SHRACK: A SELF-ORGANIZING PEER-TO-PEER SYSTEM FOR DOCUMENT SHARING AND TRACKING

by

Hathai Tanta-ngai

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy

at

Dalhousie University
Halifax, Nova Scotia
April 2010

DALHOUSIE UNIVERSITY

FACULTY OF COMPUTER SCIENCE

The undersigned hereby certify that they have read and recommend to the Faculty of Graduate Studies for acceptance a thesis entitled "SHRACK: A SELF-ORGANIZING PEER-TO-PEER SYSTEM FOR DOCUMENT SHARING AND TRACKING" by Hathai Tanta-ngai in partial fulfillment of the requirements for the degree of Doctor of Philosophy.

Dated: April 23, 2010

External Examiner: Dr. Jimmy Huang

Research Supervisors: Dr. Evangelos E. Milios

Dr. Vlado Keselj

Examining Committee: Dr. Nick Cercone

Dr. Nur Zincir-Heywood

DALHOUSIE UNIVERSITY

DATE: April 23, 2010

AUTHOR: Hathai Tanta-ngai

TITLE: SHRACK: A SELF-ORGANIZING PEER-TO-PEER SYSTEM FOR DOCUMENT SHARING AND TRACKING

DEPARTMENT OR SCHOOL: Faculty of Computer Science

DEGREE: PhD CONVOCATION: October YEAR: 2010

Permission is herewith granted to Dalhousie University to circulate and to have copied for non-commercial purposes, at its discretion, the above title upon the request of individuals or institutions.

Signature of Author

For my caring husband, Tony
my precious son, Leonard
my supportive parents, Boonchai and Suwannee
and my beloved sister, Kamonwan

Table of Contents

List of Tables

List of Figures

Abstract

Given a set of peers with overlapping interests where each peer wishes to keep track of new documents that are relevant to their interests, we propose Shrack—a self-organizing peer-to-peer (P2P) system for document sharing and tracking. The goal of a document-tracking system is to disseminate new documents as they are published. We present a framework of Shrack and propose a gossip-like pull-only information dissemination protocol. We explore and develop mechanisms to enable a self-organizing network, based on common interest of document sets among peers.

Shrack peers collaboratively share new documents of interest with other peers. Interests of peers are modeled using relevant document sets and are represented as peer profiles. There is no explicit profile exchange between peers and no global information available. We describe how peers create their user profiles, discover the existence of other peers, locally learn about interest of other peers, and finally form a self-organizing overlay network of peers with common interests.

Unlike most existing P2P file sharing systems which serve their users by finding relevant documents based on an instant query, Shrack is designed to help users that have long-term interests to keep track of relevant documents that are newly available in the system. The framework can be used as an infrastructure for any kind of documents and data, but in this thesis, we focus on research publications.

We built an event-driven simulation to evaluate the performance and behaviour of Shrack. We model simulated users associated with peers after a subset of authors in the ACM digital library metadata collection. The experimental results demonstrate that the Shrack dissemination protocol is scalable as the network size increases. In addition, self-organizing overlay networks, where connections between peers are based on common interests as captured by their associated document sets, can help improve the relevance of documents received by peers in terms of F-score over random peer networks. Moreover, the resulting self-organizing networks have the characteristics of social networks.

List of Abbreviations and Symbols Used

$C^t(p_i, p_j)$	a common interest score between a local peer p_i and a known peer p_j at time t
$C_I^t(p_i, p_j)$	an item-based quantification of $C^t(p_i, p_j)$
$C_T^t(p_i, p_j)$	a term-based quantification of $C^t(p_i, p_j)$
$D^t(p_i)$	the set of documents whose metadata was received by peer p_i up to time t, excluding the self-published documents $S^t(p_i)$
$E(p_i)$	the set of connections between provider peers of p_i
$F_c(x)$	the complementary cumulative distribution function of x
IDF_v	the estimated global IDF of term v
$K^t(p_i, p_j)$	the set of documents that a local peer p_i receives via peer p_j up to time t
K^t	the set of all peers known to p_i at time t
$K_I^t(p_i, p_j)$	the profile of p_j in the known peer profile of p_i at time t
$K_T^t(p_i, p_j)$	the term-based profile of p_j in the known peer profile of p_i at time t
$L^t(p_i)$	a union of the set of relevant documents whose metadata was received by peer p_i and the set of documents that peer p_i has published up to time t
$L_I^t(p_i)$	the item-based local peer profile of peer p_i at time t
$L_T^t(p_i)$	the term-based local peer profile of peer p_i at time t
M	a set of Shrack messages
N	the maximum number of provider peers to which each peer may connect
$N(p_i)$	the set of provider peers of p_i
P^t	the set of all documents that are published by every peer in the network up to time t
P^{τ_k}	the set of documents published during time slot τ_k

Q^t	a relevant document set
Q^t_T	a term-based profile of Q^t
$R^t(p_i)$	the set of documents that are published up to time t in the entire network and are relevant to peer p_i
$R^{\tau_k}(p_i)$	the set of documents relevant to peer p_i that are published during time slot τ_k
$S^t(p_i)$	the set of documents that peer p_i publishes up to time t
$S^{\tau_k}(p_i)$	$S^{\tau_k}(p_i)$ is the set of documents published by p_i during time slot τ_k
TF^t_v	the TF of the term v at time t
V	the number of terms in the vocabulary
X	in-degree of a node p_i in the network
β	an exploration parameter of the hybrid provider peer selection strategy
τ_k	an arbitrary time slot
$\vec{K}^t(p_i, p_j)$	a term-weight vector representing $K^t_T(p_i, p_j)$
$\vec{L}^t(p_i)$	a term-weight vector representing $L^t_T(p_i)$
$\vec{Q}^t$	a term-weight vector representing Q^t_T
$\vec{d}$	a term-frequency vector of the metadata of document d
d, d^θ	a document or metadata of a document
d_v	the frequency of the term number v in the metadata of document d
m	a Shrack message
n	the number of glossip targets
p, p_i, p_j	a peer in a peer-to-peer network
t^k_e	the end of the time slot τ_k
t^k_s	the start of the time slot τ_k
t^k_{end}	a time when the dissemination of documents in P^{τ_k} ends

t_p	a minimum period of time that dissemination of P^{τ_k} remains unchange to determine t^k_{end}
w^t_v	the term weight $TF \times IDF$ of the term v at time t
z	the number of peers in the network

ACM	Association for Computing Machinery
ANOVA	Analysis of Variance
CCDF	Complementary Cumulative Distribution Function
CCO	The Network Clustering Coefficient
CCS	The ACM Computing Classification System
CPL	The Characteristic Path Length
DHT	Distributed Hash Table
ID	Identifier
IDF	Inverse Document Frequency
P2P	Peer-to-Peer
TF	Term Frequency
TTL	Time-To-Live

Acknowledgements

This thesis would not have been possible without support from several people. I would like to express my deep gratitude to my supervisors, Dr. Evangelos E. Milios and Dr. Vlado Kešelj, for their understanding, guidance, and encouragement throughout the development of this thesis. I had such a wonderful experience working with them.

I would like to express my appreciation to my committee members, Dr. Nick Cercone and Dr. Nur Zincir-Heywood, for their thoughtful and valuable comments. My appreciation also goes to Dr. Jack Duffy for his helpful suggestions on the statistical analysis. I would also like to thank Dr. Jimmy Huang, my external examiner, for his insightful feedback. I am also thankful to Dr. Jacob Slonim and Dr. Michael McAllister for their guidance in the early stages of my study.

Throughout my study at Dalhousie University, I was fortunate to have met my friends and colleagues with whom I shared experiences and discussions, both in academic and in social life. I would also like to thank my friends and colleges at King Mongkut's University of Technology Thonburi for their encouragement and support.

I owe my foremost appreciation to my parents, Boonchai and Suwannee, and my sister, Kamonwan, for their endless love, encouragement, support, and sacrifice. The belief and trusts that they gave me empowered my spirit to pass through all obstacles in my PhD journey. I would also like to extend my thanks to all the members of my in-law family for their cheerfulness and encouragement.

Finally, my utmost gratitude to my husband, Tony, and my son, Leonard, for their unconditional love and sacrifice. Their constant encouragement and support always cheered me up when I was lost. I am thankful for their patience and company through every stages of my PhD journey.

Chapter 1

Introduction

Researchers with a common interest normally form collaborations to advance their knowledge or achieve the same goal. They may collaborate by sharing resources or ideas through discussion, or by working together to integrate their results. The collaborations are not limited to researchers in the same organization. Several Internet applications have been used to help researchers form collaborations around the world. Email, the Web, and chat are examples of such applications.

The Internet also serves as a medium for researchers to share documents of interest. Tools have been developed to help researchers find documents over the Internet. These tools include search engines, mailing lists, and on-line archives. Recently, online social bookmarking services such as Citeulike [5], Bibsonomy [4] and ResearchGATE [6] gain interests in research communities to share documents of interests among each other. However, existing tools pose a number of challenges for researchers in getting informed about newly published documents, which from now will be referred as new documents. For instance, to keep track of new documents of interest, researchers need to revisit search engines or on-line archives multiple times and issue the same query. Mailing lists can also be used for keeping track of new documents, however, researchers need to find where to post and subscribe for information. Moreover, these tools usually lack autonomous mechanisms to select documents based on individual interest. Even though Citeulike provides a watchlist option for users to keep track of new documents that are relevant to each page, the users need to manage the watchlists manually. Since researchers who work in the same area are usually interested in the same documents, an autonomous collaborative environment to share and keep track of new documents in the area will lessen effort of each researcher to individually keep track of their documents of interest.

We propose Shrack, a peer-to-peer (P2P) system for document sharing and tracking, as a framework for researchers to share and keep track of new documents among researchers who are interested in similar research areas. Each peer is associated with an individual researcher or a research organization. A peer that is used by a research organization can be viewed as a super peer—a server that supports group activities for document sharing such as a digital library or a document archive. There is neither explicit query nor explicit profile exchange in Shrack, each peer automatically learns about the user interests, and then keeps track and presents relevant documents to the user based on the user feedback and received information.

1.1 Motivation

Our motivation is to enable a direct collaboration among researchers by sharing and keeping track of documents of interest. Even though there are many tools to search for research publications, there is no autonomous system where a researcher can keep track of research publications based on individual interest. In Shrack, we aim to build an environment where the system can autonomously form groups of researchers that have similar interests to collaboratively share and keep track of new research publications among them. We design Shrack as a peer-to-peer system and use pull-only communication to disseminate information among peers. The researchers do not need to explicitly issue a query nor exchange their profiles to participate in the system. The researchers first join the system based on real world contacts and then the system will autonomously learn about the researchers' interests and form connections with other researchers having similar interests. As a result, researchers can autonomously form a collaborative group to learn about new research publications that are of interest among them.

Shrack is different from traditional search engines in that Shrack's peers periodically keep in touch with other peers to update and share new documents of interests. Shrack's peers will automatically inform their users when they learn about new documents that are of relevance to the interests of the peer users. On the other hand, in traditional search, users issue search queries to find relevant documents that are currently available in the system. To keep track of new relevant documents, the users

need to periodically revisit the search engines to issue the queries to get new results. We introduce Shrack as an alternative way for researchers to keep track of documents of interests, or somewhat similar to continuous search queries. In addition, the tracking ability also makes Shrack different from other existing P2P systems, which usually support only search services.

1.1.1 Why a Peer-to-Peer System

Shrack is designed as a peer-to-peer system because peer-to-peer systems allow researchers to directly exchange research papers of mutual interest on the Internet without relying on centralized servers. Peer-to-peer systems are robust against single points of failure and have the potential of being scalable as the system size increases. Researchers (peers) and research organizations (super peers) can share resources and form collaborations at a local level without the privacy concerns and the system management cost of centralized servers. Moreover, researchers in different organizations can individually form their own collaborations.

Between the two peer-to-peer system architectures, unstructured and structured peer-to-peer systems, we choose to design Shrack as an unstructured peer-to-peer system. An unstructured peer-to-peer system, such as Gnutella [29] and KaZaa [42], is a peer-to-peer system that does not have control over network topology and data placement. Each peer maintains a list of addresses of other chosen peers randomly or based on mutual interest.

In contrast to an unstructured peer-to-peer system, a structured peer-to-peer system has a predefined network topology that defines which peer's addresses each peer has to maintain. Due to the dynamic behaviour of peers, which can independently join or leave the system, structured peer-to-peer systems require additional maintenance cost to synchronize peers according to the predefined topology.

Moreover, most structured peer-to-peer systems [60, 62, 68, 80], which are designed based on a distributed hash table, have restrictive control over data placement on each peer according to unique keys of the data. Although structured peer-to-peer systems provide guarantee and efficient algorithms to locate data items in the system, each peer needs to know the unique keys of the data items in advance in order to acquire

the items.

We design Shrack as an unstructured peer-to-peer system because: (1) we would like Shrack to allow peers to form collaborations based on their interests, not based on a predefined structure, (2) we would like peers to keep data according to their interests not according to unique keys of the data, and (3) Shrack is built for peers to acquire new documents, therefore, it would be difficult for peers to have unique keys of the new documents in advance to locate the new documents in structured peer-to-peer systems.

1.1.2 Why Pull-Only Communication

Shrack uses pull-only communication to disseminate shared information among peers. Pull communication refers to a communication where the information transfer is initiated by a destination node or node that wants to receive information, e.g. as in the HTTP or POP protocols. On the other hand, push communication refers to a communication where the information transfer is initiated by the source node, e.g. as in the SMTP protocal for email transfer. Consequently, pull communication gives control to peers who seek information to select from which sources and when to pull shared information. The system stimulates cooperation by learning and sharing new documents within a group of researchers having common interest, which is determined and controlled by peers that seek information. Peers that provide good sources of information can build their reputation and make themselves and their work visible. In addition it is easier for researchers to publish documents at one place than push or inform potential readers in many different places or mailing lists.

With pull-only communication, peers that want to provide information would make themselves available for other peers to access the information. Peers that want to receive information, when they are on-line will seek and contact peers that provide information. However, designated peers, called super peers, can act as information aggregators and they are expected to be always on-line. Many pull-based applications are widely used in the Internet such as Web browsers, and RSS readers.

1.2 Thesis Statement

The goal of this research is to develop a peer-to-peer system framework to provide support to researchers in forming direct collaboration for document sharing and tracking in a peer-to-peer environment based on their interests. We hypothesize that:

1. Pull-only information dissemination is scalable in a large-scale peer-to-peer system.

2. Self-organizing networks based on common interest of document sets can enhance relevance of documents received by peers.

To validate the first hypothesis, we propose a pull-only information dissemination protocol to disseminate information of new documents among peers in Shrack. A simulation is built to test its scalability and performance of the protocol in large-scale networks assuming that all peers are interested in all documents published. To validate the second hypothesis, we create a user's interest model based on publication and authorship to create artificial peer users and their interests. After that, we evaluate performance of Shrack overlay networks according to three provider peer selection strategies; namely, a common interest strategy, a random strategy, and a hybrid strategy.

1.3 Contributions

We built a peer-to-peer system model that facilitates sharing and keeping track of new documents of interest among researchers. We proposed Shrack as a framework for such system. The emphasis of this research is on developing and evaluating a pull-only information dissemination protocol and self-organizing overlay strategies in the proposed framework.

We summarize the contributions of this research as follows:

1. Shrack framework: Defined a peer-to-peer framework to facilitate researchers in forming direct collaboration for document sharing and tracking based on their interests.

2. Shrack dissemination protocol: Developed a pull-only information dissemination protocol that is scalable as the network size increases.

3. Self-organizing overlay networks: Developed self-organizing overlay strategies that form groups of peers having similar interests without explicitly exchange user profiles to enhance relevance of documents received by peers.

4. Simulation Environment: Built an event-driven simulation environment to validate dissemination protocols and self-organizing overlay strategies according to the Shrack framework.

5. Evaluation Methodologies: Defined evaluation techniques and metrics to evaluate the dissemination protocol and self-organizing Shrack networks.

1.4 Overview of the Thesis

This thesis consists of ten chapters. The next chapter, Chapter 2, provides background and related work to this thesis. Chapter 3 describes the Shrack framework including defining research problems and introducing formal definitions of terminology using throughout the thesis. Chapter 4 presents the Shrack prototype system. Chapter 5 describes the experimental environment that we use to evalution the prototype system. Chapter 6 presents experiments on scalability of Shrack dissemination protocol. Chapter 7 presents experiments on self-organizing Shrack network. Chapter 8 and Chapter 9 present the effect of peer profile representation and Time-To-Live (TTL) on the self-organizing Shrack network. Finally, Chapter 10 provides conclusion and discusses future works of the thesis.

Chapter 2

Background and Related Work

This chapter provides background and related work to give readers knowledge and intuition behind this thesis. We first discuss data and document sharing networks in Section 2.1. Then we give an overview of peer-to-peer systems in Section 2.2. In Section 2.3, we provide a review of peer-to-peer systems for data sharing. In Section 2.4, we describe the original gossip protocol—a probabilistic dissemination protocol that is widely used in large-scale systems. After that, in Section 2.5, we give a review of the existing approaches to resource discovery in unstructured peer-to-peer systems. In Section 2.6 we review the Re-Coll system that motivated the pull-only dissemination protocol in Shrack. We review several systems that are based on the JXTA platform in Section 2.7. In Section 2.8, we present related work in collaborative filtering. Finally, Section 2.9 provides a summary of this Chapter.

2.1 Data and Document Sharing Networks

Several well-known centralized research article repositories or digital libraries, such as the ACM digital library [7] and IEEE Xplore [8], have been available to research communities for several years. However, additional efforts are still emerging to collect documents from many digital libraries to create either individual interest group repositories or global online repositories. Examples of such repositories include arXiv [1], Web interface CiteSeer [23], Google Scholar [2], DBLP [45], and AMS Digital Mathematics Registry (DMR) [14]. These repositories demonstrate the interest of researchers in sharing documents through the Internet. While such repositories have proved to be invaluable to the research community and demonstrated that they do scale up to certain non-trivial amount of documents, they have also exposed weaknesses of the centralized approach.

Centralized servers create a bottleneck in accessing and collecting documents. As

the number of users and documents increases, centralized servers need to increase their capacities to support users' activities requiring high bandwidth and storage. Currently, CiteSeer alleviates this problem through mirrors. In addition, scaling up CiteSeer through distributed, cooperative servers was proposed in OverCite, a cooperative digital research library [69]. However, these systems increase the performance through server-sided capacity, which require dedicate servers.

Current models of digital libraries focus on collecting documents and allowing users to search and access document collections. However, there is no guarantee that all documents will be stored in a particular digital library. Users need to visit several digital libraries to search for documents of interest. In addition, users do not know when a digital library receives new documents in its collection. Users have to regularly visit a digital library to check for the new documents, or subscribe for a particular library to be notified when the new documents arrive.

In this thesis, we are interested in designing a mechanism for researchers to collaboratively keep track of new documents among those who have common interest. In our model, researchers are keeping track of documents of interest not from a particular digital library but from a collection of documents that are of interest to the collaborative group, which can be from any digital library or any individual collection.

2.2 Introduction to Peer-to-Peer Systems

Since 2000, peer-to-peer systems or peer-to-peer networks have drawn attention to the research community as a new way of communication allowing individual computers to communicate directly with each other and to share information as resources without using special servers. Some examples of such peer-to-peer systems include Gnutella [29], Freenet [24], CAN [60], and Chord [68]. Peer-to-peer systems are known to be robust against single point of failure and have the potential of being scalable as the system size increases [47, 64, 79]. In the following, we provide a definition of peer-to-peer networks.

2.2.1 Definition of Peer-to-Peer Networks

Throughout this work, we adopt Schollmeier's definition of a peer-to-peer network, which he presented at the first international conference on peer-to-peer computing:

> A distributed network architecture may be called a Peer-to-Peer (P-to-P, P2P, ...) network, if the participants share a part of their own hardware resources (processing power, storage capacity, network link capacity, printers, ...). These shared resources are necessary to provide the service and content offered by the network (e.g. file sharing or shared workspaces for collaboration). They are accessible by other peers directly, without passing intermediary entities. The participants of such a network are thus resource (Service and content) providers as well as resource (Service and content) requestors (Servent-concept) [66].

2.2.2 Peer-to-Peer Overlay Networks

Normally, peer-to-peer networks are viewed as overlay networks. An overlay network is a network that is built on top of another network. Peers in a peer-to-peer network are connected with virtual or logical links with other peers, called neighbours. The logical links identify communication paths, which are composed of many physical links in the underlying network. There are two major peer-to-peer overlay architectures; (1) structured peer-to-peer overlay networks or structured peer-to-peer systems and (2) unstructured peer-to-peer overlay networks or unstructured peer-to-peer systems.

Unstructured Peer-to-Peer Systems In unstructured peer-to-peer systems, such as Gnutella [29], peers are normally organized into a random graph and use floods or random walks to discover data stored in the overlay network. Unstructured peer-to-peer systems support arbitrarily complex queries and do not impose any constraints on the peer topology or on data placement. However, rare data items are hard to find efficiently because they require an exhaustive search through the overlay network. Numerous works have focused on improving searches in unstructured peer-to-peer systems [22, 28, 39, 50]. Recently, semantic and social network concepts have

been adopted to create an overlay network to improve resource discovery in unstructured peer-to-peer systems. Examples of such systems include REMINDIN' [72], P2PDating [57], and Tribler [59].

Structured Peer-to-Peer Systems Structured peer-to-peer systems have a predefined network topology to specify how peers are connected. Well-known structured peer-to-peer systems, such as Chord [68], Pastry [62], Tapestry [80], and CAN [60], are designed based on Distributed Hash Tables (DHT). DHT imposes constraints on both data placement and network topology. Structured peer-to-peer systems are known to have scalable and efficient overlay routing for requests with specific keys or exact match queries. In addition, these systems are designed based on uniform resources; that is, each peer is expected to dedicate an equal amount of storage, processing, and network bandwidth. A number of works [35, 30, 70] propose to relax this assumption of uniformity to improve routing efficiency. However, a major drawback of the structured peer-to-peer systems is that although they are well-suited for exact match lookups using unique identifiers they do not directly support text search [46].

GridVine [9] is a structured peer-to-peer system that addresses this limitation by using a semantic overlay to support semantic queries. Castro et al. [21] provide a detailed comparison of the unstructured and structured peer-to-peer systems.

We design Shrack as an unstructured peer-to-peer system. The main criteria that led to this design decision are simplicity, flexibility, and the total lack of central control. We do not want to put constraints on data placement or network topology as in structured peer-to-peer systems. We believe that peers should have the freedom to connect to any peers they want and keep only information in which they are interested.

2.3 Peer-to-Peer System for Data Sharing

A peer-to-peer system is an alternative approach for data or document sharing network that allows researchers to directly share their data or document collections with each other. Each researcher has its own repository and allows others to access the shared information. Hence, the communication bandwidth and repository storage can

be distributed among researchers in the community, allowing the system to scale up as the number of researchers in the system increases. Peers form a collaborative search to find target documents in the system. For an exhaustive survey of peer-to-peer data sharing technologies, we refer the readers to the work of Androutsellis-Theotokis and Spinellis [15].

2.4 Gossip Protocol

Gossip-based probabilistic multicast protocols rely on a peer-to-peer interaction model for multicasting a message and are scalable since the load is distributed among peers in the network [43]. They use redundant messages to achieve reliability and fault tolerance. Many peer-to-peer systems use gossip-based protocols as a major protocol to exchange their expertise or disseminate information among peers [36, 57, 65, 74]. Jelasity presents a framework for peer sampling services in unstructured peer-to-peer systems using a gossip-based protocol [37], which could be used for disseminating information among peers.

follow. The original gossip protocol assumes that there is a membership protocol which provides each peer with a randomized partial knowledge of other peers in the system. A *notification* or gossip message contains an event to disseminate to the whole group. When a peer generates a notification event, a gossip protocol round is initiated. Normally a gossiping round is initiated periodically and several notifications per gossip message may be received [43].

Algorithm 1 Probabilistic gossiping algorithm at each node

```
1: Receive gossip(sender, notification);
2: if notification.getId() ∉ historyList then
3:     deliver(message);
4:     historyList.add(notification);
5:     for (i=0; i < n; i++) do
6:         choose target peers at random;
7:         send(target, myself, notification, "gossip");
```

Algorithm 1 presents the original gossip protocol, where n is the number of gossip targets, which could be a random variable. When a peer receives a gossip message

from a sender, if the notification is new (does not exist in the historyList), the notification is delivered and added to historyList. After that, the peer randomly forwards the notification to n peers.

Shrack dissemination protocol is similar to the gossip protocol in the sense that peers periodically exchange messages to update new events. However, the Shrack dissemination protocol is different from the gossip protocol in three aspects:

1. While the gossip protocol is push-based, Shrack uses a pull-based communication protocol;

2. Shrack peers only disseminate messages that are relevant to them, while in the gossip protocol peers forward every notification they receive to other peers; and

3. Shrack peers form an overlay network of peers with common interests, while peers in the gossip protocol form a randomly connected overlay network.

2.5 Resource Discovery in Unstructured Peer-to-Peer Systems

We survey several approaches to resource discovery in unstructured peer-to-peer systems such as blind search, a variety of semantic and selective overlay networks, and a community-based information dissemination network.

2.5.1 Blind Search

In order to locate a file, a peer initiates distributed search among peers in the system. Originally, in Gnutella, peers distribute queries by flooding and later randomly forwarding queries to their neighbours [15]. In *flooding*, each peer forwards queries to all peers in the neighbourhood. In *random forwarding*, each peer only forwards queries to a set of its randomly chosen neighbours. Unstructured peer-to-peer systems do not guarantee that the search returns the sought results. The results are returned as best effort. Random walks for searching achieves better results than flooding when the overlay topology is clustered and when a client re-issues the same query while its network size does not change much [28]. Several other techniques exist that improve

flooding [22, 28, 39, 50]. However, these approaches blindly forward queries among peers without taking into account the content of queries or resources of peers in the network.

content have been purposed to type of networks, peers are forwarded only to related peers. documents.

2.5.2 Predefined Semantic Overlay

Crespo and Garcia-Molina proposed a system with multiple predefined semantic overlay networks. Each peer can join many semantic overlay networks and forwards queries only to the related semantic overlay networks. Peers have to know in advance what are the existing semantic overlay networks and join each overlay network based on the document collections they contain [26].

2.5.3 Shortcut Overlay

Without a predefined semantic overlay, peers can maintain knowledge of peers in the networks and forward queries to peers that have a high potential to return search results. Haase et al. presented Bibster [32], where peers are connected randomly but each peer has knowledge of the expertise of their neighbours and forwards queries based on the expertise of their neighbours [33]. The system shows improvement over randomly forwarded queries. However, the system only learns about the expertise of their neighbours at the start of the simulation and assumes static semantic and network topology. Similarly, Sripanidkulchai et al. [67] introduce shortcuts to forward queries to peers that return successful results. Joseph [40], Cooper [25] and Balke et al. [16] create shortcuts to super-peers or knowledge sources that have local indexes of document locations. REMINDIN' [72] and Löser et al. [49] show that selective forwarding of queries to top-rank peers with some randomness helps improving the search results.

Tempich et al. [71] explore an IR-ranking strategy to select the top-rank peers for multiple keywords search. Peers are ranked based on an *inverse peer frequency* (IPF), called TFxIPF (Term Frequency-Inverse Peer Frequency) —a similar technique to TFxIDF [51]. Peers that contain all the terms in a query will have the highest

score and also terms that exist in many peers are given a less significant weight. Their experiments compare four peer ranking schemes: (1) ranking based on TFxIDF score, (2) ranking based on TF, (3) ranking based query hit, and (4) using exact matches. For the exact matches, peers only forward queries to peers that match all the terms in a query. The results show that the TFxIPF, TF and query hit schemes have comparable results that are better than exact matches.

With semantic shortcuts, peers keep expertise knowledge of their neighbours and forward queries to peers that are semantically related. They still rely on the usage of the underlying connections and the default mechanism when the shortcuts fail. Peers usually acquire peer expertise relationship based on queries and results returned. In the next subsection, we discuss self-organizing overlay networks where peers form connections to neighbours that have similar interests.

2.5.4 Self-Organizing Overlay Networks

Motivated by small-world networks [75], several self-organizing unstructured peer-to-peer systems based on similar interests were proposed to improve search performance [12, 57, 65, 73, 77]. In these networks, each peer usually has a set of peers called *known peers*, which are peers in the system of which the peer is aware. Peers exchange their expertise regularly with others peers and periodically keep interactions with their neighbours, which are peers with similar interests. Normally, each peer adjusts its neighbours according to new updated expertise information, forming a self-organizing overlay of peers with similar interests.

There are three common mechanisms to manage self-organizing overlay networks: (1) peer expertise learning, (2) peer similarity measurement, and (3) peer neighbour selection. The peer's expertise learning is a mechanism for peers to learn about interest of other peers. The peer similarity defines how to quantify the similarity between two peers. The peer neighbour selection defines how peers select their neighbourhood, which defines the overlay network.

Sellf-organizing Semantic Overlay Network network. Each peer in a self-organizing semantic overlay network periodically asks its neighbours about their expertise. If the peer learns that their neighbours have less similar interests than a threshold value, then the peer sends a *WalkMessage* containing its expertise to ask about the expertise of other peers in the network in a random manner. After a peer learns about other peers, it establishes connections to peers that have more similar interests. Expertise consists of a content item selected as a representative of interest of the peer. In the experiments, expertise is represented by terms in an ontology. The similarity between two peers is quantified by the number of edges between terms in the peers' expertise according to the ontology. The experiments show that a certain amount of clustering of peers is beneficial for query routing; over-clustering, however, (i.e., connecting to peers that are too similar) may cause performance to deteriorate [65].

Common-Interest Peer Clustering A two-layer approach to maintain a cluster of peers with common interests is proposed by Voulgaris [73]. Peers are randomly connected in one layer, and in the other layer peers are connected based on common interests. In both layers, peers run a gossip-based protocol [74] to periodically exchange their expertise with their neighbours. A peer's expertise is represented by the list of files that the peer shares. The similarity between two peers is measured by the intersection of files that both peers share.

Semantic Overlay Networks for Web Searches Semantic overlay networks for Web searches are proposed in systems such as 6Search (6S) [77] and P2PDating [57]. Each peer has its own topical crawler and a local search engine. Peers first search for a query result in their local archives, then forward the query to their neighbours for more results. In 6Search, peers implicitly learn about expertise of other peers from the exchanges of queries and responses. Peers select their neighbours based on terms in query hit. In addition to the peer neighbour list, each peer also keeps contacts of known peers, which increases connectivity among peers in the system. Known peers can be used as candidate future peer neighbours. In P2PDating, peers randomly connect to the network, exchange their profiles with other peers, and adjust their

connection to peers with similar interests. Peers periodically exchange their profiles to evaluate their similarity by choosing peers from their friend list (neighbours), their candidate list (known peers), and random peers. The similarity between peers is evaluated by quality or usefulness measures, which can be based on (1) peer behaviour, (2) overlap of peer contents, or (3) semantic similarity using Kullback-Leibler divergence. The results show that among the three similarity measures, the Kullback-Leibler divergence returns the best search results. Akavipat [12] shows that the emerging semantic communities in 6Search follow small-world properties, such that the semantic overlay network has higher clustering coefficient and lower characteristic path length than the random overlay network [76].

Tribler Tribler [59] is a social-based peer-to-peer file-sharing system for content discovery, content recommendation, and downloading. Tribler is implemented on top of the BitTorrent[1] file-sharing system. Tribler exploits social behaviour by creating an overlay network based on similar interests among peers. Tribler uses a gossip-based protocol called BuddyCast to exchange peer expertise, which is a list of recently downloaded files, to form an overlay of peers with similar interests. In BuddyCast, each peer periodically selects a peer to which to connect from a list of neighbours ranked by their common interests. The selection is probabilistic using the roulette wheel approach. The similarity between peers are measured using Pearson correlation.

Like other systems in this category, Shrack is a self-organizing overlay network. However, most of these systems focus on searching for documents in response to a query, whereas Shrack peers focus on dissemination and discovery of new documents. In addition, the Shrack dissemination messages are used for peers to learn about expertise of other peers in the system without explicitly exchanging their profiles. As a result, Shrack peers not only learn about peer expertise but about new documents of interest at the same time. Shrack peers use only pull communication and form a self-organizing community network based on common documents of interest among peers. representations and similarity measures, predefined ontology. Furthermore, the user profile representation and similarity in Shrack is flexible. We investigate

[1] `http://www.bittorrent.com`

item-based and term-based representations in the thesis, but other options are not excluded by the framework.

pull communication, whereas our proposed system, answer search queries. documents. known peers without any explicit exchange of profiles. self-organizing overlay P2P networks is the investigation of

Next, we present other community-based information dissemination peer-to-peer systems.

2.5.5 Community-Based Information Dissemination Networks

Two peer-to-peer information dissemination approaches for data sharing based on small-world networks were proposed by Iamnitchi et al. [36] and by Mitre and Navarro-Molders [52]. The networks are comprised of several clusters, where each cluster is defined as a community with overlapping data interests. The systems are built to support search and locate documents in the same cluster.

In the first approach [36], each peer periodically updates global information of all peers and document locations in the same cluster through a "gossip" protocol [43], a push-based information dissemination mechanism. Hence, each peer can immediately locate documents in the same cluster. However, only filenames and their locations are disseminated among peers. The system is built on the assumption that each peer knows how to forward the requests to search for documents outside its cluster and have a predefined cluster overlay.

In the second approach [52], populated files are replicated among peers in the same cluster. Peers use limited flooding of requests to search for files within their clusters. Each peer also needs to have knowledge of other clusters to directly issue requests for documents outside its cluster. By using replication, peers do not have to maintain global knowledge of document collections and peers in their clusters, but they need to spend time on search to locate documents of interest.

Shrack has a similar design as the first approach. However, instead of defining a rigid cluster and requiring peers to maintain global knowledge of every peer in the same cluster, Shrack peers maintain knowledge and learn about expertise of other peers locally from received information. Moreover, Shrack is specially designed for

document sharing where the metadata of documents, not only filenames and their locations, are disseminated among peers.

Cuenca-Acuna et al. [27] introduced a content-addressable publish/subscribe service for unstructured peer-to-peer Systems (PlanetP) to improve distributed search in peer-to-peer communities. Each peer in the system maintains global addresses of peers in the communities and have global knowledge of an inverted term-to-peer index, which maps terms to peers having documents containing these terms. The gossip protocol is used for peers to distribute the global index and membership changes among peers in the system. To search for documents of interest, peers forward a request to a set of peers that contain documents with the requested terms based on the inverted index and ranking algorithm. When a peer receives a request, a local search is performed to find documents that match the request and the results are forwarded to the requesting peer. The experimental results show that PlanetP search efficiency is comparable with centralized servers. Shrack is different from PlanetP in that PlanetP disseminates their expertise to create shortcuts for query routing and use a random network as the underlying overlay. On the contrary, Shrack peers disseminate new documents of interests, which can be used to learn about expertise of peers in the system and then form a self-organizing overlay based on common interests.

2.6 Re-Coll: Peer-to-Peer Document Tracking Network

A Web-based peer-to-peer support for Research Collaboration, Re-Coll, was a pilot experiment of a peer-to-peer document tracking system using pull-only communication [53]. Unlike generic peer-to-peer systems where users learn about data items of interests after they issue search queries, Re-Coll enables users that have common interests to collaboratively keep track of data items of interests when they become available in the collaborative group. Re-Coll only provides a dissemination protocol for peers to keep track of documents of interest. The user of each peer needs to explicitly manage who they want to contact. In other words, Re-Coll peers do not have a mechanism to select peer neighbours or learn about the existence of other peers in the network.

The Shrack information dissemination protocol is motivated by Re-Coll's pull-only dissemination. However, Shrack's protocol is different from Re-Coll's. In Shrack, information is disseminated based on the interest of the peer users, while in Re-Coll every peer shares all information it receives with other peers in the network. As a result, Shrack peers benefit from collaborative filtering from peers with similar interests. In addition, Shrack enables peers to learn about expertise of other peers and form a self-organizing overlay network based on common interests.

2.7 Peer-to-Peer Research Collaboration on JXTA

JXTA is an open network computing platform designed for peer-to-peer computing [3]. JXTA provides basic functions to support peer-to-peer applications such as membership management, service advertisement, indexing, and searching. Several research collaborative peer-to-peer systems are built on top of the JXTA platform. Examples of such systems are DFN [63], BIBSTER [32], Edutella [54] and OAI-P2P [10]. These systems use the JXTA platform to manage membership and information in the system. They use JXTA's open search to locate documents of interest. Each of the systems focuses on different kinds of information that are shared among peers.

The DFN Science-to-Science (S2S) [63] is a non-profit organization that provides research infrastructure in Germany. The focus of the project is on the search capability in indexing and gathering scientific information. BIBSTER [32] focuses on managing, searching and sharing of bibliographic metadata from BibTeX files and uses semantic Web techniques to enhance document searching. Edutella [54] is a peer-to-peer networking infrastructure based on RDF. Edutella peers use RDF for information exchange and querying among peers. OAI-P2P [10] is a peer-to-peer network for open archives. OAI-P2P is built on Edutella for research organizations to share their documents and support distributed search over all connected metadata repositories.

