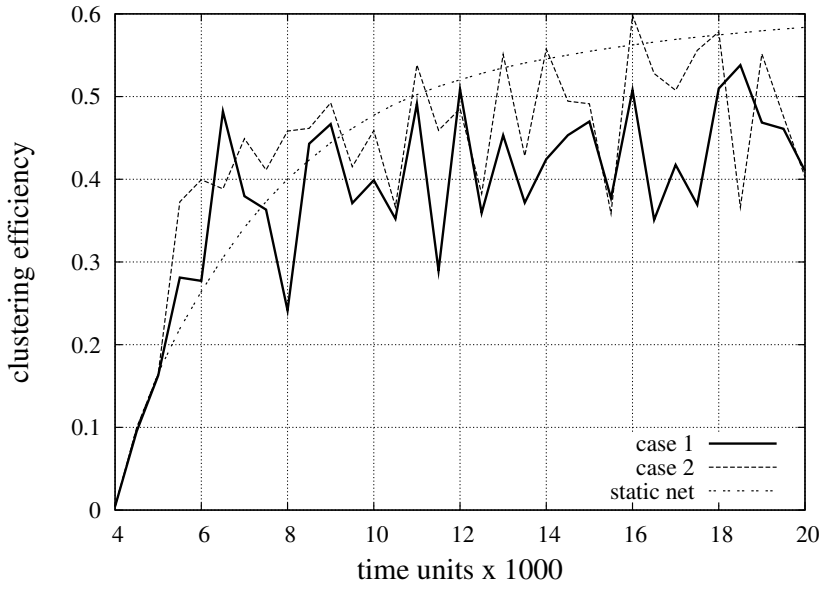


Figure 6.4: Clustering efficiency as a function of time

stance of time. In the first scenario the percentage of the peers that are logged-off reaches up to 42% (for $t = 11.5K$), while in the second scenario the percentage of the peers that are logged-off every moment is kept under 25%.

6.3.3 Evaluation Results

Rewiring Effectiveness

Figure 6.4 presents clustering efficiency as a measure of network organisation over time. The plots presented in the figure concern the two dynamic scenarios discussed in the previous section and a static network (i.e., when peers are continuously connected to the network) used as the baseline for our evaluation. We observe that during the initial convergence period ($t \leq 5K$) peers self-organise into clusters improving the clustering efficiency. In the case of the static network, rewiring finally converges towards a stable network organisation and a constant value of clustering efficiency. In the case of the dynamic network, peers arbitrarily leave and join the system and by this, links pointing to off-line peers as well as newly-connected peers emerge. As a result, peer connections continuously change and peers try (by rewiring) to recover network organisation, a situation continuously repeated.

As shown in Figure 6.4, the peer organisation protocol manages to organise the

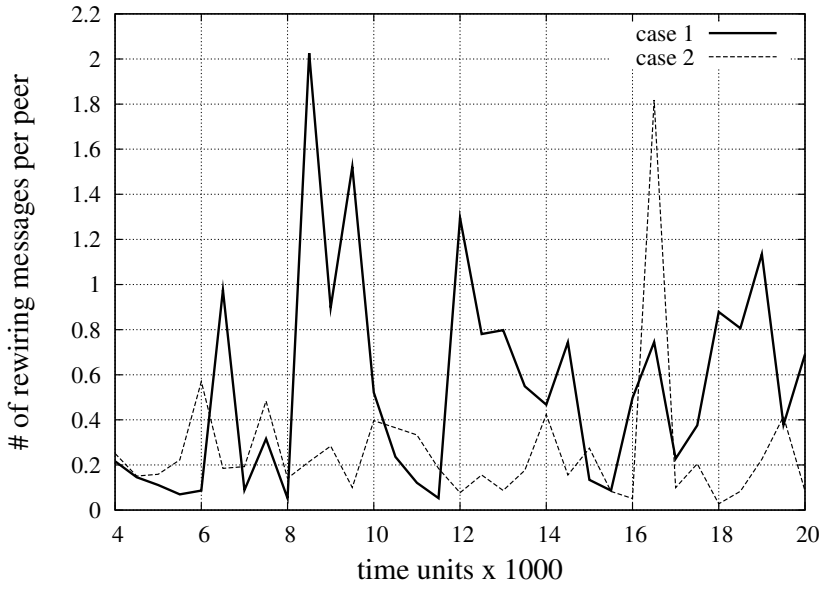


Figure 6.5: Rewiring messages as a function of time

network and clustering efficiency is eventually kept high compared to the unorganised network ($t = 4K$). Naturally, the network looses its clustering cohesion at moments of high churn, as for example in the first case when $t = 8K$ and 35% of the peers are off-line, but the network manages to quickly recover in all cases. We are thus, driven to the conclusion that i*Cluster* manages to quickly reorganise the network and keep the values of clustering efficiency high. Notice, by collating Figures 6.4 and 6.3(b), that clustering efficiency is narrowly related to the percentage of peers that are off-line; when the percentage of off-line peers increases clustering efficiency decreases, and vice versa.

In terms of rewiring messages, peer churn imposes a continuous need for peer organisation, which in turn leads to continual message traffic. Figure 6.5 presents the number of rewiring messages as a function of time. Notice that the most dynamic the network is, the most peers need to rewire their links and the more the number of the messages that traverse the network are needed. Figure 6.5 shows that 3 times more rewiring messages are sent on average in the first scenario (where off-line session lengths are larger) compared to the second scenario that is considered easier. Compared to a static network, the rewiring task needs on average 2 times more and on the worst case 7 times more messages when operating under a dynamic network. As a conclusion, i*Cluster* manages to keep the network organised at the

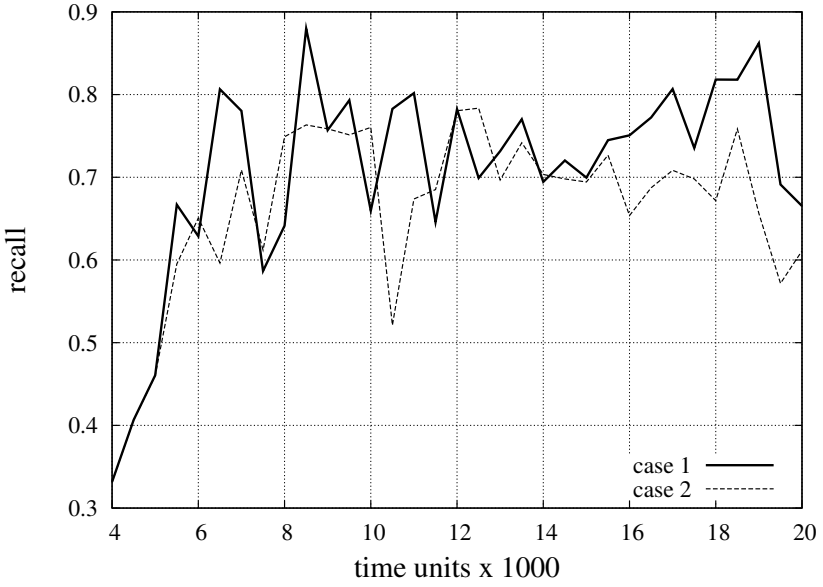


Figure 6.6: Retrieval efficiency as a function of time

exchange of high network traffic.

Retrieval Effectiveness

Figure 6.6 presents the performance of retrievals under churn as a function of time. In this figure, we observe that recall attains high values throughout the entire evolution of the system. In particular, during the initial convergence period ($t \leq 5K$), the peers self-organise into clusters and the retrieval performance is almost doubled compared to the initial unorganised network. After this point, peers leave and rejoin the network. Because of this, links pointing to off-line peers or to non-organised peers, as well as newly joined peers emerge. The arbitrarily connected peers naturally cause troubles to the retrieval performance of the network. However, Figure 6.6 shows that recall is always kept high (in the the worst case of the second scenario ($t = 10.5K$) recall is 55% better compared to the recall on an unorganised network). This remark coincides with the results presented in Figure 6.4 inducting thus, that the network recovers quickly and that i*Cluster* proves to be resilient to peer churn.

Figure 6.7 shows the number of messages per query over time for the two different scenarios. When the network is not organised ($t = 4K$), a high number of search messages are needed to retrieve the available data relevant to a query. However, this

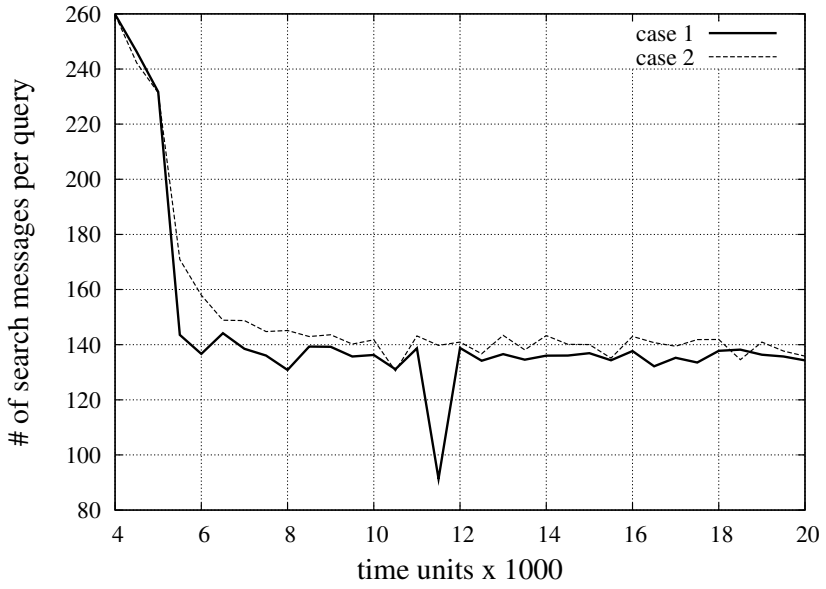


Figure 6.7: Search messages as a function of time

message overhead is decreased (more that 85%) as peers continuously get organised into clusters with similar interests. Notice though, that the number of search messages is kept low throughout the evolution of the system, regardless the continuous changes in network structure. This happens because the number of search messages is related to the number of neighbouring peers (see Section 3.3.4), that under a dynamic scenario is interpreted to the number of *available* neighbouring peers. In cases of high churn, a noticeable number of peers have gone off-line, the network size has been reduced and the overlay has lost its structure. The rewiring mechanism reinstates the network organisation, but the number of search messages is kept low since many peers are off-line and the neighborhoods are sparse populated. To show proof of concept, notice that the lower number of messages is achieved in the first scenario when the percentage of off-line peers is the higher perceived ($t = 11.5K$).

Summing Up

Summarising, the overlay structure is directly related to peer churn (Figure 6.3): the more dynamic the network is, the more often peers connect to and disconnect from the network affecting the organisation of the network. Figure 6.4 shows that i*Cluster* tries and eventually manages to retain a clustered organisation of peers by rewiring. However, Figure 6.5 indicates that this organisation does not come for free. Network

organisation affects in turn retrieval performance. i*Cluster* manages to retain high values of recall and low values of search messages per query (Figures 6.6 and 6.7) by keeping the network organised even under high peer churn.

6.4 Conclusions

We presented a model that describes peer behaviour and specified assumptions on peer arrivals and departures. Peer churn affects directly the overlay network. In turn, the arbitrarily changing overlay affects both the network organisation and the content availability. Under such a dynamic setting, i*Cluster* was utilised to organise the network and retrieve data to the interested users. To the best of our knowledge, this is first study concerning the way rewiring is affected by and deals with the dynamic behaviour of the peers.

We evaluated i*Cluster* using a real-life corpus with web data and studied the way peer churn affects the system performance. Since composite rewiring has proven efficient in terms of recall and message traffic when evaluated in a static network (Chapter 5), we used this strategy as the underlying protocol for organising the peers under such a dynamic setting. Composite strategy proved efficient in dynamic settings and i*Cluster* managed to retain a clustered organisation of peers by rewiring. Specifically, i*Cluster* exchanged increased rewiring network traffic for high retrieval performance (in terms of recall and search messages).

Chapter 7

Digital Library Use Case

In this chapter, we introduce i*ClusterDL*, a novel P2P architecture supporting self-organising digital libraries. Section 7.1 introduces the main characteristics of this digital library use case and discusses some related work in this area. A high-level view of i*ClusterDL* architecture including the involved components is presented in Section 7.2, whereas Section 7.3 explains in detail the protocols for providing retrieval and filtering functionality. A critical evaluation of the performance of i*ClusterDL* is presented in Section 7.4, followed by the conclusions in Section 7.5.

7.1 Introduction

Currently, document collections are fragmented across different Digital Libraries (DLs) due to copyright issues that prevent the owners to share their documents. To deal with this issue, a number of P2P architectures, that allow users to transparently search these data collections, have emerged as a natural solution to the problem of search in distributed digital libraries. Most of these architectures use structured overlay networks[1] (e.g., [ZTW07, TIK05, SCL+05]) as the underlying routing infrastructure. In this chapter, we introduce i*ClusterDL*, a novel P2P architecture supporting self-organising DLs that demonstrates (i) the feasibility of supporting rich query models without resorting to structured overlays and (ii) the benefits de-

[1]Structure overlay networks reside mainly on distributed hash tables (e.g., [SMLN+03, RFH+01]), which provide accurate location mechanisms.

rived from a looser organisation of system components. i*ClusterDL* is build upon a two-tier version of the i*Cluster* architecture and is designed to support retrieval and filtering functionality. i*ClusterDL* along with the protocols that regulate peer interactions are presented in [RPTW08].

i*ClusterDL* relies on the idea of organising digital libraries into semantic overlay networks. In this framework, digital libraries are linked by virtue of containing similar information. Building upon a P2P model, i*ClusterDL* consists of three general types of peers namely information providers (contributing documents to the network), information consumers (seeking for existing information) and super-peers (acting as access points for clients and providers). Super-peers self-organise into a semantic overlay network to offer a robust, fault-tolerant and scalable means for routing messages and managing queries. The description of each super-peer is derived from the descriptions of the providers connected to it, allowing super-peers to organise into clusters of similar content.

i*ClusterDL* is designed to support both *information retrieval* (IR) and *information filtering* (IF) functionality. In an IR scenario, a user poses a *one-time query* and the system returns all resources matching the query (e.g., all currently available documents relevant to the query). In an IF scenario, a user submits a *continuous query* and waits to be notified about certain future events of interest (i.e., about newly published documents relevant to the continuous query). With today's information explosion, IF becomes a necessary component for DLs since it provides the tools to keep the users informed, while not imposing the effort and cognitive overload of periodic one-time queries.

As an example of an application scenario for i*ClusterDL* let us consider a computer scientist whose main field of expertise is "information retrieval". Our scientist is mainly interested in retrieving scientific publications on his topic of interest and also, in following the work of prominent researchers in the area. He regularly uses the digital library of his department and also other digital libraries to search for new papers in the area. Even though searching for interesting papers this week turned up nothing, a search next week may return new information. Clearly, a system that is able to integrate a big number of relative sources and also capture his long-term

information need would be a valuable tool that would allow him to save both time and effort.

In our example scenario, consider a university comprised of geographically distributed campuses that maintain their local digital libraries. In the context of i*ClusterDL*, each campus implements and maintains its own super-peer, which serves as an access point for the provider representing the campus' DL and the clients deployed by end-users (e.g., students or faculty) querying for information. Other super-peers may also be deployed by larger institutions, like research centers or content providers (e.g., CiteSeer, ACM, Springer), to provide access points for their employees and make the contents of their DLs available in a timely way.

i*ClusterDL* offers an infrastructure, based on concepts from P2P systems, for organising the super-peers in a scalable, efficient and self-organising architecture. This architecture (i) allows for seamless integration of information sources (since different DLs and other content providers offer the same querying interface through the i*ClusterDL* system to the end-user), (ii) enhances fault tolerance (since it avoids centralised components that introduce bottlenecks and single points of failure) and (iii) requires no central administration authority (since each participant is responsible for the administration and maintenance of its own (super-) peers).

To the best of our knowledge, i*ClusterDL* is the first approach towards efficient organisation of digital libraries in semantic overlay networks that supports both retrieval and filtering functionality. The proposed architecture is automatic (requires no intervention and minimal administration), general (requires no previous knowledge of the DL contents and works for any type of data model or content), adaptive (adjusts to changes of DL contents), efficient (offers fast query processing) and accurate (achieves high recall).

7.2 The i*ClusterDL* Architecture

Figure 7.1 illustrates an overview of the i*ClusterDL* architecture composed of three main components: *super-peers*, *providers* and *clients*.

Super-peers represent the message routing layer of the network. Super-peers

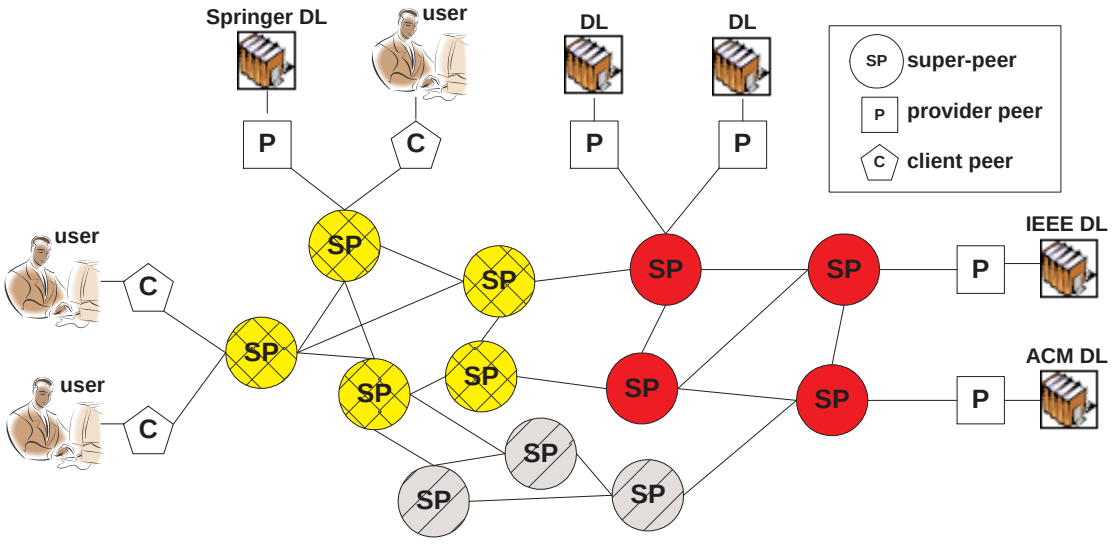


Figure 7.1: A high-level view of the i*ClusterDL* architecture

are nodes with more capabilities than provider or client peers (e.g., more cpu power and bandwidth), are responsible for serving information producers (providers) and information consumers (clients), and act as their access points to i*ClusterDL* network. Super-peers run the *peer organisation* protocol and form clusters based on their likelihood to contain similar content. Each super-peer is characterised by the information content of the providers that use it as an access point and maintains a *routing index* holding information for short-range and long-range links to other super-peers: *short-range links* correspond to *intra-cluster* information (i.e., links to super-peers with similar interests) and *long-range links* correspond to *inter-cluster* information (i.e., links to super-peers having different interests). In the i*ClusterDL* architecture, the role of super-peers is multi-fold: they act as the glue between information producers and information consumers, build a distributed self-organising repository that can be queried efficiently, store continuous queries submitted by information consumers to match them against newly published documents, and serve as a robust, fault-tolerant and scalable routing infrastructure.

Provider peers stand for information sources exposing their contents to the clients of the i*ClusterDL* network. A provider connects to the network through a super-peer (its access point). Providers store locally their documents and answer one-time queries. Each document is represented by a vector of terms in the spirit of vector space model (VSM) [Sal89], which may be either automatically (by

text analysis) or manually assigned to each document (e.g., tags or index terms). To identify its *interests*, a provider categorises its documents (each document may belong in multiple categories) using an external reference system (e.g., the ACM categorisation system), an ontology, or unsupervised clustering methods [SKK00]. Since documents are represented by term vectors, naturally a provider's interest is also represented by the *centroid* (i.e., the mean vector of the vector representations of the documents it contains). The interests of a provider are then used to determine the interests of the super-peer that acts as an access point for the provider.

Client peers provide an interface to end-users searching or subscribing for data. Clients can pose one-time queries and receive answers, or subscribe to resource publications and receive notifications about published resources that match their interests. A client connects to the network through a single super-peer (its access point). Clients can connect, disconnect or even leave the system at any time. If clients are not on-line, notifications matching their interests are stored by their access points and delivered once clients reconnect.

7.3 The i*ClusterDL* Protocols

The main idea behind i*ClusterDL* is to let super-peers (i.e., digital libraries) self-organise into clusters of similar interests and then address (or monitor) the most promising cluster of super-peers with respect to a one-time (or continuous) query. In this section, we discuss the protocols that specify how peers join and leave the network, how super-peers self-organise into clusters, and how query processing for both one-time or continuous query and document publication are carried out.

7.3.1 Provider and Client Join

The first time a provider p_i wants to connect to the i*ClusterDL* network, it has to follow the join protocol. Initially, p_i categorises its documents and stores its interests in $I(p_i)$. To join the network, p_i finds the IP address of a super-peer sp_j using out-of-band means (e.g., via a secure web site that contains IP addresses for the super-peers that are currently online) and sends to sp_j a NewP= $(id(p_i), ip(p_i), I(p_i))$ message,

where $id(p_i)$ is the identifier of p_i created when the provider bootstraps and is used to support *dynamic IP addressing* and $ip(p_i)$ is the IP address of p_i. Subsequently, sp_j adds p_i in its *provider table* (PT_j), which is a table used for identifying the providers that use sp_j as their access point. PT_j is used to associate the $id()$ of a provider peer with its last known IP address and also, stores information such as the status of the peer (connected/disconnected). Finally, sp_j sends to p_i an acknowledgement message AckP= $(id(sp_j), ip(sp_j))$. Once p_i has joined, it can use the connect/disconnect protocol described next to connect to and disconnect from the network.

Clients use a similar protocol to join the i*ClusterDL* network. When a client c_i wants to connect for the first time, it sends a NewC= $(id(c_i), ip(c_i))$ message to its access point sp_j and is subsequently added to the *client table* (CT_j) of sp_j. Similarly, CT_j is used to store contact and status information for the client peers, along with non-delivered notifications (e.g., due to a client peer being offline).

7.3.2 Provider and Client Connect/Disconnect

A provider p_i connects to the network by addressing a ConnectP= $(id(p_i), ip(p_i))$ message to its access point sp_j. If $id(p_i)$ exists in the provider table PT_j then p_i is marked as connected, otherwise this means that sp_j is not the access point of p_i and that p_i wants to *migrate* to another super-peer. In this case, the connection request is rejected and the provider p_i has to run the join protocol described in the previous section to connect and use sp_j as its access point. When a provider p_i wants to disconnect, it sends to its access point sp_j a DisconnectP= $(id(p_i), ip(p_i))$ message and sp_j marks provider p_i as disconnected in its provider table PT_j.

Clients connect to or disconnect from the network in a similar way, but sp_j has also to make sure that a disconnected client c_i will not miss notifications about resources of interest while not online. Thus, notifications for an offline client peer c_i are stored in the client table CT_j of sp_j and wait to be delivered upon reconnection of client c_i.

7.3.3 Super-Peer Join

To join the i*ClusterDL* network, a super-peer sp_i must compute the list of its interests $I(sp_i)$ as $\bigcup_{\forall p_j \in PT_i} I(p_j)$ in order to identify itself in the network. For each distinct interest $I_{ik} \in I(sp_i)$, sp_i maintains an individual routing index RI_{ik}, which contains short-range links that point to super-peers with similar interests and long-range links that point to super-peers with different interests. During the joining procedure of sp_i, the routing index is initialised as follows: sp_i collects in RI_{ik} (i) the IP addresses of s randomly selected super-peers or (ii) the routing index entries of neighbouring super-peers. These links will be refined according to the interest I_{ik} of sp_i using the super-peer organisation protocol described in the next section.