While there are many initiatives to build peer-to-peer systems for research collaborations, the existing systems focus on document searching. In Shrack, we focus on document tracking based on users' interests.

communication protocol; protocol peers form

2.8 Collaborative Filtering

Collaborative filtering is a type of a recommender system where the recommendations that a system provides to a user are based in part on the profiles of other similar users. User similarity can be measured using a similarity measure on the user profiles' feature vectors, such as the cosine similarity measure, or it can be defined explicitly through social connections between users in a social network.

2.8.1 Types of Collaborative Filtering Approaches

Collaborative filtering approaches fall under two general categories: (1) memory-based and (2) model-based. Memory-based collaborative filtering algorithms store all rating examples into memory and use them to compute user or item similarity, which is later used to rate a test item for a user. The similarity of users or items is computed by using cosine similarity of the rating vectors based on an assumption that similar users will rate items similarly and similar items will be rated by users similarly. Model-based collaborative filtering algorithms, on the other hand, use the training examples to generate a model that is able to predict the rating for items that a test user has not rated before. Breese et al. provide an empirical analysis of both types of algorithms [18].

Hybrid collaborative filtering approaches combine memory-based with model-based algorithms. The hybrid approach reduces the user dimensionality using a clustering or a classification algorithm. First, a model-based algorithm builds several models of users, where each model represents a cluster [78] or a persona [58] of similar users. Then a memory-based algorithm is applied to the models.

2.8.2 Collaborative Filtering Applications

Some of the systems that have successfully employed collaborative filtering include Tapestry [31], GroupLens [44], ReferralWeb [41], Amazon.com [48], and PipeCF [34].

Tapestry Tapestry [31] is one of the earliest applications of collaborative filtering in information filtering. Tapestry is an experimental mail system developed at Xerox

to filter electronic documents based on users' annotations. Although Tapestry does not account for user similarity, it provides an efficient and practical approach for annotating documents and messages, and for filtering incoming data based on user-defined filters. Tapestry is an example of a distributed content-based filtering system.

GroupLens GroupLens [44] provides recommendation to Usenet newsgroup readers based on explicit ratings of news items by the users. The system successfully addresses issues of sparsity, the start-up problem, and the short life-span of the news messages. Scalability, however, is cited as one of the GroupLens system's limitations that could be addressed through a distributed system.

ReferralWeb ReferralWeb [41] exploits the social graph to provide expert or topic recommendations to a user. The system architecture is distributed in the sense that each user has their own ReferralWeb agent. The agent infers the social graph automatically from the Web documents, which is accomplished by querying a search engine. The dependency on an external search engine renders the system only partially decentralized. Further, the user has little control or direct influence over the social graph and has no explicit way of providing feedback to the system to improve its performance.

Amazon.com Amazon.com is arguably the most successful commercial application of a collaborative filtering system. Instead of comparing users to each other, Amazon.com uses an item-to-item collaborative filtering system [48]. This system is able to compute all the recommendation off-line over several million items. This approach is not suitable for distributed environment because it requires a global knowledge of all items and user interactions, whereas each node in a distributed system contains only a partial knowledge of available items and other users.

PipeCF PipeCF [34] is a distributed collaborative filtering algorithm for peer-to-peer networks. It relies on a distributed memory-based approache for storing user profiles. The distributed hashing table (DHT) is used as the underlying mechanism for storing and retrieving user data in the system. Hence, the limitations of PipeCF

are those of the DHT, namely the data locality problem; that is, users cannot control where their data is stored or which data they store.

The Shrack system acts as a distributed collaborative filtering system. The documents that a peer receives are filtered by its neighbours, where the neighbours are chosen based on their similarity to the peer.

Most collaborative filtering methods have the *cold-start problem*, which refers to the situation that an item cannot be recommended unless it has been rated by a substantial number of users. However, there is no rating in Shrack. Shrack peers recommend all the documents they recieve from the neighbourhood to the peer users. The cold-start problem could refer to the situation when a peer joins the network, where (1) the neighbourhood is determined randomly, and (2) the peer does not have any knowledge of the interest of the peer user. We can alleviate these problems by: (1) users manually setting up the peer neighbourhood from known contacts that have similar interests in the real world, and (2) users presenting the set of relevant documents to the peer to initially model their interests.

2.9 Summary

This chapter provides background and related work for this thesis. First, we provide background knowledge of data and document sharing network. Then, we give an introduction to peer-to-peer systems; we define peer-to-peer networks and describe two major peer-to-peer overlay architectures. After that, we provide a review of resource discovery in unstructured peer-to-peer systems starting from blind search, semantic or selective search, resource discovery in self-organizing overlay, and information dissemination in community based peer-to-peer networks. Then, we review the Re-Coll system, which motivated the pull-only dissemination protocol in Shrack. Subsequently, we describe the original gossip protocol and present related work in collaborative filtering. In the next chapter, we present the framework of Shrack.

Chapter 3

Shrack Framework

We propose Shrack as a peer-to-peer system that facilitates researchers to share and keep track of new documents of interest. Shrack is designed based on an assumption that researchers who work in the same research area are usually interested in overlapping sets of documents. Their efforts in sharing and keeping track of new documents of interest can benefit from an autonomous collaborative environment.

In this chapter, we first define the research problems in Section 3.1. Then we define formal terminology that will be used in the rest of the thesis in Section 3.2. Then we describe the Shrack framework. We present the Shrack architecture in Section 3.3. Subsequently, we provide the characteristics of Shrack network in Section 3.4. After that, we explain the Shrack peer functionalities and introduce the Shrack information dissemination protocol in Section 3.5 and Section 3.6, respectively.

3.1 Problem Definition

The goal of this research is to develop a design and predict the performance of a peer-to-peer system to help researchers share and keep track of new documents of interest. This thesis focuses on the following research problems:

1. Making design choices to create the Shrack framework; defining peer components, peer functionality, communication protocols, and characteristics of the network.

2. Designing an information dissemination protocol using pull-only communication that is scalable as the network size increases.

3. Developing a self-organizing Shrack network, for a given set of peers with overlapping interests, to improve the dissemination performance, measured as an average F-score of documents received by peers.

3.2 Formal Definitions

Next, we briefly define terminology that we will be using in the rest of the thesis.

Definition 1 (Peer) *A peer is an individual node that connects to the network. Every peer in the network has the same functionality.*

Definition 2 (Local Peer) *A local peer is a peer under consideration.*

Definition 3 (Receiver Peer) *A receiver peer is a peer that pulls information or a peer that issues a request for information, in the context of a P2P communication session.*

Definition 4 (Provider Peer) *A provider peer is a peer from whom information is pulled or a peer that responds to a request for information. A provider peer also serves as a source of information for a local peer.*

Definition 5 (Known Peer) *A known peer is a peer in the network of which the local peer is aware. The local peer may find out about a known peer from messages that are disseminated among peers in the network or from users manually entering peers into the local peer's database.*

Definition 6 (Publisher Peer) *A publisher peer is a peer that provides the original source of a document.*

Definition 7 (Shrack Message) *A Shrack message is a message that is disseminated among peers to inform them about the new document in the network. A Shrack message contains information about the new document, message history, and message status.*

The detailed content of a Shrack Message is described in Table 4.1 (page 35).

Definition 8 (Document Metadata) *Document metadata contains information about the document. A document metadata may include title, authors and document description. Each Shrack message contains a document metadata item describing information related to the associated document being shared.*

Definition 9 (Peer User) *A peer user is a person or organization that is associated with the peer and the peer's interest captures the peer user's interest.*

Example of a peer user is a researcher, a department, a research group, or a document archive.

3.3 The Shrack Architecture

Shrack is an unstructured P2P network designed to mirror real world research collaborations, assuming that researchers who are interested in similar research areas are usually interested in overlapping sets of documents, and willing to share their resources and knowledge to keep track of new documents in the area. Each peer is associated with a peer user such as an individual researcher or a research organization. A peer that is used by a research organization can be viewed as a *super peer*—a server that supports group activities for document sharing such as a digital library or a document archive. Each peer acts on behalf of its user based on the interest of the user and when we talk about the "interest of a peer", we refer to the interest of the peer user. Each Shrack peer has the same functionality of sharing and keeping track of new documents published in the network using pull-only communications. We believe that pull communication gives control to peers who seek information to select from which sources and when to pull shared information.

The Shrack architecture from a peer's perspective is shown in Figure 3.1. The *Shrack network* is a peer-to-peer network that supports collaboration by keeping track of new documents injected by peers into the network. The details of the Shrack network are described in Section 3.4. *Shared directory* is a directory that contains Shrack messages that are made accessible to other peers. *Peer's archive* is a local repository containing documents and document metadata that are of interest to the peer. *Provider peer list* maintains the addresses of peers from which the local peer will pull Shrack messages, referred to as *provider peers*. The provider peers could be viewed as a peer's neighbourhood in the context of a generic peer-to-peer system, but we use the terminology "provider peers" to emphasize relationship of peers according

to pull communications. *Incoming links* are connections established by other peers through a server port to pull Shrack messages from the local peer. *Outgoing links* are connections established by the local peer to provider peers from which the peer pulls Shrack messages. *User's feedback* is a user interface to present document metadata of new documents to and receive relevant feedback from the user.

Each peer keeps track of new documents from Shrack messages that are disseminated among peers through pull connections. Two main modules are included in the Shrack architecture to enable self-organizing overlay network among peers with similar interest, namely, *knowledge integrator* and *provider peer selection*.

3.3.1 Knowledge Integrator Module

Each peer has a knowledge integrator module to analyze information received from interacting with the peer user and other peers. The knowledge integrator module uses this information to build a knowledge base of documents and peers in the network. The knowledge integrator module also maintains information about interests of the peer user and interests of known peers. When a peer presents document metadata to its associated user, the knowledge integrator module can explicitly or implicitly learn about the user interests. For example, the knowledge integrator module can implicitly learn about interest of the user from a set of documents that the user requests to download. The knowledge integrator learns about interests of known peers from received Shrack messages that are propagated among peers that have similar interests. The knowledge integrator module should have an ability to build a knowledge base that integrates information from document analysis, user analysis, and peer analysis.

The document integrator module can analyze information related to the document such as the document content, document metadata, or the citation graph and the co-authorship graph of documents to model relationship of documents. The user analysis finds the interest of the peer user, which will be used to create a local peer profile. The local peer profile can be used by the document metadata filtering module to indicate document metadata that are of interest to the peer user or by the provider peer selection module to find peers that have common interest with the peer user.

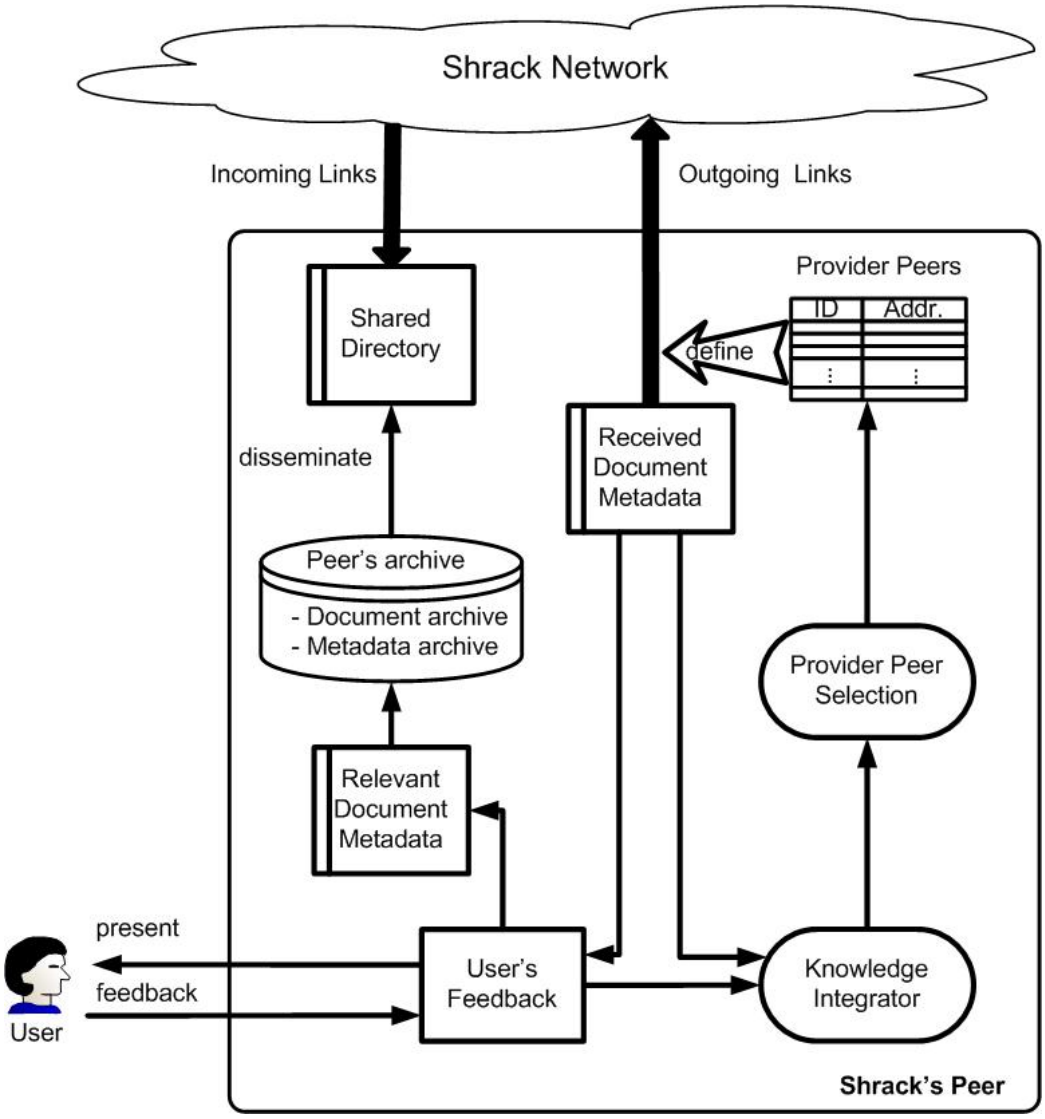


Figure 3.1: Shrack peer architecture

Similarly, the peer analysis finds the interests of known peers and creates a local view of interests of known peers and the relationship between peers.

3.3.2 Provider Peer Selection Module

The provider peer selection module evaluates and selects provider peers as sources of information that are of interest to the local peer —creating a self-organizing network based on common interest. Each peer selects its provider peers from among known peers with common interests. The interests of a known peer can be identified from the information embedded in Shrack messages processed by the knowledge integrator module. Each peer periodically calls the provider peer selection module to change or update the list of provider peers.

We will discuss the the knowledge integrator module and the provider peer selection module in detail in Chapter 4 (page 33).

Next, we describe the property of the Shrack network.

3.4 Shrack Network

We assume that Shrack peers are autonomous in that the local peer determines which peers it wishes to get connected to and which information it wishes to share with other peers. Hence, Shrack is modelled as an unstructured peer-to-peer system and it is a purely decentralized network. There is no central directory that maintains global knowledge. Shrack topology is organically formed according to peer interests and collaboration. Peers having common interests and who are willing to collaborate establish pull connections to one another, which are represented as directed edges from receiver peers to provider peers, as shown in Figure 3.2. Each Shrack peer can act as both a receiver peer and a provider peer. We use different terminology to differentiate the role of a peer at a particular time.

Receiver peers maintain contacts to and initiate pull communication (a *pull request*) with their provider peers. Provider peers provide information (a *pull response*) to the receiver peers if they wish to collaborate. Peers voluntarily join and leave the system without notifications. The Shrack network is dynamic, unbounded and may

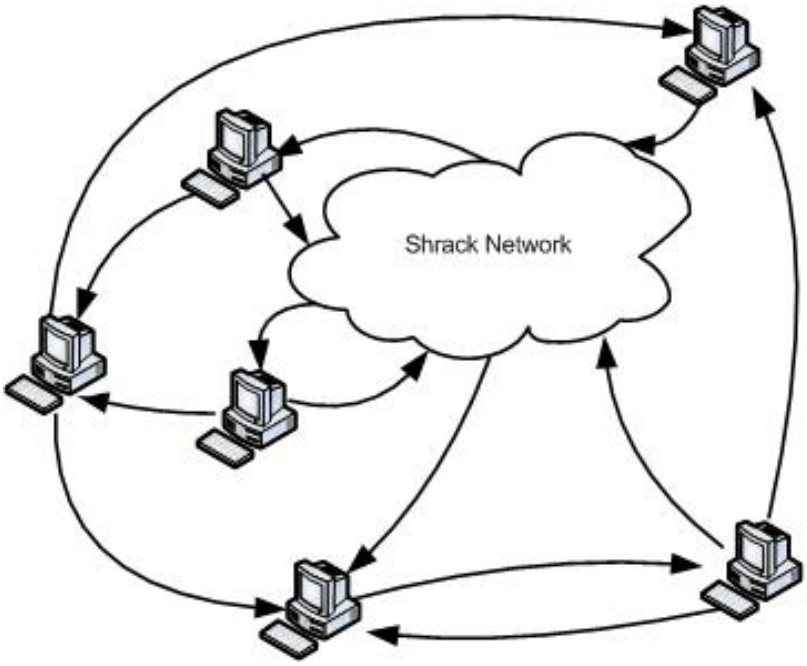


Figure 3.2: Shrack network

contain cycles. It is unbounded in the sense that no peer can store information about the complete global network topology, which may be considered infinite from our point of view.

3.5 Peer Functionality

The four basic peer functions are: join, leave, publish, and retrieve. The details are described below.

Join

To join a Shrack network, a new peer firstly obtains a Shrack contact address from potential provider peers with whom it has real world collaboration, or it learns about from other means of public media, such as web pages or blogs. Users can exchange Shrack addresses similarly to exchanging their email addresses or telephone numbers. We call these potential provider peers *start-up provider peers*. At least one start up provider peer is required. After obtaining the addresses of the start up provider peers, the new peer issues pull requests to pull Shrack messages that have been disseminated

in the system from them. The new peer then uses the information contained in the Shrack messages to learn about other peers in the system, which become its known peers, and which later can be used as its new provider peers. The provider peers respond to the pull request of the new peer by not only sending back the Shrack messages but also keeping the contact address of the new peer in the list of their known peers. Later, the provider peers can use the contact address to pull back information from the new peer. In this way, the information of the new peer will be disseminated to other peers in the system. As a result, Shrack peers learn about each other in the process of disseminating Shrack messages.

Leave

There is no explicit notification when a peer leaves the Shrack network. Each peer can leave the network any time without notice. Other peers will learn and assume that a peer has left the network, if the peer is in their list of provider peers and they get no response after multiple pull requests. Then the peer that left will be removed from their list of provider peers and their list of known peers and finally it will be removed from the network.

Publish

To publish a document, a publisher peer creates a Shrack message containing document metadata and pushes the message into its shared directory. The document metadata are propagated by means of receiver peers pulling information from the publisher peer and then taking on the role of provider peers. With successive pulls, the document metadata will be disseminated to connected peers. When a receiver peer receives new document metadata items, the document metadata filtering module will automatically classify document metadata as relevant to the peer's interest or not, based on the local peer's profile. The relevant document metadata are stored in the peer's archive, and presented to the peer user.

An example of what happens when a peer p_1 publishes a document d and its document metadata item is disseminated to peer p_2 by successive pulling of Shrack messages is shown in Figure 3.3.

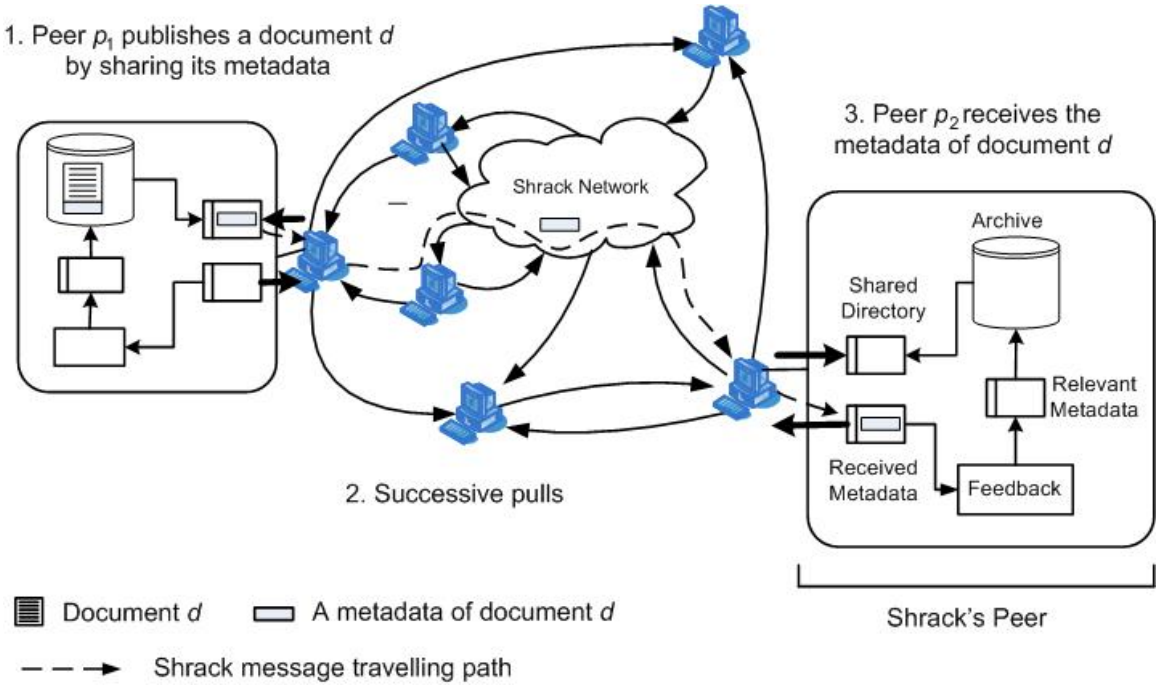


Figure 3.3: Peer p_1 publishes a document d and its document metadata is disseminated to peer p_2

Retrieve

The peer can automatically download the relevant documents, or the user can later retrieve documents of interest using information about the document's location contained in the Shrack message.

3.6 Information Dissemination Protocol

We propose a pull-only information dissemination protocol for peers to share and keep track of new documents among other peers that have common interest. Periodically, each peer gets connected to its provider peers to request or pull Shrack messages from the provider peers. Upon receiving Shrack messages, the peer disseminates the received Shrack messages by inserting them into its shared directory for other peers to pull. With successive pulls, Shrack messages will be disseminated to all connected

peers. Shrack messages should contain enough information about documents for the peer users to decide whether the documents are relevant to their interest. In addition, Shrack messages should also contain enough information for peers to automatically perform their operations with minimum effort.

With pull-only communication, peers in the network cannot push Shrack messages to other peers. The messages will be transferred only by requests. In this way, receiver peers can avoid receiving some non-relevant information or receiving information when they are not ready. Moreover, provider peers need to maintain their reputation by not disseminating spams or junk documents, so that other peers will keep pulling the messages from them. Peers that do not maintain their reputation will be automatically disconnected as other peer lose interest in pulling their shared messages. As a result, their publications will be hardly visible by other peers. Details of the information dissemination protocol will be presented in the next chapter, Section 4.1.

The Shrack protocol only exchanges metadata about documents, not the documents themselves. As a result, the overhead and bandwidth utilization of the underlying network is minimal. However, if someone publishes a document that becomes popular and many peers in the system wish to download this document, then the publisher peer may experience a high level of network traffic. This problem can be mitigated by either hosting the document by a super peer, or by using an approach similar to BitTorrent for distributing documents.

3.7 Summary

In this chapter, we present the Shrack framework. We define the research problems and formal definition of terminology using in this thesis. We describe the shrack architecture, shrack network, and peer functionality. In addition, we provide an overview of the Shrack information dissemination protocol. In the next chapter, we present a prototype system of Shrack.

Chapter 4

Shrack Prototype System

This chapter presents a prototype system of the Shrack framework focusing on the following goals:

1. to develop a scalable information dissemination protocol using pull-only communication; and

2. to enable a self-organizing network to improve the quality of documents received by peers based on common interests derived from the overlap in the set of documents of interests.

First, we describe the information dissemination protocol and define Shrack messages in Section 4.1. Subsequently, we present the knowledge integrator in Section 4.2, discussing how peers learn about the interests of their associated users, how peers discover the existence of other peers in the network and learn about their interests based on information carried in Shrack messages, and how to quantify the common interests between peers. After that, we present the provider peer selection module in Section 4.3, investigating three different provider peer selection strategies.

4.1 Information Dissemination Protocol

We develop the Shrack information dissemination protocol using pull-only communication. Peers in the network cannot push information to other peers. The information is transferred only on request. In Shrack, it is not the user that issues requests, but the user's peer periodically pulls Shrack messages from its provider peers. In standard P2P systems, the peer executes a search when the user issuse a search query. The difference is that in Shrack the information has already been collected by the user's peer when the user needs it. We claim that pull-only information dissemination is

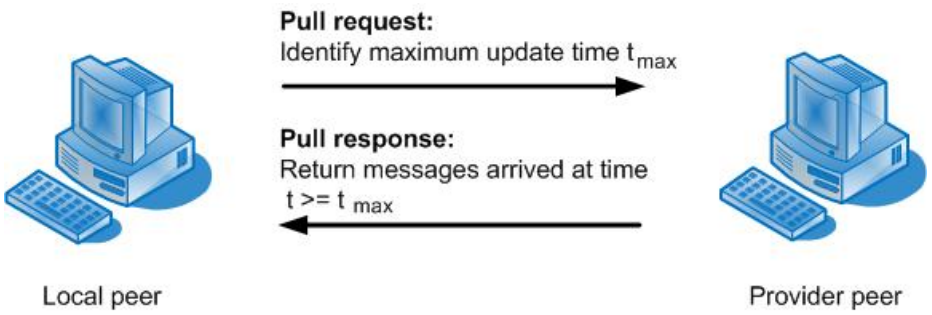


Figure 4.1: Shrack pull request and pull response

suitable for peers to keep tracks of events that are of interest to the peer user over a long period of time, when the events do not require immediate notifications. As a result, it is suitable for keeping track of new documents, since the interests of researchers (peer users) usually persist over a long period of time and change slowly with time. In addition, with pull-only communication, the receiver peers can choose when and from where to pull information.

The Shrack information dissemination protocol is simple. Peers disseminate shared information in the form of Shrack messages by exchanging pull messages, namely *pull requests* and *pull responses*. Each type of pull messages is defined next.

Definition 10 (Pull Request) *A pull request is initiated by a receiver peer to request pulling Shrack messages from a provider peer. A pull request contains the contact of the receiver peer and an* update time *identifying the oldest Shrack messages the receiver peer wants to pull from the provider peer's shared directory.*

Definition 11 (Pull Response) *A pull response is a response to a pull request from a provider peer. The pull response contains a set of Shrack messages that arrive at the provider peer at time equal to or later than the update time given in the pull request.*

A pull request is initiated by a receiver peer to pull Shrack messages from a provider peer. A pull response is a response from the provider peer to the pull request containing a set of Shrack messages. Each peer records an arriving time stamp of each Shrack message, indicating a time when the message arrives. When a receiver peer wants to pull Shrack messages from a provider peer, the receiver peer initiates a pull

request with an *update time* to the provider peer. Upon receiving the pull request, the provider peer, if it wants to collaborate, responds with a pull response, which contains a set of Shrack messages with arriving time stamps after the update time back to the receiver peer. In general, the update time is the time of the previous pull request of the the receiver peer sent to the provider peer. Figure 4.1 shows message exchanges when a local peer issues a pull request to a provider peer.

4.1.1 Shrack Messages

Table 4.1: Shrack message fields

Fields	Descriptions
Message ID	Unique message identifier
Publisher Peer ID	Contact of the peer that creates the Shrack Message such as a URI or an IP address
Document ID	Unique document identifier
Document Metadata	Data about the document such as title author, abstract, keywords, and publication date
Visited Peers	A list of IDs of peers the message has travelled through on its way from the publisher peer to the local peer
Time-To-Live (TTL)	The number of hops after which the message will be discarded

Each Shrack message contains information about the shared documents and status of the message to manage the dissemination. We define Shrack message fields in Table 4.1 as consisting of a message identifier, publisher peer identifier, document identifier, document metadata, visited peers, and Time-To-Live value. *Message ID* is a unique identifier created from a combination of a publisher peer ID and document ID to identify the message in a unique manner. *Publisher Peer ID* is the contact of the peer that creates the Shrack message such as a URI or an IP address. Publisher Peer ID allows receiver peers to learn about the original source of information, so that they can use this information to retrieve the whole document. *Document ID* is a unique identifier to identify a document, such as a digital object identifier (DOI)[1].

[1]http://www.doi.org/

Document Metadata contains summary information related to the document, which help users determine whether they are interested in retrieving the whole document. For a research publication, the document metadata may include the title, author, abstract, keywords, and date of the publication. *Visited Peers* is a list of peer IDs of peers that the message has travelled through along the path from the publisher peer to the local peer. This list of visited peers contains the message propagation history allowing receiver peers to learn about the existence of peers in the system. *Time-To-Live (TTL)* identifies the number of hops after which the message will be discarded. The initial value of the TTL is a system-wide predefined value. The TTL is used as part of a mechanism to ensure that the dissemination of a Shrack message will be terminated.

4.1.2 Pull Procedure

Shrack peers use Shrack messages to learn about new documents and the existence of other peers in the network, as well as the interests of these peers. Each peer only stores and disseminates messages that are of interest to the peer user. These messages reside in peer's shared directory for other peers to pull. As a result, each peer filters out irrelevant messages for other peers that have common interest, creating an implicit community filtering or recommendation system. The pull procedure describes how a peer p_i pulls and processes Shrack messages. The pseudocode of pull procedure is presented in Algorithm 2. We also present how peer profile learning and provider peer selection processes are incorporated in the pull procedure.

We define *pull interval* as a predefined interval, specifying how frequently a peer wishes to pull its provider peers to learn about new documents. In this work, we assume that peers pull Shrack messages from all of their provider peers using the same pull interval. In practice, pull intervals might be different. A peer p_i pulls Shrack messages by issuing a pull request to each provider peer (line 2). Upon receiving a pull response, the peer p_i updates the last pull time of p_j to the current time (line 3).

For each Shrack message in the pull response (line 4), the peer checks if the message contains new document metadata to the peer p_i by comparing the document metadata's identifier with the peer's history list (line 6). If the identifier of the

Algorithm 2 Pull Procedure of peer p_i

```
1: for each provider peer p_j of peer p_i do
2:     peer p_i issues a pull request to pull a set of Shrack messages, M, from p_j's
       shared directory that arrived at p_j after the last time p_i pulled Shrack messages
       from p_j;
3:     update last pull time of p_j;
4:     for each Shrack message m ∈ M do
5:         d is a document metadata embedded in m;
6:         if d is new to p_i then
7:             if d is relevant to the p_i's interest then
8:                 keep d in the local archive;
9:                 add arriving time stamp for m;
10:                decrease the m.ttl by one;
11:                if m.ttl is greater than 0 then
12:                    append p_i's ID to the Visited Peers list of m;
13:                    add m to the share directory of p_i;
14:    update local profile of p_i
15:    update known peer profile of p_i
16: update list of provider peers of p_i
```

document metadata is not already in the history list, it will be detected as a new document metadata. If the new document metadata d is relevant to peer p_i, d will be added to the peer's local archive (line 8). Then, for each message m containing relevant document metadata d, the receiver peer p_i records an arrving time stamp for m (line 9) and decreases the TTL of m by one (line 10). If the TTL of m is greater than zero, the peer appends its peer contact ID to m and places the message m in its shared directory for further dissemination (line 12-13). After that, the peer updates its local profile according to the set of new relevant documents (line 14). Subsequently, the peer updates its known peer profile of peers in the visited list of the received Shrack messages (line 15). At the end of each pull cycle, the receiver peer evaluates its known peers and updates its provider peers (line 16).

A Bloom filter [17] is a good candidate to be used as a history list of received document metadata. The Bloom filter is a randomized data-structure for concisely representing a set in order to support approximate membership queries. The space efficiency is achieved at the cost of a small probability of false positives. Bloom filters have been used in many network applications [20]. We describe how a peer updates

its local profile and known peer profile in Section 4.2, and how a peer updates the list of provider peers in Section 4.3. Note that for simplicity, we assume that a peer gets immediate feedback from its user on the relevance of the new documents based on their metadata. In practice, the peer has to wait for feedback from the user.

4.2 Knowledge Integrator Module

To enable a self-organizing network based on common interests among peers, each peer needs to have the ability to learn about the interest of its associated user. In addition, each peer also needs to have the ability to learn about the existence of other peers and their interests. Furthermore, each peer needs the ability to quantify the common interests between its associated user and other peers' users. These abilities are captured by the peers' knowledge integrator module.

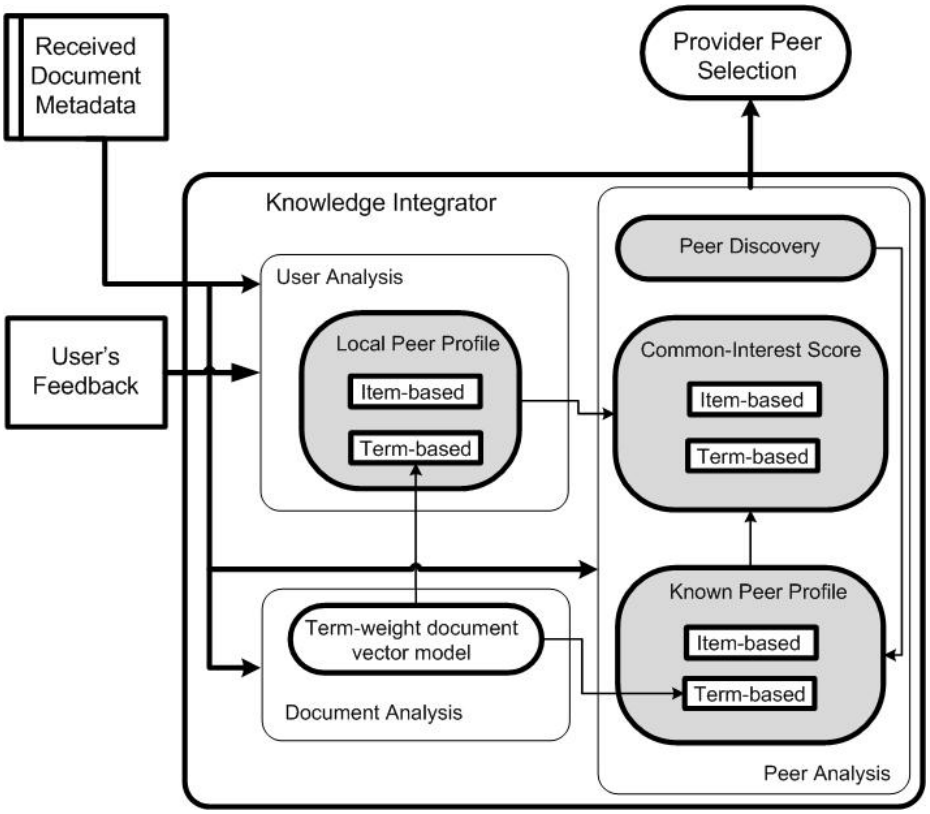


Figure 4.2: The prototype of the knowledge integrator module

We show a diagram of the prototype of the knowledge integrator module in Figure 4.2. As mentioned in Section 3.3.1 (page 26), the knowledge integrator module maintains information about interests of its associated user and interests of other peers. The knowledge integrator also maintains a list of existing peers of which the local peer is aware, called known peers. Each peer has a *peer discovery* module to learns about its known peers. The interests of the peer's associated user are represented by *a local peer profile*, and a peer represents its knowledge about interests of other peers in the *known peer profile* module. Each peer discovers its known peers and learns about their interest locally, based on information carried in Shrack messages that propagate through the network. There is no explicit profile exchange between peers. We model both local peer profiles and known peer profiles with two representations:

1. Item-based peer profiles: The set of documents represents a peer profile

2. Term-based peer profiles: A term-weight vector is used to represent peer profiles

The level of common interests between a local peer and its known peers is quantified by the *common-interest score* module based on the similarity between their corresponding profiles. The *document analysis* module handles the term-weight vector representation of documents.

Since the set of documents that are available in the network changes with time as documents get published over time, we define in Section 4.2.1 sets of documents in relation to time. After that, we describe how a peer models a term-based peer profile from a given relevant document set in Section 4.2.2. Subsequently, we explain how a peer discovers the existence of other peers in Section 4.2.3. Then, we describe how a peer creates its local peer profile and its known peer profile in Section 4.2.4, and 4.2.5, respectively. Finally, we explain how a peer quantifies the common interest between itself and its known peers in Section 4.2.6.

4.2.1 Document Set Definitions

We now give definitions of various peer-related document sets that explicitly incorporate the notion of time. This is necessary because the peer-to-peer network is a

dynamic system that changes over time. From now, the term "document" also refers to the metadata of the document, unless specified differently.

Definition 12 (Peer-Published Document Set) *$S^t(p_i)$ is the set of documents that peer p_i publishes up to time t.*

The peer published document set represents a set of documents that a peer publishes up to time t.

Definition 13 (System-Published Document Set) *P^t is the set of all documents that are published by every peer in the network up to time t; that is,*

$$P^t = \cup_{i=1}^{z} S^t(p_i) \tag{4.1}$$

where z is the number of peers in the network.

The system published document set is a set of all documents that are available in the system at time t.