7.3.4 Super-Peer Organisation

Super-peer organisation proceeds by establishing new connections and by discarding old (or outdated) ones, producing this way clusters of super-peers with similar interests. Each super-peer sp_i *periodically* initiates a *rewiring procedure*. For each interest I_{ik}, sp_i computes the intra-cluster similarity NS_{ik} as:

$$NS_{ik} = \frac{1}{s} \cdot \sum_{\forall sp_j \in RI_{ik}} sim(I_{ik}, I_{jy}), \tag{7.1}$$

where s is the number of short-range links (super-peers in the neighborhood of sp_i) with respect to interest I_{ik}, sp_j is a super-peer that is on-line contained in the RI_{ik}, I_{jy} is the interest of sp_j stored in RI_{ik} and $sim()$ can be any appropriate similarity function between peer interests. If NS_{ik} is greater than a threshold θ then sp_i does not need to take any further action since it is surrounded by similar super-peers. Otherwise, sp_i initiates a cluster refinement process by issuing FindPeers= $(id(sp_i), ip(sp_i), I_{ik}, L, \tau_R)$ message, where L is a list initially empty and τ_R is the time-to-live (TTL) of the message. Notice that both θ and τ_R are system parameters that are tuned upon system bootstrapping.

A super-peer sp_j receiving the message computes the similarity between each one of its interests I_{jx}, $1 \leq x \leq n$, with interest I_{ik} in FindPeers() message, appends

Procedure Rewiring($sp_i, I_{ik}, \tau_R, \theta, m$)
A procedure initiated by a super-peer sp_i whenever its neighborhood similarity NS_{ik} drops below θ.

input: super-peer sp_i with interest I_{ik} and routing index RI_{ik}
output: updated routing indexes

1: compute $NS_{ik} = \frac{1}{s} \cdot \sum_{\forall p_j \in RI_{ik}} sim(I_{ik}, I_{jy})$
2: **if** $NS_{ik} < \theta$ **then**
3: $L \leftarrow \{\ \}$
4: create FindPeers()
5: $sp_l \leftarrow sp_i$
6: **repeat**
7: send FindPeers() to m random neighbours of sp_n
8: let sp_j be a neighbour of sp_n receiving FindPeers() and I_{jy} the interest of sp_j that is most similar to I_{lk}
9: sp_j generates a random number x
10: **if** $x \leq \varrho$ **then**
11: update RI_{jy} with information from L
12: $L \leftarrow L :: \langle ip(sp_j), I_{jy} \rangle$
13: $sp_l \leftarrow$ every sp_j receiving FindPeers()
14: $\tau_R \leftarrow \tau_R - 1$
15: **until** $\tau_R = 0$
16: return list L to sp_i
17: update RI_{ik} with information from L

Figure 7.2: The super-peer rewiring protocol

to L the interest I_{jy} resulted in the maximum similarity value, reduces τ_R by 1 and forwards FindPeers() message to its neighbouring super-peers. When $\tau_R = 0$, the FindPeers() message is sent back to the message initiator sp_i. During the message forwarding procedure, a message recipient sp_j chooses to forward FindPeers() message to a set of m randomly chosen super-peers contained in sp_j's routing index RI_{jy}. To further clarify the rewiring procedure, Figure 7.2 presents the pseudocode for super-peer organisation.

A super-peer sp_j receiving FindPeers() message may also collect with proba-

bility 50% (remember the refinement parameter $\varrho = 0.5$ described in Section 5.2) information about new super-peers with similar interests by examining the interests of previous message recipients. This new information is then used to update the routing index RI_{jy} of sp_j by replacing old short-range links corresponding to super-peers with less similar interests with new links corresponding to super-peers with more similar interests.

7.3.5 IR: Processing One-Time Queries

Let us assume that a client c_i wants to submit a query q, where q is a term vector. Initially, c_i sends a SUBMITQ= $(id(c_i), ip(c_i), q)$ message to its access point sp_j. Upon receival of a SUBMITQ() message, sp_j compares q against its interests $I_{jx}(1 \leq x \leq n)$ and selects the interest I_{jy} for which $I_{jy} = max(sim(q, I_{jx}))$ holds (i.e., the most similar interest to the query). If $sim(q, I_{jy}) \geq \theta$, then sp_j creates a QUERY= $(id(sp_j), ip(sp_j), id(c_i), ip(c_i), q, \tau_b)$ message, where τ_b is the query TTL, and forwards it to all its short-range links in RI_{jy}. In this way, peer sp_j broadcasts the message to its neighborhood. This broadcasting happens because sp_j has identified that the query is close to its interest and by this its is implied that the query may also be close to the interests of sp_j's neighbours.

If $sim(q, I_{jy}) < \theta$, sp_j forwards the QUERY() message with TTL τ_f to the m super-peers contained in its routing index with interests most similar to q. In this case m is usually small and this query forwarding technique is referred as *fixed forwarding*, since the query is forwarded through a limited number of paths until it finds a similar cluster. All forwarding super-peers execute the aforementioned protocol and reduce τ_f by one at each step of the forwarding procedure. Notice that the value of the TTL in the case of broadcasting is not the same with that of fixed forwarding; typically the TTL is smaller when broadcasting (since we need to reach super-peers only a few hops away) and larger when performing a fixed forwarding (since we need to explore regions of the network that are possibly far away from the initiating super-peer).

Apart from query forwarding, each super-peer sp_k, for which $sim(q, sp_k) \geq \theta$

holds, applies the following procedure for retrieving documents similar to q. It constructs a FINDRES= $(id(sp_j), ip(sp_j), q)$ message and sends it to all providers with interests similar to q, by examining its provider table PT_k. Once a provider peer p_l receives a FINDRES() message, it matches q against its local document collection to retrieve all documents matching q. The provider peer ranks the local results according to their relevance to the query, creates a result list r_l of the form $\langle d, m(d), Sim(q,d)\rangle$, where d is a pointer to a document, $m(d)$ are metadata about d and $sim(q,d)$ is the similarity between q and d, and sends a RETRES= $(id(p_l), ip(p_l), r_l)$ message to sp_j (notice that sp_j is the super-peer that initiated the querying procedure on behalf of client c_i). In this way, sp_j collects the local result lists of the relevant providers contacted and uses them to compute a final result list R that is sent to the client peer c_i and presented to the user.

7.3.6 IF: Processing Continuous Queries

In the following, the protocols of Section 7.3.5 are adjusted appropriately to support IF functionality.

Subscribing with a Continuous Query: Let us assume that a client c_i wants to submit a continuous query cq. Initially, c_i sends a SUBMITCQ= $(id(c_i), ip(c_i), cq)$ message to its access point sp_j. Super-peer sp_j initiates a CQUERY= $(id(sp_j), ip(sp_j), id(c_i), ip(c_i), cq, \tau_q)$ message. The message is forwarded in the super-peer network following the mechanism described in Section 7.3.5. A super-peer sp_k that receives a continuous query cq similar to some of its interests (i.e., $sim(cq, I_{ky}) \geq \theta$), stores cq in its local continuous query data structures to match it against future publications. Super-peer sp_k will utilise these data structures at publication time to find quickly all continuous queries that match a publication. This can be done using an appropriate local filtering algorithm such as SQI [YGM99]. Figure 7.3 presents the pseudocode for continuous query routing.

Publishing a New Document: Publications in i*ClusterDL* are kept locally at each provider in the spirit of [ZTW07]. This lack of publication dissemination mechanism is a design decision that avoids document-granularity dissemination (e.g.,

Procedure CQuery_Routing($cq, sp_i, \tau_b, \tau_f, \theta, m$)
A super-peer sp_i compares the continuous query cq towards its interests, decides whether to store it in its local continuous query data structures and forwards cq to the super-peer network.

input: query q issued by super-peer sp_i
output: updated continuous query data structures

1: compare cq against interests I_{ix}, where $1 \leq x \leq n$
and select I_{iy} that is the interest most similar to cq
2: initiate message CQUERY()
3: **if** $sim(cq, I_{iy}) \geq \theta$ **then**
4: store cq in local data structures
5: forward CQUERY() to all short-range links in RI_{iy}
6: $\tau_b \leftarrow \tau_b - 1$
7: **else**
8: forward CQUERY() to m neighbours of sp_i that are the most similar to cq
9: $\tau_f \leftarrow \tau_f - 1$
10: **do** the same for the neighbours of sp_i
11: **repeat** until $\tau_f = 0$ or $\tau_b = 0$

Figure 7.3: The continuous query routing protocol

as in [ZTW07]) and offers increased scalability by trading recall. Thus, only the corresponding super-peers (i.e., those indexing a continuous query cq) can notify a client c_i, although provider peers using *other* super-peers as access points may also publish relevant documents. When a provider peer p_j wants to publish a new document d to the network, it sends the document to its access point sp_k and then, the super-peer sp_k is responsible for matching d against its local continuous query database to decide which continuous queries match d and thus, which clients should be notified.

At pre-specified intervals or when the document collection of a provider peer p_j has changed significantly, p_j recomputes its interests and informs its corresponding access point by sending a RefreshInt= $(id(p_j), ip(p_j), I(p_j))$ message. Subsequently, the super-peer that acts as the access point of p_j changes the respective record in its provider table and refreshes the list of p_j's interests.

Notification Delivery: Let us assume that a super-peer sp_i has to deliver a notification for a continuous query cq to the client c_j. It creates a NOTIFY= $(id(p_l), ip(p_l), d, m(d), cq)$ message, where d is a pointer to the document matching cq, $m(d)$ are metadata about d and p_l is the provider that published d, and sends it to c_j. If c_j is not online, then s_i sends the message to the access point sp_k of c_j, using $ip(sp_k)$ associated with cq. Super-peer sp_k is then responsible for storing the message in its client table and delivering it to client peer c_j upon reconnection.

7.3.7 Discussion

i*ClusterDL* is highly dynamic as it allows for random insertions or deletions of all peer types, as well as for insertions or deletions of new documents. All peer insertions/deletions are performed in a local manner (affecting only entries in local super-peer data structures in the case of provider or client peers and super-peer routing indexes in the case of super-peers) and the network self-organises to a new stable state after a few iterations of the rewiring protocol. In this way, i*ClusterDL* is based solely on local interactions, requiring no previous knowledge of the network structure or of the overall content in the network. The messaging cost to maintain the network clustered is dampened at query time by the fast and efficient search mechanism. To further improve the search mechanism, i*ClusterDL* maintains a fixed number of long-range links (i.e., links to other clusters) in the routing indexes of the super-peers. These links provide shortcuts to other clusters and prevent them from forming disconnected communities that are inaccessible by others.

Notice that methods assuming one interest per super-peer [NWQ+02, Sch04] (specialisation assumption) will not perform well under a digital library setting: the description of a super-peer would either reflect the contents of its strongest interest (i.e., the interest of the provider with the largest document collection) ignoring all other interests, or result in a single category representing the averaging over the document collections of the super-peer's providers. This would result in poor retrieval performance as queries (even very specific ones) would be addressing highly incoherent clusters of super-peers. To avoid this in i*ClusterDL* providers and super-peers use multiple interests obtained by document categorisation.

7.4 Experimental Evaluation

In this section, we evaluate the proposed protocols using two real-life corpora with web and medical data and compare them against a baseline flooding approach.

7.4.1 Performance Measures

As it is typical in the evaluation of information retrieval systems, performance is measured in terms of network traffic and retrieval effectiveness. The *network traffic* is measured by recording the number of messages exchanged by the super-peers during rewiring or querying. The retrieval effectiveness is evaluated using *recall* as the percentage of qualifying answers retrieved with respect to the total number of qualifying answers in the network. The filtering effectiveness is evaluated using *recall* as the percentage of total number of notifications received with respect to the total number of published documents matching a subscription in a time window.

7.4.2 Experimental Testbed

Dataset. The first dataset contains over 556,000 web documents from the TREC-6[2] collection belonging in 100 categories. Notice that this dataset has been previously used by Xu and Croft [XC99] to evaluate information retrieval algorithms over distributed document collections for scenarios similar to the ones of the digital library domain. The second dataset is a subset of the OHSUMED TREC[3] document collection that contains over 30,000 medical articles from 10 different categories as suggested in [SKK00]. The one-time and continuous queries that were employed are strong representatives of the document categories (i.e., the topics of the categories).

Setup. We consider N loosely-connected super-peers. Each provider is specialised in one category, while we pose no restrictions on which providers connect to a super-peer. Thus, a super-peer may have providers with different interests and be subsequently part of many different clusters. Each super-peer periodically tries to find better neighbours by initiating the rewiring procedure. The base unit for time used

[2]http://boston.lti.cs.cmu.edu/callan/Data/
[3]http://trec.nist.gov/data/t9_filtering.html

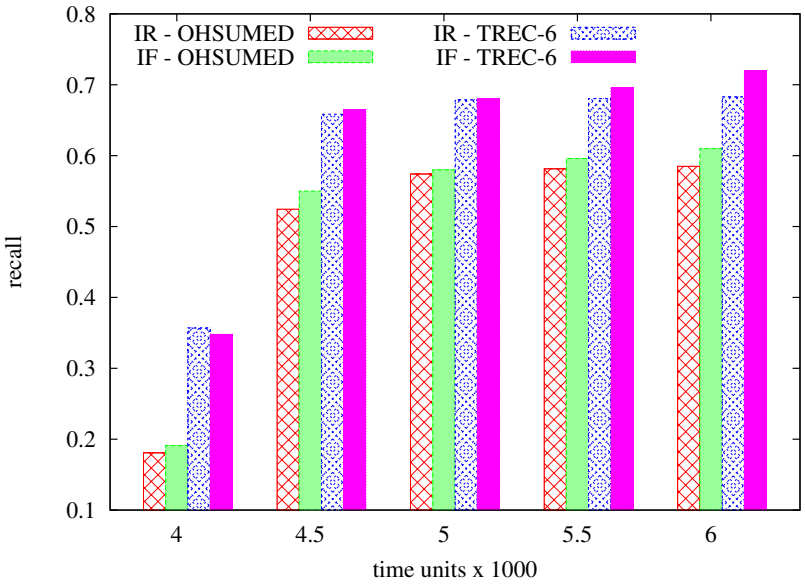


Figure 7.4: Recall over time for both IR and IF scenarios using two real-life corpora

is the period t. The start of the rewiring procedure for each peer is randomly chosen from the interval $[0, 4K \cdot t]$ and its periodicity is randomly selected from a normal distribution of $2K \cdot t$, in the spirit of [Sch04]. We start recording the network activity at time $4K \cdot t$, when all super-peers have initiated at least once the rewiring procedure. The size of the network is 2,000 super-peers. The baseline parameter values used for this set of experiments are $\theta = 0.9$, $\tau_R = 4$, $\tau_b = 2$ and $\tau_f = 8$.

The simulator used to evaluate i*ClusterDL* is implemented in C/C++ and all experiments were run on a Linux machine. Our results were averaged over 25 runs (5 random initial network topologies and 5 runs for each topology).

7.4.3 Evaluation Results

Retrieval and Filtering Effectiveness

Figure 7.4 illustrates the performance of i*ClusterDL* as a function of time for both datasets. Due to the similar query routing protocol, recall for both retrieval and filtering scenarios has similar behaviour. When the super-peer network is not yet fully organised into clusters of similar super-peers ($t = 4K$) the queries cannot be routed towards the appropriate super-peers, thus reaching low recall values (around 35% for the TREC-6 corpus and 20% for the OHSUMED corpus). When the network

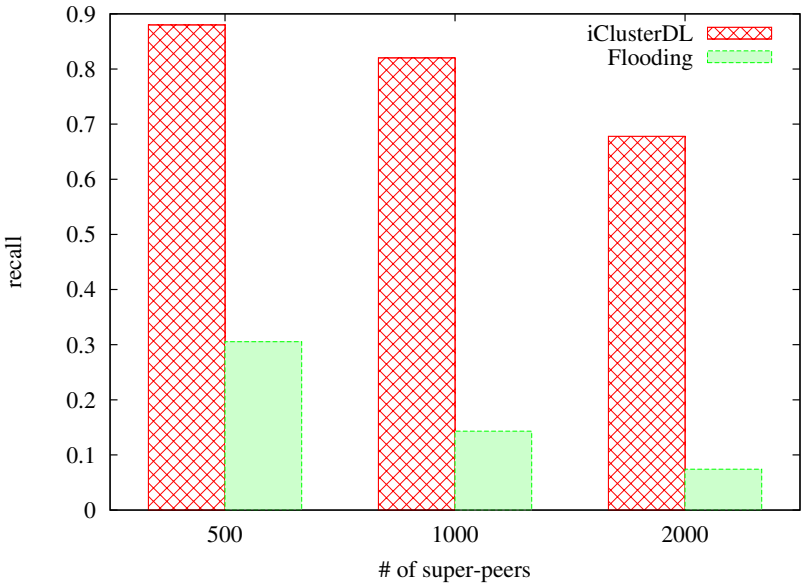


Figure 7.5: Recall for i*ClusterDL* and Flooding using the same number of messages (TREC-6)

becomes organised into cohesive clusters ($t \geq 6K$), i*ClusterDL* achieves high values of recall (over 60%) for all examined scenarios.

Figure 7.5 illustrates a comparison of the performance of i*ClusterDL* against the flooding approach limited to the same number of messages, for different sizes of the network using the TREC-6 corpus (the results are similar for the OHSUMED dataset). i*ClusterDL* demonstrates significant performance improvement over flooding, which increases with network size reaching up to 8 times better recall. In fact, i*ClusterDL* scales up well for large networks (only 20% decrease of recall for 400% increase in network size).

Message Costs

Figure 7.6 shows the number of messages needed over time for the self-organisation of the super-peers. Initially, the network presents a message overhead in terms of organisation messages, which is greatly reduced after the network organises into coherent clusters. The organisation messages for TREC-6 are higher due to the higher number of clusters created, which in turn is an effect of the higher number of document categories. However, after the organisation of the super-peers, the rewiring protocol is able to maintain an effective super-peer organisation at a small

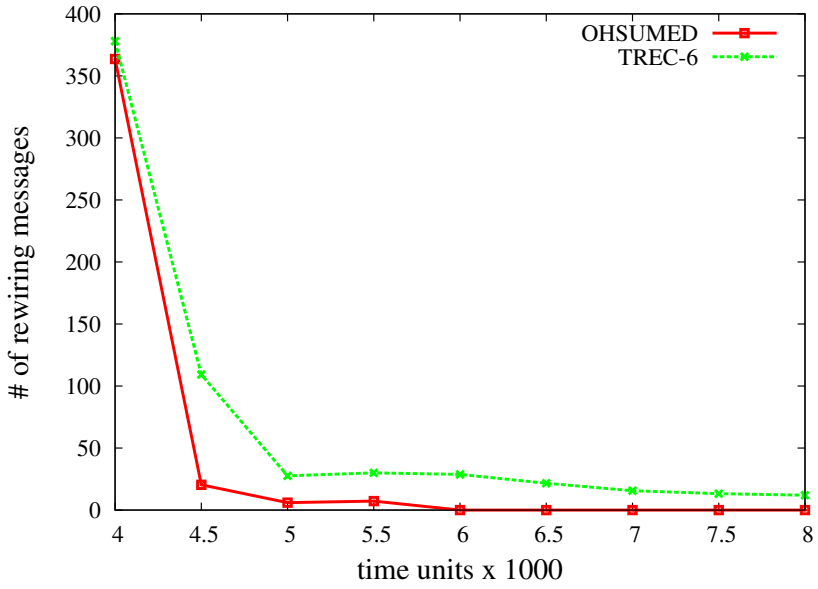


Figure 7.6: Organisation messages for different corpora

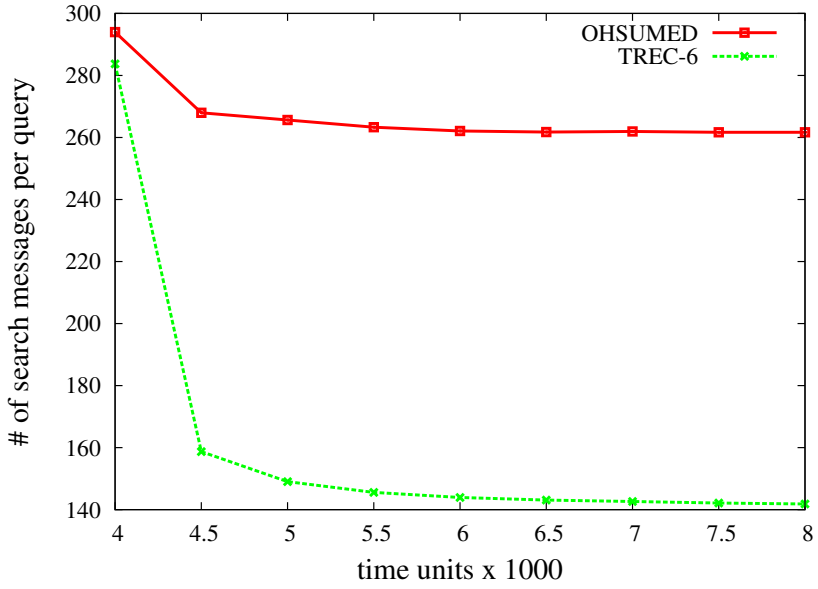


Figure 7.7: Search messages per query for different corpora

communication cost.

Figure 7.7 shows the number of messages per (continuous) query over time. When the network is not yet organised into coherent neighborhoods (left-most points in the x-axis), i*ClusterDL* needs high number of search messages to achieve the recall shown in Figure 7.4. However, this message overhead is decreased (over 12% decrease for the OHSUMED corpus and 100% decrease for the TREC-6 corpus) as the super-peers get organised into clusters with similar interests (right-most points

in the x-axis). The search messages for the OHSUMED corpus are higher due to the higher number of peers per cluster. Figures 7.7 and 7.4 demonstrate that i*ClusterDL* manages to effectively organise the network, as retrieval and filtering performance improves for much less communication overhead.

7.5 Conclusions

In this chapter, we introduced i*ClusterDL*, a novel P2P architecture supporting self-organising digital libraries that demonstrates (i) the feasibility of supporting rich query models without resorting to structured overlays and (ii) the benefits derived from a looser organisation of system components. i*ClusterDL* is build upon a two-tier version of the i*Cluster* architecture and is designed to support retrieval and filtering functionality under a single unifying framework. We evaluated i*ClusterDL* using two real-life corpora with web and medical documents.

To the best of our knowledge, i*ClusterDL* is the first comprehensive architecture that exploits semantic overlay networks to provide a framework for self-organised and self-managed digital libraries that can offer a rich quiver of tools to end-users. Our architecture maintains a flat structure of clusters with no representatives and the communication can be routed through any peer within a cluster, thus avoiding bottlenecks. Moreover, i*ClusterDL* architecture supports IF functionality (in addition to IR) at no extra communication cost. The proposed architecture is automatic (requires no intervention and minimal administration), general (requires no previous knowledge of the DL contents and works for any type of data model or content), adaptive (adjusts to changes of DL contents), efficient (offers fast query processing) and accurate (achieves high recall).

Chapter 8

Epilogue

This last chapter presents a brief summary (Section 8.1) and highlights the main contributions (Section 8.2) of the research conducted in this thesis. Section 8.3 discusses possible extensions to the presented work and elaborates on open research questions including load balancing, replication and caching and the applicability of our work to modern application domains.

8.1 Overview

Recently, the P2P paradigm has moved towards *distributed data management systems* that are able to offer *information retrieval* or *information filtering* functionality. The main advantage of P2P is its ability to handle huge amounts of data in a decentralised and self-organising manner. The characteristics of P2P offer high potential benefit for information systems regarding scalability, efficiency, and resilience to failures and dynamics. Such an information management system can potentially benefit from the intellectual input of a large user community participating in the data sharing network. Peers are *organised* based upon the idea of sharing data using loose P2P architectures assuming neither a specific network structure nor total control over the location of the data. These P2P architectures offer better support of semantics due to their ability to provide mechanisms for approximate, range, or text queries, and emphasise on peer autonomy.