Definition 14 (Peer-Relevant Document Set) *$R^t(p_i)$ is the set of documents that are published up to time t in the entire network and are relevant to peer p_i; that is,*

$$R^t(p_i) = \{d \mid d \in P^t \text{ and } d \text{ is relevant to } p_i\} \tag{4.2}$$

A peer relevant document set of peer p_i, $R^t(p_i)$, is the global set of relevant documents that are available in the system at time t. However, at a given time t, a peer p_i may not be aware of all documents in $R^t(p_i)$, because peer p_i might not have connections to some peers that publish documents in the relevant set, or due to propagation delay of the relevant Shrack messages. However, we assume that peers are always interested in the documents that they publish themselves, i.e., $S^t(p_i) \subseteq R^t(p_i)$.

Definition 15 (Peer-Received Document Set) *$D^t(p_i)$ is the set of documents whose metadata was received by peer p_i up to time t, excluding the self-published documents $S^t(p_i)$.*

Note that a Peer-Received document Set of peer p_i, $D^t(p_i)$, includes all documents whose metadata was received by p_i up to time t, which may be relevant or not relevant to p_i. Moreover, a Peer-Relevant Document Set of peer p_i is generally not a subset of, and not equal to a Peer-Received Document Set of peer p_i, i.e., $R^t(p_i) \not\subseteq D^t(p_i)$, due to propagation delay or network connections.

4.2.2 Term-Weight Document Metadata Model

A term-based peer profile is modeled by using the document metadata contents of a given relevant document set. In general, the metadata of document consists of mulitple fields to describe information about the associated document; however, we select three fields to model peer profiles, namely "Title", "Abstract", and "Keywords". Since the information contained in document metadata is not very long and the selected fields represent the concepts of the documents, we give equal importance weight to terms present in the selected fields. As a result, we view document metadata as an unstructured document which is a concatenation of the terms in "Title", "Abstract", and "Keywords" fields.

The metadata of document d is represented by a vector $\vec{d} =< d_1, d_2, d_3, ..., d_V >$ where d_v is the term frequency of the term v in $\vec{d}$ and V is the number of terms in the vocabulary. When we specify "term v", where $1 \leq v \leq V$, we mean term number v in the vocabulary. However, if one wishes to give a weight to terms in each field differently, one way to handle this is suggested by [61] , which is simply combining the term frequencies of the different fields by forming a linear weighted combination of the corresponding fields.

At a given time t, a peer models a term-based profile after a given relevant document set Q^t, denoted Q^t_T. The peer views the document set Q^t as a single document, called a profile document. The term-based peer profile Q^t_T, is represented by a term-weight vector $\vec{Q}^t$; that is,

$$Q^t_T = \vec{Q}^t =< w^t_1, w^t_2, w^t_3, ..., w^t_V > \tag{4.3}$$

where w^t_v is the term weight $TF \times IDF$ of the term v in $\vec{Q}^t$ and V is the number of

terms in the vocabulary. The term-frequency weight TF_v^t of each term v at time t is computed from the profile document represented by the following equation:

$$TF_v^t = \frac{\sum_{d \in Q^t} d_v}{\sum_{d \in Q^t} |\vec{d}|} \tag{4.4}$$

where $d \in Q^t$.

We assume there is a representative document collection available to estimate a global IDF of each term v, denoted IDF_v. Subsequently, each peer computes the IDF_v from the representative document collection; a maximum value of the global IDF is assigned to terms that do not exist in the representative document collection. Hence, the weight w_v^t for each term in $\vec{Q}^t$ is defined as

$$w_v^t = \frac{\sum_{d \in Q^t} d_v}{\sum_{d \in Q^t} |\vec{d}|} \times IDF_v \tag{4.5}$$

where $d \in Q^t$.

To incrementally update the term-based profile Q_T^t with a new relevant document d^θ, denoted Q_T^{t+1}, the document profile is modified by adding d^θ in Q^t, denoted Q^{t+1}. Then, a new term-weight vector $\vec{Q}^{t+1}$ is computed. We denoted w_v^{t+1} as a new term-weight of term v in the new term-weight vector $\vec{Q}^{t+1}$. Formally,

$$w_v^{t+1} = \frac{\sum_{d \in Q^t} d_v + d_v^\theta}{\sum_{d \in Q^t} |\vec{d}| + |\vec{d^\theta}|} \times IDF_v \tag{4.6}$$

where $d \in Q^t$ and $Q^{t+1} = Q^t \cup d^\theta$

We discuss how a peer creates its local peer profile and known peer profile in Section 4.2.4 and Section 4.2.5, respectively. Next, we describe how a peer learns about an existance of other peers in the system.

4.2.3 Peer Discovery

Each local peer learns about the existence of other peers from the list of visited peers in the received Shrack messages. The list of visited peers maintains contacts of peers that the message has travelled through, from the publisher peer to the local peer.

Furthermore, a local peers also learns about a new peer from a pull request. When the new peer issues a pull request to the local peer, the request contains a contact of the new peer as the receiver peer. When a local peer learns about a new peer, either from received Shrack messages or pull requests, the local peer adds the new peer into its known peer profiles. The initial profile of the new peer contains only its contact. The local peer later learns about interests of the new peer from associated Shrack messages.

4.2.4 Local Peer Profile

A local peer profile represents the interests of the local peer's user. Each user can explicitly give a set of terms or a set of documents to initialize the peer profile, or let the peer learn about the user's interest incrementally from the user feedback, implicitly or explicitly. We can get an implicit feedback from a user by assuming that documents that the user requests to download or documents that the user keeps in the local archive are the documents that are of interest to the user, to which we refer as *relevant documents*. In addition, a peer can implicitly include documents that it self-publishes in the set of relevant documents. On the other hand, documents that the user does not request to download or keep in the local achieve are referred to as *non-relevant documents*.

Since each peer is only aware of documents whose metadata they received or published up to a given time, we define a set of relevant documents of a local peer p_i as a function of time, called *a local peer relevant document set* $L^t(p_i)$.

Definition 16 (Local Peer Relevant Document Set) *$L^t(p_i)$ is a union of the set of relevant documents whose metadata was received by peer p_i and the set of documents that peer p_i has published up to time t; that is,*

$$L^t(p_i) = S^t(p_i) \cup (R^t(p_i) \cap D^t(p_i)). \tag{4.7}$$

Each peer incrementally updates its local peer profile when a peer publishes or learns about a new document d that is relevant to user's interests.

We explore two peer profile representations: an item-based local peer profile and

a term-based local peer profile. The item-based local peer profile uses only document identifiers to represent a local peer profile. The term-based local peer profile uses a term-weight vector of relevant documents to represent a local peer profile.

Item-based Local Peer Profile

In the item-based local peer profile, we model peer profiles using only a set membership of a local peer relevant document set without taking into account the contents of the document metadata. Hence, the item-based local peer profile $L_I^t(p_i)$ of peer p_i at time t is defined as the local peer relevant document set $L^t(p_i)$.

$$L_I^t(p_i) = L^t(p_i) \tag{4.8}$$

When a peer p_i publishes or learns about a new document, d that is relevant to its interest, the peer adds d into its local peer profile.

$$L_I^{t+1}(p_i) = L_I^t(p_i) \cup \{d\} \tag{4.9}$$

Term-based Local Peer Profile

In the term-based local peer profile, each peer modeles its profile according to the term-weight document metadata model as discussed in Section 4.2.2 (page 41). The local peer relevant document set $L^t(p_i)$ is used as the relevant document set Q^t to create the profile document. As a result, the term-based local peer profile of peer p_i at time t, $L_T^t(p_i)$ is represented by a term-weight vector $\vec{L}^t(p_i)$; that is,

$$L_T^t(p_i) = \vec{L}^t(p_i) =< w_1^t, w_2^t, w_3^t, ..., w_V^t > \tag{4.10}$$

where term-weight w_v^t for each term v in $\vec{L}^t(p_i)$ is defined as Equation 4.5, where $Q^t = L^t(p_i)$.

When a peer publishes or learns about a new document d^θ, a new term weight vector $\vec{L}^{t+1}(p_i)$ is computed to represent the new term-based local peer profile L_T^{t+1}. The new term-weight w_v^{t+1} of term v in $\vec{L}^{t+1}(p_i)$ is computed by Equation 4.6, where

$Q^{t+1} = L^{t+1} = L^t \cup \{d^\theta\}$.

4.2.5 Known Peer Profiles

A known peer profile contains a set of profiles of known peers. The profile of each known peer represents the interests of the known peer that the local peer observes. Since peers only disseminate document metadata that are relevant to their interests to other peers by inserting them into the peer's share directory, the list of visited peers in each Shrack message represents a list of peers that are interested in the associated document.

Each peer learns about the interests of other peers when it receives a Shrack message that contains the other peers in the list of visited peers. The local peer learns that the visited peers are interested in the associated document. Hence, the local peer uses the associated document to update the profiles of the visited peers. In general, each peer can receive Shrack messages that contain the document metadata of a given document from many different paths. Each peer uses the list of visited peers associated with a given document that it receives to build its local knowledge of peers and uses the metadata of these documents to create the profile of each known peer.

For a local peer p_i, we define a set of documents that are relevant to the interest of its known peer p_j at a given time t as *a known peer relevant document set* $K^t(p_i, p_j)$.

Definition 17 (Known Peer Relevant Document Set) *$K^t(p_i, p_j)$ is a set of documents that a local peer p_i receives via peer p_j up to time t; that is, for each document $d \in K^t(p_i, p_j)$, the peer p_j is in the list of visited peers associated with d; that is,*

$$K^t(p_i, p_j) = \{d \mid d \in D^t(p_i) \cup S^t(p_i) \text{ and } p_j \text{ is in the list of visited peers of } d\} \quad (4.11)$$

The known peer relevant document set $K^t(p_i, p_j)$ is created locally in peer p_i and may not contain all documents that are relevant to peer p_j at a given time t. Each local peer uses the known peer relevant document set to create the profile of that peer.

In addition to peers in the visited list of received Shrack messages, each peer also learns about the existence of other peers when they request to pull from its shared directory. Each peer keeps the contact information of new peers, who issue pull requests from its shared directory, as its known peers with empty profiles. Empty profiles are used because the peer does not yet have knowledge of relevant documents to the new peers, until it pulls from their shared directory.

Similarly to the local peer profile, we also model the two known peer profile representations: item-based and term-based.

Item-Based Known Peer Profile

An item-based known peer profile contains a set of documents that are relevant to each known peer according to the information that the local peer receives. The known peer relevant document set $K^t(p_i, p_j)$ is represented as the profile of p_j in the known peer profile of p_i at time t, $K_I^t(p_i, p_j)$.

$$K_I^t(p_i, p_j) = K^t(p_i, p_j) \tag{4.12}$$

The item-based known peer profile of p_i consists of a set $K_I^t(p_i, p_j)$, where $p_j \in K^t$ and K^t is the set of all peers known to p_i at time t.

When a local peer pulls Shrack messages from its provider peers, every Shrack message that the local peer receives will be used to update its known peer profile. For each message, m, the local peer uses the associated document metadata to update the known peer profile of peers in the visited list, as shown in Algorithm 3.

Algorithm 3 p_i.updateKnownPeerProfile(m)

1: d is a document metadata embedded in a message m
2: **for** each visited peer p_j in m **do**
3: $\quad K^{t+1}(p_i, p_j) = K^t(p_i, p_j) \cup \{d\}$

When a local peer learns about a new peer from a pull request to pull messages from the local peer shared directory, the local peer creates the new profile with an empty set.

Term-Based Known Peer Profile

Each peer models its term-based known peer profile of each known peer according to the term-weight document metadata model in Section 4.2.2. For each peer p_j known to p_i at time t, its relevant document set $K^t(p_i, p_j)$ is represented as the profile document of p_j. The term-based profile of p_j in the known peer profile of p_i at time t, $K_T^t(p_i, p_j)$, is represented by a term-weight vector $\vec{K}^t(p_i, p_j)$. That is,

$$K_T^t(p_i, p_j) = \vec{K}^t(p_i, p_j) =< w_1^t, w_2^t, w_3^t, ..., w_V^t > . \tag{4.13}$$

where w_v^t is the term weight $TF \times IDF$ of the term v in $\vec{K}^t(p_i, p_j)$ and V is the number of terms in the vocabulary. The term-weight w_v^t is defined by Equation 4.5, where $Q^t = K^t(p_i, p_j)$. As a result, the term-based known peer profile of a local peer p_i consists of a set of vector $\vec{K}^t(p_i, p_j)$, where $p_j \in K^t$ and K^t is the set of all peers known to p_i at time t.

When a local peer pulls a Shrack message m from its provider peer, the local peer p_i includes the associated document metadata d^θ in the profile document of known peer p_j that exists in the list of visited peers. Each metadata of document d^θ is represented by a vector $\vec{d^\theta}$ and used to update a new term-based profile of known peer p_j, $K_T^{t+1}(p_i, p_j)$, represented by $\vec{K}^{t+1}(p_i, p_j)$. The new term-weight w_v^{t+1} of term v in $\vec{K}^{t+1}(p_i, p_j)$ is computed by Equation 4.6, where $Q^{t+1} = K^{t+1}(p_i, p_j) = K^t(p_i, p_j) \cup \{d^\theta\}$. The pseudo code for updating the known peer profile of peer p_i is shown in Algorithm 4.

The local peer also learns about the existence of other peers when they request

Algorithm 4 p_i.updateKnownPeerProfile(m)

1: $\vec{d^\theta}$ is a term-weight vector representing metadata of document d^θ in a message m
2: **for** each visited peer p_j in m **do**
3: | **for** each term v in $\vec{d^\theta}$ **do**
4: | | $w_v^{t+1} = \dfrac{\sum d_v + d_v^\theta}{\sum |\vec{d}| + |\vec{d^\theta}|} \times IDF_v$, where $d \in K^t(p_i, p_j)$

to pull from its shared directory and initializes the new profile with a zero vector.

4.2.6 Common Interest Score

To create self-organizing networks based on common interests, each peer connects to provider peers that have common interests to its local peer profile. We quantify the common interests between a local peer and a known peer as *a common interest score*.

Definition 18 (Common Interest Score) *$C^t(p_i, p_j)$ is a quantification of the common interests between a local peer p_i and a known peer p_j at time t*

The common-interest score is a scalar value and is calculated locally by a local peer, based on received information. The objective of the common-interest score is to quantify the quality of known peers according to the interests of a local peer, which is then used for peer ranking in the provider peer selection module. A peer p_i updates the common interest score $C^t(p_i, p_j)$ each time when the profile of its known peer p_j, $K^t(p_i, p_j)$, is updated.

The common interest score $C^t(p_i, p_j)$ does not satisfy the commutative property. Since each peer creates its known peer profile using information that it receives locally, for a given document metadata d that is relevant to both p_i and p_j, there may exist a dissemination path of d from p_j to p_i but not from p_i to p_j. The definition of the common interest score $C^t(p_i, p_j)$ is defined according to the type of peer profiles (item-based or term-based peer profiles), which are described next.

Item-Based Common-Interest Score

We quantify the item-based common-interest score based on a modification of the *Jaccard index* by measuring the similarity of an item-based local peer profile, $L_I^t(p_i)$, and an item-based profile of each known peer p_j, $K_I^t(p_i, p_j)$. The Jaccard index is a well-known similarity metric for sample sets and can be expressed as an extension of the cosine similarity measure to binary attributes.

In addition, we assume that a new peer p_j who pulls shared messages from a local peer p_i would tentatively have a common interest with the local peer, hence we set its common-interest score to 1, the maximum value. As a result, the local peer will select the new peer p_j as one of its provider peers the next time the local peer updates its provider peers to learn about the interests of p_j. After the local peer

pulls information from p_j, the common-interest score between the local peer p_i and the peer p_j will be computed regularly, as defined in Equation 4.14. In other words, this mechanism enables a peer p_j to introduce itself to p_i, by initiating a pull request to p_i.

The item-based common-interest score, $C_I^t(p_i, p_j)$, is defined as,

$$C_I^t(p_i, p_j) = \begin{cases} 1 \text{ if } p_j \text{ is new to } p_i \\ \dfrac{|L_I^t(p_i) \cap K_I^t(p_i, p_j)|}{|L_I^t(p_i) \cup K_I^t(p_i, p_j)|} \text{ otherwise} \end{cases} \tag{4.14}$$

Term-Based Common-Interest Score

We quantify the term-based common-interest score based on the cosine similarity measure, a measure of similarity between two vectors, which is widely used to compute the similarity of documents in a term-weight vector representation. The term-based common-interest score $C_T^t(p_i, p_j)$ of peer p_i and peer p_j, according to peer p_i at time t, is measured by the cosine similarity of the term-based local peer profile, $L_T^t(p_i)$, and the term-based profile of known peer p_j, $K_T^t(p_i, p_j)$.

We also set a common-interest score of a new peer p_j who requests to pull shared messages from a local peer p_i to 1 encouraging p_i to add p_j as a provider peer the next time p_i update its list of provider peers. Since, $L_T^t(p_i)$ is represented by $\vec{L}^t(p_i)$, and $K_T^t(p_i, p_j)$ is represented by $\vec{K}^t(p_i, p_j)$, the term-based common-interest score, $C_T^t(p_i, p_j)$, is defined as,

$$C_T^t(p_i, p_j) = \begin{cases} 1 \text{ if } p_j \text{ is new to } p_i \\ \dfrac{\vec{L}^t(p_i) \cdot \vec{K}^t(p_i, p_j)}{|\vec{L}^t(p_i)||\vec{K}^t(p_i, p_j)|} \text{ otherwise} \end{cases} \tag{4.15}$$

The rest of this chapter describes several provider peer selection strategies.

4.3 Provider Peer Selection Module

Each peer maintains a list of provider peers as sources of information from which the peer pulls information. To build a list of provider peers, a local peer initializes the list

with its start-up provider peers, as described in Section 3.5 (page 29). After successive pulls, the local peer uses information contained in the received Shrack messages to build its known peer profile and later selects some of them as its provider peers. We assume that a group of peers that have long term common interests are most likely to be interested in documents that are published by peers in the group. Ideally, for each local peer, its provider peers all together provide all the new documents published in the system that are of interest to the local peer. However, we cannot partition peers into disjoint groups because each peer may have multiple interests. Hence, each peer needs to select its set of provider peers locally. In addition, in a large unstructured peer to peer environment, it is expensive for a peer to pull messages from every peer in the system. As a result, we explore three provider peer selection strategies, namely: *common interest strategy*, *random strategy*, and *hybrid strategy* to select the best potential provider peers.

The following sections describe how peers get connected based on different provider peer selection strategies. The parameter N defines the maximum number of provider peers to which each peer may connect. Effectively, N is a property of Shrack network that determines the number of neighbours or the neighbourhood size.

4.3.1 Random Strategy

Since the Gossip protocol [43], a well-known scalable and reliable dissemination protocol for a large-scale network, disseminates information among peers based on random connections, a random strategy is selected as a baseline. In this strategy, a peer simply updates its provider peers by randomly selecting N known peers as its new provider peers without considering the common-interest score between the local peer and each known peer.

4.3.2 Common Interest Strategy

In the common interest strategy, a peer gets connected based on common interests with its known peers. We quantify the common interests between a local peer p_i and a known peer p_j at time t with the common-interest score $C^t(p_i, p_j)$ as previously defined in Section 4.2.6. When a peer wants to update its provider peers, the top-N

ranked known peers according to their common interest scores will be selected as a new set of provider peers. Ties in the scores are resolved by a random selection.

4.3.3 Hybrid Strategy

The hybrid strategy is chosen to reduce the effect of the greedy behaviour of the common interest strategy by allowing peers to also randomly explore peers in the network. In this strategy, each peer selects its provider peers from its top-ranked known peers with probability $1-\beta$, or randomly from its known peers with probability β, where β is an exploration parameter and $0 \leq \beta \leq 1$. The β parameter indicates how much peers want to explore the network. When β equals 0, the hybrid strategy behaves like the common interest based strategy. As β increases, the hybrid strategy behaves more like the random strategy.

We implement the hybrid strategy based on a random rewiring procedure for creating a mathematical model of a small-world network [76]. Each peer starts by selecting the top-N ranked known peers according to their common interest score as a new set of provider peers (the same process as in the common interest strategy). After that, for each selected provider peer, with a probability of β, the peer replaces the selected provider peer with a known peer chosen uniformly at random over the entire set of known peers, without replacement (i.e., duplicate selection is not possible). Otherwise, with probability of $1 - \beta$, the peer leaves the selected provider peer as is.

4.4 Summary

This chapter describes a prototype system of the Shrack Framework. We present the Shrack information dissemination protocol using pull-only communication. The Shrack dissemination protocol is similar to the Gossip protocol [43], however, peers only disseminate information that they are interested to other peers. In the Gossip protocol, peers disseminate information to other peers randomly. We describe how Shrack peers can form a self-organizing overlay networks to disseminate information among peers with similar interest. The knowledge integrator module describes how

peers integrate information they receive from their users and other peers to create a local knowledge of peers in the networks and identify peers with similar interest to their users. We explore two peer profile representations; item-based and term-based profile representations. The provider peer selection module describes how peers connect to other peers to create self-organizing overlay network. We explore three provider peer selection strategies; namely, common-interest, random, and hybrid strategies. In the next chapter, we present an experimental environment to evaluate the performance of the Shrack prototype system.

Chapter 5

Experimental Environment

This chapter describes an experimental environment to validate the Shrack framework. We create a Shrack simulator to realize the prototype system as discussed in the previous chapter. We also create simulated users to simulate interests of users in the network. We describe an authorship user interest model in Section 5.1. Then, we define the performance evaluation metrics in Section 5.2. After that, we give a brief introduction to our Shrack simulator in Section 5.3.

5.1 Authorship User Interest Model

We propose an authorship user interest model to create a simulated user associated with each peer in the simulated environment. The interests of peers are defined by the interests of the associated user. Each simulated user defines a set of documents that the associated peer publishes, $S^t(p_i)$, and a set of peer-relevant document set, $R^t(p_i)$, during the simulation, where t equals the total simulation time. Users may have overlapping interests. Therefore, we should not partition a set of documents to a group of peers. A certain degree of overlaped should be introduced. There exist some P2P data placement techniques for simulated environments, such as (1) using a focused crawler to create a document collection of each peer given a seed URL and topics [13], or (2) partitioning a fixed data set with a standard clustering algorithm and assigning a set of clusters to each peer with a sliding window technique to create an overlap data distribution [55]. However, although these techniques may be used to identify a set of document of which each peer should keep track during the simulation, there is no model for determining which peers should publish which documents.

We create the authorship user interest model from a collection of documents containing information about the interests of their authors. In our experiments, we

use the ACM Digital Library [1] metadata collection. Each author in the collection is viewed as a simulated user associated with a peer in the simulation. The list of documents that each user has published is the set of documents that the associated peer publishes. The interest of each user is modeled based on the ACM Computing Classification System (CCS)[2] according to the real ACM metadata collection. Each document in the ACM collection has been assigned to one or more CCS classes by the document authors. We assume that the interest of each user is identified by the work that he/she publishes, which could be associated with multiple classes. In our case, the interest of each user is described by the CCS classes of the documents that he/she has published. Subsequently, the set of documents of which a peer p_i should keep track during the simulation or the relevance documents is the documents in the peer-relevant document set excluding the documents in the peer-published document set, $R^t(p_i) \setminus S^t(p_i)$.

To simulate the user's feedback, we assume that users will give feedback according to their interest. When a peer receives Shrack messages, the messages containing metadata of document that is relevant to the associated user are inserted into the local peer's share directory for further dissemination, while non-relevant messages are discarded.

Table 5.1: An example of an authorship user interest model

Peer	Author	Publication	Documment Class	$R^t(p_i) \setminus S^t(p_i)$
p_1	a_1	d_1	g_1	d_5, d_7
		d_2	g_1	
p_2	a_2	d_3	g_2	d_6, d_7
		d_4	g_2	
p_3	a_3	d_5	g_1	d_1, d_2
		d_6	g_2	d_3, d_4
		d_7	g_1, g_2	

An example of an authorship user interest model is shown in Table 5.1. In this

[1] `http://portal.acm.org/dl.cfm`
[2] `http://www.acm.org/about/class/1998`

example, the set of documents consisting of seven publications $(d_1, d_2, \ldots, d_7)$ published by three authors (a_1, a_2, a_3). There are four document classes $(g_1, g_2, g_3,$ and $g_4)$. From this model, we can create three simulated users, namely a_1, a_2, and a_3, associated with three peers, namely p_1, p_2, p_3, respectively. The peer p_1, p_2, and p_3 are responsible to publish the document set of $\{d_1, d_2\}$, $\{d_3, d_4\}$, and $\{d_5, d_6, d_7\}$, respectively. Based on their publications, peer p_1 is interested in document group g_1; peer p_2 is interested in document group g_2; and peer p_3 is interested in document group g_1 and g_2; respectively. The $R^t(p_i) \setminus S^t(p_i)$ shows a set of documents of which each peer should keep track, which is excluding the documents that the peer self-publishes. In case of peer p_1, the relevant set is $\{d_5, d_7\}$ because d_5 and d_7 are labelled with class g_1, which is the interest class of peer p_1. Similarly, the set of documents that peers p_2 and p_3 should keep track of during the simulation are $\{d_6, d_7\}$, and $\{d_1, d_2, d_3, d_4\}$, respectively.

The main advantage of the authorship user interest model is that the overlap of interests of the users are created naturally based on the classes of documents they published.

5.2 Performance Evaluation Metrics

We evaluate the performance of self-organizing Shrack networks based on the *relevance of received document metadata* and the *dissemination speed and distance of relevant document metadata* to the local peer. Moreover, we analyse the properties of the resulting networks to determine whether they have properties of social networks.

As previously mentioned, in these experiments we would like to evaluate the performance of the system according to relevant documents a peer received based on the standard information retrieval metrics; namely, precision, recall, and F-score. Traditionally, these performance metrics are used to measure the performance of information retrieval systems based on a set of queries and returned results. Comparing with information retrieval systems, we can view the interest of each Shrack peer as a set of queries, and view the set of documents a peer receives as returned results. However, in Shrack, each document takes time to be disseminated. So, we divide the simulation times into time slots, define a set of documents published during each time

slot, allow time for their document metadata to be disseminated and then measure the performance of the system per publication time slot, when the dissemination ends.

We previously defined sets of documents with time t in Definition 12 through Definition 14. In this section, we extend these definitions to define sets of documents published during an arbitrary time slot $\tau_k = (t_s^k, t_e^k)$, where t_s^k and t_e^k are the start and the end of the time slot τ_k.

Definition 19 *$S^{\tau_k}(p_i)$ is the set of documents published by p_i during time slot τ_k; i.e., $S^{\tau_k}(p_i) = S^{t_e^k}(p_i) \setminus S^{t_s^k}(p_i)$.*

Definition 20 *P^{τ_k} is the set of documents published during time slot τ_k; i.e., $P^{\tau_k} = P^{t_e^k} \setminus P^{t_s^k}$.*

Definition 21 *$R^{\tau_k}(p_i)$ is the set of documents relevant to peer p_i that are published during time slot τ_k; i.e., $R^{\tau_k}(p_i) = R^{t_e^k}(p_i) \setminus R^{t_s^k}(p_i)$.*

The performance of the system with respect to a set of documents P^{τ_k} published during time slot τ_k is measured when the dissemination of documents in P^{τ_k} ends, denoted t_{end}^k. We determine the end of dissemination time t_{end}^k heuristically as the time when metadata of documents in P^{τ_k} have not been exchanged by peers for a predefined duration t_p. In practice, we monitor precision and recall of documents published in each time slot periodically as time progresses, and record the values that remain stable for a minimum period of time t_p.

Next, we define measurement metrics to evaluate the quality of received document metadata in Section 5.2.1 and the dissemination speed and distance in Section 5.2.2. After that, in Section 5.2.3, we describe measurement metrics to observe network properties.

5.2.1 Quality of Received Documents

We measure the quality of document metadata that a peer receives up to some point in time based on standard information retrieval quality measures: precision, recall, and F-score. A high precision value is indicative that the majority of documents that

a peer receives are actually relevant to the peer. Recall, on the other hand, measure coverage; that is, the percentage of interesting documents in the system that reach the peer. The F-measure facilitates comparing parameters and algorithm by combining precision and recall in a single numeric value. In the current system, Shrack peers present all the documents its received to the users without ranking. The precision, recall, and F-score are chosen to measure relevance of received documents. When the system incorporates ranking, other measurement measures that widely use to measure search results, e.g. mean average precision (MAP) [51], could be used. Next, we formally define precision, recall, and F-score.

Definition 22 (Precision) *$Precision^{\tau_k}(p_i)$ is the fraction of documents published during time slot τ_k, excluding self-published documents, received by peer p_i that are relevant to p_i over all the documents that are published by other peers in time slot τ_k that peer p_i receives. That is,*

$$Precision^{\tau_k}(p_i) = \frac{|R^{\tau_k}(p_i) \cap D^{t^k_{end}}(p_i)|}{|P^{\tau_k} \cap D^{t^k_{end}}(p_i)|} \tag{5.1}$$

Definition 23 (Recall) *$Recall^{\tau_k}(p_i)$ is the fraction of documents published during time slot τ_k relevant to peer p_i, excluding self-published documents, that are received by p_i over all the documents that are relevant to the interest of p_i and are published by other peers in time slot τ_k. That is,*

$$Recall^{\tau_k}(p_i) = \frac{|R^{\tau_k}(p_i) \cap D^{t^k_{end}}(p_i)|}{|R^{\tau_k}(p_i) \setminus S^{\tau_k}(p_i)|} \tag{5.2}$$

Definition 24 (F-score) *$Fscore^{\tau_k}(p_i)$ is the harmonic mean of $Precision^{\tau_k}(p_i)$ and $Recall^{\tau_k}(p_i)$. That is,*

$$Fscore^{\tau_k}(p_i) = \frac{2 \cdot Precision^{\tau_k}(p_i) \cdot Recall^{\tau_k}(p_i)}{Precision^{\tau_k}(p_i) + Recall^{\tau_k}(p_i)} \tag{5.3}$$

Note that, the standard measurement metrics—precision, recall and F-score—assume that there is a returned set of documents and a ground truth of relevant document set. In Shrack, for each peer, we could face a situation where we can not

measure the relevance of messages a peer receives, which we define as undefined states as following:

1. Undefined precision: $Precision^{\tau_k}(p_i)$ is undefined if peer p_i does not receive documents published in a particular time slot τ_k; i.e., $P^{\tau_k} \cap D^{t^k_{end}}(p_i) = \emptyset$.

2. Undefined recall: $Recall^{\tau_k}(p_i)$ is undefined if there are no relevant documents for peer p_i that are published in a particular time slot τ_k; i.e., $R^{\tau_k}(p_i) = \emptyset$.

3. Undefined F-score: $Fscore^{\tau_k}(p_i)$ is undefined if either of $Precision^{\tau_k}(p_i)$ or $Recall^{\tau_k}(p_i)$ are undefined.

We refer to a precision, recall, and F-score of the system at each publishing time slot τ_k, denoted $Precision^{\tau_k}$, $Recall^{\tau_k}$ and $Fscore^{\tau_k}$, as an average $Precision^{\tau_k}(p_i)$, an average $Recall^{\tau_k}(p_i)$ and an average $Fscore^{\tau_k}(p_i)$ over all peers in the network in each time slot, excluding undefined states, respectively. In addition, we measure the performance of the system by average precision, recall and F-score overall interested time slots.

5.2.2 Dissemination Speed and Distance

The dissemination speed and distance of relevant documents are measured in terms of pull delay and path length of relevant documents published at a given time slot that a peer receives, which are defined as follows:

Definition 25 (Relevant Pull Delay) *Given a peer p_i and a time slot τ_k, the relevant pull delay of documents published during time slot τ_k of p_i, $RelPullDelay^{\tau_k}(p_i)$, is defined as the average of the time delay from when d is published until p_i first observes d over all such documents d that are relevant to p_i that p_i receives; i.e., $d \in R^{\tau_k}(p_i) \cap D^{t^k_{end}}(p_i)$.*

Definition 26 (Relevant Path Length) *Given a peer p_i and a time slot τ_k, the relevant path length of documents published during time slot τ_k of p_i, denoted $RelPathLength^{\tau_k}(p_i)$, is defined as the average hop count over all such document metadata $d \in R^{\tau_k}(p_i) \cap D^{t^k_{end}}(p_i)$ that are relevant to p_i and p_i receives for the first time.*

We use the terms $RelPullDelay^{\tau_k}$ and $RelPathLength^{\tau_k}$ to refer to the average $RelPullDelay^{\tau_k}(p_i)$ and the average $RelPathLength^{\tau_k}(p_i)$ over all peers in each time slot, respectively. A peer is considered to experience better performance if it has lower relevant pull delay and lower relevant path length.

We introduce the relevant pull delay to represent the average time it takes for peers to learn about new documents of interest that are available in the system. The relevant path length is introduced to observe the average distance that relevant messages are travelled around the network.

5.2.3 Self-Organizing Network Property

We analyse the resulting network topologies to determine whether they form a social network by examining their clustering coefficients, characteristic path lengths, and degree distributions, which are defined as follow.

Definition 27 (Clustering Coefficient) *The network clustering coefficient, CCO, is the average of the clustering coefficient of peer p_i, $CCO(p_i)$, over all peers in the network. For a directed graph, $CCO(p_i)$ is defined as follow:*

$$CCO(p_i) = \frac{|E(p_i)|}{|N(p_i)|(|N(p_i)| - 1)} \tag{5.4}$$

where $E(p_i)$ is the set of edges between neighbours of p_i, and $N(p_i)$ is the set of neighbours of p_i.

The $CCO(p_i)$ is the number of edges actually existing between nodes in the neighbours of peer p_i, $N(p_i)$, divided by the number of all possible edges that could exist between nodes in $N(p_i)$. In our case, $E(p_i)$ is the set of connections between provider peers of p_i, and $N(p_i)$ is the set of provider peers of p_i.

Definition 28 (Characteristic Path Length) *the characteristic path length, CPL, of a network is the average length of the shortest paths between any two peers in the network.*

In the case of a disconnected network, the characteristic path length is the average shortest path between any two peers in the largest strongly connected component.

In addition, we consider only directed path between two peers according to pull connections.

Definition 29 (Degree Distribution) *The degree distribution of a network is defined as the complementary cumulative distribution function (CCDF) of the in-degree X of peers in the network, where the in-degree X of a peer p_i is the number of peers for which p_i is a provider peer. The CCDF is defined as,*

$$F_c(x) = Prob(X > x) \tag{5.5}$$

where $Prob(X > x)$ is the probability that the random variable in-degree X is greater than some value x.

We measure only the in-degree distribution. Since every peer has the same fixed number of provider peers, the out-degree distribution is not considered.

Many studies [11, 56, 75] report that social networks usually have small-world properties such as large clustering coefficient, small characteristic path length, and power-law scaling in degree distributions. Watts and Strogatz [75] show that small-world networks are highly clustered like regular lattices with a much higher clustering coefficient than random graphs of the same parameter, but have small characteristic path length similar to random graphs.

5.3 ShrackSim: A Shrack Simulator

ShrackSim is an event-based simulator that is developed on top of PeerSim [38], a Java based peer-to-peer simulator released to the public under the GNU General Public License (GPL).

We implemented ShrackSim to follow the structure of PeerSim such that it is modelled based on components, which make it easy to quickly test and modify Shrack's protocols and modules. Users can set up simulation parameters through a configuration file, which allows dynamic loading of components. ShrackSim provides several predefined objects to monitor the properties that users are interested in during a simulation, such as evaluation metrics and network properties. The monitor objects

can be also configured through the configuration file. ShrackSim is extensible and can be modified through PeerSim components, which provide different pluggable building blocks. We build ShrackSim to facilitate our and other researchers need to quickly test and evaluate different Shrack protocols and modules.

5.3.1 An Overview of PeerSim

PeerSim is a peer-to-peer simulator that is started under the EU projects BISON[3] and DELIS[4]. We give a synopsis of PeerSim documentation to introduce the readers to the concept of PeerSim on which ShrackSim relies. Detailed documentation of PeerSim can be found on the PeerSim project Web page[5].

PeerSim is written in Java and supports two simulation models: a cycle-based model and an event-based model. In the cycle-based model, events or components are scheduled per cycle, until a given number of cycles, or until a component decides to end the simulation. In the event-based model, everything works exactly the same way as in the cycle based model, except time management and the way control is passed to the control components. In the cycle-based model, by default, controls are called in each cycle. In the event-based model, the controls have to be scheduled explicitly. We develop ShrackSim on top of the event-based model, which allow us to write controls that are specific to events or messages and schedule them explicitly.

5.3.2 PeerSim Event-Based Simulation

The five main components behind the PeerSim event-based model that one used in ShrackSim are: *Node*, *Protocol*, *Event-Driven Protocole*, *Linkable Protocol*, and *Control*. Each of these components is described below.

Node The P2P network is composed of nodes. A node is a container of protocols. The Node provides access to the protocols it holds, and to a fixed identifier (ID) of the node.

[3]http://www.cs.unibo.it/bison/
[4]http://delis.upb.de/
[5]http://peersim.sourceforge.net/

Protocol Protocols are run by every node in the network. The two primary types of protocols are event-driven protocols and linkable protocols.