The main challenge addressed in this thesis is to improve retrieval performance

(in terms of improved retrieval accuracy and reduced network load) by means of rewiring. Most of the research proposals assume a fixed rewiring procedure for creating semantic overlay networks and search the network by exploiting certain architectural or modelling aspects of this organisation. The work presented in this thesis carries the existing research one step further and suggests that the rewiring strategy employed by a system can affect its retrieval performance. The notions of *peer rewiring* and *network performance* are investigated and taken into account in designing an effective (in terms of retrieval performance) rewiring strategy. A number of choices in designing a rewiring strategy are considered and all are implemented and evaluated in a realistic setting using real-world data and queries.

The research presented in this thesis offers the ability to identify the functional issues related to the design and the performance of any rewiring strategy by answering fundamental questions. The different rewiring strategies are thoroughly discussed and evaluated using real-world data and queries. Based on insight acquired by answering these questions, this thesis proposes optimisations to the rewiring protocol used both for information retrieval and information filtering functionality in a fully decentralised and dynamic setting. The most promising rewiring strategy is finally designed based on the evaluation of the alternative network configurations and on the analysis of their performance.

8.2 Contributions

This thesis presents i*Cluster* , a decentralised P2P architecture that ensures scalability, robustness and fault-tolerance for providing improved network performance by means of an effective (in terms of retrieval accuracy) rewiring strategy. The rewiring strategies are proposed, discussed and evaluated in this work.

Similarly to unstructured P2P systems, i*Cluster* keeps a completely decentralised architecture where no peer is more important than any other. Similarly to SON-based systems, i*Cluster* applies a periodic rewiring protocol to cluster peers with similar interests and a fireworks-like technique to route queries within the network. The interests of a peer are identified by its local document collection and each peer has a tunable and dynamic number of interests depending on its capabilities,

collection size and content diversity. i*Cluster* is the first system to *overcome the specialisation assumption* common in semantic overlay networks. i*Cluster* architecture along with the corresponding protocols is presented in [RP08b].

Contrary to other approaches, the focus of this thesis is on the rewiring protocol that is (periodically) executed independently by each peer for the purpose of improving network organisation. We *introduced the functional issues* related to peer rewiring and investigated the way that the rewiring strategy employed by a system can *affect retrieval performance*. This study is thoroughly presented in [RPT09].

Specifically, we considered and discussed a number of choices in designing a rewiring strategy by answering the following fundamental questions: (i) what is an appropriate forwarding strategy for rewiring messages within a SON, (ii) what information should be used to update peer connections, (iii) should peer connections be bi-directional or just one-directional, (iv) how coherent and well-connected should a cluster of similar peers be? We considered all the possible design choices turned up by these questions and implemented different rewiring protocols. These rewiring protocols were thoroughly discussed and evaluated using real-world data and queries. Based on insight acquired by answering the above questions and on the analysis of the experimental results, we proposed optimisations to the rewiring protocol and sketched the most promising rewiring strategy.

This thesis *introduced a network clustering measure*, called clustering efficiency κ, in a attempt to describe and quantify the connectivity of the network in a formal way. An experimental evaluation investigated the usefulness of the proposed measure in estimating the connectivity of the underlying network when using different rewiring strategies. This part of our work is presented in [RP08a].

In real-world applications a peer joins the network when a user starts the application and a peer leaves the system when the user exits the application. Contrary to other approaches, we took into account these *user-driven dynamics* of peer participation in both the design and evaluation of the P2P system proposed in this thesis. We actually moved towards a realistic scenario where peers join, participate, leave and re-join the network. i*Cluster* was utilised to organise the network and retrieve data to the interested users under such a dynamic setting. We considered that peers

and data follow appropriate dynamic models that exist in the literature, studied the way peer churn affects the system performance and identified which rewiring protocol performs well under dynamic networks.

The main contributions provided by the evaluation of i*Cluster* under a dynamic scenario are the following: (i) a novel study concerning a P2P architecture and its related protocols supporting retrieval functionality in a distributed P2P dynamic environment and (ii) an analysis of the experimental results when testing i*Cluster* under a dynamic setting in order to use them as a feedback and identify a rewiring protocol that performs well under dynamic networks.

Finally, this thesis *examines a digital library use case*, called i*ClusterDL*, as an application scenario. i*ClusterDL* is a novel P2P architecture supporting self-organising digital libraries that demonstrates (i) the feasibility of supporting rich query models without resorting to structured overlays and (ii) the benefits derived from a looser organisation of system components. i*ClusterDL* was build upon a two-tier version of the i*Cluster* architecture and designed to support retrieval and filtering functionality under a single unifying framework. i*ClusterDL* is presented in [RPTW08].

To the best of our knowledge, i*ClusterDL* is the first approach towards efficient organisation of digital libraries in semantic overlay networks that supports both retrieval and filtering functionality. The proposed architecture is automatic (requires no intervention and minimal administration), general (requires no previous knowledge of the DL contents and works for any type of data model or content), adaptive (adjusts to changes of DL contents), efficient (offers fast query processing) and accurate (achieves high recall).

8.3 Open Questions

The topics discussed in this section are not only relevant within the scope of this thesis, but are also interesting in the context of P2P resource sharing in general. The discussion will focus on topics relevant to the efficiency and effectiveness of data sharing, and will not go into security or privacy issues that, although of high

importance, are independent of the proposed solutions.

In the remainder of this section will discuss open questions in the fields of *load balancing*, *replication and caching* and the applicability of our work to recent *computer paradigms*.

8.3.1 Load balancing

An open problem related to this thesis is *load balancing* in a distributed P2P setting. Load balancing is important for networks where it is difficult to predict the number of issued requests and is a critical issue for the efficient operation of the P2P network. Load balancing can be defined as distributing processing and communication activity evenly across a computer network so that no single device is overwhelmed [PGSM05]. In a P2P network, the load of a peer is determined by the number of issued requests and is directly related to the stored data items of the peer. The total load of the P2P network is defined as the sum of the loads of all the peers participating in the network. Considering a load balanced network, the load of a peer should be around $1/N$ of the total load when considering a network of N peers[1]. A peer is referred to as *overloaded* or *heavy* if it has a significantly higher load compared to the load it would have in a load balanced network. Respectively, a peer is *light* if it has significantly less load than the load it would have in a load balanced network.

As presented in the previous chapters of this thesis, the i*Cluster* approach utilises a semantic overlay network as the underlying infrastructure. The following questions related to load balancing in semantic self-organising P2P networks arise and have to be addressed in the future:

- Rewiring aims at the reconstruction of peer linkage based on peer similarity as follows: each peer periodically (and independently from the other peers) sends a message within the network following a specific routing strategy. By this process peers discover and collect in their routing indices information about other peers. The rewiring messages traversing the network impose peers with extra communication load. The goal of load balancing is to *distribute the*

[1]When a peer is considered equal with respect to its processing power and storage capacity.

rewiring communication activity evenly among peers. This problem could be seen as solved by relying on a random decision making routing strategy. However, we should consider the dependence of the rewiring on both the initial network topology and the population of the peer clusters. The task of designing a routing strategy that distributes rewiring communication activity evenly among peers, but without drawing away the initial scope of rewiring (i.e., bring in contact similar peers) is not trivial if we consider (i) the challenge to retain the small-world properties of the network, (ii) the high rates of peer join and leave (churn) and (iii) the wide variation in peer network and storage capacity (heterogeneity).

- When a peer issues a query, it forwards it to all its neighbours (i.e., similar peers) when the query is similar to this peer's interests. Otherwise, the peer sends the query to the m most similar (to the query) peers stored in its routing index. The querying messages traversing the network impose peers with communication and processing load. The goal of load balancing is to *distribute querying communication activity evenly* among peers. The querying load of a peer is determined by the number of received requests and is straightly related to the interests of the peer and the entries in its routing index. Since the interests of a peer are determined by the data items this peer contributes to the network, the focus is put on the construction of the routing indices. To achieve the task of querying load balancing the rewiring strategy should thus, avoid the appearance of the same few peers in many routing indices. Particularly, concerning the forwarding of a query to dissimilar peers in the network (i.e., when a query is not similar to the interests of the issued peer), the rewiring strategy should ensure the use of different peers as long-range links in different routing indices, so that to avoid using the same cluster gateways and by this, abstain bottlenecks when trying to reach a peer cluster.

- It is expected that some topics of interest will be more popular than others. Peers being part of popular clusters (i.e., contributing data items belonging to popular categories) will be thus, heavy-loaded receiving many requests by interested users. Cluster load balancing aims to *distribute the load evenly*

among the clusters of the network. The research presented in [TXKN03] is a prominent work addressing this issue. There, inter-cluster load balancing is achieved by associating the document categories with clusters of peers in a manner that ensures a fair distribution of the document-category popularity to the clusters of peers. The proposed approach is a fine load balancing solution in the case where the data categories and the documents popularity are given and known to the peers. An interesting extension would be to allow the peers decide their data categories and exchange this information with their neighboring peers prior to forming clusters, like in i*Cluster* approach. Controlling the abstraction of grouping (i.e., how general or specialised document clusters will be) could possibly help towards the direction of cluster load balancing.

8.3.2 Replication and Cashing

An interesting open question for future work involves dealing with *replication and caching*. *Replication* is the process of sharing information so as to ensure consistency between redundant resources, to improve reliability, fault-tolerance, or accessibility. It could be data replication if the same data is stored on multiple sources, or computation replication if the same computing task is executed many times. Typically, the access to a replicated entity is uniform with the access to a single non-replicated entity. The replication itself should be transparent to an external user. Replication mechanisms can improve resilience and query efficiency in distributed systems. Specifically in P2P networks, replicated data is stored at multiple peers so that it can be accessed by the user even when some of the copies are not available due to peer failures. However, replication entails various costs such as storage for holding replicas and communication overheads for ensuring replica consistency. Additionally, the overall effectiveness of replication is heavily dependent on the topology of the overlay network.

In computer science, a *cache* is a collection of data duplicating original values stored elsewhere or computed earlier. The idea of using a cache is based on the fact that the original data may be expensive to fetch (owing to longer access time) or to compute, compared to the cost of reading the cache. In other words, a cache

is a temporary storage area where frequently accessed data can be stored for rapid access. Once the data is stored in the cache, future use can be made by accessing the cached copy rather than re-fetching or recomputing the original data, so that the average access time is shorter. In P2P networks, peers storing copies of the data items they have requested is the standard approach for caching.

Both caching and replication are two popular topics in distributed information management systems. In Freenet[2], data is replicated at each peer through the path where the search message traverses. Lv et al. [LRS02] revisit Gnutella-like P2P networks and investigate wether these networks can be more scalable by introducing some characteristics, such as content replication at a fair fraction of peers. Cohen and Shenker [CS02] propose and analyse three replication strategies for blind search (uniform strategy, proportional strategy and square-root strategy) and prove that the square-root strategy can minimise the search size. In [CRB+03], Chawathe et al. propose a model to include replication mechanisms (among other features) in Gnutella-like P2P networks. Sripanidkulchai et al. [SMZ03] and Markatos [Mar02] study the characteristics of search messages and propose some query cache policies to reduce message cost in Gnutella-like P2P networks. In [TK05], Tewari and Kleinrock investigate the effect of the number of file replicas on search performance in unstructured P2P networks, and show that flooding-based search is optimised when the number of replicas is proportional to the file request rates. Recently, Hassan et al. [HRM+08] explore a multi-objective optimisation approach for replica management and propose two novel algorithms for deciding the number of replicas as well as their placement within the overlay.

In the context of semantic overlay networks, it not obvious either how or what to cache or replicate. Constraints are imposed by the existence of peer interests and the clustering of peers. In the setting of this thesis, the most benefits out of replication and caching emerge from applying them to the content contributed in the network, or even to a distributed table storing peer statistics (useful in a highly dynamic setting).

[2] `http://freenet.sourceforge.net/`

8.3.3 Other Computing Paradigms

An interesting direction for future research covers the use of the presented P2P approach on other computer paradigms, such as cloud computing and social networking. These computer paradigms are not only interesting within the scope of this thesis, but are also relevant in the context of the P2P technology in general. In what follows, we define cloud computing and present the new twist inserted in it by the use of P2P technology. Next, we discuss the way user preferences defined in social networking and overlay creation in P2P networks are interconnected.

Cloud Computing

Cloud computing stands for *Internet-based* development and use of *computer technology*. It concerns a mode of computing where IT-related capabilities are provided *as a service* via the web, allowing users and developers to access and utilise technology-enabled services in the cloud without knowledge of, expertise with, nor control over the technology infrastructure that supports them. Cloud computing is a general concept that incorporates software as a service, Web 2.0 and other recent well-known technology trends, where the common theme is relying on the Internet for satisfying the computing needs of the users.

Cloud computing can be seen as a sophisticated evolution of the centralised client-server model: a "server farm" with hundreds or thousands of cheap low-end machines that acts as a centralised computing resource is employed. The cloud typically resides within a company's organisation (e.g., Google) and cloud applications are usually products (e.g., Gmail). By this, companies provide via web the functionality also found within an installed application, but without the need to update or maintain the software from the end-user side. Although, some of the companies use geographically distributed data centers, it is not the case for the majority of the clouds. The use of a single data center leads to single points of failure with serious downtimes. P2P technology can insert a new twist on cloud computing.

The idea of using the SON-based technology for cloud computing has been widely discussed recently, while some of the cloud providers incorporate overlay networks in

their services. Skype[3] develops and operates a VoIP computer telephony program build on top of a hybrid P2P architecture that uses the computers of both end-users and data centers in order to decentralise the network and to help ensure a high uptime percentage. Once a user logs in to Skype, its computer becomes part of the network (i.e., its computer is seen as a node so a bit of its bandwidth and CPU is used to help the rest of the Skype network). Super nodes are also used for management, messaging, monitoring etc. Super nodes in Skype reside on huge pipes such as Universities and ISPs. i*Cluster* protocols could be used to provide both an efficient organisation of user nodes, as well as management and messaging solutions eliminating in this way, the need of data centers.

Wuala[4] is an online storage system that allows its users to store, backup and access files from anywhere, and to share files with friends, groups and the world. Wuala, based on a P2P cloud storage solution, uses some of the disk space on the computers of its users. Files are encrypted on the user's computer, chopped up into little pieces and uploaded to both Wuala's servers and other users' computers to provide a redundant storage solution (called *social grid storage*). Wuala users can choose to contribute idle resources to the system or buy additional online storage. Wuala users could be clustered in similar groups according to their preferences or acquaintances in the spirit of i*Cluster*, and exploit these similarities when storing or accessing files.

Faroo[5] has launched a search engine that lets users search and browse the web via unstructured peer-to-peer technology. Faroo's search engine uses distributed crawling, indexing and ranking to provide relevant search terms. Whenever a user opens a web page in a browser, the page automatically gets inserted into the distributed index of the P2P network, as opposed to storing the search index on a central server. The index is replicated across multiple nodes in order to create redundancy and prevent information loss.

Metaaso mermaid[6] is a line of products based on P2P technology, is completely

[3]https://developer.skype.com/
[4]http://www.wuala.com/
[5]http://www.faroo.com/
[6]http://mermaid.metaaso.com/

server-less and allows users to broadcast movies, webcam feeds, audio feeds, news and files over their own overlay network. The Mermaid products are pure P2P and do not require any major infrastructure to scale to millions of people. The idea is that the collective power of all the computers connected to the network is harnessed in order to provide the aforementioned broadcast services. In such an application domain, techniques from semantic overlay networks could fit well for providing management and messaging solutions.

In general, introducing SON-based technology in cloud computing can improve reliability and scalability. Running away from single points of failure and using the collective power of the users can also prove cost effective, since the end-users' resources are used and there is no need to build and maintain expensive data centers. However, building clouds on top of semantic overlay networks is not obvious. Security and privacy issues that have to be tackled arise, as well as issues concerning the regulation and the management of the network traffic, the distributed indexing and the searching task. Regarding the i*Cluster* approach presented in this thesis, it could provide elegant solutions for organising and searching the services distributed over the peers in the network.

Social networks

A *social network* is a structure made of individuals or organisations (nodes), which are tied (connected) by one or more specific types of interdependency, such as friendship, kinship, financial exchange, dislike etc. From a networking point of view, nodes are the individual actors within the network and ties are the relationships between the actors. The resulting graph-based structures are often very complex. Research on social networks is extended on several fields, concerning sociology, biology, economics, information science etc., and has become a popular topic of speculation and study. There is a fairly extensive list of social networking websites, including

among other Buzznet[7], Classmates.com[8], Facebook[9], Flickr[10], hi5[11], LinkedIn[12] and MySpace[13], where registered users share photos and videos, chat, or exchange personal and professional information with other users standing for friends, relatives, or colleagues.

In the setting of this thesis, overlay creation in P2P networks and social networking can be nicely interconnected. On the one hand, social networking can insert an interesting perspective in P2P information sharing systems by enhancing the creation of overlay networks. On the other hand, SON-based technology can insert a new twist in social networks by moving them towards distributed solutions. In the following, interesting issues introduced by the interconnection of social and semantic overlay networks are elaborated:

- Combining file sharing applications with social networks enables people to create a trusted network of their friends. LimeWire[14], a popular P2P file sharing system, recently tried to incorporate social elements that will enable users to share files with friends on their buddy lists. LimeWire calls this new type of network a "personal sharing network". The idea is that users are more likely to want and feel comfortable with content from people they know, so trusted context is added to user searches for content. Tribler[15] [PGW+08], which constitutes a set of extensions to BitTorrent[16] network, is also a social-based P2P file sharing paradigm that exploits social phenomena. Tribler maintains and uses social information in content discovery, content recommendation and downloading, in an attempt to make users feel comfortable within the network and to improve download performance.

  However, most of the current P2P file sharing systems treat their users as

[7] http://www.buzznet.com/
[8] http://www.classmates.com/
[9] http://www.facebook.com/
[10] http://www.flickr.com/
[11] http://hi5.com/
[12] http://www.linkedin.com/
[13] http://www.myspace.com/
[14] http://www.limewire.com/
[15] http://tribler.org/trac
[16] http://www.bittorrent.com/

anonymous and disregard any social relationships that may exist between them. Though, social affinities (e.g., friendship) and the existence of communities of users with similar interests may be exploited in such systems to increase their usability and performance. In the context of i*Cluster*, the overlay creation is based on user preferences. i*Cluster* extracts peer interests by grouping the documents these peers contribute to the network and explores these interests to create connections among peers. The social affinities could insert one more interesting dimension when creating peer linkage that worth exploring.

- Most of the popular social networking systems are built on top of big centralised warehouses, and users are connected to these systems following a client-server model. The data (e.g., files, media, urls) provided by the users are stored somewhere centrally in the system and the communication between users is conducted through the server of the system. This way users share their data and communicate having always a central point in between. SON-based technology can insert a twist in social networks: users can keep their data on their computers and communicate directly without the need of a central server. The idea of moving towards distributed social networking can improve reliability and scalability, can eliminate serious bottlenecks, point of failures, or security threats, and can amplify the interpersonal nature of social networking.

  Imeem[17] and Krawler[18] are two systems that launch P2P-based social networking. Imeem is a hybrid P2P social network that integrates IM, blogging, file and photo sharing features. Users of Imeem create their own personal private networks in a way that they can bridge the gap between storing everything on their computers and not having to completely give up control of the content they want to share. Each user defines "circles of trust" and allows access to its content only to certain individuals. Krawler is a tool for creating, sharing, searching and securely managing communities and content. It uses a hybrid

[17] http://www.imeem.com/
[18] http://www.krawler.com/

P2P social networking platform that lets users view not just their friends' profiles, but also their shared content.

The concepts incorporated by the above systems sound interesting, though these works are rather preliminary. Research towards distributed social networks has still many question to answer and many problems to solve. In the setting of this thesis, ideas concerning the organisation of peers, the routing of messages and the searching of data could be used to built a distributed semantic overlay social network.

References

[AAG+05] K. Aberer, L. O. Alima, A. Ghodsi, S. Girdzijauskas, M. Hauswirth, and S. Haridi. The Essence of P2P: A Reference Architecture for Overlay Networks. In *Procedings of the International Conference on Peer-to-Peer Computing (P2P)*, August 2005.

[ACMD+03] K. Aberer, P. Cudré-Mauroux, A. Datta, Z. Despotovic, M. Hauswirth, M. Punceva, and R. Schmidt. P-Grid: A Self-organizing Structured P2P System. *SIGMOD Record*, 32(2), September 2003.

[ACMH03] K. Aberer, P. Cudré-Mauroux, and M. Hauswirth. The Chatty Web: Emergent Semantics Through Gossiping. In *Proceedings of the 12th International World Wide Web Conference (WWW)*, Budapest, Hungary, 20-24 May 2003.

[AEABH03] L. O. Alima, S. El-Ansary, P. Brand, and S. Haridi. DKS(N, k, f): A Family of Low Communication, Scalable and Fault-Tolerant Infrastructures for P2P Applications. In *Proceedings of the International Symposium on Cluster Computing and the Grid (CCGRID)*, pages 344–350, 2003.

[AGH04] L. O. Alima, A. Ghodsi, and S. Haridi. A Framework for Structured Peer-to-Peer Overlay Networks. In *Global Computing 2004*, volume 3267 of *LNCS*, pages 223–250, 2004.

[AMA+08] F. Atalla, D. Miranda, J. Almeida, M. A. Goncalves, and V. Almeida. Analyzing the Impact of Churn and Malicious Behavior on the Quality of Peer-to-Peer Web Search. In *Proceedings of the 23rd Annual ACM*

Symposium on Applied Computing (SAC), Fortaleza, Ceará, Brazil, 16–20 March 2008.

[AP04] S. S. R. Abidi and X. Pang. Knowledge Sharing Over P2P Knowledge Networks: A Peer Ontology And Semantic Overlay Driven Approach. In *Proceedings of the International Conference on Knowledge Management*, Singapore, 13-15 December 2004.

[AT05] I. Aekaterinidis and P. Triantafillou. Internet Scale String Attribute Publish/Subscribe Data Networks. In *Proceedings of the 14th International Conference on Information and Knowledge Management (CIKM)*, pages 44–51, 2005.

[BBR+06] R. Baldoni, S. Bonomi, A. Rippa, L. Querzoni, S. T. Piergiovanni, and A. Virgillito. Evaluation of Unstructured Overlay Maintenance Protocols under Churn. In *Procceedings of the 26th International Conference on Distributed Computing Systems Workshops (ICDCS Workshops)*, Lisboa, Portugal, 4–7 July 2006.

[BG87] D. Bertsekas and R. Gallager. *Data Networks*. Prentice Hall, 1987.

[BGW08] U. Brandes, M. Gaertler, and D. Wagner. Engineering Graph Clustering: Models and Experimental Evaluation. *ACM Journal of Experimental Algorithmics*, 12(1.1), 2008.

[BKK+03] H. Balakrishnan, M. F. Kaashoek, D. R. Karger, R. Morris, and I. Stoica. Looking up Data in P2P Systems. *Communications of the ACM*, 46(2):43–48, 2003.

[BMT+05] M. Bender, S. Michel, P. Triantafillou, G. Weikum, and C. Zimmer. MINERVA: Collaborative P2P Search. In *Proceedings of the 31st International Conference on Very Large Databases (VLDB)*, Trondheim, Norway, 30 August - 2 September 2005.

[BS07] S. Blanas and V. Samoladas. Contention-Based Performance Evaluation of Multidimensional Range Search in Peer-to-Peer Networks. In *Proceedings of the 2nd International Conference on Scalable Information Systems (InfoScale)*, volume 304, 2007.

[CGM02] A. Crespo and H. Garcia-Molina. Routing Indices for Peer-to-Peer Systems. In *Proceedings of the 22nd IEEE International Conference on Distributed Computing Systems (ICDCS)*, pages 23–32, Vienna, Austria, July 2002.

[CGM03] A. Crespo and H. Garcia-Molina. Semantic Overlay Networks for P2P Systems. Technical Report, Stanford University, May 2003.

[CHM+02] I. Clarke, T. W. Hong, S. G. Miller, O. Sandberg, and B. Wiley. Protecting Free Expression Online with Freenet. *IEEE Internet Computing*, 6(1):40–49, January 2002.