Event-Driven Protocol The Event-Driven Protocol is a type of Protocol that is designed to run in the event-driven model. This protocol is responsible for generating events to be added to the event queue, and may also handle the processing of these events.

Linkable Protocol The Linkable Protocol provides a service to other protocols to access a set of neighbour nodes. The instances of the same linkable protocol class over the nodes define an overlay network.

Control Controls are scheduled for execution at certain points during the simulation. Typically, Controls observe the simulation statistics or modify the simulation variables.

When setting up a new simulation, the following general steps are taken:

1. Choose a simulation length;
2. Choose the number of nodes;
3. Choose one or more protocols with which to experiment, and initialize them;
4. Choose one or more Control objects to monitor the properties you are interested in and to modify some parameters during the simulation; and
5. Run the simulation by invoking the Simulator with a configuration file that contains the above information.

5.3.3 Life Cycle of a PeerSim Event-Based Simulation

The life cycle of a PeerSim event-based simulation is depicted in Figure 5.1. The first step is to read the configuration file. The configuration contains all the simulation parameters concerning all the objects involved in the experiment. Then the simulator sets up the network, initializing the nodes in the network and the protocols in them.

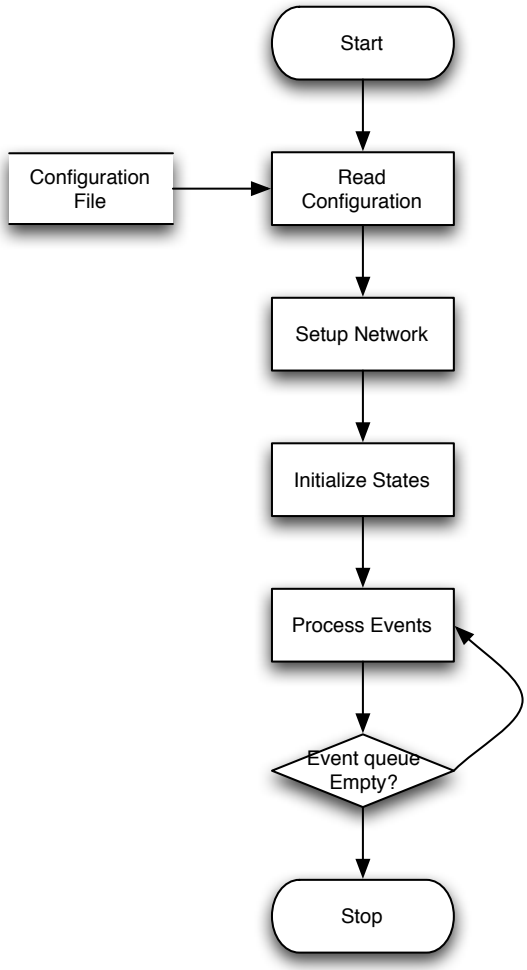


Figure 5.1: A flow chart of the PeerSim simulation life cycle

Each node has the same kinds of protocols; that is, instances of protocols form an array in the network, with one instance in each node. The instances of the nodes and the protocols are created by cloning. That is, only one instance is constructed using the constructor of the object, which serves as a prototype, and all the nodes in the network are cloned from this prototype.

At this point, initialization needs to be performed that sets up the initial states of each protocol and the simulation event queue. The initialization phase is carried out by Control objects that are scheduled to run only at the beginning of each experiment. In the configuration file, the initialization components are easily recognizable by the

init prefix.

After initialization, the execution of the simulation is driven by controls that generate events and send events (messages) to protocols. Each object in PeerSim (controls and protocols) is assigned a Scheduler object which defines when they are executed according to the event queue. The simulation stops when the event queue is empty (nothing left to do) or if all the events in the queue are scheduled for a time later than the specified end time.

5.3.4 Main Components of ShrackSim

ShrackSim is an implementation of the Shrack framework and architecture in simulated environment on top of PeerSim. Thus, ShrackSim consists of the following main components:

1. Shrack node
2. Simulated User model
3. Shrack protocol
4. Neighbourhood protocol
5. Observers

Next, we describe each of the ShrackSim components and their role in the simulation of a Shrack network.

Shrack Node A Shrack node represents a peer in a Shrack network and is used to compose an overlay network. In ShrackSim, the Shrack node acts as a container for the various protocols, same as PeerSim. In addition, each Shrack node is associated with a predefined simulated user. Shrack nodes also keep status and local statistics about their peer, which can be inspected by observers.

Simulated User Model The user model represents a simulated user that is associated with a Shrack peer. The user model consists of a user identifier, a list of

publications that the associated peer will publish during the simulation, and a list of interest (publication) groups of which the peer should keep track.

Shrack Protocol The Shrack protocol is the main event-driven protocol that implements the Shrack dissemination protocol, which processes pull requests and pull responses. It implements the Pull Procedure represented in Section 4.1.2 (page 36). The Shrack protocol is responsible for issuing pull requests and responding to pull requests from other peers. It maintains a representation of the local peer profile described in Section 4.2.4 (page 43), such as item-based and term-based representations. Further, it implements several neighbour selection strategies such as the random, top-rank, and hybrid strategies, described in Section 4.3 (page 49). It relies on the Neighbourhood protocol to identify the provider peers and determine where to send pull requests. It addition, it uses information from the user model to generate publication events; that is, inject new documents in to the network.

Neighbourhood Protocol The neighbourhood protocol is an implementation of the linkable protocol in PeerSim. It defines the Shrack overlay network that maintains a list of provider peers, a list of known peers. It maintains a representation of the known peer profiles described in Section 4.2.5 (page 45), such as item-based and term-based representations.

Observers An observer is an implementation of a PeerSim Control that is used to observe the status of protocols and components in the network. The observers are run periodically at a predefined step. The observers used in our simulations collect several statistical about the network, messages exchanged, and relevance of the received documents. The measured metrics, described in Section 5.2 (page 55), are:

- the average precision, recall, and F-score of received documents;
- the relevant pull delays;
- the relevant path length;

- the characteristic path length of the network;
- the clustering coefficient of nodes in the network; and
- the in-degree statistics of nodes.

Further implementation details including class diagrams, structure of the configuration file, and instructions on executing a simulation are given in Appendix D.

5.4 Summary

This chapter describes an experimental environment used to validate the Shrack framework. First, we explain how to create simulated users in the experiments by introducing an authorship user interest model. Then, we define the performance evaluation metrics to evaluate our prototype system according to the quality of received documents, the dissemination speed and distance, and the property of networks. Finally, we give an overview of the Shrack simulator.

Chapter 6

Shrack Dissemination Protocol

In this chapter, we study the performance of the Shrack information dissemination protocol with respect to the increase of network size. We define the experiment hypotheses and performance evaluation metrics in Section 6.1 and Section 6.2, respectively. After that, we define the experiment parameters in Section 6.3. We describe the experimental design in Section 6.4. We evaluate the performance of the dissemination protocol in two scenarios, first, when peers form collaboration directly without super peers, and second, when peers form collaboration through super peers. We present the experiments and discuss the results of the first and second scenarios in Section 6.5 and 6.6, respectively. At the end, we provide the summary in Section 6.7.

6.1 Experiment Hypotheses

The goal of these experiments is to study the scalability of the Shrack dissemination protocol. We observe the performance of Shrack dissemination protocol as the network size increases with the following hypotheses.

H6.1 We hypothesize that the Shrack dissemination protocol is scalable as the network size increases.

H6.2 We hypothesize that super peers can help improve the dissemination performance for normal peers in the system.

For simplicity, we assume that all peers are interested in all documents that are available in the system.

Table 6.1: Performance metrics for experiments on scalability of the Shrack dissemination protocol

Metric	Brief Definition
Pull Delay	The average time delay when a document published until a receiver peer first observes the document
Path Length	The average number of hops a message travels from the publisher peer to a receiver peer
Coverage	The percentage of peers that receives dissemination messages
Pull Load	The average number of messages transferred for each pull response
New Messages	The average number of messages in each pull response that are new to the receiver peer

6.2 Performance Evaluation Metrics

The performance metrics for this experiments are summarized in Table 6.1. The path length and coverage are standard metrics to measure message propagation performance. We use pulll delay to measure average time delay that a message propagate to receiver peers. We introduce pull load and new messages to measure communication load. We consider the scalability of the dissemination protocol in terms of dissemination speed and distance as describe in Section 5.2.2, namely *relevant pull delay* and *relevant path length*. Since in this experiment all documents are relevant documents, we refer to the relevant pull delay and the relevant path length as *pull delay* and *path length*, for short, respectively. We define the scalability of the dissemination protocol as follows.

Definition 30 (Scalability of dissemination protocol) *A dissemination protocol is scalable if the average pull delay over every peer is a logarithmic function of the network size.*

The pull delay is affected by the pull interval and the number of peers in the system. We are interested in assessing the behaviour of the pull delay as the number of peers in the system increases. With a large number of peers, the delay could be

prohibitively large, e.g., it could be in the order of months or years, and this is why we believe that the pull delay metric is meaningful.

Since every peer is interested in every documents in the system, the precision of the received messages always equals to 1.0. We replace the measure in terms of quality of received documents—precision, recall and f-score—with a dissemination coverage. The dissemination coverage is defined as the average number of peers that receive or learn about documents that are available in the system. We define the dissemination coverage as following:

Definition 31 (Dissemination Coverage) *The dissemination coverage of a document d is the ratio between the number of peers that receives the metadata of document d to the network size.*

In addition, we also measure the number of Shrack messages that are transferred from a provider peer to a receiver peer for each pull response, called *pull load*, and the number of messages in the pull load that are new to the receiver peer, called *new messages*.

6.3 Parameter Definitions

The experimental parameters are defined in Table 6.2.

Table 6.2: Input parameters for experiments on scalability of the Shrack dissemination protocol

Parameter	Definition
Pull interval	Time interval between successive pulls by a peer
Publishing rate	Average number of documents that a peer publishes per time unit
Number of provider peers	Number of provider peers that each peer connects to pull Shrack messages at each pull time
TTL	The initial TTL value

The pull interval, number of provider peers, and TTL are standard parameters for pull-based protocols. The TTL is a system-wide predefined value. Although the publishing rate is not a parameter of the Shrack dissemination protocol, it is a part of our experimental environment; representing peer behaviours.

6.4 Experimental Design

The experiments are conducted in our simulated environment, ShrackSim, as described in Chapter 5. We define *cycle* as a time measurement unit which could be of any fixed duration. Each cycle has 1,000 simulation clock units in PeerSim. In these experiments, we focus only on the dissemination protocol. We do not incorporate the self-organizing model into the system. The network is static, that is, every peer stays continuously in the network and the overlay networks are predefined. The pull delay depends on pull intervals and network communication delays. We assume that the point-to-point network communication delay between two peers is negligible because it is several orders of magnitude smaller than the pull interval. For instance, the point-to-point network delay could be in orders of milliseconds, while the pull interval could be several hours or days. Hence, in our experiments, pull delays depend only on the pull intervals.

We evaluate the performance of the information dissemination protocol in two scenarios. First, we test its performance on a uniform model, where there are no super peers in the system. Second, we test the performance in a super-peer model, where peers form collaboration through their super peers, and compare the results with the uniform model. We describe the experiment setup and discuss the performance of the dissemination protocol in the uniform model in Section 6.5 and the super-peer model in Section 6.6.

6.5 Shrack Dissemination Protocol in the Uniform Model

This experiment aims to test Hypothesis H6.1; that is, the Shrack dissemination protocol is scalable as the network size increases. We first describe the experimental setup and then discuss the performance of the dissemination protocol.

6.5.1 Experiment Setup

We evaluate the system in a uniform model where peers have similar resources and there are no super peers in the overlay network. Two peer-to-peer network models are used for testing the performance of the dissemination protocol: (1) a small-world network and (2) a random network. A small-world network is likely to be a more accurate model for social networks than a random network, as the latter induces a network topology without preferential attachment. In the first case, we connect provider peers based on the small-world model of Watts and Strogatz (WS) [76]. In the second case, provider peers are assigned randomly. For both topologies, we ensure that every peer in the network is a provider peer of at least one peer in the network, which guarantees that every peer is pulled by at least one peer during the simulation.

Table 6.3: Parameter setup for experiments on the scalability of the Shrack dissemination protocol in the uniform model

Parameter	Value
Number of provider peers	3 - 20 (power-law distribution; with exponent equal to 2.7)
Publishing rate	1 document per 30 cycles (Poisson distribution)
Pull interval	2 cycles (periodically)
TTL	20 hops
Overlay network	random, small-world

A summary of the parameter setup for the experiment is shown in Table 6.3. In both models, the power-law distribution is used to model the number of provider peers of each peer, where the exponent of the power law is 2.7 (based on [19]), and the minimum and the maximum size of the provider peers are 3 and 20, respectively. Each peer publishes documents over time according to a Poisson distribution with an average publishing rate of one document per 30 cycles, and periodically pulls Shrack messages every 2 cycles (but at a different, random starting time for each peer). The maximum hop count (TTL) is 20. We vary the network size and measure the

performance metrics observed by each peer and averaged over all peers.

6.5.2 Experiment Results

The dissemination pull delay and path length of both random and small-world networks are depicted in Figure 6.1 and Figure 6.2, respectively. The results are presented on a logarithmic scale of network size. In both networks, the results support the hypothesis that the Shrack dissemination protocol is scalable, since the average pull delay increases logarithmically as the size of the network increases. The performance in terms of dissemination path length correlates with the pull delay. The average path length also follows a logarithmic function of the network size. When the size of the network is greater than 100, the random networks shows smaller pull delay and smaller path length than the small-world networks.

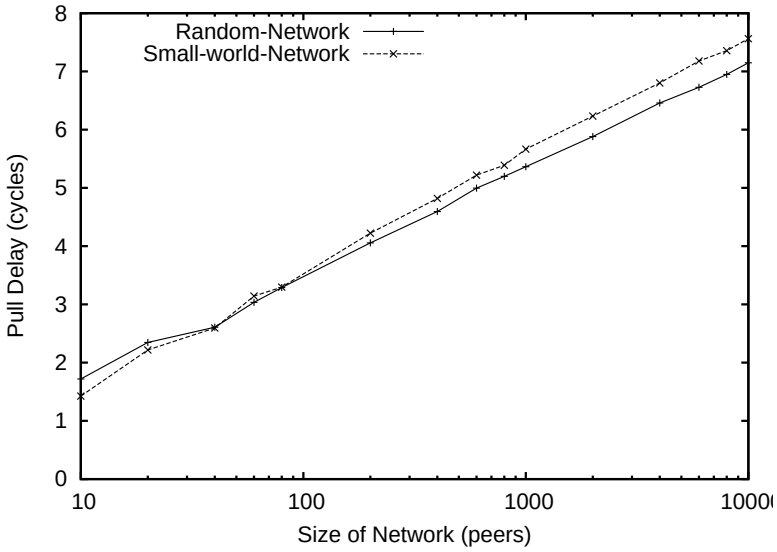


Figure 6.1: The pull delay of Shrack dissemination protocol on a uniform model as a logarithmic function of network size.

The dissemination coverage is presented in Figure 6.3. In both networks, the system always has 100% dissemination coverage. The result indicates that the dissemination protocol is reliable; that is, every message is disseminated to every peers in the network. The average pull load and new messages per pull response are reported as a function of the network size in Table 6.4. The results show that both

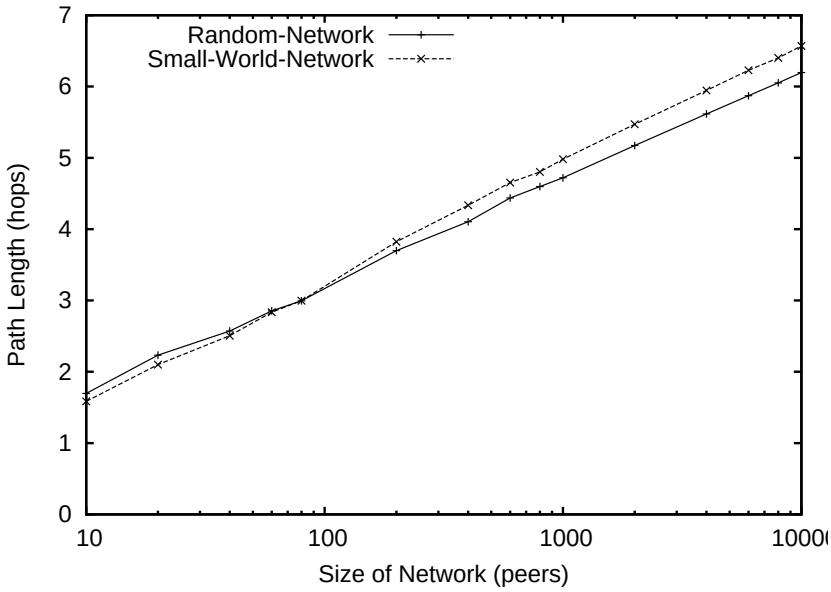


Figure 6.2: The dissemination path length of Shrack messages on a uniform model as a logarithmic function of network size.

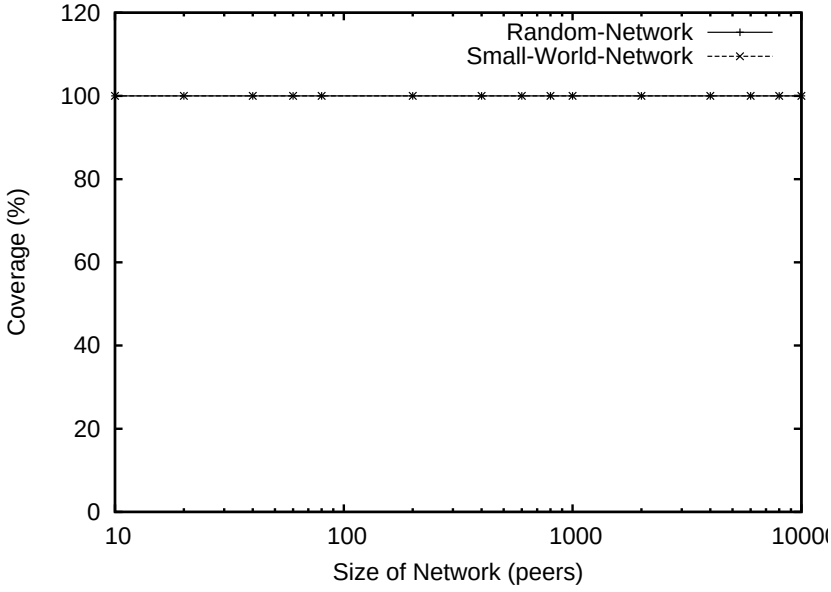


Figure 6.3: Dissemination coverage of Shrack dissemination protocol on a uniform model as a function of network size.

Table 6.4: The Pull Load and New messages per pull response of Shrack dissemination protocol on a uniform model as a function of network size

Network Size	Random Network		Small-World Network	
	Pull Load	New Messages	Pull Load	New Messages
10	0.43	0.08	0.55	0.10
20	1.20	0.24	1.26	0.26
40	2.02	0.33	2.51	0.41
60	3.56	0.61	3.96	0.69
80	4.77	0.86	5.23	0.90
200	11.74	2.25	12.82	2.42
400	25.11	4.66	23.61	4.46
600	37.15	7.03	35.92	6.66
800	48.62	9.03	47.76	8.77
1000	60.91	11.28	60.39	11.08
2000	119.78	22.16	121.68	22.62
4000	243.95	45.01	236.67	43.81
6000	357.81	65.55	352.38	65.17
8000	471.53	86.50	470.90	86.57
10000	589.06	108.71	586.01	107.88

Note: Pull Load and New Messages are defined in Table 6.1.

networks give the same characteristics of pull load and new messages. Each pull response receives constantly higher pull load than new messages. On average, for each pull response, the pull load and new messages increase linearly as the network size increases. To measure the *message overhead*, we compute the number of new messages per pull load. In other words, the message overhead is a number of copies of messages that are disseminated among peers in the network for a given message. The results show that the system has a constant message overhead of 5.47 on average, independent of the network size.

6.5.3 Discussion

The behaviour of the average pull delay is a direct consequence of the way Shrack messages propagate throughout the network. In general, more than one peer pulls shared directory of a given peer. Hence, Shrack messages propagate down in a tree

topology through the network at an exponential rate, where the dissemination path length represents the dept of the tree. This argument is supported by the observation that the average dissemination path length also follows a logarithmic function of the network size.

Our experiments were conducted in the ideal situation where peers can respond to every pull request. Hence, the experimental results show that the system is highly scalable. Although the propagation path of Shrack messages is similar to query routing in flooding-based networks, the behaviour is rather different, as explained below.

Flooding-based message query routing is a push-based protocol, which requires peers to forward messages immediately when a message arrives. In our protocol, peers pull information from the shared directory of other peers, and the information is replicated in several peers having similar interests. Peers can choose to pull information when they are ready and can select from which peers they want to pull. Moreover, in general P2P file-sharing systems, the number of original messages equals to the number of requests generated by each peer. In Shrack, the number of original root messages equals to the number of publications. Queries in general file-sharing systems might be for the same document but they are created multiple times for different requests, which results in creating multiple root messages. In Shrack, only new publications will be propagated as root messages among peers.

Our dissemination protocol is more similar to the gossip protocol [43] than the flooding-based message query-routing. The gossip protocol is known to be scalable and reliable for message dissemination in large-scale networks for group communication. Our proposed protocol differs from the gossip protocols in that peers self-organize into groups of peers having similar interests and disseminate only relevant messages, rather than push messages to randomly chosen peers and disseminate every message received. The self-organizing property of the Shrack networks and the corresponding performance metrics are investigated in Chapter 7.

In terms of message transfers, our dissemination protocol is also scalable, because each peer only receives a fixed number of messages (maximum one copy of metadata of each document) from each of its provider peers. The provider peers only send a pull

response once per pull request. As a result, the number of dissemination messages is bounded by the number of provider peers and not the size of the network. This argument is supported by the experimental results that the system has a constant message overhead independent of the network size. The results show that the pull load increases linearly as the size of the network increases, because in our experimental setup every peer has the same publication rate. Hence, the number of new documents in the system increases linearly as the network size increases, and the pull load increases as the number of new documents in the network increases. Although, duplicate copies of messages introduce the message overhead in the system, they also help the system achieve fault-tolerance in dynamic environment, where peers may leave or fail. However, further analysis is required to determine the optimal message overhead.

6.6 Shrack Dissemination Protocol in the Super-Peer Model

In this section, we conduct experiments to test Hypothesis H6.2 that states that super peers can help improve the dissemination performance for normal peers in the system. Super peers are peers that have high resources and availability. They can be thought of as peers that are associated with research organizations that maintain a collection of documents of interest to researchers in the organization. They only communicate with normal (not super) peers in their organization. Normal peers are viewed as researchers in the organization, which can form collaboration with peers within and outside the research groups. We study the effect of super peers on local group collaborations in the small-world network model.

Next, we describe the experimental setup and then discuss the results.

6.6.1 Experiment Setup

The overlay network in the super-peer model is defined as follows. Each normal peer is associated with only one super peer. Each super peer periodically pulls messages only from the peers in its local organization, with the same pull interval as normal peers. Each normal peer also periodically pulls document metadata from its super

peer, as well as from other peers in the network. An example of a super-peer overlay network is shown in Figure 6.4.

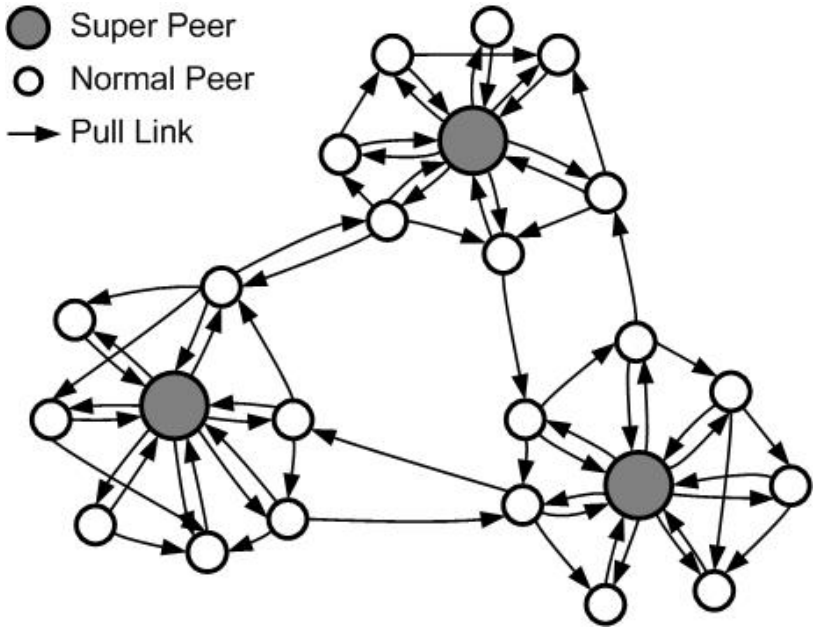


Figure 6.4: Example of a super-peer overlay network

In the experiments, each normal peer has the same parameter setup as in previous experiments as defined in Table 6.3. Only normal peers publish documents. Super peers only support the collaboration among normal peers and build the organization document collections. We test the performance of the system with collaboration of super peers in two configurations called *fix-super-peer* and *vary-super-peer* configurations. We summarize the experimental parameter setup for super peers in Table 6.5.

In the fix-super-peer configuration, the number of super peers is fixed at 10 independent of the network size, and the number of normal peers associated with each super peer is the same. In the vary-super-peer configuration, the number of super peers increases as the size of the network increases. We fix the number of normal peers associated with each super peer at 500, regardless of all network size. In both configurations, normal peers have the same configuration as the previous experiment presented in Table 6.3. Each peer must have one super peer as its provider peer. We vary the number of normal peers, and measure the dissemination characteristics,

Table 6.5: Parameter setup for experiments on scalability of the Shrack dissemination protocol in the super peer model*

Notation	Overlay Network	Number of super peers
No-super-peer	small-world	0
Fix-super-peer	small-world	fixed (10)
Vary-super-peer	small-world	vary (1 for every 500 normal peers)

* The parameter setup for normal peers is the same as in the previous experiment, presented in Table 6.3.

comparing them with the small-world network in the uniform model.

6.6.2 Experimental Results

The comparison of pull delay and path length of the super-peer models with the uniform model in the small-world network observed by normal peers are depicted in Figure 6.5 and Figure 6.6, respectively. The results show that in both configurations the super-peer models help improve the dissemination performance for normal peers in the system. The average pull delay and path length observed by normal peers in the super-peer models are smaller than in the uniform model and highly depend on the number of super peers. In the fix-super peer configuration, normal peers observe a near-constant average pull delay and path length in all network sizes. In the vary-super peer configuration, normal peers observe an increase of pull delay and path length as the network size increases.

The comparison of the average pull load and new messages per pull response observed by normal peers in the super-peer models and the uniform model is presented in Table 6.6 and 6.7, respectively. The results show that normal peers observed the same pull load and new messages in all configurations. This shows that super peers do not help in reducing the dissemination pull load observed by normal peers and, at the same time, they do not introduce additional pull load to the normal peers, which means that the existance of super peers does not affect the pull load of normal peers.

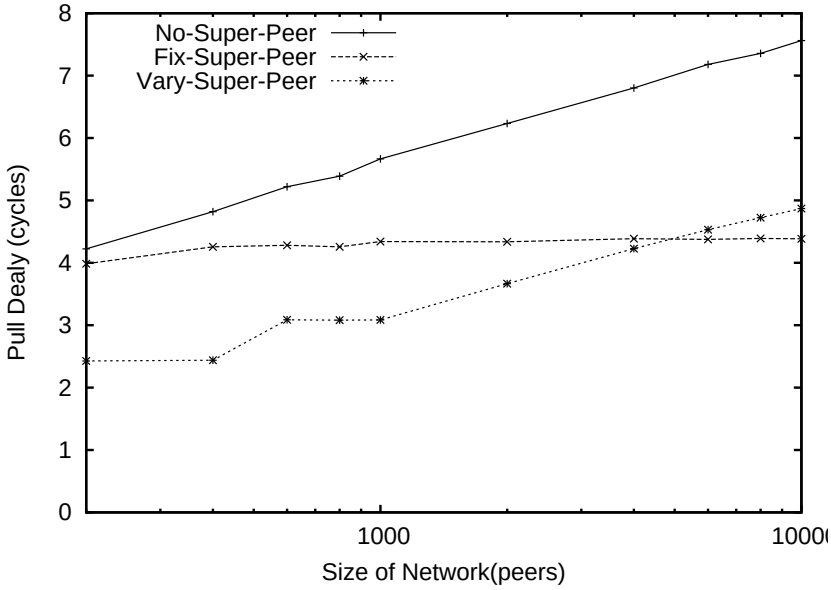


Figure 6.5: Dissemination pull delay observed by normal peers in super-peer models

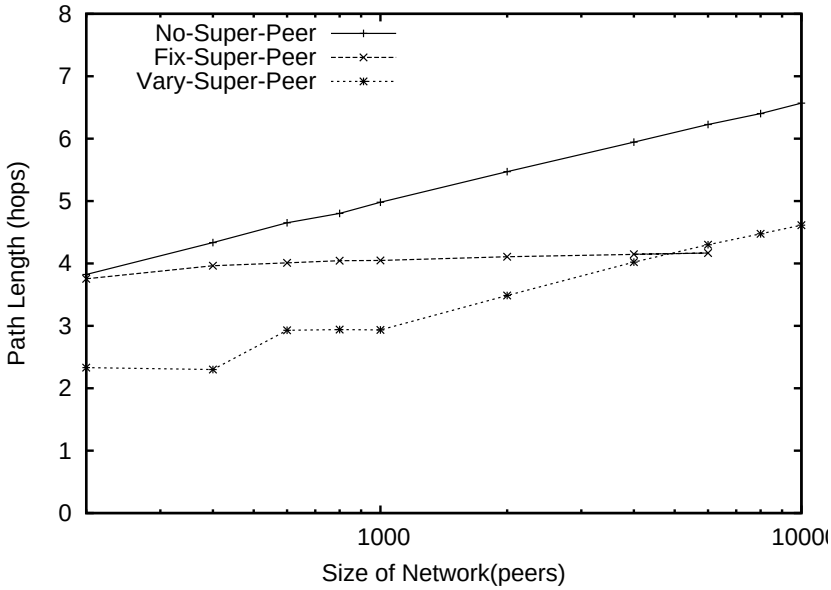


Figure 6.6: Dissemination path length observed by normal peers in super-peer models

Next, we present the dissemination performance observed by super peers, comparing between the fix-super-peer configuration and the vary-super-peer configuration. Figure 6.7 presents the dissemination pull delay. Super peers in the fix-super-peer

Table 6.6: The average pull load observed by normal peers in super-peer models

Network size	No super peers Pull Load	Fix super peers Pull Load	Vary super peers Pull Load
200	12.82	13.54	12.62
400	23.61	24.52	25.04
600	35.92	36.96	37.26
800	47.76	47.45	51.17
1000	60.39	60.42	63.06
2000	121.68	122.38	123.40
4000	236.67	242.90	241.49
6000	352.38	364.73	364.57
8000	470.90	n/a*	484.35
10000	586.01	n/a*	611.24

* n/a = no data available.

Table 6.7: The average new messages observed by normal peers in super-peer models

Network size	No super peers New Messages	Fix super peers New Messages	Vary super peers New Messages
200	2.42	2.54	2.37
400	4.46	4.63	4.69
600	6.66	6.86	6.91
800	8.77	8.72	9.36
1000	11.08	11.13	11.62
2000	22.62	22.80	22.95
4000	43.81	44.89	44.68
6000	65.17	67.33	67.34
8000	86.57	n/a*	89.00
10000	107.88	n/a*	112.44

* n/a = no data available.

configuration observe a nearly constant pull delay, while super peers in the vary-super-peer configuration observe a logarithmically increasing pull delay as the network size increases. The dissemination path length observed by super peers correlates with the pull delay as presented in Figure 6.8.

The dissemination pull load and the number of new messages per pull response

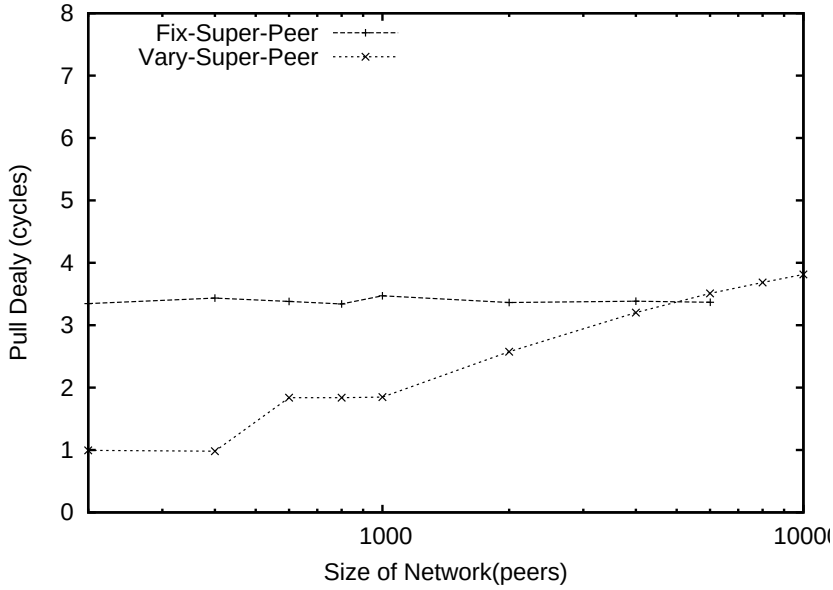


Figure 6.7: Dissemination pull delay observed by super peers

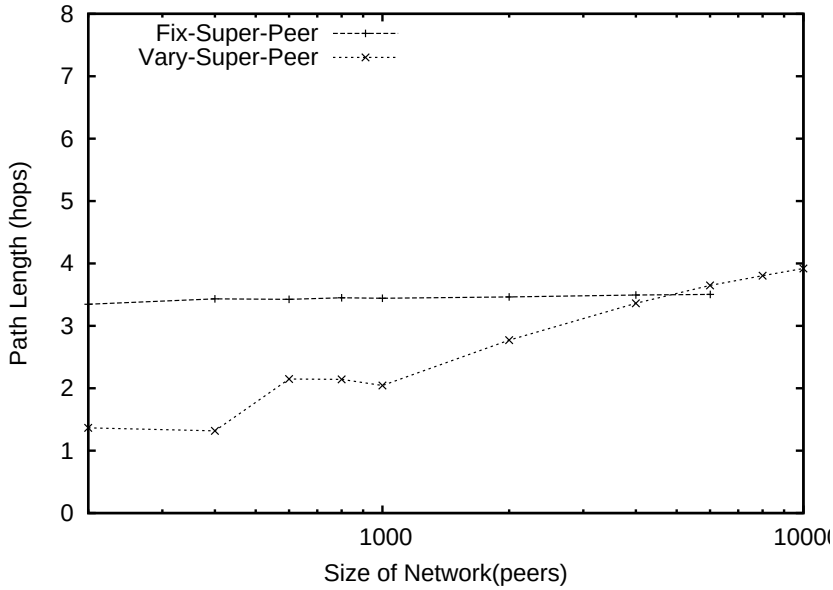


Figure 6.8: Dissemination path length observed by super peers

observed by super peers in the fix-super-peer and vary-super-peer configuration are presented in Table 6.8 and Table 6.9, respectively. The results shows that super peers in both configurations experience the same pull load per pull response. However,

the number of new messages per pull response of super peers in the fix-super-peer configuration does not change, while the number of new messages per pull response of super peers in the vary-super-peer configuration increases as the network size increase. We compute the message overhead, which shows that the message overhead of super peers correlates with the number of their associated normal peers or group size.

Table 6.8: Comparison of pull load, new message per pull response, and message overhead of super peers with a fixed number of super peers

Normal peers	**Group size***	**Pull Load**	**New Message**	**Overhead**
200	**20**	13.50	0.69	**19.45**
400	**40**	24.44	0.64	**38.39**
600	**60**	36.78	0.63	**57.99**
800	**80**	47.40	0.61	**77.11**
1000	**100**	59.89	0.62	**96.29**
2000	**200**	121.61	0.63	**192.21**
4000	**400**	242.91	0.63	**385.26**
6000	**600**	363.75	0.63	**578.26**

* Group size is the number of normal peers associated with each super peer.

Table 6.9: Comparison of pull load and new messages per pull response, and message overhead of super peers with vary number of super peers

Normal peers	**Group size***	**Pull Load**	**New Message**	**Overhead**
200	**200**	12.46	0.07	**191.67**
400	**400**	24.79	0.06	**385.58**
600	**300**	37.09	0.13	**288.19**
800	**400**	51.01	0.13	**387.10**
1000	**500**	62.28	0.13	**478.12**
2000	**500**	122.99	0.26	**480.83**
4000	**500**	241.23	0.50	**481.35**
6000	**500**	363.99	0.76	**480.85**
8000	**500**	483.24	1.01	**479.96**
10000	**500**	608.37	1.26	**481.10**

* Group size is the number of normal peers associated with each super peer.

6.6.3 Discussion

The experiment results show that super peers help reduce the dissemination path length, and consequently the pull delay of peers in the same groups, which eventually effects the dissemination path length between peers in the network. Hence, the pull delay and path length correlate with the number of super peers or collaborative groups in the network. We present the number of normal peers and the number of super peers in the fix-super-peer and vary-super-peer configurations in Table 6.10.