[CIKN04] P. A. Chirita, S. Idreos, M. Koubarakis, and W. Nejdl. Publish/Subscribe for RDF-based P2P Networks. In *Proceedings of the European Semantic Web Symposium (ESWS)*, 2004.

[CRB+03] Y. Chawathe, S. Ratnasamy, L. Breslau, N. Lanham, and S. Shenker. Making Gnutella-like P2P Systems Scalable. In *Proceedings of the ACM SIGCOMM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication*, Karlsruhe, Germany, 25–29 August 2003.

[CS02] E. Cohen and S. Shenker. Replication Strategies in Unstructured Peer-to-Peer Networks. In *Proceedings of the ACM SIGCOMM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication*, Pittsburgh, Pennsylvania, USA, 19–23 August 2002.

[DNV06] C. Doulkeridis, K. Noervaag, and M. Vazirgiannis. Scalable Semantic Overlay Generation for P2P-based Digital Libraries. In *Procedings of the 10th European Conference on Research and Advanced Technology for Digital Libraries (ECDL)*, 2006.

[FC06] B. Forstner and H. Charaf. Analytical Model for Semantic Overlay Networks in Peer-to-Peer Systems. In *Proceedings of the 5th WSEAS International Conference on Software Engineering, Parallel and Distributed Systems*, pages 82–87, Madrid, Spain, 2006.

[FHJS02] A. Fronczak, J. A. HoImage, M. Jedynak, and J. Sienkiewicz. Higher Order Clustering Coefficients in Barabási-Albert Networks. *Physica A: Statistical Mechanics and its Applications*, 316(1-4), 2002.

[FLSC04] M. Feldman, K. Lai, I. Stoica, and J. Chuang. Scalable and Robust Incentive Techniques for P2P Networks. In *Proceedings of the ACM Conference on Electronic Commerce (EC)*, New York, May 2004.

[GDS+03] K. P. Gummadi, R. J. Dunn, S. Saroiu, S. D. Gribble, H. M. Levy, and J. Zahorjan. Measurement, Modeling, and Analysis of a Peer-to-Peer File-Sharing Workload. In *Proceedings of the 19th ACM Symposium on Operating Systems Principles (SOSP)*, Bolton Landing, NY USA, 19-22 October 2003.

[GKM03] A. J. Ganesh, A.-M. Kermarrec, and L. Massoulié. Peer-to-Peer Membership Management for Gossip-Based Protocols. *IEEE Transactions on Computers*, 52(2), February 2003.

[GLBM01] P. Golle, K. Leyton-Brown, and I. Mironov. Incentives for Sharing in Peer-to-Peer Networks. In *Proceedings of the ACM Conference on Electronic Commerce (EC)*, Tampa, Florida, 14–17 October 2001.

[GM02] H. Garcia-Molina. Peer-to-Peer Data Management. Keynote address at International Conference on Data Engineering (ICDE), 26 February - 1 March 2002. Available from `http://www-db.stanford.edu/peers/`.

[GMY02] H. Garcia-Molina and B. Yang. Efficient Search in Peer-to-Peer Networks. In *Proceedings of 22nd IEEE International Conference on Distributed Computing Systems (ICDCS)*, 2002.

[GSE+00] J. Gray, P. Sundaresan, S. Englert, K. Baclawski, and P. J. Weinberger. Quickly Generating Billion-Record Synthetic Databases. In *Proceedings of ACM International Conference on Management of Data (SIGMOD)*, pages 243–252, Dallas, Texas, 14-19 May 2000.

[HAH07] H. F. Hansen, C. A. Andresen, and A. Hansen. A Quantitative Measure for Path Structures of Complex Networks. *EPL: A Letters Journal Exploring the Frontiers of Physics*, 78, May 2007.

[HB07] A. N. Hidayanto and S. Bressan. Towards a Society of Peers: Expert and Interest Groups in Peer-to-Peer Systems. In *Proceedings of the OTM Workshops on Mobile and Networking Technologies for Social Applications (MONET)*, volume 1, pages 487–496, Vilamoura, Portugal, 2007.

[HCW05] D. Hughes, G. Coulson, and J. Walkerdine. Free Riding on Gnutella Revisited: the Bell Tolls? *IEEE Distributed Systems Online*, 6(6), June 2005.

[HK01] J. Han and M. Kamber. *Data Mining: Concepts and Techniques*, chapter 8: Cluster Analysis. Academic Press, 2001.

[HLY06] K. Y. K. Hui, J. C. S. Lui, and D. K. Y. Yau. Small-World Overlay P2P Networks: Construction, Management and Handling of Dynamic Flash Crowds. *Computer Networks*, 50(15):2727–2746, 2006.

[HRM+08] O. Al-Haj Hassan, L. Ramaswamy, J. A. Miller, K. Rasheed, and E. R. Canfield. Replication in Overlay Networks: A Multi-objective Optimization Approach. In *Proceedings of the 4th International Conference on Collaborative Computing: Networking, Applications and Worksharing (CollaborateCom)*, pages 512–528, Orlando, FL, USA, 13–16 November 2008.

[IKT04] S. Idreos, M. Koubarakis, and C. Tryfonopoulos. P2P-DIET: An Extensible P2P Service that Unifies Ad-hoc and Continuous Querying in Super-Peer Networks. In *Proceedings of the ACM Conference on Management of Data (SIGMOD)*, Paris, France, 13-18 June 2004.

[ITKD04] S. Idreos, C. Tryfonopoulos, M. Koubarakis, and Y. Drougas. Query Processing in Super-Peer Networks with Languages Based on Information Retrieval: the P2P-DIET Approach. In *Proceedings of the International Workshop on Peer-to-peer Computing and Databases (P2P&DB)*, Herakion, Greece, March 2004.

[JM04] M. Jelasity and A. Montresor. Epidemic-Style Proactive Aggregation in Large Overlay Networks. In *Proceedings of the 24th International*

Conference on Distributed Computing Systems (ICDCS), 2004.

[JMB06] G. P. Jesi, A. Montresor, and O. Babaoglu. Proximity-Aware Superpeer Overlay Topologies. In *Proceedings of the International Workshop on Self-Managed Systems & Services (SelfMan)*, Dublin, Ireland, June 2006. Springer-Verlag.

[KA04] A. Kementsietsidis and M. Arenas. Data Sharing Through Query Translation in Autonomous Sources. In *Proceedings of the 30th International Conference on Very Large Databases (VLDB)*, Toronto, Canada, 2004.

[KHG08] S. Y. Ko, I. Hoque, and I. Gupta. Using Tractable and Realistic Churn Models to Analyze Quiescence Behavior of Distributed Protocols. In *Proceedings of the IEEE Symposium on Reliable Distributed Systems (SRDS)*, pages 259–268, Naples, 6–8 October 2008.

[KJ04] I. Klampanos and J. Jose. An Architecture for Information Retrieval over Semi-Collaborating Peer-to-Peer Networks. In *Proceedings of the 19th Annual ACM Symposium on Applied Computing (SAC)*, 2004.

[KP08] G. Koloniari and E. Pitoura. Recall-based Cluster Reformulation by Selfish Peers. In *Proceedings of the ICDE Networking Meets Databases Workshop (NetDB)*, pages 200–205, Cancoon, Mexico, April 2008.

[KTS07] V. Kantere, D. Tsoumakos, and T. K. Sellis. Semantic Grouping of Social Networks in P2P Database Settings. In *Proceedings of 18th International Conference on Database and Expert Systems Applications (DEXA)*, Regensburg, Germany, 3-7 September 2007.

[KTSR09] V. Kantere, D. Tsoumakos, T. Sellis, and N. Roussopoulos. GrouPeer: Dynamic Clustering of P2P Databases. *Information Systems*, 34(1):62–86, 2009.

[LC03] J. Lu and J. Callan. Content-Based Retrieval in Hybrid Peer-to-Peer Networks. In *Proceedings of the 12th ACM International Conference on Information and Knowledge Management (CIKM)*, pages 199–206, 2003.

[LC04] J. Lu and J. Callan. Merging Retrieval Results in Hierarchical Peer-to-Peer Networks. In *Proceedings of the ACM Conference on Research and Development in Information Retrieval (SIGIR)*, pages 472–473, 2004.

[LC05] J. Lu and J. Callan. Federated Search of Text-Based Digital Libraries in Hierarchical Peer-to-Peer Networks. In *Proceedings of the 27th European Conference on Information Retrieval (ECIR)*, 2005.

[LLS04] M. Li, W.-C. Lee, and A. Sivasubramaniam. Semantic Small World: An Overlay Network for Peer-to-Peer Search. In *Proceedings of the 12th IEEE International Conference on Network Protocols (ICNP)*, 2004.

[LNBK02] D. Liben-Nowell, H. Balakrishnan, and D. R. Karger. Analysis of the Evolution of Peer-to-Peer Systems. In *Proceedings of the 21st ACM Symposium on Principles of Distributed Computing (PODC)*, pages 233–242, Monterey, California, 21–24 July 2002.

[LP07] A. Linari and M. Patella. Metric Overlay Networks: Processing Similarity Queries in P2P Databases. In *Proceedings of the 5th International Workshop on Databases, Information Systems and Peer-to-Peer Computing (DBISP2P)*, 2007.

[LRS02] Q. Lv, S. Ratnasamy, and S. Shenker. Can Heterogeneity Make Gnutella Scalable? In *Proceedings of the 1st International Workshop on Peer-to-Peer Systems*, pages 94–103, 2002.

[LSK+05] J. Li, J. Stribling, F. Kaashoek, R. Morris, and T. Gil. A Performance vs. Cost Framework for Evaluating DHT Design Tradeoffs under Churn. In *Proceedings of the 24th Conference on Computer Communications (INFOCOM)*, 2005.

[LT05] A. Loser and C. Tempich. On Ranking Peers in Semantic Overlay Networks. In *Proceedings of the 3rd Conference on Professional Knowledge Management - Experiences and Visions (WM)*, Kaiserslautern, Germany, 10-13 April 2005.

[LW06] A. Linari and G. Weikum. Efficient Peer-to-Peer Semantic Overlay Networks Based on Statistical Language Models. In *Proceedings of the International Workshop on Information Retrieval in Peer-to-Peer Networks (P2PIR)*, pages 9–16, Arlington, Virginia, USA, 2006. ACM.

[LWSN03] A. Loser, M. Wolpers, W. Siberski, and W. Nejdl. Semantic Overlay Clusters within Super-Peer Networks. In *Proceedings of the 29th VLDB International Workshop on Databases, Information Systems and Peer-to-Peer Computing (DBISP2P)*, Berlin, Germany, September 2003.

[LYRL07] D. Leonard, Z. Yao, V. Rai, and D. Loguinov. On Lifetime-Based Node Failure and Stochastic Resilience of Decentralized Peer-to-Peer Networks. *IEEE/ACM Transactions on Networking*, 15(5):644–656, June 2007.

[Mar02] E. P. Markatos. Tracing a Large-Scale Peer to Peer System: An Hour in the Life of Gnutella. In *Proceedings of the 2nd IEEE International Symposium on Cluster Computing and the Grid (CCGRID)*, Berlin, Germany, 21–24 May 2002.

[MGGP08] B. Mitra, S. Ghose, N. Ganguly, and F. Peruany. Stability Analysis of Peer-to-Peer Networks Against Churn. *Pramana - Journal of Physics*, 71(2):263–273, August 2008.

[MKL+02] D. S. Milojicic, V. Kalogeraki, R. Lukose, K. Nagaraja, J. Pruyne, B. Richard, S. Rollins, and Z. Xu. Peer-to-Peer Computing. Technical Report HPL-2002-57, HP Laboratories, Palo Alto, March 2002.

[MNR02] D. Malkhi, M. Naor, and D. Ratajczak. Viceroy: a Scalable and Dynamic Emulation of the Butterfly. In *Proceedings of the ACM Symposium on Principles of Distributed Computing (PODC)*, pages 183–192, 2002.

[MRT+05] A. Mostéfaoui, M. Raynal, C. Travers, S. Patterson, D. Agrawal, and A. El Abbadi. From Static Distributed Systems to Dynamic Systems. In *Proceedings of the IEEE Symposium on Reliable Distributed Systems (SRDS)*, pages 109–118, Orlando, 26–28 October 2005.

[MSZ05] S. Merugu, S. Srinivasan, and E. Zegura. Adding Structure to Unstructured Peer-to-Peer Networks: the Use of Small-World Graphs. *Parallel and Distributed Computing*, 65(2):142–153, 2005.

[NSC02] C.H. Ng, K.C. Sia, and C.H. Chang. Advanced Peer Clustering and Firework Query Model in the Peer-to-Peer Network. In *Proceedings of the 11th International World Wide Web Conference (WWW)*, May 2002. Poster.

[NWQ+02] W. Nejdl, B. Wolf, C. Qu, S. Decker, M. Sintek, A. Naeve, M. Nilsson, M. Palmer, and T. Risch. EDUTELLA: a P2P Networking Infrastructure based on RDF. In *Proceedings of 11th International World Wide Web Conference (WWW)*, Honolulu, Hawaii, USA, 7-11 May 2002. ACM Press.

[Ora01] Andy Oram. *Peer-to-Peer : Harnessing the Power of Disruptive Technologies*. Amazon, 2001.

[PGSM05] C. Pairota, P. García, A. F. Gómez Skarmetab, and R. Mondéjar. Towards New Load-Balancing Schemes for Structured Peer-to-Peer Grids. *Future Generation Computer Systems*, 21(1):125–133, January 2005.

[PGW+08] J. A. Pouwelse, P. Garbacki, J. Wang, A. Bakker, J. Yang, A. Iosup, D. H. J. Epema, M. J. T. Reinders, M. R. van Steen, and H. J. Sips. TRIBLER: A social-Based Peer-to-Peer system. *Concurrency and Computation: Practice and Experience*, 20(2):127–138, 2008.

[PLM+08] W. Penzo, S. Lodi, F. Mandreoli, R. Martoglia, and S. Sassatelli. Semantic Peer, Here are the Neighbors you Want! In *Proceedings of the International Conference on Extended Database Technology (EDBT)*, 2008.

[PMW07] J. X. Parreira, S. Michel, and G. Weikum. p2pDating: Real Life Inspired Semantic Overlay Networks for Web Search. *Information Processing and Management*, 43(1):643–664, January 2007.

[PS05] T. G. Papaioannou and G. D. Stamoulis. An Incentives' Mechanism Promoting Truthful Feedback in Peer-to-Peer Systems. In *Proceedings*

of the International Workshop on Global and Peer-to-Peer Computing (GP2PC), 2005.

[RC04] M. E. Renda and J. Callan. The Robustness of Content-Based Search in Hierarchical Peer-to-Peer Networks. In *Proceedings of the International Conference on Information and Knowledge Management (CIKM)*, pages 562–570, 2004.

[RD01] A. Rowstron and P. Druschel. Pastry: Scalable, Distributed Object Location and Routing for Large-Scale Peer-to-Peer Systems. In *Proceedings of IFIP/ACM International Middleware Conference*, pages 329–350, Heidelberg, Germany, November 2001.

[Ree01] W. J. Reed. The Pareto, Zipf and other Power Laws. *Economics Letters*, 74(1):15–19, December 2001.

[RFH+01] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker. A Scalable Content-Addressable Network. In *Proceedings of the ACM Annual Conference of the Special Interest Group on Data Communication (SIGCOMM)*, San Diego, California, 27-31 August 2001.

[RMJ07] M. J. Rattigan, M. Maier, and D. Jensen. Graph Clustering with Network Structure Indices. In *Proceedings of the 24th Annual International Conference on Machine Learning (ICML)*, 20-24 June 2007.

[RP08a] P. Raftopoulou and E. G. M. Petrakis. A Measure for Cluster Cohesion in Semantic Overlay Networks. In *Proceedings of the 6th International Workshop on Large-Scale Distributed Systems for Information Retrieval (LSDS-IR)*, Napa Valley, California, October 2008. ACM.

[RP08b] P. Raftopoulou and E. G. M. Petrakis. iCluster: a Self-Organising Overlay Network for P2P Information Retrieval. In *Proceedings of the 30th European Conference on Information Retrieval (ECIR)*, Glasgow, Scotland, 30 March - 3 April 2008.

[RPT09] P. Raftopoulou, E. G. M. Petrakis, and C. Tryfonopoulos. Rewiring Strategies for Semantic Overlay Networks. *Journal of Distributed and Parallel Databases*, 26(2-3):181–205, December 2009.

[RPTW08] P. Raftopoulou, E. G. M. Petrakis, C. Tryfonopoulos, and G. Weikum. Information Retrieval and Filtering over Self-Organising Digital Libraries. In *Proceedings of the 12th European Conference on Research and Advanced Technology for Digital Libraries (ECDL)*, Aarhus, Denmark, 14-19 September 2008.

[Sal89] G. Salton. *Automatic Text Processing: The Transformation Analysis and Retrieval of Information by Computer*. Addison-Wesley, 1989.

[SBDZ08] J. Sedmidubsky, S. Barton, V. Dohnal, and P. Zezula. Adaptive Approximate Similarity Searching through Metric Social Networks. In *Proceedings of the 24th International Conference on Data Engineering (ICDE)*, pages 1424–1426, April 2008. Poster.

[Sch04] C. Schmitz. Self-Organization of a Small World by Topic. In *Proceedings of the 1st International Workshop on Peer-to-Peer Knowledge Management (P2PKM)*, Boston, MA, August 2004.

[Sch05] C. Schmitz. Towards Self-Organizing Communities in Peer-to-Peer Knowledge Management. In *Proceedings of ESWC 2005 Workshop on Ontologies in Peer-to-Peer Communities*, Heraklion, Greece, May 2005.

[SCL+05] J. Stribling, I. Councill, J. Li, M. Kaashoek, D. Karger, R. Morris, and S. Shenker. Overcite: A Cooperative Digital Research Library. In *Proceedings of 4th International Workshop on Peer-to-Peer Systems (IPTPS)*, 2005.

[SDCM06] J. Sacha, J. Dowling, R. Cunningham, and R. Meier. Discovery of Stable Peers in a Self-Organising Peer-to-Peer Gradient Topology. In *Proceedings of the 6th International DAIS Conference*, volume 4025 of *LNCS*, pages 70–83, 2006.

[SGG02] S. Saroiu, P. K. Gummadi, and S. D. Gribble. A Measurement Study of Peer-to-Peer File Sharing Systems. In *Proceedings of SPIE/ACM Conference on Multimedia Computing and Networking (MMCN)*, 2002.

[SHA95] Secure Hash Standard. Technical Report 180-1, National Institute of Standards and Technology, 1995.

[SKK00] M. Steinbach, G. Karypis, and V. Kumar. A Comparison of Document Clustering Techniques. In *Proceedings of KDD Workshop on Text Mining*, 2000.

[SKT09] J. Shin, T. Kim, and S. Tak. P2P Incentive Mechanism for File Sharing and Cooperation. In *Proceedings of the International Conference in Computational Science and Its Applications (ICCSA)*, volume 5592, pages 912–923, Suwon, Korea, 29 June–1 July 2009.

[SL06] C. Schmitz and A. Loser. How to Model Semantic Peer-to-Peer Overlays? In *Proceedings of the International Workshop on Information Retrieval in Peer-to-Peer Networks (P2PIR)*, Dresden, 2006.

[SMLN+03] I. Stoica, R. Morris, D. Liben-Nowell, D. R. Karger, M. Frans Kaashoek, F. Dabek, and H. Balakrishnan. Chord: A Scalable Peer-to-Peer Lookup Protocol for Internet Applications. *IEEE/ACM Transactions on Networking*, 11(1):17–32, February 2003.

[SMZ03] K. Sripanidkulchai, B. Maggs, and H. Zhang. Efficient Content Location using Interest-Based Locality in Peer-to-Peer Systems. In *Proceedings of the IEEE Conference on Computer Communications (INFOCOM)*, volume 3, pages 2166–2176, San Francisco, CA, March 2003.

[SR06] D. Stutzbach and R. Rejaie. Understanding Churn in Peer-to-Peer Networks. In *Proceedings of the Internet Measurement Conference (IMC)*, pages 189–201, Rio de Janeiro, Brazil, October 25-27 2006.

[SRS05] D. Stutzbach, R. Rejaie, and S. Sen. Characterizing Unstructured Overlay Topologies in Modern P2P File-Sharing Systems. In *Proceedings of the Internet Measurement Conference (IMC)*, Berkeley, USA, 19-21 October 2005.

[TIK05] C. Tryfonopoulos, S. Idreos, and M. Koubarakis. LibraRing: An Architecture for Distributed Digital Libraries Based on DHTs. In *Proceedings of the 9th European Conference on Research and Advanced*

Technology for Digital Libraries (ECDL), pages 25–36, Vienna, Austria, September 2005.

[TK05] S. Tewari and L. Kleinrock. Analysis of Search and Replication in Unstructured Peer-to-Peer Networks. In *Proceedings of the ACM International Conference on Measurements and Modeling of Computer Systems (SIGMETRICS)*, Banff, Alberta, Canada, 6–10 June 2005.

[TR03] D. Tsoumakos and N. Roussopoulos. A Comparison of Peer-to-Peer Search Methods. In *Proceedings of the International Workshop on the Web and Databases (WebDB)*, pages 61–66, 2003.

[TXD03] C. Tang, Z. Xu, and S. Dwarkadas. Peer-to-Peer Information Retrieval Using Self-Organizing Semantic Overlay Networks. In *Proceedings of the ACM Annual Conference of the Special Interest Group on Data Communication (SIGCOMM)*, Karlsruhe, Germany, August 2003.

[TXKN03] P. Triantafillou, C. Xiruhaki, M. Koubarakis, and N. Ntarmos. Towards High Performance Peer-to-Peer Content and Resource Sharing Systems. In *Proceedings of the 1st International Conference on Innovative Data Systems Research (CIDR)*, January 2003.

[VSI07] S. Voulgaris, M. Steen, and K. Iwanicki. Proactive Gossip-based Management of Semantic Overlay Networks. *Concurrency and Computation: Practice and Experience*, 19(17), 2007.

[Wat99] D.J. Watts. *Small Worlds - The Dynamics of Networks between Order and Randomneess*. Princeton University Press, Princeton, New Jersey, 1999.

[WS98] D.J. Watts and S.H. Strogatz. Collective Dynamics of 'Small-World' Networks. *Nature*, 393:440–442, 4 June 1998.

[XC99] J. Xu and W.B. Croft. Cluster-Based Language Models for Distributed Retrieval. In *Proceedings of the 22th Annual International ACM Conference on Research and Development in Information Retrieval (SIGIR)*, 1999.

[YGM99] T.W. Yan and H. Garcia-Molina. The SIFT Information Dissemination System. In *Proceedings of the International Conference on Transactions on Database Systems (TODS)*, 1999.

[YGM01] B. Yang and H. Garcia-Molina. Comparing Hybrid Peer-to-Peer Systems. In *Proceedings of the International Conference on Very Large Databases (VLDB)*, pages 561–570, 2001.

[YGM03] B. Yang and H. Garcia-Molina. Designing a Super-Peer Network. In *Proceedings of the International Conference on Data Engineering (ICDE)*, March 2003.

[YLWL06] Z. Yao, D. Leonard, X. Wang, and D. Loguinov. Modeling Heterogeneous User Churn and Local Resilience of Unstructured P2P Networks. In *Proceedings of the 14th IEEE International Conference on Network Protocols (ICNP)*, pages 32–41, Santa Barbara, California, USA, 12-15 November 2006.

[ZHS+04] B. Y. Zhao, L. Huang, J. Stribling, S. C. Rhea, A. D. Joseph, and J. Kubiatowicz. Tapestry: A Resilient Global-Scale Overlay for Service Deployment. *IEEE Journal on Selected Areas in Communications*, 22(1), January 2004.

[ZTW07] C. Zimmer, C. Tryfonopoulos, and G. Weikum. MinervaDL: An Architecture for Information Retrieval and Filtering in Distributed Digital Libraries. In *Proceedings of the 11th European Conference on Research and Advanced Technology for Digital Libraries (ECDL)*, pages 25–36, Budapest, Hungary, September 2007.

Index