Table 6.10: The number of super peers in the network with fix-super peer and vary super peer configurations

Normal peers	**Fix super peer Super peers**	**Vary super peer Super peers**
200	10	1
400	10	1
600	10	2
800	10	2
1000	10	2
2000	10	4
4000	10	8
6000	10	12
8000	10	16
10000	10	20

In the fix-super-peer configuration, peers experience nearly constants pull delay and path length because there are always 10 collaborative groups in the system. In the vary-super-peer configuration, the pull delay and path length are about the same when the network size equals to 200 and 400, since both networks contain one collaborative group. Similarly, when the network size is between 600 and 1,000 peers, the networks contain two collaborative groups, and the pull delay and path length are nearly constant. When the network size increases from 2,000 to 10,000 peers, the pull delay and path length increase as the number of collaborative groups increases.

Although, the fix-super-peer configuration can improve the dissemination performance to be nearly constant independent of the network size, its super peers have to

handle more load as the network size increases. Even though the pull load per pull response is the same for both configurations, the total pull load for each pull interval is different. Since, in each pull interval, super peers pull messages from all of their associated normal peers, the total pull load for each pull interval of each super peer correlates with the size of collaborative groups. As a result, the total pull load of super peers in the fix-super-peer configuration increases as the network size increases, while the total pull load of super peers in the vary-super-peer configuration remains constant. Subsequently, the message overhead of super peers in the fix-super-peer configuration increases as the size of the network increases, while the message overhead of super peers in the vary-super-peer configuration remain constant as presented in Table 6.8 and 6.9, respectively.

The experiment results also show that super peers have to handle a large amount of overhead, which increases linearly as the size of collaborative groups increases. This overhead could be reduced in several ways, since super peers do not need to pull the same copy of a given message from every normal peers in the associated groups. For example, in each pull interval, each super peer could only pull messages from a randomly chosen subset of the normal peers. In addition, each super peer can reduce the update time for each pull request to pull only a few latest messages that normal peers receive. For normal peers, their message overhead remains the same in all configurations, because in the experiments, they all have the same parameter setup for the number of provider peers.

6.7 Summary

In this chapter, we present experimental evaluation of the Shrack dissemination protocol. We evaluate the dissemination protocol based on scalability criteria. The experimental results support our hypotheses that:

- H6.1: The Shrack dissemination protocol is scalable as the network size increases.
- H6.2: Super peers can help improve the dissemination performance for normal peers in the system.

In the uniform model, the dissemination pull delay follows a logarithmic function of the network size. In addition, the messages overhead correlates with the number of provider peers, not the size of the network. In the super-peer models, the normal peers observe a smaller pull delay and path length when they form collaborative groups through super peers.

These experiments are setup in ideal environment to understand the behaviour of the dissemination protocol. The network topologies are predefined in static overlay networks and all the peers are connected. In addition, peers always respond to all requests and are interested in all documents available in the system. In the next chapter, we study the system where peers are self-organizing and peers are only interested in a subset of documents available in the system.

Chapter 7

Self-Organizing Shrack Networks

In the previous chapter, we evaluate the performance of the dissemination protocol where we assume that every peer is interested in every document published in the system. In this chapter, we focus on users' interests and self-organizing networks. Given a set of peers with overlapping interests where each peer wishes to keep track of new documents that are relevant to their interests, we study the behaviour of the Shrack prototype where peers form self-organizing network based on common interest of document set. Next, we present the experiment hypotheses and the road map of this chapter.

7.1 Experiment Hypotheses and Road Map

There are two hypotheses for experiments in this chapter defined as follows.

H7.1 We hypothesize that self-organizing Shrack networks based on common interest of document set can enhance quality of documents received by peers in terms of F-score over random connected networks.

H7.2 We hypothesize that the resulting self-organizing networks based on common interest of document set have the characteristics of social networks.

This two hypotheses are tested on the same experimental sets using different performance metrics. The road map of this chapter is summarized in Table 7.1. The first hypothesis is measured in the aspect of the quality of received documents. The second hypothesis is measured in the aspect of the self-organizing network property. In addition, we also observe the performance of the system in the aspect of dissemination speed and distance. We provide a summary of the performance metrics for each

Table 7.1: The road map of experiments on self-organizing Shrack networks

.

Hypothesis 1	**Hypothesis 2**
Performance Metrics (Section 7.2)	
Data Set Preparation (Section 7.3)	
Experiment Setup Section 7.4	
Quality of Received Documents (Section 7.5) Dissemination Speed and Distance (Section 7.6)	Self-Organizing Network Property (Section 7.7)
Summary (Section 7.8)	

aspect in Section 7.2. In Section 7.3, we describe the dataset preparation. Then, we explain the experiment setup in Section 7.4. We present the experimental results and discussion in the aspect of the quality of received documents, the dissemination speed and distance and the self-organizing network property in Section 7.5, 7.6, and 7.7. Finally, we provide a summary in Section 7.8.

7.2 Performance Metrics

The summary of performance metrics in the aspect of the quality of received documents, the dissemination speed and distance and the self-organizing network property is presented in Table 7.2. The definition of each performance metrics is defined in Section 5.2 (page 55).

7.3 Dataset Preparation

In this section, we explain the preparation of the experiment dataset including the simulated users and the term-based peer profile representation.

Table 7.2: The summary of peformance metrics in the aspect of the quality of received documents, the dissemination speed and distance and the self-organizing network property

Performance	Metrics
Quality of received documents	Precision Recall F-score
Dissemination speed and distance	Relevant pull delay Relevant path length
Self-organizing network property	Clustering Coefficient Characteristic Path Length Degree Distribution

7.3.1 Simulated Users

We create our simulated users from authors who published documents in class H.3.3, information search and retrieval, in the year 2008, in the ACM metadata collection. We use a set of documents in class H.3.3 that these authors published since the year 2000 as our document dataset. There are 7 subclasses in class H.3.3. These subclasses are used to define interests of the artificial users and document classes in our simulation. Each document can be labelled by multiple subclasses, which define the interests of the document's authors. We select the top 1,000 authors according to the number of documents they published as the set of the artificial users. From these authors, we have 1,639 documents in our simulation.

We show the number of users and documents in each subclass in Table 7.3. Out of 1,000 users, 38% are interested in one subclass, 31% are interested in two subclasses, the rest are interested in three or more subclasses. The majority of documents, 72%, are labelled with one subclass.

7.3.2 Term-based Peer Profile Representation

For term-based profiles, we estimate the global inverse document frequency (IDF) of each term from the ACM document metadata collection that was published before

Table 7.3: The number of users and documents in each subclass

Subclass Name	#Users	#Docs
Clustering	235	222
Information filtering	268	238
Query formulation	327	354
Relevance feedback	148	151
Retrieval models	579	629
Search process	437	493
Selection process	144	110

year 2000. Note that our test set is the documents in class H.3.3, that are published since year 2000. We precompute feature vectors for each peer based on their interests, selecting top 200 terms ranked by their TFxIDF weight. However, the weigh of each term in the profile vector is computed locally and dynamically according to the messages each peer receives.

7.4 Experimental Setup

We use ShrackSim, discussed in Chapter 5 (page 53), to study the performance of self-organizing Shrack networks. We define a "cycle" as a time measurement unit, which could be of any fixed duration. Each cycle has 1,000 simulation clock units. We assume that the point-to-point network communication delay between two peers is negligible. We run the simulation on a network with 1,000 peers with the initial connections formed randomly. Initially, peers only have contacts of peers that they get connected to, then they learn about other peers in the network and form a self-organizing network as described in the prototype system, Chapter 4 (page 33).

We conducted two sets of experiments:

1. *Item-based Profile:* Each peer represents its local and known peer profiles using the item-based profile representation.

2. *Term-based Profile:* Each peer represents its local and known peer profiles using the term-based profile representation.

Table 7.4: Experimental parameter setup for self-organizing Shrack network

Property	Value
Peer Profile representation	Item-based Term-based
Provider peer selection	Common Interest Random Hybrid1, Hybrid2, Hybrid3
System Publishing rate	1 documents per 4 cycles (Poisson distribution)
Pull Interval	20 cycles (periodically)
TTL	8
Size of provider peers	3-15
Maximum Update	160 cycles
Number of random seeds	10

The summary of the experiment parameter setup is presented in Table 7.4. Each set of experiments uses five provider peer selection strategies, namely the common-interest strategy, the random strategy and the hybrid strategy with $\beta \in 10^{-1}, 10^{-2}, 10^{-3}$. In all experiments, documents are published in the system by the peer associated with the first author. The publishing time of documents in the system follows a Poisson distribution with an average publishing rate of 1 document per 4 cycles. The document publication time in the simulation is independent of the time of the actual ACM publication. The pull interval is fixed at 20 cycles with a different (random) starting time for each peer. The experiments are conducted with a TTL value of 8 (selected experimentally) for experiments with a limited TTL.We vary the number of provider peers from 3 to 15. Each peer sets the maximum update time to 160 cycles—the product of the TTL (8) and the pull interval (20 cycles). Each configuration is tested with 10 simulations, with different random seeds to initialize the network connections.

We evaluate the performance of the system as a function of time using a sliding window. The parameter t_p of the criterion determining dissemination end is set at 200

cycles. We use a window size of 400 cycles in duration, with 200 cycles overlap between successive windows; i.e., $t_s^k = \{0, 200, 400, ..., 5400\}$, where $k = \{1, 2, 3, ..., 29\}$. We ignore the last 4 time slots, τ_{30} through τ_{33}, because the simulation ends before the end of the dissemination of documents published in these time slots. The network property is measured every 200 cycles. Each performance metric for a given observation time, except for the degree distribution, is averaged over the 10 simulations, for each configuration.

To measure the performance and network property of Shrack networks in each configuration, we average each evaluation metric, except the degree distribution, over the last 10 time slots, τ_{20} through τ_{29}. The results are presented as a function of the number of provider peers.

We compare the performance and property of the Shrack network among five provider peer selection strategies, namely the common-interest strategy, the random strategy, the hybrid strategy with $\beta = 10^{-1}$, the hybrid strategy with $\beta = 10^{-2}$, and the hybrid strategy with $\beta = 10^{-3}$. We present the experimental results from two set of experiments; (1) item-based profiles with a limited TTL and (2) term-based profile with a limited TTL. We denote the item-based profile and term-based profile representation by the similarity measure it uses to quantify the common-interest score, which are *Jaccard* and *Cosine*.

Table 7.5 presents the notation of provider peer selection strategies with item-based or term-based profile representation. Note that the random provider peer selection strategy does not require peer profiles. The experimental results and discussion are divided under three groups; (1) the quality of received document metadata, (2) the dissemination speed and distance of relevant document metadata, and (3) the self-organizing network property.

7.5 Quality of Received Documents

In this section, we present experiment results and discussion in terms of precision, recall, and F-score.

Table 7.5: Notation of provider peer selection strategy with item-based or term-based profile representation

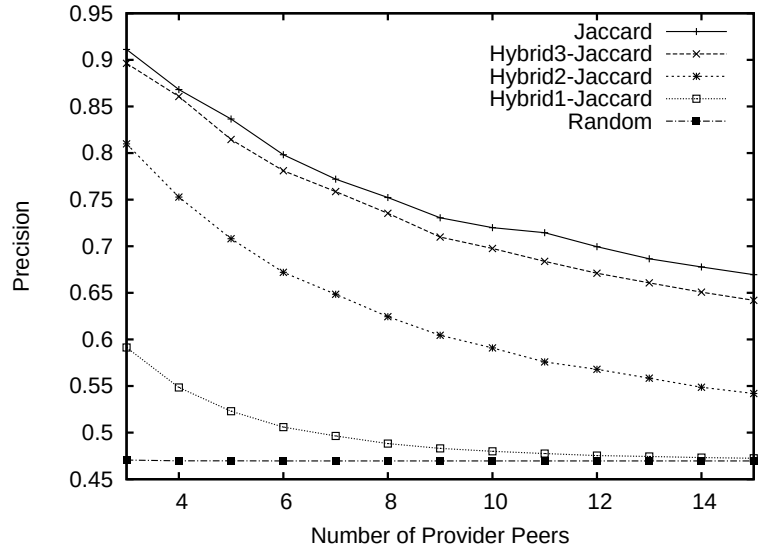
Notation	Provider peer selection strategy	Profile representation
Jaccard	Common interest	Item-based
Hybrid1-Jaccard	Hybrid with $\beta = 10^{-1}$	Item-based
Hybrid2-Jaccard	Hybrid with $\beta = 10^{-2}$	Item-based
Hybrid3-Jaccard	Hybrid with $\beta = 10^{-3}$	Item-based
Cosine	Common interest	Term-based
Hybrid1-Cosine	Hybrid with $\beta = 10^{-1}$	Term-based
Hybrid2-Cosine	Hybrid with $\beta = 10^{-2}$	Term-based
Hybrid3-Cosine	Hybrid with $\beta = 10^{-3}$	Term-based
Random	Random	-

7.5.1 Experiment Results

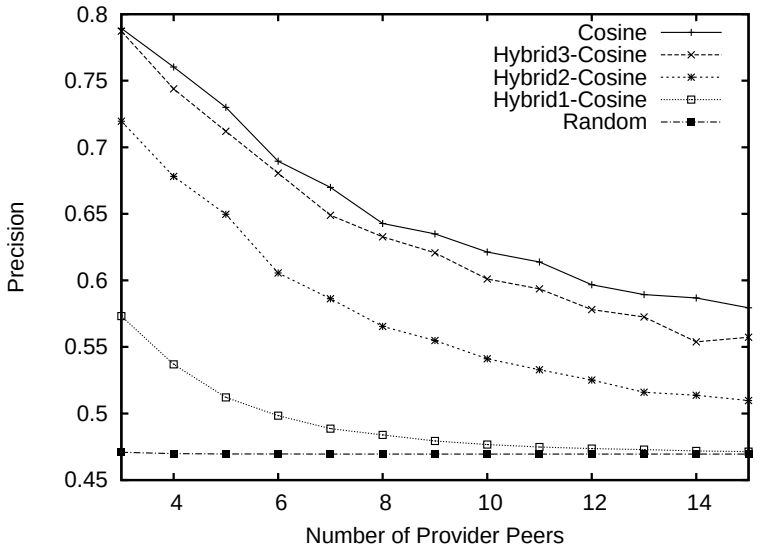
Precision The precision of document metadata that Shrack peers received, averaged over all peers is presented in Figure 7.1. Figure 7.1 (a) shows the precision of the system when the item-based profile representation is used. Figure 7.1 (b) shows the precision of the system when the term-based profile representation is used.

The experimental results are very consistent in both sets of experiments. The common interest strategy always gives the best precision compared with other types of provider peer selection strategies. On the other hand, the random strategy always gives the lowest precision. The hybrid strategy shows the mixture effects of the common interest and random strategy and gives the precision in between the precision of the common interest and random strategy. As the value of the exploration parameter β increases, the hybrid strategy behaves more like the random strategy. In both sets of experiments, Hybrid3 gives higher precision than Hybrid2 and both of them give higher precision than Hybrid1.

We did a ANOVA to test for statistical significance of peer selection strategy on the precision. There is a significant main effect of peer selection strategy on the precision ($F = 26877.7$, $df = 4$ and 5940, $p < 0.001$, partial $\eta^2 = 0.948$). The Scheffé



(a)



(b)

Figure 7.1: The quality of document received in terms of precision as the number of provider peers increases using different provider peer selection strategies; (a) an item-based profile representation; (b) a term-based profile representation

post hoc testing indicates that the common interest strategy gives the best precision followed by Hybrid3, Hybrid2, Hybrid1, and Random.

As the number of provider peers increases, the precision of every provider peer selection strategy decreases, except for the random strategy. In the random strategy,

the precision remain unchanged as the number of provider peers increases. However, in all configurations, as the number of provider peers increases, the order of performance in terms of precision remains unchanged. The ANOVA and Scheffé post hoc testing indicate that there is a statistically significant effect of the number of provider peers on the precision ($F = 4487.8$, $df = 5$ and 5940, $p < 0.001$, partial $\eta^2 = 0.791$).

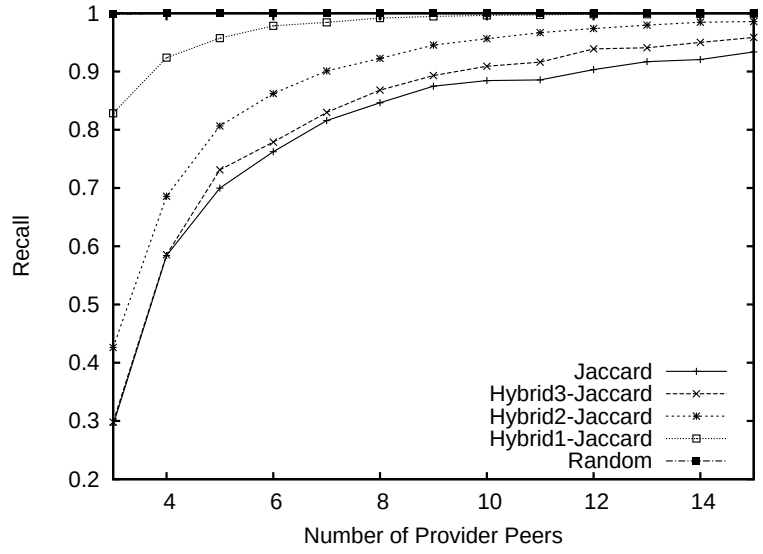
The partial η^2 indicates that the selection strategy is the best prediction of the precision (partial $\eta^2 = 0.948$). The next best prediction is the number of provider peers (partial $\eta^2 = 0.791$). The complete ANOVA test is presented in Appendix B Table B.1 (page 162).

Recall The quality of received document metadata in terms of recall, averaged over all peers over the last 10 time slots is presented in Figure 7.2. Figure 7.2 (a) shows the recall of the system when the item-based profile representation is used. Figure 7.2 (b) shows the recall of the system when the term-based profile representation is used.

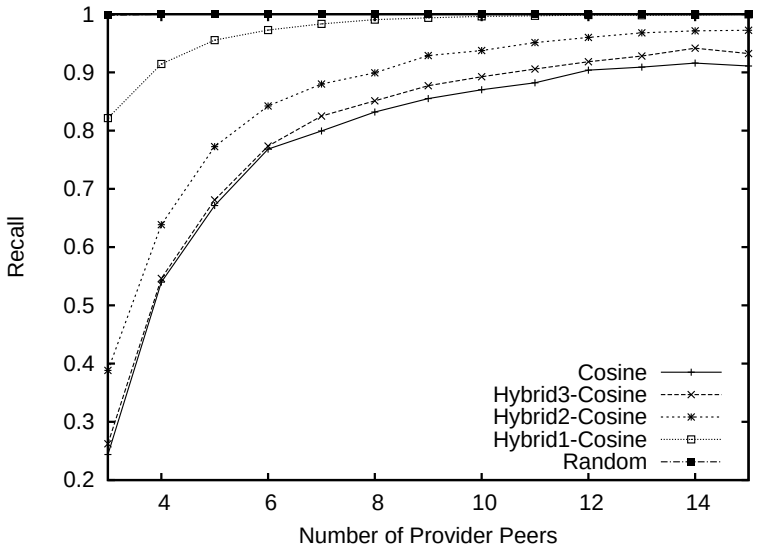
Both sets of experiments give similar results, which shows a reverse order of performance in terms of recall, compared with the precision. In terms of recall, the random strategy always gives the best and nearly perfect performance; i.e., its recall is always very close to 1.0 (always greater than 0.998), in all configurations. Conversely, the common interest strategy gives the lowest recall. Similar to the precision performance, the hybrid strategy gives the mixture effects of the common interest and random strategy. All hybrid strategies give recall in between the recall of the common interest and random strategy. As the value of β increases, the hybrid strategy gains higher recall and behaves more like the random strategy.

The ANOVA and Scheffé post hoc testing indicate that there is a statistically significant main effect of the selection strategy on the recall ($F = 11129.7$, $df = 4$ and 5940, $p < 0.001$, partial $\eta^2 = 0.882$). The Random strategy gives the best recall followed by Hybrid1, Hybrid2, Hybrid3, and the common interest strategy.

Unlike the recall of the random strategy, the recall of the common interest and hybrid strategies are sensitive to the increment of the number of provider peers. As the number of provider peers increases, the recall of the common interest and hybrid strategies increases. The ANOVA and Scheffé post hoc testing indicate that there



(a)



(b)

Figure 7.2: The quality of document received in terms of recall as the number of provider peers increases using different provider peer selection strategies; (a) an item-based profile representation; (b) a term-based profile representation

is a statistically significant main effect of the number of provider peers on the recall ($F = 9131.2$, $df = 5$ and 5940, $p < 0.001$, partial $\eta^2 = 0.885$).

The partial η^2 shows that the selection strategy and the number of provider peers are the best prediction of recall with partial η^2 equals 0.882 and 0.885, respectively.

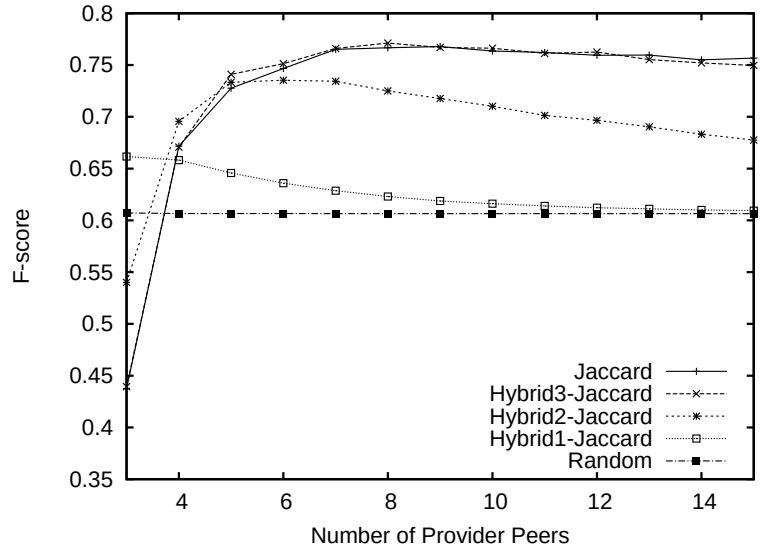
This results indicate that the selection strategy and the number of provider peers account for more than 80% of the overall variance of recall, which is considered to be a large effect size. The complete ANOVA test is presented in Appendix B Table B.2 (page 163).

F-score The performance in terms of F-score, which is a harmonic mean of precision and recall is depicted in Figure 7.3. Figure 7.2 (a) shows the F-score when the item-based profile representation is used. Figure 7.2 (b) shows the F-score of the system when the term-based profile representation is used.

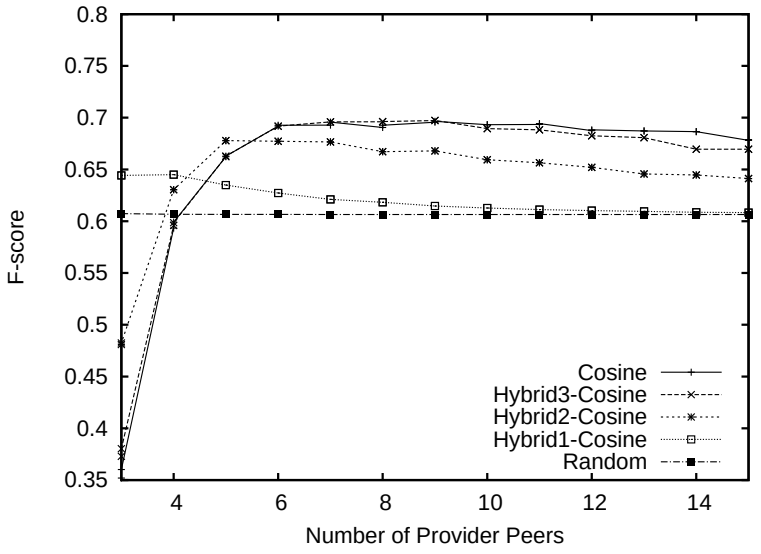
In both sets of experiments, when the number of provider peers equals three, the common interest and the hybrid with β equals 10^{-3} (Hybrid3) and 10^{-2} (Hybrid2) give very low F-score, while the hybrid with β equals 10^{-1} (Hybrid1) gives the best F-score followed by the random strategy. As the number of provider peers increases, the common interest strategies, Hybrid3 and Hybrid2 gain more recall than the loss of precision, as a result, their F-score increase. The F-score of Jaccard, Hybrid3-Jaccard, and Hybrid2-Jaccard overcome the F-score of Hybrid1-Jaccard and Random, when the number of provider peers is greater than or equals to four. Similarly, the F-score of Cosine and Hybrid3-Cosine overcome the F-score of Hybrid1-Cosine and Random when the number of provider peers is greater than or equals to five. The F-score of Hybrid2-Cosine overcomes the F-score of Random and Hybrid1-Cosine when the number of provider peers is greater than or equals to four and five.

Since, the number of provider peers does not affect its precision and recall, the random strategy always gives the same F-score, which is 0.607 on average, as the number of provider peers increases. For Hybrid1, the increment of recall as the number of provider peers increases does not have a sufficient effect to overcome the loss of precision, as a result, its F-score decreases closer to the F-score of the random strategy, as the number of provider peers increases.

When the recall of the common interest strategy and the hybrid strategy approaches 1.0, the increment of recall is not sufficiently larger than the loss of precision. As a result, we can notice the slight decline of their F-score, e.g., when the size of provider peer is greater than or equals to ten in Jaccard and Hybrid3-Jaccard.



(a)



(b)

Figure 7.3: The quality of document received in terms of F-score as the number of provider peers increases using different provider peer selection strategies; (a) an item-based profile representation; (b) a term-based profile representation

The results from the ANOVA testing for statistical significance indicate that there is a significant main effect of the selection strategy on the F-score ($F = 7224.9$, $df = 4$ and 5940, $p < 0.001$, partial $\eta^2 = 0.830$). The Scheffé post hoc testing shows that the common interest strategy and Hybird3 give the best F-score followed

by Hybrid2, Hybrid1, and Random. There is no significant difference between the common interest strategy and Hybrid3. In their best performance, when the number of provider peers equals nine and eight, Jaccard and Hybrid3-Jaccard give a 26.5% and 27.0% improvement in the F-score over Random, respectively. In addition, when the number of provider peers equals nine, Cosine and Hybrid3-Cosine give a 14.8% improvement in the F-score over Random.

The ANOVA and Scheffé post hoc testing also indicates that there is a statistically significant effect of the number of provider peers on the F-score ($F = 557.651$, $df = 5$ and 5940, $p < 0.001$, partial $\eta^2 = 0.319$). The system achieves the best F-score when the number of provider peers equals six to eight, and there is no significant difference between them.

The selection strategy is the best prediction of the F-score with partial η^2 equals 0.830, which accounted for more than 80% of the overall variance of F-score. The next best is the profile representation with partial η^2 equals 0.513 [1]. The number of provider peers is the third to affect F-score with partial η^2 equals 0.319, which accounted less than 50% of the overall variance of F-score. The complete ANOVA test is presented in Appendix B Table B.3 (page 164).

7.5.2 Discussion

The results show that peers in the common interest and hybrid strategy with β less than 10^{-2} can automatically form a self-organizing network of peers with common interest. By sharing and disseminating only messages that are of interest, peers automatically filter out messages that are not relevant to their group, creating a community filtering system.

In general, the hybrid strategy provides mixture effects of the common interest and the random strategies. When the exploration parameter β equals 0, the hybrid strategy behaves like the common interest strategy. As the β parameter increases, the hybrid strategy behaves more like the random strategy. In addition, when the β parameter equals 1, the hybrid strategy behaves like the random strategy.

Since, peers in the common interest strategy try to connect to provider peers with

[1] The effect of profile representation will be disscussed in the next chapter

high common interests, their provider peers help the local peers filter out non-relevant documents. As a result, the networks with common interest strategy gives the best precision.

The relatively low recall of the common interest strategy can be attributed to peers not receiving relevant document metadata from some peers that have low common interests. Another possible reason is the group of peers having common interests becoming disconnected from the network, either by a small number of TTL, a small number of provider peers, or a characteristic of the cluster that they form. Peers with multiple interests may have difficulty in keeping track of documents in a sparse interest group, because the common interest strategy favours peers with high common interest. A possible solution is for the user to use multiple peers, one for each interest. Thus, peers responsible for sparse interests will perform a random walk until they find relevant peers. When the number of provider peers increases, each peer connects with a larger varieties of provider peers, the recall increases while the precision decreases.

The networks with random strategy behaves like a standard gossip protocol—a probabilistic reliable dissemination in large-scale systems [43]. With the random strategy, peers receive document metadata of all documents available in the networks (recall of 1.0). However, since not all the documents are relevant to all peers, the average precision of the random strategy is lower than the other strategies. The random strategy is good for disseminating information to every peer in the network. In our experiments, the number of provider peers does not affect the performance of the random strategy, because every peer in the random strategy is already connected since the number of provider peers equals three, and Shrack messages can be disseminated to every peer with in the predefined TTL (eight).

Since, in our experiments, the system with common interest strategy and the hybrid strategy gain more precision than the loss of recall comparing with the random strategy, they have better performance in terms of F-score than the random strategy.

We anticipated that the hybrid strategy with very small β, i.e., $\beta = 10^{-3}$ or Hybrid3 in our experiment, would reduce the effect of the greedy behaviour in the common interest strategy, sufficiently increasing the recall and as a result F-score over the common interest strategy. Experimental results show, however, that for number of

provider peers greater than five, Hybrid3 does not provide a significant improvement in the F-score over the common interest strategy. This shows that increasing the provider peer diversity by increasing the number of provider peers results in better overall performance than by creating some random connections.

7.6 Dissemination Speed and Distance

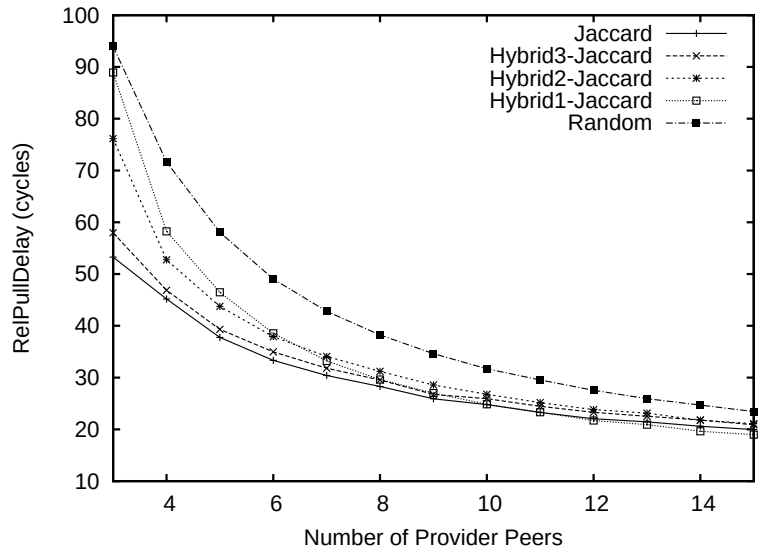
The dissemination speed and distance of relevant document metadata are measured in terms of relevant pull delay and relevant path length, averaged over all peers over the last 10 time slots.

7.6.1 Experiment Results

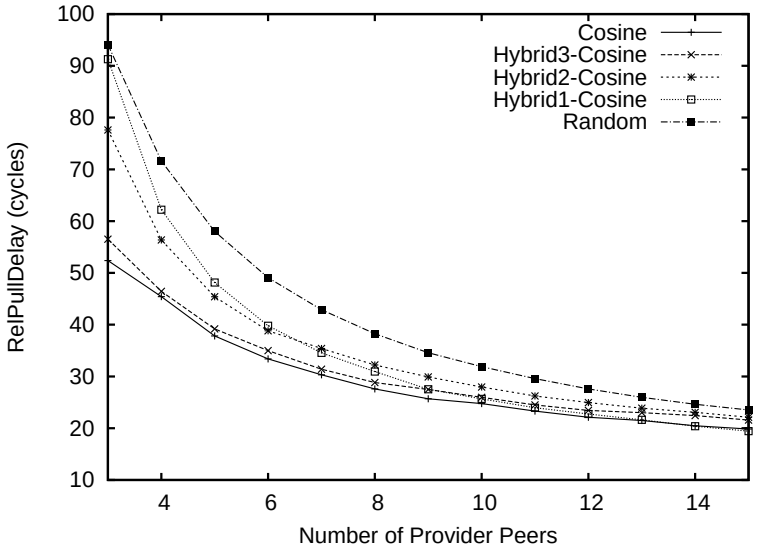
Relevant Pull Delay The relevant pull delay of the system as the number of provider peers increases using five provider peer selection strategies is presented in Figure 7.4. Figure 7.4 (a) shows the relevant pull delay of the system when the item-based profile representation is used. Figure 7.4 (b) shows the relevant pull delay of the system when the term-based profile representation is used.

The relevant pull delay indicates the average dissemination time delay from time when a document is published until relevant peers first observe its metadata, which is defined in Section 5.2.2 (page 58). The experimental results show that in both the item-based and the term-based profile representations, the random strategy always has the highest relevant pull delay and the common interest strategy always has the lowest relevant pull delay. The hybrid strategy has the relevant pull delay in-between the relevant pull delay of the random and the common interest strategy; i.e., the relevant pull delay increases as the exploration parameter β increases.

The system is considered to have better performance if it has lower relevant pull delay. The ANOVA testing for statistical significance shows that there is a significant main effect of the selection strategy on the relevant pull delay ($F = 6778.1$, $df = 4$ and 5940, $p < 0.001$, partial $\eta^2 = 0.820$). The Scheffé post hoc testing indicates that the common interest strategy gives the best relevant pull delay followed by Hybrid3, Hybrid1, Hybrid2, and Random.



(a)



(b)

Figure 7.4: The relevant pull delay as the number of provider peers increases using different provider peer selection strategies; (a) an item-based profile representation; (b) a term-based profile representation

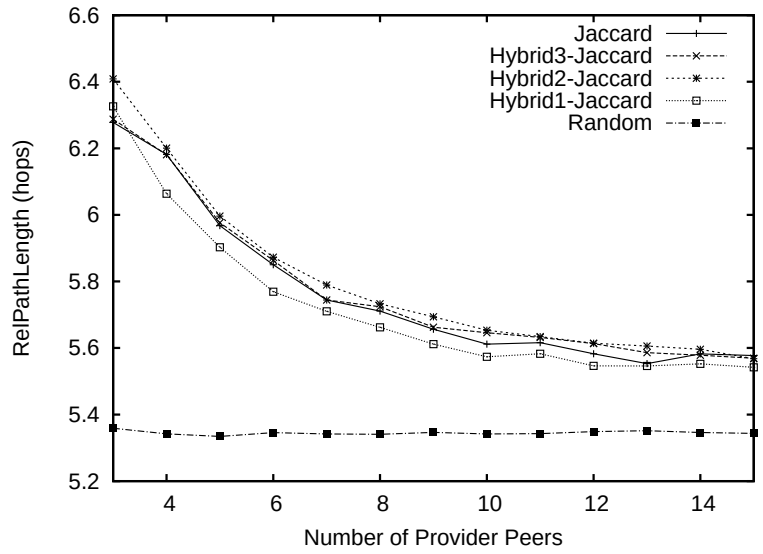
The ANOVA testing for statistical significance shows that there is a significant main effect of the number of provider peers on the relevant pull delay ($F = 45308.2$, $df = 5$ and 5940, $p < 0.001$, partial $\eta^2 = 0.974$). The Scheffé post hoc testing indicates that as the number of provider peers increases, the relevant pull delay of all provider peer selection strategies decreases. On average, the difference of the relevant pull delay of each strategy is also reduced, as the number of provider peers increases. When the number of provider peers is greater than or equals to six, the difference of the relevant pull delay among each strategy is less than one pull interval (20 cycles).

The partial η^2 indicates that the number of provider peers is the best predictor of the relevant pull delay with partial η^2 equals 0.974. The selection strategy is the next best predictor of the relevant pull delay with partial η^2 equals 0.820. The complete ANOVA test is presented in Appendix B Table B.4 (page 165).

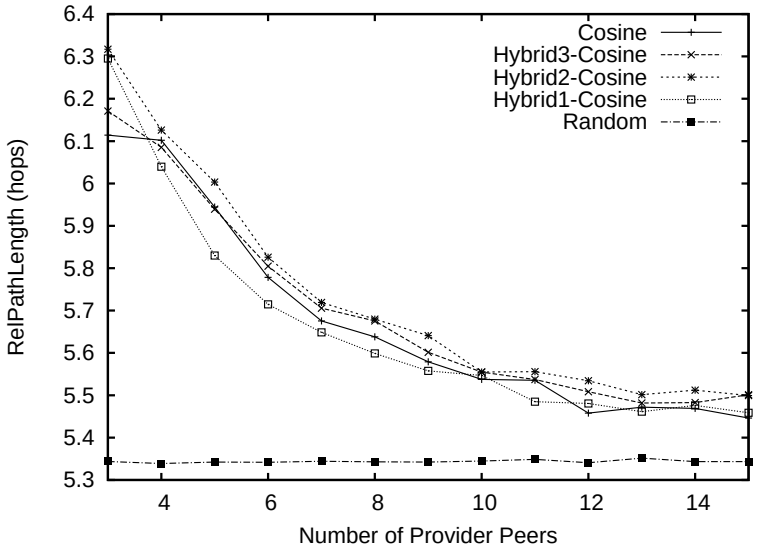
Relevant Path Length The relevant path length of the system as the number of provider peers increases using five provider peer selection strategies is presented in Figure 7.5. Figure 7.5 (a) shows the relevant path length of the system when the item-based profile representation is used. Figure 7.5 (b) shows the relevant path length of the system when the term-based profile representation is used. The relevant path length indicates the average distance of peers to the source of relevant document metadata, which is defined in Section 5.2.2 (page 58).

The experimental results show that in both the item-based and the term-based profile representations, the random strategy always has the lowest relevant path length. In addition, the number of provider peers does not affect the relevant path length of the random strategy. For the other strategies, their relevant path length exhibit a negative correlation with the number of provider peers.

The results from the ANOVA and Scheffé post hoc testing indicate that there is a statistically significant effect of the selection strategy ($F = 5947.3$, $df = 4$ and 5940, $p < 0.001$, partial $\eta^2 = 0.800$) and the number of provider peers ($F = 5829.1$, $df = 4$ and 5940, $p < 0.001$, partial $\eta^2 = 0.831$) on the relevant path length. The partial η^2 indicates that the number of provider peers is the best to predict the relevant path length with partial η^2 equals 0.831. However, the relevant path length is not



(a)



(b)

Figure 7.5: The relevant path length as the number of provider peers increases using different provider peer selection strategies; (a) an item-based profile representation; (b) a term-based profile representation

significantly different, when the number of provider peers is greater than or equals to 12. The selection strategy is the next best predictor of the relevant path length with partial η^2 equals 0.80. The complete ANOVA test is presented in Appendix B Table B.5 (page 166).

In general, the random strategy gives the best relevant path length, which is less than the hybrid and the common interest strategy. Overall, although they are statistically different, the difference between the average relevant path length of all strategies is very small, which is less than one hop.

7.6.2 Discussion

The speed and distance performance shows that, even though, the networks with common interest strategy have higher relevant path length, they have statistically lower relevant pull delay than the networks with random strategy. This behaviour shows that the networks with common interest strategy disseminate relevant messages faster than the random strategy, i.e., disseminating messages through multiple hops with less pull delay. This is because peers in the networks with common interest strategy usually pull and receive relevant messages from peers in the same group, while peers in the networks with random strategy get relevant messages from connecting to many different random peers according to their dynamic topology. Every time a peer updates its provider peers, a peer with the random strategy will connect to other peers randomly, while a peer with the common interest strategy tend to connect to the same group of provider peers. However, this property has less effect on the system as the number of provider peers increases. As the number of provider peers increases, each peer gets connected to more provider peers, and as a result, it receives more messages for each pull interval and messages are disseminated faster. Overall, the difference of the performance in terms of speed and distance of all provider peer selection strategies is very small. Especially, since we assume that the time that a peer learns about relevant documents is not critical in our application.

7.7 Self-Organizing Network Property

We measure the property of network in terms of the clustering coefficient, the characteristic path length, and the degree distribution, to determine whether they exhibit a social network property. We provide definitions of each measurement metric in Section 5.2.3 (page 59).

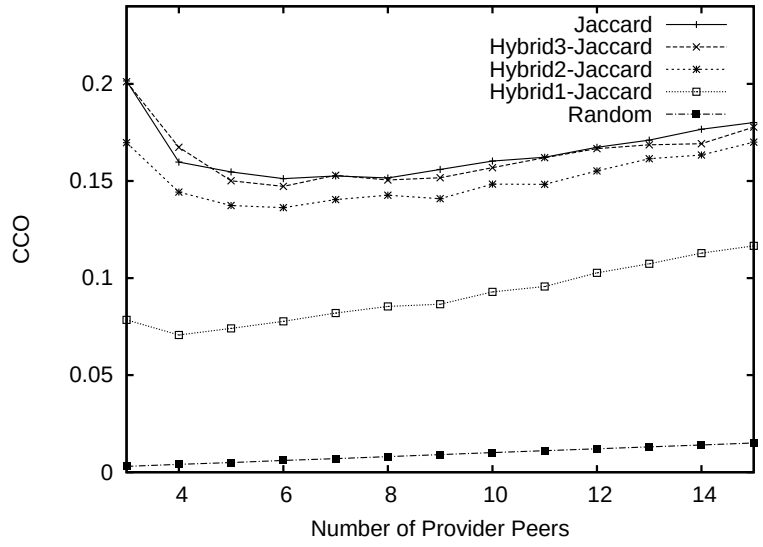
7.7.1 Experiment Results

Clustering Coefficient The clustering coefficient CCO of the networks in each provider peer selection strategy are shown in Figure 7.6. Figure 7.6 (a) presents the clustering coefficient of the networks when the item-based profile representation is used. Figure 7.6 (b) presents the clustering coefficient of the networks when the term-based profile representation is used.

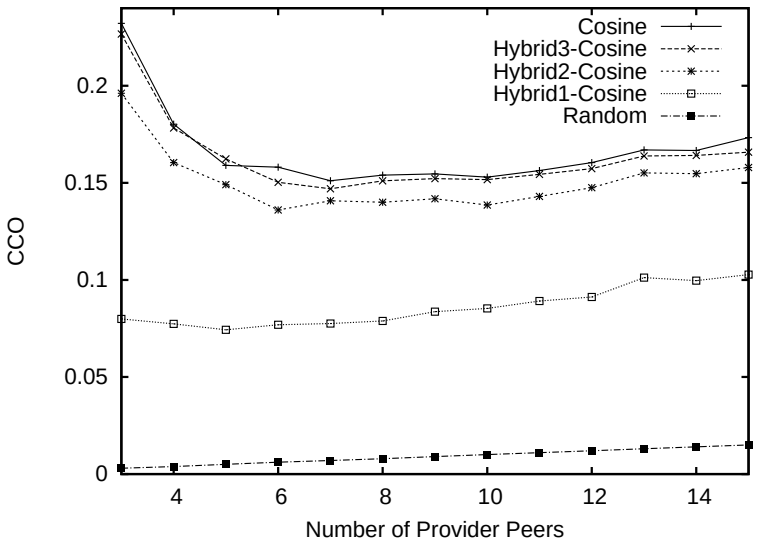
In general, the networks of the common interest strategy and the hybrid strategy with $\beta = 10^{-3}$ have similar clustering coefficients, which are higher than the clustering coefficients of the other strategies. As the exploration parameter β increases the clustering coefficient of the networks of the hybrid strategy decreases toward that of the random network. The random network has the lowest clustering coefficient.

The results from ANOVA indicate that there is a significant main effect of peer selection strategy on the clustering coefficient ($F = 51822.7$, $df = 4$ and 5940, $p < 0.001$, partial $\eta^2 = 0.972$). The results from the Scheffé post hoc testing indicates that the networks of the common interest strategy gives the highest clustering coefficient followed by the networks of Hybrid3, Hybrid2, Hybrid1, and the random strategy.

Initially when the number of provider peers increases, the clustering coefficients of the common interest strategy networks diminishingly drops and, after that, they slightly increase as the number of provider peers increases. For Jaccard, its network clustering coefficient diminishingly drops when the number of provider peers increases from three to six, and then start to increase slightly when the number of provider peers is greater than eight. For Cosine, its network clustering coefficient diminishingly drops when the number of provider peers increases from three to seven, and start to



(a)

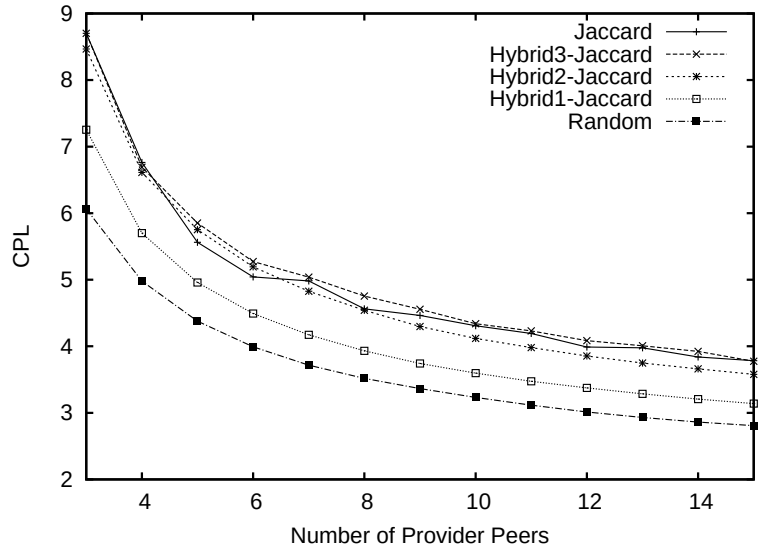


(b)

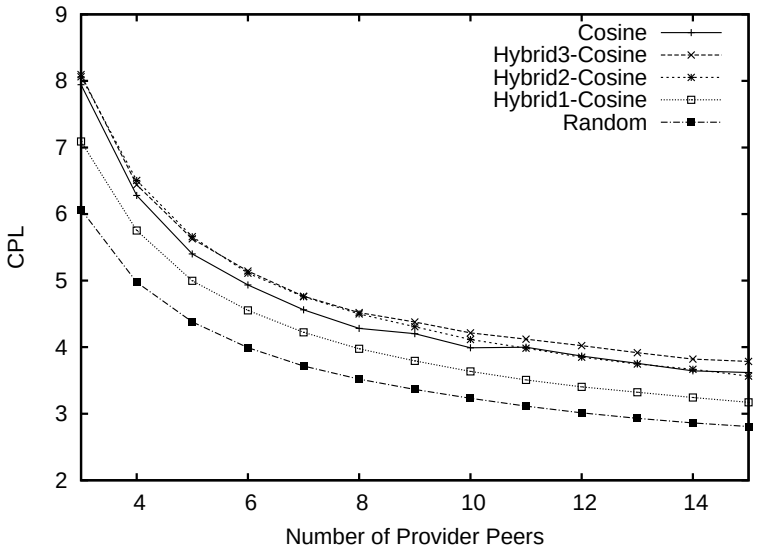
Figure 7.6: The network clustering coefficient as the number of provider peers increases using different provider peer selection strategies; (a) an item-based profile representation; (b) a term-based profile representation

increase slightly when the number of provider peers is larger than ten. The clustering coefficient of the random network constantly slightly increases as the number of provider peers increases. The hybrid gives the mixture effects between the common interest and the random strategy. The results from the ANOVA and Scheffé post hoc

testing indicate that there is a statistically significant effect of the number of provider peers on the clustering coefficient ($F = 491.2$, $df = 5$ and 5940, $p < 0.001$, partial $\eta^2 = 0.293$).



(a)



(b)

Figure 7.7: The network characteristic path length as the number of provider peers increases using different provider peer selection strategies; (a) an item-based profile representation; (b) a term-based profile representation

The partial η^2 indicates that the peer selection strategy is the best to predict the

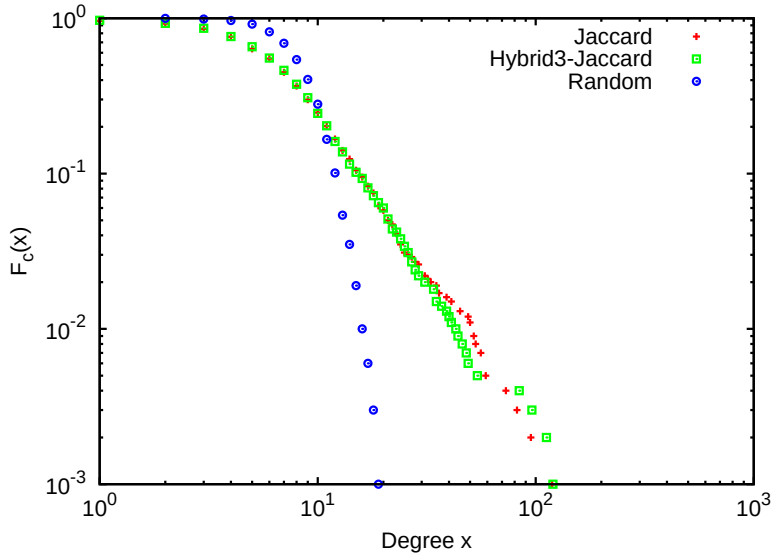
clustering coefficient with partial η^2 equals 0.972. The number of provider peers is the second best to predict the clustering coefficient with partail η^2 equals 0.293. However, less than 30% of the overall variance of the clustering coefficient is accounted for by the variance between groups formed by the number of provider peers. The complete ANOVA test is presented in Appendix B Table B.6 (page 167).

Characteristic Path Length The characteristic path length CPL of the networks in each provider peer selection strategy are shown in Figure 7.7. Figure 7.7 (a) presents the characteristic path length of the networks when the item-based profile representation is used. Figure 7.7 (b) presents the characteristic of the networks when the term-based profile representation is used.

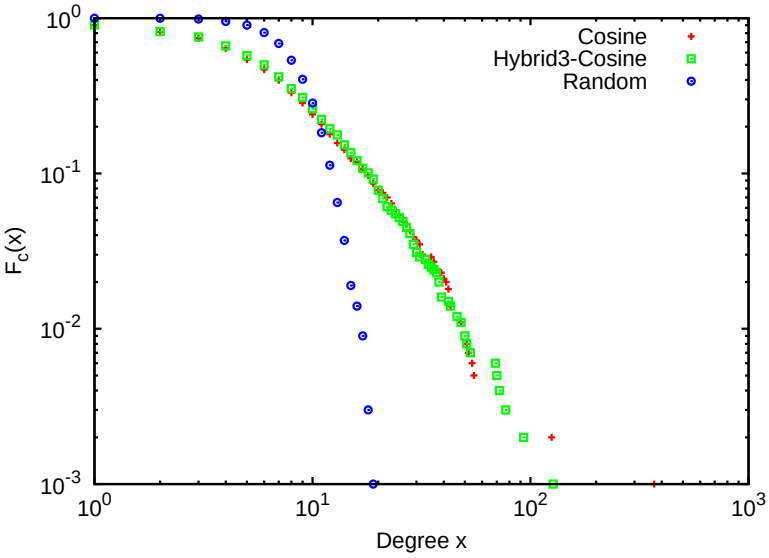
Although, the results show that all networks exhibit similar characteristic path length, the ANOVA test indicates that there is a statistically significant effect of peer selection strategy on the characteristic path length ($F = 5525.0$, $df = 4$ and 5940, $p < 0.001$, partial $\eta^2 = 0.788$). The results from the Scheffé post hoc testing indicate that the random strategy network has the smallest characteristic path length compared with other networks. The networks of the common interest strategy and the hybrid strategy with β equals 10^{-3} and 10^{-2} have similar characteristic path length, which are higher than the characteristic path length of the hybrid strategy with $\beta = 10^{-1}$ and the random strategy. There is no significant difference between the common interest strategy and the hybrid strategy with β equals 10^{-2}.

The results form the ANOVA and Scheffé testing also indicate that the effect of the number of provider peers is a statistically significant on the characteristic path length ($F = 17483.3$, $df = 4$ and 5940, $p < 0.001$, partial $\eta^2 = 0.936$). All networks show a negative correlation with the number of provider peers; i.e., the characteristic path length of all networks diminishingly decreases as the number of provider peers increases.

The partial η^2 indicates that the number of provider peers is the best predictor for the characteristic path length with partial η^2 equals 0.936. The next best predictor is the peer selection strategy with partial η^2 equals 0.788. The complete ANOVA test is presented in Appendix B Table B.7 (page 168).



(a)



(b)

Figure 7.8: Example of the in-degree complementary cumulative distribution function of the networks under different provider peer selection strategies; (a) an item-based profile representation; (b) a term-based profile representation

Degree Distribution Example of the in-degree complementary cumulative distribution function (CCDF) of the networks under different provider peer selection strategies on a log-log scale is presented in Figure 7.8. Figure 7.8 (a) presents the in-degree complementary cumulative distribution function of the networks when the

item-based profile representation is used. Figure 7.8 (b) presents the in-degree complementary cumulative distribution function of the networks when the term-based profile representation is used. We select, for each provider peer selection strategy, an instance of the network that is evolved from the same initial topology at the observation time slot τ_{29}. Every network has the same predefined number of provider peers, which is eight.

We observe that the in-degree distribution of the networks using the common interest strategy and the hybrid with $\beta = 10^{-3}$ follows the power law distribution fairly closely. When in-degree is greater than 3, the in-degree distribution of their networks fits a power law function with $\alpha = 2.1$. Conversely, the network with the random strategy does not follow the power law distribution as closely; even for in-degree values greater than 9, its in-degree distribution fits a power law function with $\alpha = 9.6$ which is outside the typical range of $2 > \alpha > 3$ for social networks.

7.7.2 Discussion

The analysis of the network properties gives evidence that the networks with the common interest strategy and the hybrid strategy have statistically different characteristics than the networks with the random strategy. The experimental results show that the network with common interest strategy and the hybrid strategy with β equals 10^{-3} (Hybrid3) are self-organizing into a topology that follows the characteristics of social networks, namely the small-world property [11, 56, 75]. Particularly, with the same configuration, the common interest strategy and Hybrid3 networks have significantly larger clustering coefficient than the random strategy network, have small characteristic path length similar to the random strategy network, and have power-law scaling in degree distribution. In addition, the difference of their characteristic path lengths also agrees with the small-world property [75] that, the small world network (our common interest strategy) has higher characteristic path length than the random network with the same configuration.

The clustering coefficient also indicates that when the number of provider peers are very small (less than six), the networks with the common interest strategy have very high clustering path length, which decreases as the number of provider peers

increase. We conjecture that when the number of provider peers equals three, there are multiple small disconnected clusters of peers in the network. On average, peers have high clustering coefficient, as the probability of provider peers of each peer get connected is higher, when the size of cluster is smaller. As the number of provider peers increase, the size of cluster is larger creating more variety of peers in each cluster, the probability of provider peers of each peer get connected decreases. As a result, the clustering coefficient decreases. When all the peers get connected the size of cluster is the same, but as the number of provider peers increases, the probability of provider peers of each peer to be interconnected increase. As a result, the clustering coefficient increases. A similar behaviour is shown in the network with random strategy. Since, peers in the random strategy are all connected when the number of provider peers equals three, the network clustering coefficient increases as the number of provider peers increases.

7.8 Summary

This chapter presents experimental evaluation of self-organizing Shrack network based on the provider peer selection strategies with item-based and term-based profile representation. We assume that each peer has a set of predefined interests and measures the performance of Shrack network in terms of the quality of received document metadata. We also evaluate dissemination speed and distance of relevant document metadata. Furthermore, we observes the property of self-organizing networks and compare them with the property of social networks. The results in both set of experiments support the experiment hypotheses that:

- **H7.1:** Self-organizing Shrack networks based on common interest of document set can enhance quality of documents received by peers in terms of F-score over random connected networks.
- **H7.2:** The resulting self-organizing networks based on common interest of document set have the characteristics of social networks.

The experimental results show that for both item-based and term-based profile representation, the common-interest strategy statistically outperforms the random

strategy in terms of F-score. The hybrid strategy provides a mixture effect of the comon-interest and random strategy. When the exploration parameter β equals 10^{-3}, the hybrid strategy gives similar performance as the common-interest strategy, as β increases, the hybrid strategy behaves more like the random strategy.

The performance in terms of dissemination speed and distance of relevant document metadata shows that although they are statistically different, all strategy have very similar characteristic. They all have similar relevant pull delay and relevant path length. The analysis of the self-organizing network property shows that the network with the common-interest strategy and the hybrid strategy with a small β follows the same characteristic as the social networks, while the network with the random strategy does not follow as closely.

Chapter 8

Effect of Peer Profile Representation

In this chapter, we discuss the effects of peer profile representation on the self-organizing Shrack network. We describe an experiment setup and performance metrics in Section 8.1. After that, we provide experiment results and discussion of the system performance in regard of the effect of peer profile representation. We present the results and discussion of the performance in the aspet of the quality of received documents, the dissemination speed and distance of relevant documents, and the self-organizing network property in Section 8.2, Section 8.3, and Section 8.4, respectively, Finally, we provide a summary of this chapter in Section 8.5.

8.1 Experiment Setup and Performance Metrics

We discuss the effect of peer profile representation using the results from the same sets of experiments in the previous chapter (Chapter 7). In the previous chapter, we present and discuss the experiment results in regards to different provider peer selection strategies. In this chapter, we focus on the comparison of the item-based and term-based profile representations. We use the same notation as defined in Table 7.5 (page 92) to refer to each set of experiments. The item-based profile is denoted as Jaccard and the term-based profile is denoted as Cosine. We graphically depict the effect of peer profile representation in the common interest strategy and the hybrid strategy with $\beta = 10^{-2}$, for the clarity of the presentation. The hybrid strategy with $\beta = 10^{-3}$ gives similar results as the common interest strategy, and the hybrid strategy with $\beta = 10^{-1}$ provides results closer to the random strategy. Furthermore, the peer profile representation does not affect the performance of the random strategy, because it does not use knowledge embedded in peer profiles. We discuss the results in terms of the quality of received document metadata, the dissemination speed and distance of relevant document metadata, and the self-organizing network property.

8.2 Quality of Received Documents

The comparison between the item-based and term-based profile representations of the quality of received document metadata in terms of precision, recall, and F-score, is presented in Figures 8.1, 8.2, and 8.3, respectively. In general, for each provider peer selection strategy, both profile representations have a similar behaviour in all performance metrics, as the number of provider peers increases.

8.2.1 Experiment Results

Precision In terms of precision, as shown in Figure 8.1, for each provider peer selection strategy, the item-based profile representation always gives better performance than the term-based profile representation. For the hybrid strategy, as the exploration parameter β increases the difference of the performance in term of precision between the item-based and term-based profile representation decreases and is more sensitive to the increment of the number of provider peers. The results from the ANOVA test indicate that the effect of the profile representation is statistically significant on the precision ($F = 8327.8$, $df = 1$ and 5940, $p < 0.001$, partial $\eta^2 = 0.584$). The partial η^2 indicates that the profile representation accounted for more than 50% of the overall variance of precision. We can notice that as the number of provider peers increases, the difference of precision between Hybrid2-Jaccard and Hybrid2-Cosine decreases. On average, over all the different numbers of provider peers, Jaccard and Hybrid2-Jaccard give 15.7% and 9.3% improvement over Cosine and Hybrid2-Cosine, respectively.

Recall The performance of the item-based and term-based profile representation for each strategy in terms of recall is shown in Figure 8.2. Unlike precision, the profile representation does not have major effect on the recall, both of them give similar performance. Although, the results from the ANOVA test indicate that there is a statistically significant effect of the profile representation on recall ($F = 230.4$, $df = 1$ and 5490, $p < 0.001$, partial $\eta^2 = 0.037$), the partial $\eta^2 = 0.37$ indicates that the profile representation accounted for less than 5% of the overall variance of recall. This shows that the profile representation does not practically affect the recall. In

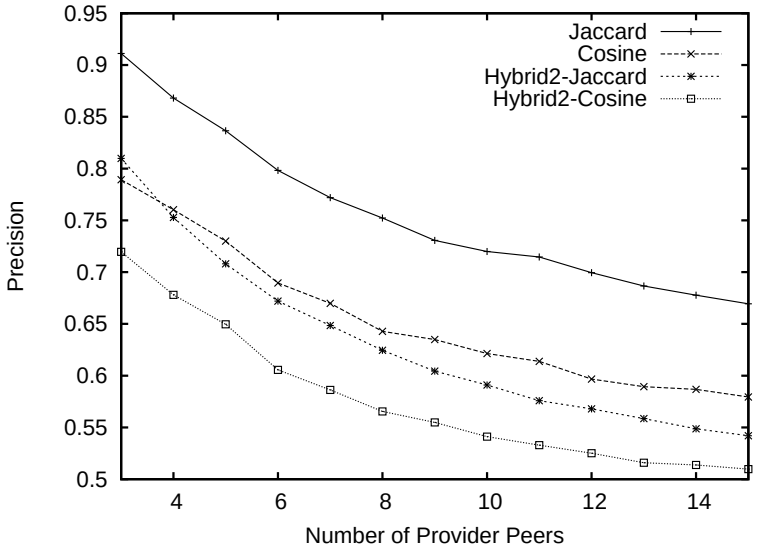


Figure 8.1: The quality of document received in terms of precision as the number of provider peers increases using item-based and term-base profile representations

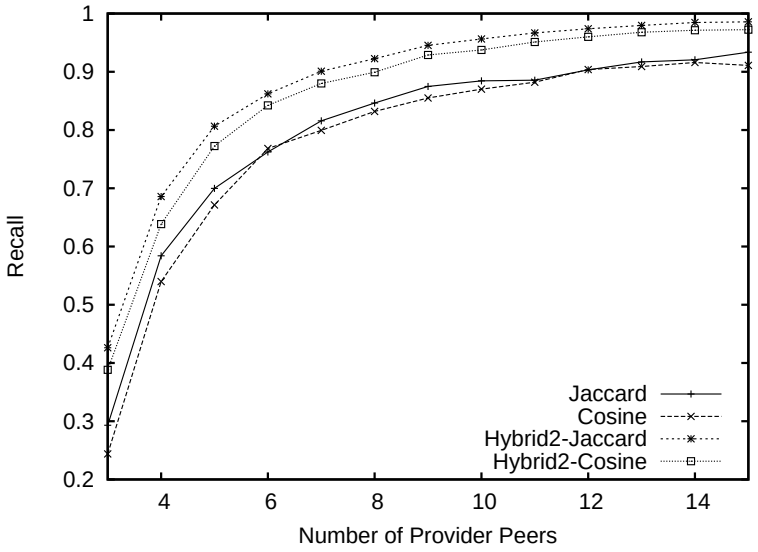


Figure 8.2: The quality of document received in terms of recall as the number of provider peers increases using item-based and term-base profile representations

general, the item-based profile representation gives slightly better performance than the term-based profile representation. On average, over all networks with the provider peers of size greater than four, Jaccard and Hybrid2-Jaccard gives 1.4% and 2.1% improvement in terms of recall over Cosine and Hybrid2-Cosine, respectively.

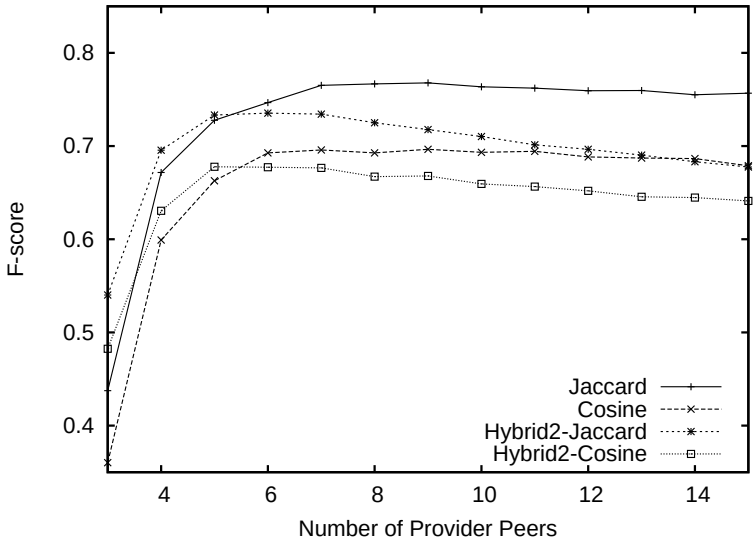


Figure 8.3: The quality of document received in terms of F-score as the number of provider peers increases using item-based and term-based profile representations

F-score Since the item-based profile representation gives a significantly better precision than the term-based profile representation, and both of them provide similar results on recall, for each provider peer selection strategy, the item-based profile representation gives a better F-score. Figure 8.3 shows the performance comparison between the two profile representations in term of F-score. For the hybrid strategy, the improvement of the item-based profile representation on the F-score over the term-based profile representation decreases, as the β parameter increases. The results from the ANOVA test indicate that the effect of the profile representation is statistically significant on the F-score ($F = 6247.7$, $df = 1$ and 5490, $p < 0.001$, partial $\eta^2 = 0.513$). The partial $\eta^2 = 0.513$ indicates that the profile presentation is a good predictor of the F-score, which accounted for more than 50% of the overall variance of the F-score. On average, over all number of provider peers, Jaccard and Hybrid2-Jaccard gives 11.1% and 8.0% improvement in term of F-score over Cosine and Hybrid2-Cosine.

8.2.2 Discussion

The experimental results show that the item-based profile representation gives a better model of Shrack peers than the term-based profile representation. Recall that our peer users are modelled after the authorship of documents in ACM dataset and the interests of the peer users are modelled after the ACM Computing Classification System (CCS). Our experiment results indicate that the item-based profile representation gives a better model to represent the common interest of authors who publish documents in ACM dataset given that the authors are interested in receiving all documents published in the same class. However, in practice, each user might not be interested in all the documents published in the same class. Experiments with real users will provide more evidence on the performance on peer profile representation. Our experimental results provide evidence that our provider peer selection strategy gives very consistent results for both item-based and term-based profile representations.

The item-based profile representation gives a direct mapping of peer common interest according to the relevant document set. On the other hand, the term-based profile representation provides an abstraction of common interest according to the terms in the relevant documents. Both of them show that Shrack peers can incrementally and locally create interest of peers in the networks from information contained in Shrack messages.

In the item-based profile representation, peers directly connect to provider peers that disseminate document metadata of documents in the overlap set of interest, as a result, its gives higher precision than the term-based profile representation. On the other hand, in the term-based profile representation, peers try to connect to provider peers that disseminate messages on the same topic of interests, as a result, it gives more variety of documents on the same topic and gives higher recall than the item-based profile representation. However, since the item-based profile representation gains more performance in terms of precision than the loss of recall, it gives a better performance in terms of F-score.

8.3 Dissemination Speed and Distance

Figures 8.4 and 8.5 show the comparison between the performance of the item-based and term-based profile representations for the common interest strategy and the hybrid strategy with $\beta = 10^{-2}$, in terms of relevant pull delay and the relevant path length.

8.3.1 Experiment Results

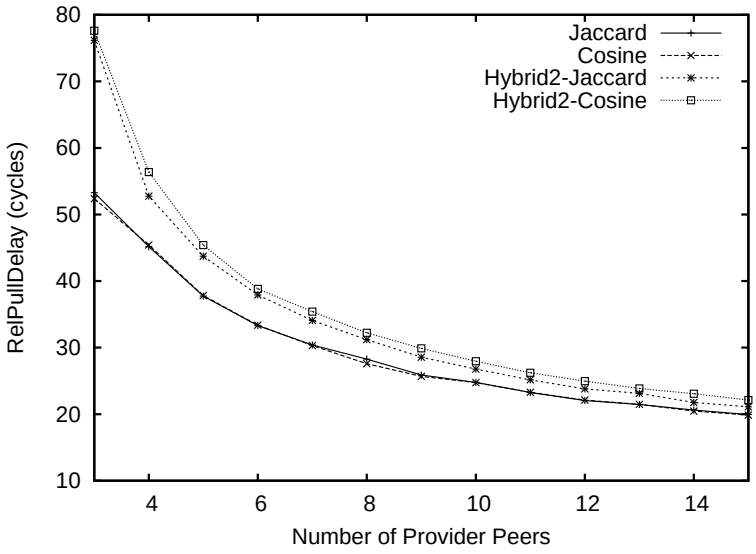


Figure 8.4: The relevant pull delay as the number of provider peers increases using item-based and term-base profile representations

Relevant Pull Delay The results show that, overall, both profile representation give a similar relevant pull delay. However, for the hybrid strategy, as the β parameter increases, we notice that the item-based profile representation gives lower relevant pull delay than the term-based profile representation, such that, the relevant pull delay of Hybrid2-Jaccard is smaller than Hybrid2-Cosine. The results from the ANOVA test show that the effect of the profile representation is statistically significant on the relevant pull delay ($F = 149.8$, $df = 1$ and 5940, $p < 0.001$, partial $\eta^2 = 0.025$). However, the partial η^2 indicates that the profile representation only accounted for

2.5% of the overall variance of the relevant pull delay. This shows that the profile representation does not practically affect the relevant pull delay.

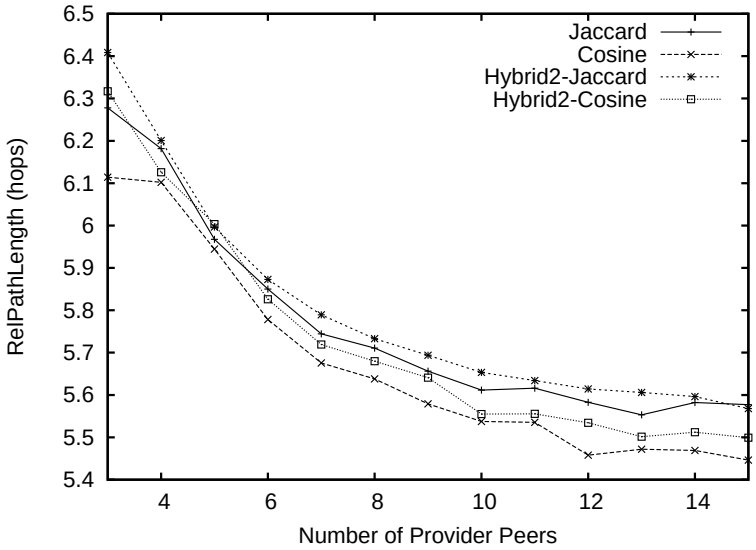


Figure 8.5: The relevant path length as the number of provider peers increases using item-based and term-base profile representations

Relevant Path Length The relevant path lengths of both profile representations are very similar, as shown in Figure 8.5. The results for the ANOVA test indicate that the effect of profile representation is statistically significant on the relevant path length ($F = 950.5$, $df = 1$ and 5940, $p < 0.001$, partial $\eta^2 = 0.138$). In general, the item-based profile representations have larger relevant path length than the term-based profile representations. Furthermore, the item-based profile representations with both provider peer selections have larger relevant path length than the relevant path length of the term-based profile representation, i.e., both Jaccard and Hybrid2-Jaccard have larger relevant path length than Cosine and Hybrid2-Cosine. The partial η^2 indicates that the profile representation does not practically affect the relevant path length compared with the number of provider peers and the peer selection strategy. The profile representation only accounted for 13.8% of the overall variance of the relevant path length, while the number of provider peers and the peer selection strategy accounted for more than 80% of the overall variance of the relevant path length.

8.3.2 Discussion

The experimental results show that both profile representations provide the same characteristics in the aspects of dissemination speed and distance. The performance of the system is more sensitive to the type of provider peer selection strategy than the type of peer profile representation. However, we can notice that both profile representation have similar relevant pull delay, but, the item-based profile representation gives a slightly better relevant path length than the term-based profile representation, which means that the item-based profile representation forms a better group of peers according to their relevant document set.

8.4 Self-Organizing Network Property

The comparison of self-organizing network property of networks evolving from the same provider peer selection strategy, using item-based and term-based profile representation is shown in Figures 8.6, 8.7, and 8.8.

8.4.1 Experiment Results

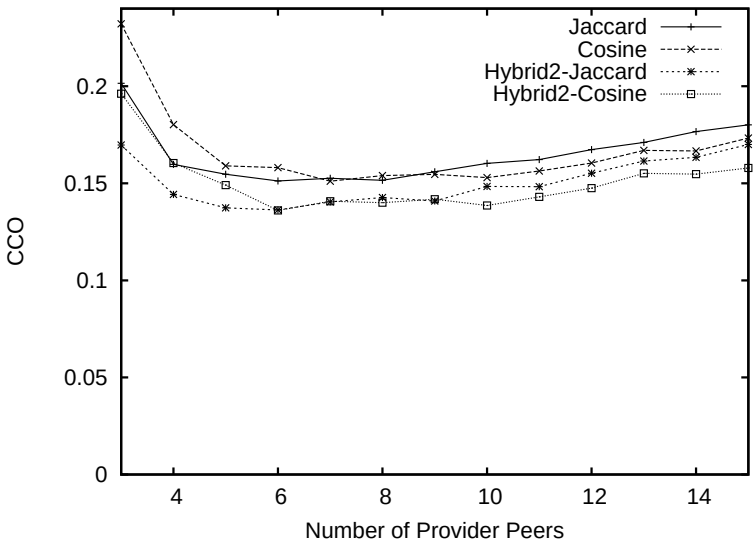


Figure 8.6: The network clustering coefficient as the number of provider peers increases using item-based and term-base profile representations

Clustering Coefficient Figure 8.6 shows that Jaccard and Cosine have similar network clustering coefficient and, similarly, Hybrid2-Jaccard and Hybrid2-Cosine have similar network clustering coefficient. However, when we look closely, when the number of provider peers is less than seven, the networks with item-based profile representation have slightly higher clustering coefficient. As the number of provider peers increases, the difference of the network clustering coefficient between the two types of profile representation decreases. In addition, when the number of provider peers is greater than nine, the networks with term-based profile representation turn to have slightly higher clustering coefficient than the networks with item-based profile representation.

The results from the ANOVA test indicate that there is a statistically significant effect of profile representation on the clustering coefficeint ($F = 35.0714$, $df = 1$ and 5940, $p < 0.001$, partial $\eta^2 = 0.006$). However, the partial η^2 indicates that the type of profile representation does not practically affect the clustering coefficeint. Less than 1% of the overall variance of the clustering coefficient is accounted for by the variance between groups formed by the type of the profile representation.

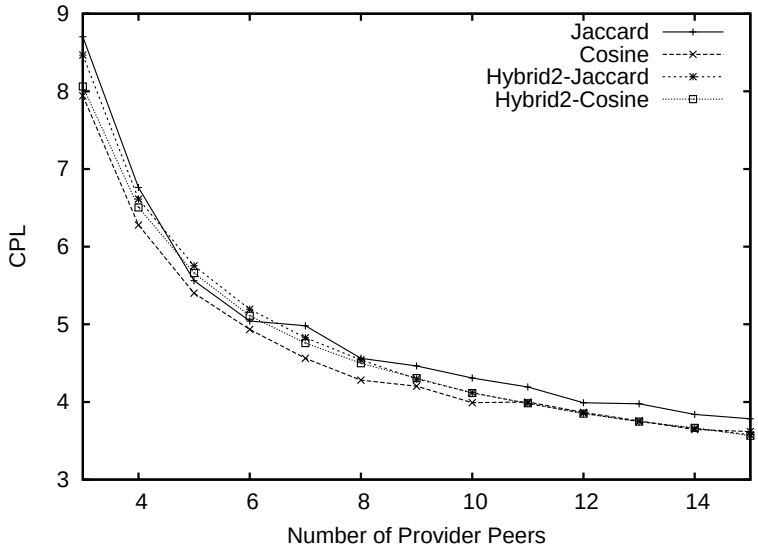


Figure 8.7: The network characteristic path length as the number of provider peers increases using item-based and term-base profile representations

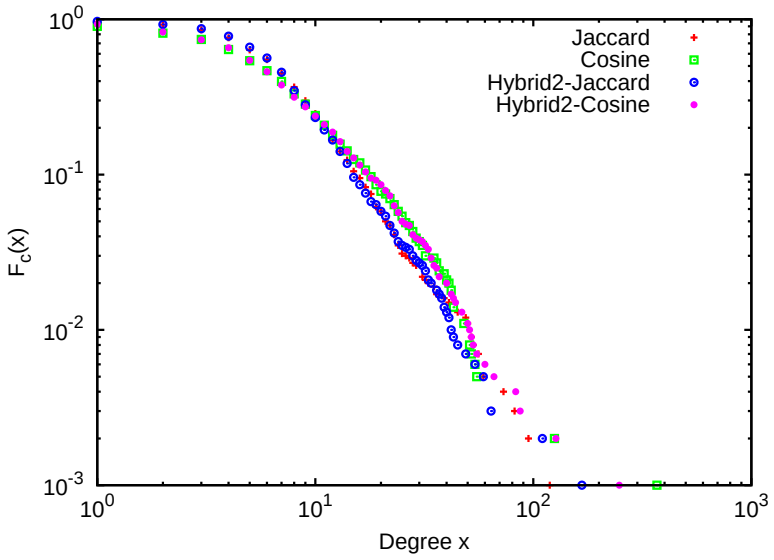


Figure 8.8: Example of the in-degree complementary cumulative distribution function of the networks with item-based and term-based peer profile representation

Characteristic Path Length The comparison of the characteristic path length of the networks is presented in Figure 8.7. The experimental results shows that all of the networks have very similar characteristic path length. The results from the ANOVA test suggest that there is a statistically significant effect of profile representation on the characteristic path length ($F = 179.8$, $df = 1$ and 5940, $p < 0.001$, partial $\eta^2 = 0.029$). However, the partial η^2 indicates that the profile representation only accounted for less than 3% of the overall variance of the characteristic path length, which infers that the profile representation does not practically affect the characteristic path length.

Degree Distribution Figure 8.8 graphically confirms that the type of peer profile representation has less effect on the characteristic of the self-organizing network than the type of provider selection strategy. The in-degree distribution of networks with the same provider peer selection strategy fits nicely with the same power law distribution, regardless of the type of peer profile representation.

8.4.2 Discussion

The experimental results show that the self-organizing networks that evolve from the same provider peer selection strategy have the same network property regardless of the type of profile representation, in all measurement metrics. This confirms that the difference of the item-based and term-based profile networks is how peers define their common interests with other peers. However, the property of networks in terms of clustering coefficient shows that cluster of peers in the item-based profile representation is less sensitive to the increment of the group size. As the size of provider peer increases, the networks of peers with item-based profile representation have slightly higher clustering coefficient than the networks of peers with term-based profile representation. On the other hand, this characteristic supports the argument that clusters of peers in the term-based profile representation have more variety than peers in the item-based profile representation. As a result, the term-based profile representation gives better recall than the item-based profile representation.

8.5 Summary

In this chapter, we discuss the effect of peer profile representation on the self-organizing Shrack network. The experimental results show that both item-based and term-based profile representations are meaningful to represent interests of peers. The results for the ANOVA test indicate that the effect of profile representation is statistically significant on all measurement metrics. However, the partial η^2 indicates that the profile representation only practically affects the precision and F-score with partial η^2 greater then 0.80. This statistical results support our observation that, in general, for each provider peer selection strategy, both profile representations have similar behaviour in all performance metrics. However, the item-based profile representation gives a better model of Shrack peers than the term-based profile representation, yielding significantly better performance in terms of precision. Subsequently, the item-based profile representation provides higher F-score than the term-based profile representation.

Chapter 9

Effect of Time-To-Live (TTL)

This chapter discusses effects of the time-to-live (TTL) on the performance and property of the Shack networks, for each provider peer selection strategy. We explore the performance of the system with unlimited TTL, where there is no TTL to limit the dissemination of Shrack messages. We present an experiment setup and performance metrics in Section 9.1. We present the results and discussion of the performance in the aspects of the quality of received documents, the dissemination speed and distance of relevant documents, the dissemination cost, and the self-organizing network property in Section 9.2, Section 9.3, Section 9.4, and Section 9.5, respectively, Finally, we provide a summary of this chapter in Section 9.6.

9.1 Experimental Setup and Performance Metrics

We repeat the previous experiments with unlimited TTL by setting the initial TTL to 400. In the previous experiments, the initial TTL is set to eight. The summary of the experimental parameter setup is presented in Table 9.1.

The same notation as in the previous experiments is used to refer to the type of provider peer selection strategy and the profile representation for each experiment configuration. In addition, for the same parameter setup but using unlimited TTL, a suffix "-400", will be added. For example, "Jaccard-400" refers to a set of experiments using a common interest strategy with item-based profile representation and with unlimited TTL.

In addition to the quality of received documents, and the dissemination speed and distance of relevant documents, we also measure the dissemination cost. The dissemination cost is measured in terms of average pull load per pull interval to observe the cost introduced by the protocol with unlimitted TTL. We defined the pull load in Table 6.2, which is the number of messages transferred for each pull response.

Table 9.1: Experimental parameter setup to study the effect of Time-To-Live (TTL)

Property	Value
Peer Profile representation	Item-based Term-based
Provider peer selection	Common Interest Random Hybrid1, Hybrid2, Hybrid3
System Publishing rate	1 documents per 4 cycles (Poisson distribution)
Pull Interval	20 cycles (periodically)
TTL	**400**
Size of provider peers	3-15
Maximum Update	160 cycles
Number of random seeds	10

The pull load per pull interval will account for all of the messages transferred in each pull interval.

Recall that with limited TTL (TTL=8), the systems using the hybrid strategy with $\beta = 10^{-3}$ (Hybrid3) give very similar performance as the systems using the common interest strategy. In addition, there is no statistically difference between this two provider peer selection strategies. For clarity of the graphical presentation, we do not include Hybrid3 with limited TTL in the graphical results. Reader can use the result of Jaccard and Cosine as a reference for Hybrid3-Jaccard and Hybrid3-Cosine respectively. Furthermore, the TTL does not affects the performance of the system using the random strategy. As a result, we do not include the results of Random and Random-400 for further discussion. For, Hybrid2 and Hybrid1, the effects of the TTL on the system performance and network property are similar to Hybrid3 with smaller degrees as the exploration parameter β increases. We did an ANOVA statistical analysis to test for statistical significance of the effect of TTL on the common interest strategy and the hybrid strategy with $\beta = 10^{-3}$ (Hybrid3).

The results of the complete full factorial model of ANOVA test are presented in Appendix C.

Hence, the rest of this chapter presents a comparison of the systems using the common interest strategy with limited and unlimited TTL, and the hybrid strategy with $\beta = 10^{-3}$ with limited and unlimited TTL, as representative results. The notation of the system disscussed in this chapter is presented in Table 9.2. The complete experimental results of experiments on unlimited TTL are presented in Appendix A. Next, we discuss the results in terms of the quality of received document metadata, the dissemination speed and distance of relevant document metadata, the dissemination cost, and the self-organizing network property.

Table 9.2: Notation for the analysis on the effects of TTL

Notation	Selection strategy	Profile	TTL
Jaccard	Common interest	Item-based	Limited
Jaccard-400	Common interest	Item-based	Unlimited
Hybrid3-Jaccard	Hybrid with $\beta = 10^{-3}$	Item-based	Limited
Hybrid3-Jaccard-400	Hybrid with $\beta = 10^{-3}$	Item-based	Unlimited
Cosine	Common interest	Term-based	Limited
Cosine-400	Common interest	Term-based	Unlimited
Hybrid3-Cosine	Hybrid with $\beta = 10^{-3}$	Term-based	Limited
Hybrid3-Cosine-400	Hybrid with $\beta = 10^{-3}$	Term-based	Unlimited

9.2 Quality of Received Documents

We present experiment results and discuss the effect of TTL in regards of the quality of received document metadata in terms of precision, recall, and F-score.

9.2.1 Experiment Results

Precision We present the effect of TTL in term of precision in Figure 9.1. The results show that in both profile representations, the common-interest strategy and Hybrid3 with unlimited TTL give higher precision than the system with limited TTL.

The results from the ANOVA test indicate that the effect of TTL is statistically significance on the precision ($F = 8024.0$, $df = 1$ and 4752, $p < 0.001$, partial $\eta^2 = 0.628$). The partial η^2 of the effect of TTL on the precision is greater than 0.50, which indicates that the overall variance of the precision is moderately influenced by the TTL. The full analysis indicates that the type of profile representation is the main effect on the precision for the system using the common interest and Hybrid3 strategies, with the partial η^2 equals 0.830. On average, when the number of provider peers is greater than or equals eight, Jaccard-400 and Cosine-400 give 18.0% and 11.6% improvement over Jaccard and Cosine, in terms of precision, respectively. In addition, Hybrid3-Jaccard-400 and Hybrid3-Cosine-400 give 15.7% and 9.8% improvement over Hybrid3-Jaccard and Hybrid3-Cosine, respectively.

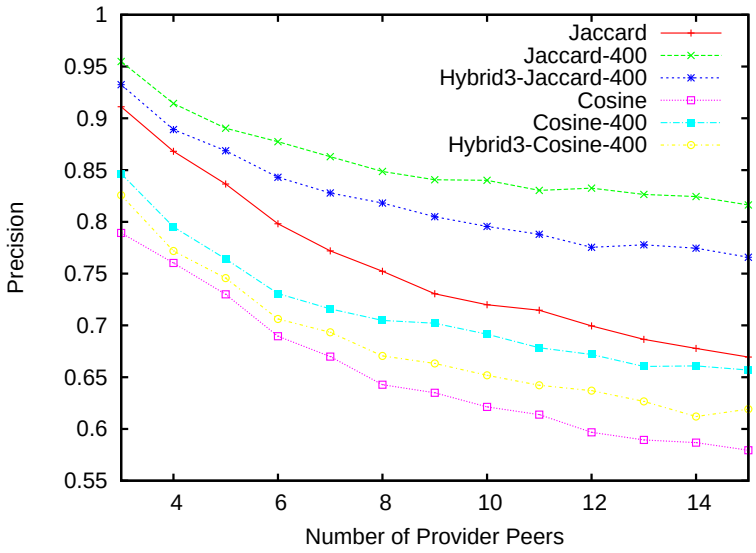


Figure 9.1: The comparison of the Shrack networks with limited and unlimited TTL in term of precision as the size of provider peer increases.

Recall In general as depicted in Figure 9.2, the systems with unlimited TTL provide very small improvement in terms of recall over the systems with limited TTL. Furthermore, for the common interest strategy with item-based profile representation, the results show reverse effects. When the number of provider peers is greater than six, Jaccard-400 gives lower recall than Jaccard. On average, when the number of

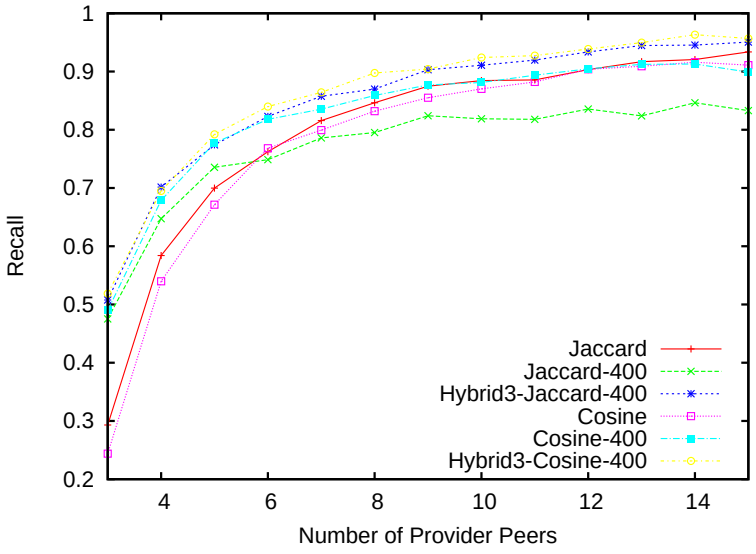


Figure 9.2: The comparison of the Shrack networks with limited and unlimited TTL in term of recall as the size of provider peer increases.

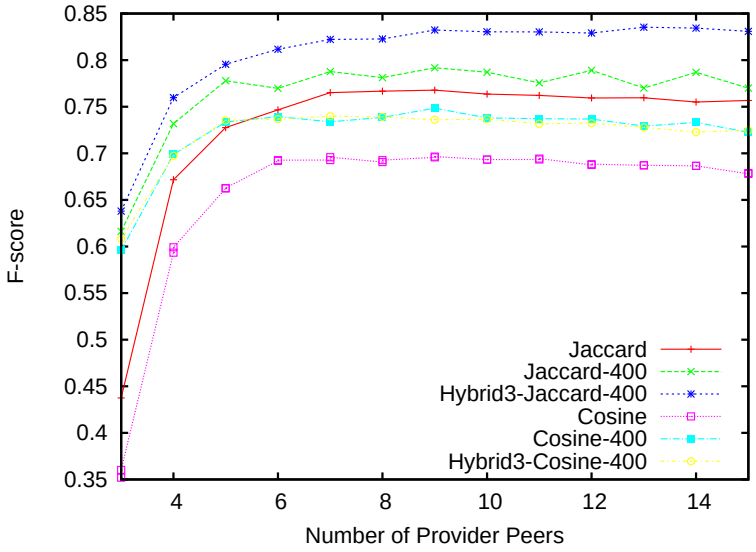


Figure 9.3: The comparison of the Shrack networks with limited and unlimited TTL in term of F-score as the size of provider peer increases.

provider peers is greater than or equals eight, Hybrid3-Jaccard-400, Cosine-400, and Hybrid3-Cosine-400 give 0.05%, 0.9% and 3.0% improvement over Hybrid3-Jaccard, Cosine, and Hybrid3-Cosine in terms of recall, respectively. On the other hand, Jaccard outperforms Jaccard-400 by 7.9%, on average, when the number of provider

peers is greater than or equals to eight.

The results from the ANOVA test indicates that the effect of TTL is statistically significance on the recall ($F = 377.4$, $df = 1$ and 4752, $p < 0.001$, partial $\eta^2 = 0.074$). However, the partial η^2 suggests that the TTL does not practically affect the recall. TTL only accounted for 7.4% of the overall variance of the recall. The best predictor for the recall for the system using the common-interest and Hybrid3 strategies is the number of provider peers, with partial $\eta^2 = 0.884$.

F-score The comparison of performance in term of F-score is depicted in Figure 9.3. The systems with unlimited TTL outperforms the systems with a limited TTL. When the number of provider peers is greater or equals eight, Jaccard-400 and Hybrid3-Jaccard-400 outperform Jaccard and Hybrid3-Jaccard by 2.6% and 9.2%, respectively. In addition, Cosine-400 and Hybrid3-Cosine-400 outperform Cosine and Hybrid3-Cosine by 6.6% and 6.9%, respectively.

The results from the ANOVA test indicates that the effect of TTL is statistically significance on the F-score ($F = 5175.1$, $df = 1$ and 4752, $p < 0.001$, partial $\eta^2 = 0.521$). The partial η^2 suggests that the TTL is the second best predictor that affects F-score with partial $\eta^2 = 0.521$. TTL accounted for more than 50% of the overall variance of the F-score. The best predictor for the F-score for the system using the common-interest and Hybrid3 strategies is the profile representation, with partial $\eta^2 = 0.647$.

9.2.2 Discussion

With unlimited TTL, the systems not only allow messages to reach more relevant peers, but also allow peers to select better provider peers that have common interest with the local peers. As a result, the systems with unlimited TTL have statistically higher precision than the system with limited TTL. However, by having provider peers that are too similar in common interests causes negative effects in terms of recall to the common interest strategy with item-based profile representation (Jaccard-400). When the number of provider peers is greater than five, Jaccard-400 gives less recall than Jaccard. This is because, with unlimited TTL, peers discover and have more

knowledge about other peers in the network, they can pick a better top-most common interest provider peers.

For item-based profile representations, the common interest profile is represented by a set of relevant documents. When the provider peers are too similar, they disseminate a highly overlapped set of documents to a local peer. As a result, a local peer receives less variety of documents, which affects the system in terms of recall. On the other hand, the term-based profile representation is not as sensitive to the provider peers with high common interest as the item-based profile representation, because a local peer semantically abstract the common interests from document contents. As a result, their provider peers disseminate documents that are conceptually relevant to the local peers, which provides more variety of documents than provider peers that disseminate the same documents in the item-based profile representation. The results show a small randomness can solve this problem. With unlimited TTL, the hybrid with β equals 10^{-3} with item-based profile representation (Hybrid3-Jaccard-400) does not have a problem with having too similar in common interests of provider peers. As a result, overall, Hybrid3-Jaccard-400 gives the best performance in terms of F-score.

9.3 Dissemination Speed and Distance

We present the effect of TTL in terms of the relevant pull delay and relevant path length.

9.3.1 Experiment Results

The comparison of the relevant pull delay and relevant path length is shown in Figures 9.4 and 9.5, respectively. The results show that the relevant pull delay and relevant path length of the systems with unlimited TTL are higher than the systems with limited TTL. However, in general, the difference is small; i.e., their relevant pull delay is less than one pull interval, on average, and their relevant path length is less than three hops, on average.

The effects of TTL is the highest when the number of provider peers equals three. On average, the systems with unlimited TTL have about 20 cycles larger relevant pull

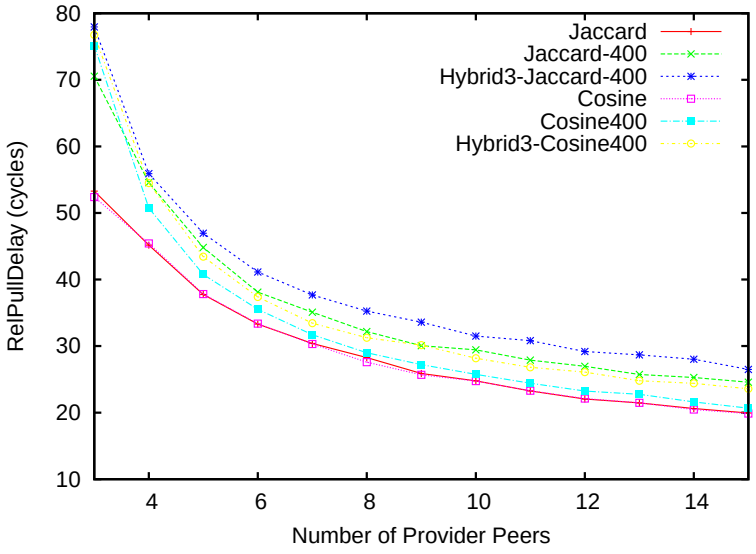


Figure 9.4: The comparison of the Shrack networks with limited and unlimited TTL in term of relevant pull delay as the size of provider peer increases.

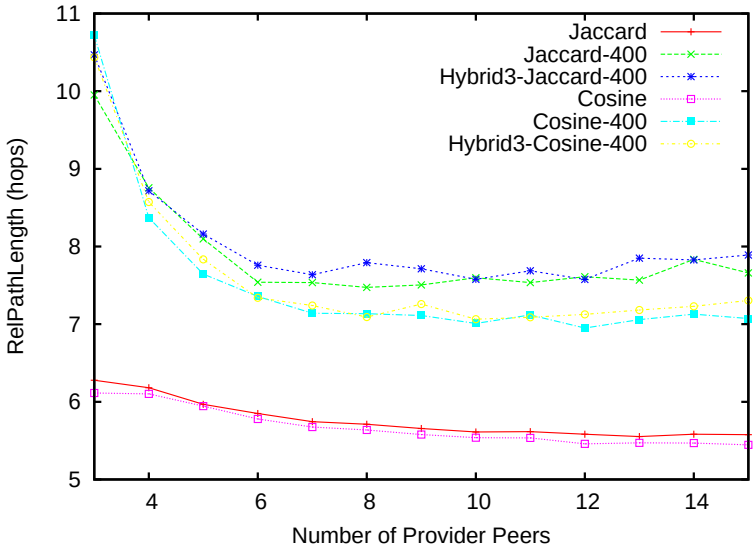


Figure 9.5: The comparison of the Shrack networks with limited and unlimited TTL in term of relevant path length as the size of provider peer increases.

delay and four hops longer relevant path length than the systems with the limited TTL. When the number of provider peers is larger than or equals eight, the systems with unlimited TTL have 3.5 cycles larger relevant pull delay and 1.8 hops longer relevant path length than the systems with the same experiment parameters, but

their TTL equals eight.

The results from the ANOVA test indicate that the effect of TTL is statistically significance on the relevant pull delay ($F = 5685.4$, $df = 1$ and 4752, $p < 0.001$, partial $\eta^2 = 0.545$) and the relevant path length ($F = 57161.4$, $df = 1$ and 4752, $p < 0.001$, partial $\eta^2 = 0.923$) of the systems using the common-interest and Hybrid3 provider peer selection strategies. The partial η^2 suggests that the TTL accounted for more than 50% and 90% of the overall variance of the relevant pull delay and the relevant path length, respectively. In addition, TTL is the best predictor of the relevant path length and it is the second best predictor of the relevant pull delay. The best predictor of the relevant pull delay is the number of provider peers (with partial $\eta^2 = 0.956$).

9.3.2 Discussion

Originally, TTL is introduced as part of a mechanism to ensure that the dissemination of Shrack messages will be terminated and to avoid their circulation in the network forever. The initial value of the TTL is a system-wide predefined value, which needed to be tuned for each community. As shown in the experiments, when the number of provider peers is small, the predefined TTL equals eight that is used in the previous experiments is actually too small for a message to disseminate to relevant peers for a system using a common interest or Hybrid3 strategy . On average, when the number of provider peers equals three, the relevant path length of the common interest strategy networks with unlimited TTL are greater than ten, which is higher than the predefined value of TTL.

In general, when the number of provider peers is greater than five, on average, the common interest strategy networks with unlimited TTL have the relevant path length closer to eight. However, the TTL does not affect the network greatly, because their characteristic path length is less than six. On average, there exists a path that is less than eight for a message to be disseminated to relevant peers. On the other hand, when the number of provider peers equals three, their characteristic path length is greater than or equal to eight. As a result, we can see a strange drop of the relevant path length of the common interest strategy networks when the number of provider

peers equals three, comparing with other type of networks. Since the systems with unlimited TTL have comparatively higher relevant path length than the limited TTL, they have higher relevant pull delay, but relatively small (less than 10 cycles or half of pull interval on average).

9.4 Dissemination Cost

The dissemination cost is measured in term of average pull load for each pull interval average over all peers, which from now will be refered as pull load.

9.4.1 Experiment Results

We present the comparison of dissemination cost in terms of pull load in Figure 9.6. We also include pull load of the Shrack networks with random selection strategy. We give labels of the random strategy with limited and unlimited TTL as Random and Random-400, respectively. The results show that the systems with unlimited TTL have higher pull load than the systems with limited TTL. However, the systems using the common-interest strategy and Hybrid3 with unlimited TTL have significantly less pull load than the random strategy with TTL equals eight. On average, when the number of provider peers is greater than or equals eight, Jaccard-400 and Cosine-400 have 14.2% and 20.8% higher pull load that Jaccard and Cosine, respectively. In addition, Hybrid3-Jaccard-400 and Hybrid3-Cosine-400 have 16.5% and 21.5% higher pull load than Hybrid3-Jaccard and Hybrid3-Cosine, respectively, on average, when the number of provider peers is greater than or equals eight.

The results from the ANOVA test indicate that the effect of TTL is statistically significant on the pull load ($F = 156.2$, $df = 1$ and 4752, $p < 0.001$, partial $\eta^2 = 0.032$) of the systems using the common-interest and Hybrid3 provider peer selection strategies. However, the partial η^2 suggests that the TTL does not practically affect the pull load; TTL only accounted for 3.2% of the overall variance of pull load. The best predictor of the pull load is the number of provider peers (with partial $\eta^2 = 0.418$).

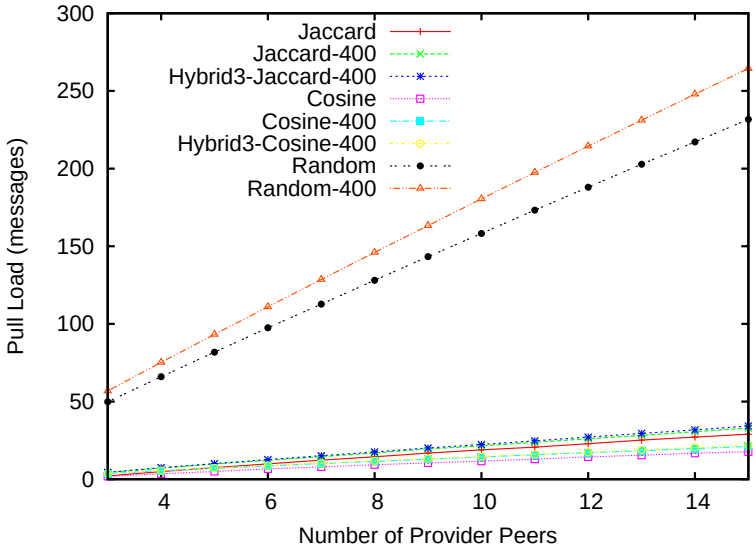


Figure 9.6: The comparison of the Shrack networks with limited and unlimited TTL in term of dissemination load per pull interval as the size of provider peer increases.

9.4.2 Discussion

The experimental results show that the systems with unlimited TTL have higher dissemination cost than the systems with limited TTL. With unlimited TTL, the system allows Shrack messages to be disseminated to all connected peers with common interests without the control of number of hops after which the messages should be discarded. As a result, there are more messages transferred among peers in the networks. Shrack peers in the system with unlimited TTL control the number of messages transfers by (1) discarding non-relevant messages to the local peers, (2) discarding duplicate messages they received, and (3) transferring messages only since the update time included in the pull request.

Peers in the random strategy do not get benefits from the provider peers to filter out (discard) non-relevant messages. Hence, they have more pull load from non-relevant messages than peers in the common-interest strategy. In addition, recall that each pull request includes an update time which is the time of the previous pull request that a local peer sent to the provider peer. The update time defines the oldest Shrack messages the provider peer should send back to the local peer. Since, the system using the common-interest strategy and Hybrid3 usually pull messages

from the same group of provider peers, their network topology are less dynamic than the network topology of peers in the random strategy. Hence, the average update interval in each pull request in the common-interest networks is less than peers in the update interval in the random networks, and as a result, peers in the common-interest strategy have less messages transferred among peers on average. Note that in our experiments, peers in the random strategy usually have the update time bounded by the predefined maximum update time. This could be fine tuned and further analysed for an optimal maximum update time for the random strategy for those who are interested in this strategy.

9.5 Self-Organizing Network Property

We present the effect of TTL in terms of clustering coefficient, characteristic path length, and degree distribution.

9.5.1 Experiment Results

In general, the networks with limited and unlimited TTL all have similar clustering coefficient, characteristic path length, and in-degree distribution. However, the systems with unlimited TTL do have slightly higher clustering coefficient and characteristic path length than the system with limited TTL, as shown in Figures 9.7 and 9.8, respectively.

Figure 9.7 shows that, for any number of provider peers, Hybrid3-Jaccard-400 and Jaccard-400 have larger clustering coefficient than Jaccard, and Hybrid3-Cosine-400 and Cosine-400 have larger clustering coefficient than Cosine. In addition, Figure 9.8 shows that, Hybrid3-Jaccard-400 and Jaccard-400 have slightly larger characteristic path length than Jaccard, and Hybrid3-Cosine-400 and Cosine-400 also have slightly larger characteristic path length than Cosine, in any number of provider peers. Furthermore, the results shows that TTL does not affect the in-degree distribution of the networks. Figure 9.9 shows that Jaccard, Jaccard400, and Hybrid-Jaccard400 have the same in-degree distribution.

The results for the ANOVA test indicate that the effect of TTL is statistically

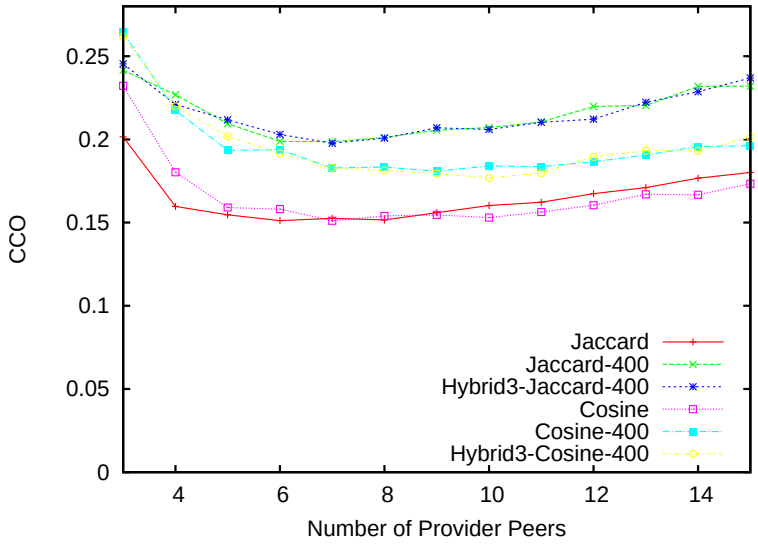


Figure 9.7: The comparison of the property of Shrack networks with limited and unlimited TTL according to the network clustering coefficient.

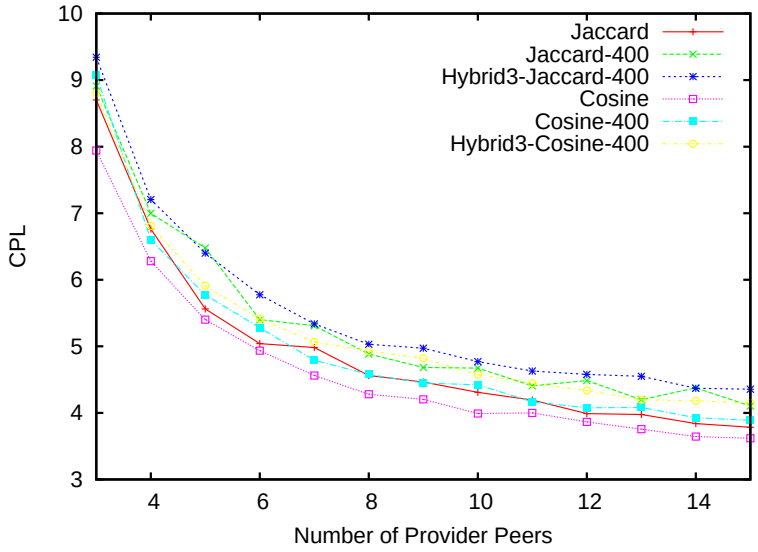


Figure 9.8: The comparison of the property of Shrack networks with limited and unlimited TTL according to the network characteristic path length.

significance on the clustering coefficient ($F = 10709.1$, $df = 1$ and 4752, $p < 0.001$, partial $\eta^2 = 0.693$) and the characteristic path length ($F = 1185.3$, $df = 1$ and 4752, $p < 0.001$, partial $\eta^2 = 0.200$). The partial η^2 suggests that the TTL is the main effect on the clustering coefficient, which accounted for 69.3% of the overall variance

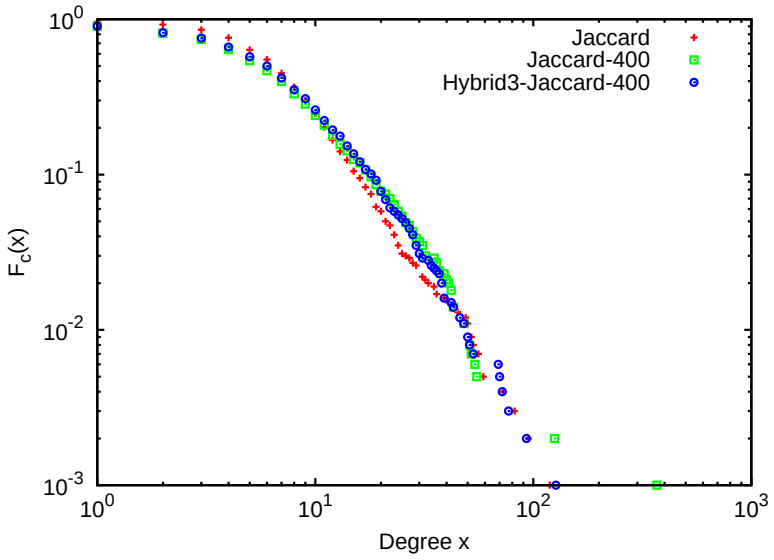


Figure 9.9: The comparison of the property of Shrack networks with limited and unlimited TTL according to the network in-degree distribution.

of the clustering path length. On the other hand, the TTL moderately affects the characteristic path length, which accounted for 20% of the overall variance of the characteristic path length. The best predictor for the characteristic path length is the number of provider peers (with partial $\eta^2 = 0.858$).

9.5.2 Discussion

Experimental results show that the TTL significantly affects the clustering coefficient of the self-organizing property of networks. For all types of provider peers and profile representations, the systems with unlimited TTL gives better performance than the systems with limited TTL. However, their self-organizing network properties are still the same. This indicates that the system does not require TTL to limit the number of hops a message should be disseminated. Peers can form a self-organizing filtering community and discard non-relevant messages from the system. In addition, the system also has other mechanisms to terminate messages including a history list and an update field in pull request. The history list provides a mechanism to discard messages that a local peer has previously seen and the update field can discard old messages from the system.

9.6 Summary

In this chapter, we discuss the effect of TTL on the self-organizing Shrack network. We compare the system between limited TTL and unlimited TTL. The experimental results show that with unlimited TTL, the systems allow messages to reach more relevant peers and allow peers to select better provider peers that have common interests with the local peers. However, having provider peers that are too similar may cause negative effects in terms of recall.

In general, the system with unlimited TTL significantly improves the performance in term of precision and recall, as a result, it gives higher F-score than the system with limited TTL. This shows that the Shrack peers can eliminate non-relevant messages and perform better without TTL. The statistic analysis also shows that the main effect on the dissemination cost is the number of provider peers not the TTL. In addition, the TTL does not significantly effect the property of self-organizing networks.

Chapter 10

Conclusion

We design Shrack with a motivation to reduce a tedious task of users that need to constantly search and keep track of new information from different sources. We expect that Shrack will broaden the information sources and narrow down the information views for users; that is, Shrack will increase visibility for users to keep track of information in which they are interested with minimum effort.

We summarize the contributions of this work in Section 10.1 and provide future work directions in Section 10.2.

10.1 Contributions

We propose a novel framework for a peer-to-peer collaborative environment that supports document sharing and tracking, namely Shrack. Shrack peers autonomously learn about the interests of their users and form a collaborative network with other peers. The peers keep track of new documents that are published in the system and are potentially of interest to their users. Shrack is different from existing document tracking tools such as RSS, Newsgroups or Mailing lists in that (1) Shrack peers find information that is of interest to the users automatically from different sources without an explicit subscription and (2) each user receives information based on his/her individual interests. As a result, each user in Shrack can receive information from a larger spectrum but unique to his/her interests.

Shrack extends the utility of existing peer-to-peer file sharing systems beyond the support for instant querying. We add functionality of the existing peer-to-peer document sharing systems to support document tracking based on a long-term interest of the user. Although in this thesis, we focus on document tracking, instant querying can be seamlessly incorporated in Shrack by issuing the query among peer groups.

In this thesis, we define the Shrack framework and present a Shrack prototype

system and simulation to validate the thesis' hypotheses. The hypotheses of this thesis are (1) a pull-only information dissemination is scalable in a large-scale P2P system and (2) self-organizing networks based on common interest of document sets can enhance relevance of documents received by peers in terms of F-score. To validate our hypotheses, we develop a pull-only information dissemination protocol and self-organizing network strategies in Shrack, build a simulation environment to study behaviour of Shrack, and define evaluation techniques and metrics to evaluate the performance of the dissemination protocol and the Shrack self-organizing networks.

The contributions of this thesis can be summarized in five topics as follows:

1. Shrack Framework;
2. Shrack Dissemination Protocol;
3. Self-Organizing Shrack Network;
4. Simulation Environment; and
5. Evaluation Methodologies.

A summary of each topic are presented next.

10.1.1 Shrack Framework

We define the Shrack architecture form a peer's perspective including peer components and main modules. We discuss characteristics of the Shrack network and define peer functionality. The framework can be used for any kind of data, however, we focus on sharing and tracking of research publications.

10.1.2 Shrack Dissemination Protocol

We develop the Shrack information dissemination protocol using pull-only communication. The dissemination protocol is tested in a scenario where all peers are interested in all document, and peers are connected in random network and small-world network configurations. The results support the first hypothesis that *a pull-only information dissemination is scalable in a large-scale P2P system.* In both network configurations,

the Shrack information dissemination protocol is scalable as the average pull delay of the system follows a logarithmic function of the network size. The Shrack dissemination protocol shows similar characteristics as the standard gossip-like protocol. Moreover, the experimental results show that super peers can improve the average dissemination speed of the system.

10.1.3 Self-Organizing Shrack Network

We develop peer-selection strategies based on common interests among peers. Peers are connected based on their observed common interests. There is no explicit profile exchange between peers and no global information available. As part of developing self-organizing networks, we develop (1) a method for a peer to discover other peers in the network, (2) a peer profile learning algorithm whereby a peer learns the interests of other peers in the network, and (3) a common-interest score between peers whereby item-based and term-based common-interest scores are explored. We describe a strategy for peers to discover the existence of other peers and learn about their interests locally, based on information carried in the document metadata that propagates through the network.

We compare our proposed common interest strategy with a randomly connected network. The results support the second hypothesis that *self-organizing network based on common interest of document sets can enhance relevance of documents received by peers in term of F-score.* Based on simulated environment using the ACM digital library metadata, the experimental results demonstrate that the proposed strategy gives noticeable improvement in the dissemination performance in terms of F-score up to 27% and 14.8% over a random network using an item-based Jaccard index and a term-based cosine similarity as a common interest score respectively.

Note that, with the common interest strategy, the system loses its reliability as each peer connects only to peers that have highly similar interests. Consequently, the peer may not receive all the relevant documents. These effects are shown in the dissemination performance in terms of recall, which is less than 1. However, the recall increases as the size of peer neighbourhood increases.

We also demonstrate that our peer selection strategies create self-organizing networks that follow the characteristics of social networks, namely the small-world property [11, 56, 75]. Particularly, our self-organizing networks—Jaccard, Cosine, Hybrid-Jaccard3 and Hybrid-Cosine3 networks—have significantly larger clustering co-efficient than random networks, small characteristic path length similar to random networks, and power-law scaling in degree distribution.

10.1.4 Simulation Environment

We develop an event-based simulation called ShrackSim. ShrackSim provides a simulation environment to test and study the behaviour of Shrack networks. ShrackSim is developed on top of PeerSim—a Java based peer-to-peer simulation. ShrackSim follows the structure of PeerSim such that ShrackSim is modelled based on components, and make it easy to quickly test and modify Shrack's protocols and modules. Users can set up simulation parameters through a configuration file, resulting in dynamic loading of components. ShrackSim provides several predefined objects to monitor the properties in which users are interested during the simulation, such as evaluation metrics and network properties. ShrackSim is extensible and can be modified through PeerSim components, which provide different pluggable building blocks.

10.1.5 Evaluation Methodology

We define and develop techniques to evaluate Shrack in simulated environment. Since it is difficult to find a large group of users to evaluate and understand the behaviour of Shrack users and the Shrack network, we introduce an artificial user model called *authorship user interest model.* The authorship user interest model is created from an existing document collection using authorship information to model simulated users based on documents they publish and documents of interest.

We devise a methodology to study and evaluate Shrack behaviour. We divide the simulation time into time slots and observe the evolution of the dissemination performance according to each time slot. The dissemination performance of a set of documents published during a given time slot is measured when the dissemination of these documents ends, which is determined by using a heuristic criterion.

Since information in a Shrack network is dynamic—sets of relevant documents for each peer change with time—standard information retrieval metrics, namely precision, recall, and F-score, can not be directly applied to evaluate the quality of documents that Shrack peers receive. We define the quality of documents that Shrack peers receive by modifying the standard information retrieval metrics such that precision, recall, and F-score are measured during the time duration when documents are published.

10.2 Future Work

There are many aspects that can be explored and incorporated in Shrack for future study and improvement.

Shrack Application The next step of this work is developing a Shrack application to be deployed and tested with real users. The application would allow the users to adjust their peer parameters such as profile representations, pull intervals, and number of provider peers. This flexibility would facilitate the creation of super peers, leading to an effective document distribution and tracking peer-to-peer network.

Dynamic Number of Provider Peers In the current model, the number of provider peers is a predefined system parameter. An interesting extension to this model would allow Shrack peers to dynamically adjust the number of provider peers according to documents received and availability of resources.

User Interface and Document Filtering A primary objective of a peer is to recommend to its user documents that the user finds of interest. Once the peer acquires candidate documents from the Shrack network, it would be beneficial to filter them before presenting them to the user. We are interested in two particular filtering techniques: content-based filtering and collaborative filtering. The content-based filtering approach computes the similarity between each document and the user's profile, and only recommends to the user documents that exceed a certain threshold. The collaborative filtering approach utilizes the information already stored in the

peer about other peers in the system and applies traditional collaborative filtering algorithms to recommend documents to its user.

An alternative to using a similarity threshold is to rank the documents based on their similarity and present the documents to the user in an ordered list. Measurement metrics such as mean average precision (MAP) [51] can be used to compare different similarity measures or ranking algorithms.

Security and Access Control The Shrack framework gives peers full control over what document metadata they share. Peers may implement different forms of security and access control as required. For instance, a publisher may authenticate its shared document metadata to protect the publisher's reputation. Access to the documents' full-text can be restricted to a group of peers. For instance, a peer hosted by a university laboratory may restrict full access to members of the laboratory or collaborating organizations. Such restrictions aim at protecting the intellectual property rights of the individuals or the organization. At a more personal level, Shrack can be tailored to the needs of users concerned with privacy and security of information exchanged, especially when Shrack is used to share sensitive information. For instance, secure communication channels, such as TLS, ssh, or HTTPS, can be used to for secure peer-to-peer communication. If authentication is required, standard authentication methods can be used, such as Kerberos.

Social Network Analysis Once Shrack is deployed in the real world, it would open the door for social network analysis. Common properties investigated in social network analysis including identifying hubs, experts, dynamically formed interest groups, and the social network connectivity. Further, we are interested in analyzing the impact of introducing super-peers in a self-organizing Shrack network.

Mathematical Analysis From a theoretical perspective, we would like to analyze the behaviour of a Shrack network using probabilistic modelling. Particularly, we are interested in modelling the behaviour of the network as a function of publication rate, network size, the number of provider peers, and the pull interval. Furthermore, we are interested in modelling the dynamic nature of peers in terms of joining and

leaving the network, publishing new metadata, and changes in the interests of the peer users.

Alternative Sharing Model This work assumes that users share documents that they author themselves. Following this assumption, a user model is created using the ACM Digital Library metadata collection whereby users are modelled after authors in the collection. An alternative user model is to assume that users are readers who share documents authored by others that they read and find interesting. The ACM Digital Library metadata collection can be used to model such users by exploiting the citation graph contained within the collection. We can still model users after authors in the collection; however, for each publication in the collection, the users share (publish in Shrack) the documents cited by this publication, rather than the publication itself.

Modelling Changes in User Interests One of the assumptions of the Shrack system is that users have persistent long-term interests. In practice, user interests may change over time. To account for gradual changes in a user's interests, we can revise the profile representations to utilize only a certain number of the most recent relevant documents. This approach would account for changes in the interests of the local user, as well as the interests of users associated with other peers. We are interested in exploring other approaches that can account for short-term spikes in user interests, as well as recurring interests.

Bibliography

[1] CoRR: Computing Research Repository. `http://arxiv.org/corr/home/`, Accessed May 2006. The Cornell University Library.

[2] Google Scholar. `http://scholar.google.com/`, Accessed May 2006. Google Inc.

[3] JXTA. `http://www.jxta.org/`, Accessed May 2006. CollabNet, Inc.

[4] Bibsonomy. `http://www.bibsonomy.org/`, Accessed April 2009.

[5] Citeulike. `http://www.citeulike.org`, Accessed April. 2009.

[6] ResearchGATE: scientific network. `http://www.citeulike.org`, Accessed June. 2009.

[7] The ACM Digital Library. `http://portal.acm.org/`, Accessed Jan. 2010.

[8] The IEEE Xplore Digital Library. `http://ieeexplore.ieee.org/`, Accessed Jan. 2010.

[9] Karl Aberer, Philippe Cudré-Mauroux, Manfred Hauswirth, and Tim Van Pelt. Gridvine: Building internet-scale semantic overlay networks. In *International Semantic Web Conference*, pages 107–121, 2004.

[10] Benjamin Ahlborn, Wolfgang Nejdl, and Wolf Siberski. OAI-P2P: A Peer-to-Peer Network for Open Archives. In *ICPP Workshops*, pages 462–468, 2002.

[11] Yong-Yeol Ahn, Seungyeop Han, Haewoon Kwak, Sue Moon, and Hawoong Jeong. Analysis of topological characteristics of huge online social networking services. In *WWW'07: Proceedings of the 16th international conference on World Wide Web*, pages 835–844, New York, NY, 2007. ACM.

[12] R. Akavipat, L.-S. Wu, F. Menczer, and A.G. Maguitman. Emerging semantic communities in peer web search. In *P2PIR '06: Proceedings of the international workshop on Information retrieval in peer-to-peer networks*, pages 1–8, New York, NY, USA, 2006. ACM.

[13] R. Akavipat, L.-S. Wu, F. Menczer, and A.G. Maguitman. Emerging semantic communities in peer web search. In *P2PIR '06: Proceedings of the international workshop on Information retrieval in peer-to-peer networks*, pages 1–8, New York, NY, 2006. ACM.

[14] American Mathematical Society. AMS Digital Mathematics Registry. `http://www.ams.org/dmr/`, Accessed Feb. 2010.

[15] Stephanos Androutsellis-Theotokis and Diomidis Spinellis. A survey of peer-to-peer content distribution technologies. *ACM Comput. Surv.*, 36(4):335–371, 2004.

[16] Wolf-Tilo Balke, Wolfgang Nejdl, Wolf Siberski, and Uwe Thaden. Progressive distributed top-k retrieval in peer-to-peer networks. In *ICDE '05: Proceedings of the 21st International Conference on Data Engineering*, pages 174–185, Washington, DC, USA, 2005. IEEE Computer Society.

[17] Burton H. Bloom. Space/time trade-offs in hash coding with allowable errors. *Communicaton of ACM*, 13(7):422–426, 1970.

[18] John S. Breese, David Heckerman, and Carl Kadie. Empirical analysis of predictive algorithms for collaborative filtering. In *Proceedings of the 14^{th} Conference on Uncertainty in Artificial Intelligence*, pages 43–52, 1998.

[19] Andrei Broder, Ravi Kumar, Farzin Maghoul1, Prabhakar Raghavan, Sridhar Rajagopalan, Raymie Stata, Andrew Tomkins, and Janet Wiener. Graph Structure in the Web: Experiments and Models. In *9th World Wide Web Conference*, May 2000.

[20] Andrei Broder, Michael Mitzenmacher, and Andrei Broder I Michael Mitzenmacher. Network applications of bloom filters: A survey. In *Internet Mathematics*, pages 636–646, 2002.

[21] M. Castro, M. Costa, and A. Rowstron. Peer-to-peer overlays: structured, unstructured, or both? Technical report, Microsoft Research, 2004.

[22] Yatin Chawathe, Sylvia Ratnasamy, Lee Breslau, Nick Lanham, and Scott Shenker. Making gnutella-like p2p systems scalable. In *SIGCOMM '03: Proceedings of the 2003 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 407–418, New York, NY, USA, 2003. ACM.

[23] CiteSeer.IST (Scientific Literature Digital Library). `http://citeseer.ist.psu.edu/`, Accessed May 2006. School of Information Sciences and Technology, Penn State.

[24] Ian Clarke, Oskar Sandberg, Brandon Wiley, and Theodore W. Hong. Freenet: A Distributed Anonymous Information Storage and Retrieval System. *Lecture Notes in Computer Science*, 2009:46–66, 2001.

[25] Brian F. Cooper. Guiding queries to information sources with infobeacons. In *Middleware '04: Proceedings of the 5th ACM/IFIP/USENIX international conference on Middleware*, pages 59–78, New York, NY, USA, 2004. Springer-Verlag New York, Inc.

[26] Arturo Crespo and Hector G. Molina. Semantic overlay networks for p2p systems. Technical report, Computer Science Department, Stanford University, 2002.

[27] Francisco Matias Cuenca-Acuna, Christopher Peery, Richard P. Martin, and Thu D. Nguyen. PlanetP: Using Gossiping to Build Content Addressable Peer-to-Peer Information Sharing Communities. In *Twelfth IEEE International Symposium on High Performance Distributed Computing (HPDC-12)*. IEEE Press, June 2003.

[28] Christos Gkantsidis, Milena Mihail, and Amin Saberi. Random walks in peer-to-peer networks: algorithms and evaluation. *Performance Evaluation*, 63(3):241–263, 2006.

[29] Gnutella.com. `http://www.gnutella.com/`, Accessed May 2006. OSMB, LLC.

[30] P. Brighten Godfrey and Ion Stoica. Heterogeneity and load balance in distributed hash tables. In *PROC. OF IEEE INFOCOM*, 2005.

[31] David Goldberg, David Nichols, Brian M. Oki, and Douglas Terry. Using collaborative filtering to weave an information tapestry. *Commun. ACM*, 35:61–70, December 1992.

[32] Peter Haase, Jeen Broekstra, Marc Ehrig, Maarten Menken, Peter Mika, Michal Plechawski, Pawel Pyszlak, Björn Schnizler, Ronny Siebes, Steffen Staab, and Christoph Tempich. Bibster — A Semantics-Based Bibliographic Peer-to-Peer System. In Sheila A. McIlraith, Dimitris Plexousakis, and Frank van Harmelen, editors, *Proceedings of the Third International Semantic Web Conference (ISWC 2004)*, pages 122–136, Hiroshima, Japan, November 2004. Springer.

[33] Peter Haase, Ronny Siebes, and Frank van Harmelen. Peer selection in peer-to-peer networks with semantic topologies. *Lecture Notes in Computer Science*, 3226, 2004.

[34] Peng Han, Bo Xie, Fan Yang, and Ruimin Shen. A scalable p2p recommender system based on distributed collaborative filtering. *Expert Systems with Applications*, 27(2):203 – 210, 2004.

[35] Jingfeng Hu, Ming Li, Weimin Zheng, Dongsheng Wang, Ning Ning, and Haitao Dong. Smartboa: Constructing p2p overlay network in the heterogeneous internet using irregular routing tables. In *the 3rd International Workshop on Peer-to-Peer System, IPTPS'04*, 2004.

[36] Adriana Iamnitchi, Matei Ripeanu, and Ian Foster. Locating Data in (Small-World?) Peer-to-Peer Scientific Collaborations. In *1st International Workshop on Peer-to-Peer Systems (IPTPS'02)*, 2002.

[37] Mark Jelasity, Rachid Guerraoui, Anne-Marie Kermarrec, and Maarten van Steen. The Peer Sampling Service: Experimental Evaluation of Unstructured Gossip-Based Implementations. In *ACM/IFIP/USENIX 5th International Middleware Conference*, 2004.

[38] Márk Jelasity, Alberto Montresor, Gian Paolo Jesi, and Spyros Voulgaris. The Peersim simulator. `http://peersim.sf.net`.

[39] Shudong Jin and Hongbo Jiang. Novel approaches to efficient flooding search in peer-to-peer networks. *Comput. Netw.*, 51(10):2818–2832, 2007.

[40] Sam Joseph. Neurogrid: Semantically routing queries in peer-to-peer networks. In *Revised Papers from the NETWORKING 2002 Workshops on Web Engineering and Peer-to-Peer Computing*, pages 202–214, London, UK, 2002. Springer-Verlag.

[41] Henry Kautz, Bart Selman, and Mehul Shah. Referral web: combining social networks and collaborative filtering. *Commun. ACM*, 40:63–65, March 1997.

[42] KaZaa.com . `http://www.kazaa.com`, Accessed May 2006. Sharman Networks.

[43] Anne-Marie Kermarrec, Laurent Massoulie, and Ayalvadi J. Ganesh. Probabilistic Reliable Dissemination in Large-Scale Systems. *IEEE Transactions on Parallel and Distributed Systems*, 14(3):248–258, March 2003.

[44] Joseph A. Konstan, Bradley N. Miller, David Maltz, Jonathan L. Herlocker, Lee R. Gordon, and John Riedl. Grouplens: applying collaborative filtering to usenet news. *Commun. ACM*, 40:77–87, March 1997.

[45] Michael Ley. Computer Science Bibliography. `http://dblp.uni-trier.de/`, Accessed May 2006. University of Trier, CS Department (Trier, Germany).

[46] Jinyang Li, Boon Thau Loo, Loo Joseph, Joseph M. Hellerstein, David R. Karger, Robert Morris, and M. Frans Kaashoek. On the feasibility of peer-to-peer web indexing and search. In *Proceeding of the International workshop on Peer-to-Peer Systems, IPTPS03*, pages 207–215, 2003.

[47] David Liben-Nowell, Hari Balakrishnan, and Davide Karger. Analysis of the evolution of peer-topeer systems. In *the Twenty-Firstst Annual Symposium on Principles of Distributed Computing*, 2002.

[48] Greg Linden, Brent Smith, and Jeremy York. Amazon.com recommendations: Item-to-item collaborative filtering. *IEEE Internet Computing*, 7:76–80, 2003.

[49] Alexander Löser, Christoph Tempich, Bastian Quilitz, Steffen Staab, Wolf Tilo Balke, and Wolfgang Nejdl. Searching dynamic communities with personal indexes. In Yolanda Gil, Enrico Motta, V. Richard Benjamins, and Mark A. Musen, editors, *Proc. 4th International Semantic Web Conference, ISWC 2005*, volume 3729 of *LNCS*, pages 491 – 505, Galway, Ireland, NOV 2005. Springer-Verlag GmbH.

[50] Qin Lv, Pei Cao, Edith Cohen, Kai Li, and Scott Shenker. Search and replication in unstructured peer-to-peer networks. In *ICS '02: Proceedings of the 16th international conference on Supercomputing*, pages 84–95, New York, NY, USA, 2002. ACM.

[51] Christopher D Manning., Prabhakar Raghavan, and Hinrich Schütze. *Introduction to Information Retrieval*. Cambridge University Press, 1 edition, July 2008.

[52] Jose Mitre and Leandro Navarro-Moldes. P2P Architecture for Scientific Collaboration. In *the 13th IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises (WET ICE'04)*, 2004.

[53] Pradeep Monga. Web-based P2P Support for Research Collaboration. Master's thesis, Faculty of Computer Science, Dalhousie University, 2005.

[54] Wolfgang Nejdl, Boris Wolf, Changtao Qu, Stefan Decker, Michael Sintek, Ambjörn Naeve, Mikael Nilsson, Matthias Palmér, and Tore Risch. EDUTELLA: a P2P Networking Infrastructure Based on RDF. In *WWW*, pages 604–615, 2002.

[55] Thomas Neumann, Matthias Bender, Sebastian Michel, and Gerhanrd Weikum. A reproducible benchmark for p2p retrieval. In *Proceedins of the First International Workshop on Performance and Evaluation of Data Management Systems, EXPDB2006*, June 2006.

[56] M. E. J. Newman. Scientific collaboration networks. i. network construction and fundamental results. *Physical Review E*, Volume 64, June 2001.

[57] Josiane Xavier Parreira, Sebastian Michel, and Gerhard Weikum. p2pDating: Real life inspired semantic overlay networks for web search. *Information Processing and Management*, 43(3):643–664, 2007.

[58] David M. Pennock, Eric Horvitz, Steve Lawrence, and C. Lee Giles. Collaborative filtering by personality diagnosis: A hybrid memory- and model-based approach. In *Proceedings of the 16th Conference on Uncertainty in Artificial Intelligence*, UAI-2000, pages 473–480, 2000.

[59] J. A. Pouwelse, P. Garbacki, J. Wang, A. Bakker, J. Yang, A. Iosup, D. H. J. Epema, M. Reinders, M. R. van Steen, and H. J. Sips. Tribler: a social-based peer-to-peer system: Research articles. *Concurr. Comput. : Pract. Exper.*, 20(2):127–138, 2008.

[60] Sylvia Ratnasamy, Paul Francis, Mark Handley, and Richard Karp. A Scalable Content-Addressable Network. In *Proceedings of the ACM SIGCOMM*, pages 161–172, August 2001.

[61] Stephen Robertson, Hugo Zaragoza, and Michael Taylor. Simple bm25 extension to multiple weighted fields. In *CIKM '04: Proceedings of the thirteenth ACM international conference on Information and knowledge management*, pages 42–49, New York, NY, USA, 2004. ACM.

[62] Antony I. T. Rowstron and Peter Druschel. Pastry: Scalable, Decentralized Object Location, and Routing for Large-Scale Peer-to-Peer Systems. In *Proceedings of the IFIP/ACM International Conference on Distributed Systems Platforms Heidelberg*, pages 329–350. Springer-Verlag, 2001.

[63] R.Werlen. DFN Science-To-Science (S2S): Peer-to-Peer Scientific Research. In *the Terena Networking Conference (TNC 2003)*, 2003.

[64] Stefan Saroiu, P. Krishna Gummadi, and Steven D. Gribble. A measurement study of peer-to-peer file sharing systems. In *Multimedia Computing and Networking (MMCN)*, 2002.

[65] Christoph Schmitz. Self-organization of a small world by topic. In *Proceeding of the 1st International Workshop on Peer-to-Peer Knowledge Management*, 2004.

[66] R. Schollmeier. A definition of peer-to-peer networking for the classification of peer-to-peer architectures and applications. *Peer-to-Peer Computing, IEEE International Conference on*, 0:0101, 2001.

[67] Kunwadee Sripanidkulchai, Bruce Maggs, and Hui Zhang. Efficient content location using interest-based. In *in INFOCOM*, 2003.

[68] Ion Stoica, Robert Morris, David Liben-Nowell, David R. Karger, M. Frans Kaashoek, Frank Dabek, and Hari Balakrishman. Chord: A Scalable Peer-to-Peer Lookup Protocol for Internet Applications. *IEEE/ACM Transactions on Networking*, 11(1):17–32, February 2003.

[69] Jeremy Stribling, Isaac G. Councill, Jinyang Li, M. Frans Kaashoek, David R. Karger, Robert Morris, and Scott Shenker. OverCite: A Cooperative Digital Research Library. In *IPTPS*, pages 69–79, 2005.

[70] Hathai Tanta-ngai and Michael MaAllister. A peer-to-peer expressway over chord. *Mathematical and Computer Modelling*, 44(7-8):659–677, 2006.

[71] C. Tempich, A. Löser, and J. Heizmann. Community Based Ranking in Peer-to-Peer Networks. In *OTM Confederated International Conferences CoopIS, DOA, and ODBASE*, pages 1261–1278, Agia napa, Cyprus, October 2005. Springer.

[72] Christoph Tempich, Steffen Staab, and Adrian Wranik. Remindin': semantic query routing in peer-to-peer networks based on social metaphors. In *WWW '04: Proceedings of the 13th international conference on World Wide Web*, pages 640–649, New York, NY, 2004. ACM Press.

[73] Spyros Voulgaris. Epidemic-style management of semantic overlays for content-based searching. In *Euro-Par 2005, Parallel Processing, 11th International Euro-Par Conference*, pages 1143–1152. Springer, 2005.

[74] Spyros Voulgaris, Maarten van Steen, and Konrad Iwanicki. Proactive gossip-based management of semantic overlay networks. *Concurrency and Computation: Practice and Experience*, 19(17), Aug 2004.

[75] Duncan J. Watts. Networks, Dynamics, and the Small-World Phenomenon. *American Journal of Sociology*, 105(2):493–527, September 1999.

[76] Duncan J. Watts and Steven H. Strogatz. Collective Dynamics of "Small-World" Networks. *Nature*, 393(4):440–442, June 1998.

[77] Le-Shin Wu, Ruj Akavipat, and Filippo Menczer. 6s: Distributing crawling and searching across web peers. In *Web Technologies, Applications, and Services*, pages 159–164, 2005.

[78] Gui-Rong Xue, Chenxi Lin, Qiang Yang, WenSi Xi, Hua-Jun Zeng, Yong Yu, and Zheng Chen. Scalable collaborative filtering using cluster-based smoothing. In *Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '05, pages 114–121, New York, NY, USA, 2005. ACM.

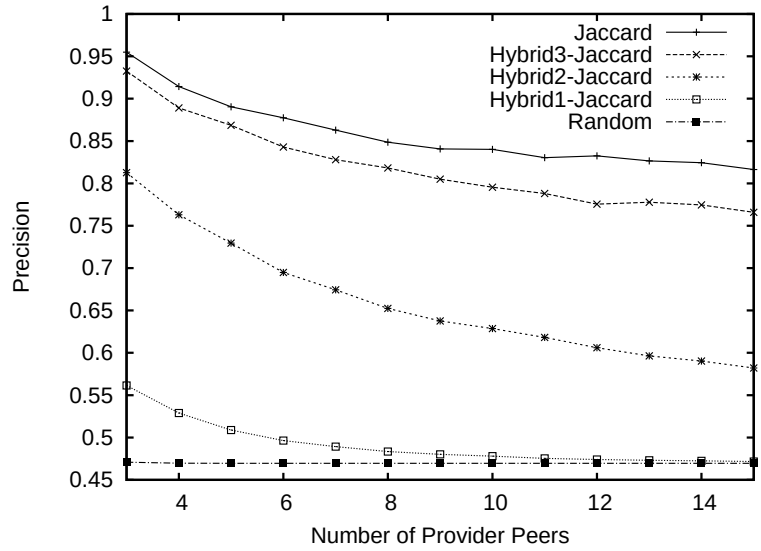
[79] Xiangying Yang and Gustavo de Veciana. Performance of peer-to-peer networks: service capacity and role of resource sharing policies. *Perform. Eval.*, 63(3):175–194, 2006.

[80] B. Zhao, Ling Huang, Jeremy Stribling, Sean C. Rhea, Anthony D. Joseph, and John D. Kubiatowicz. Tapestry: A Resilient Global-Scale Overlay for Service Deployment. *IEEE Journal on Selected Areas in Communications*, 22(1):41–53, 2004.

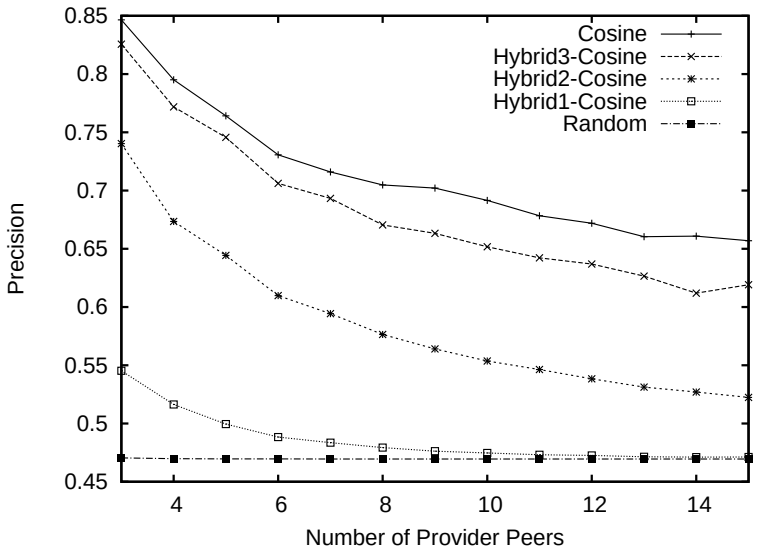
Appendix A

Complete Experimental Results of Self-Organizing Shrack Networks with Unlimited TTL

In this appendix, we present the complete results from experiments with unlimited TTL. We compare the results of the system with unlimited TTL among different provider peer selection strategies with item-based and term-based profile representations. The results are presented in the aspects of the quality of received documents, the dissemination speed and distance of relevant documents and the self-organizing network property.

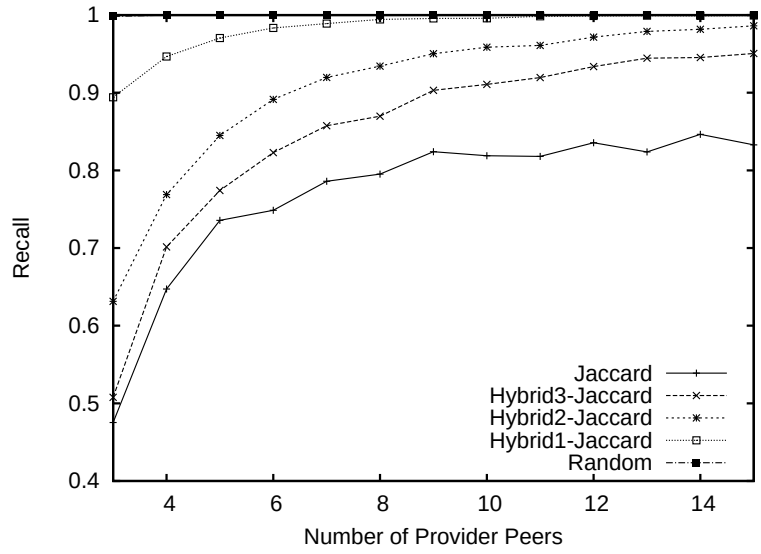


(a)

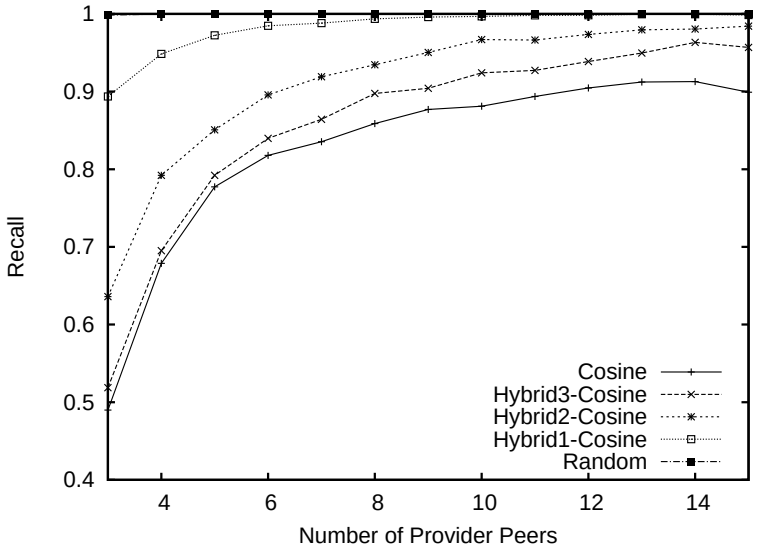


(b)

Figure A.1: The quality of documents received in terms of precision as the number of provider peers increases using different provider peer selection strategies with unlimited TTL: (a) an item-based profile representation; (b) a term-based profile representation

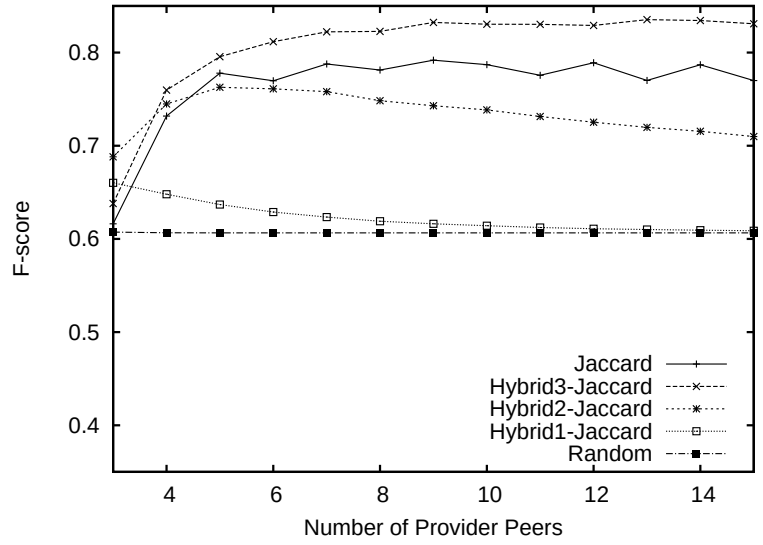


(a)

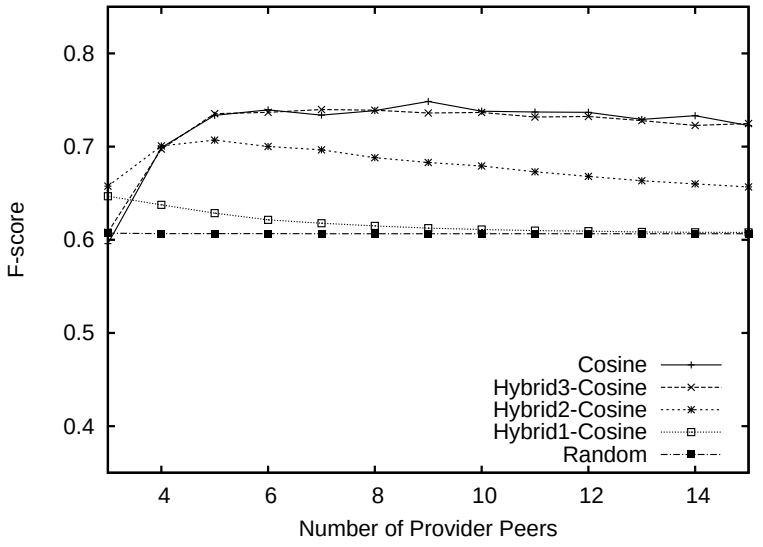


(b)

Figure A.2: The quality of documents received in terms of recall as the number of provider peers increases using different provider peer selection strategies with unlimited TTL: (a) an item-based profile representation; (b) a term-based profile representation

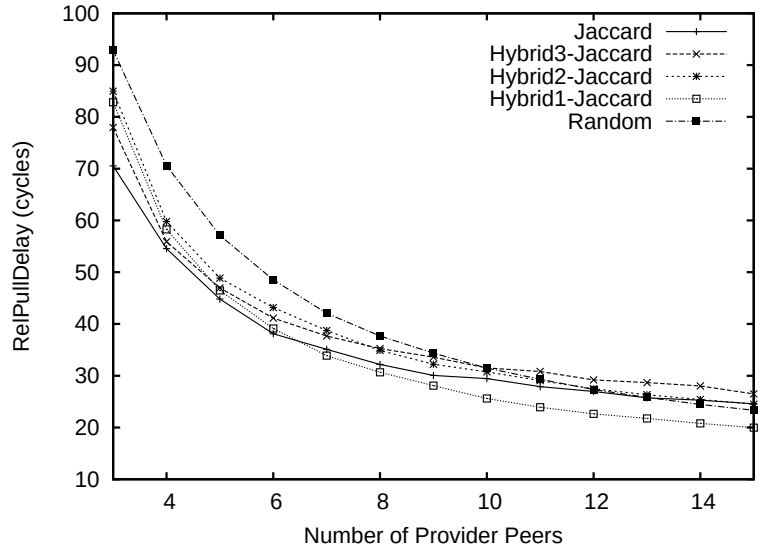


(a)

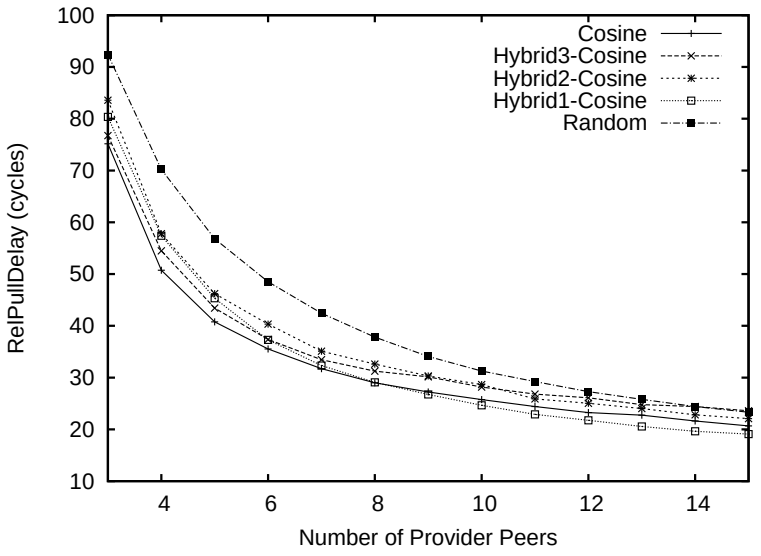


(b)

Figure A.3: The quality of documents received in terms of F-score as the number of provider peers increases using different provider peer selection strategies with unlimited TTL: (a) an item-based profile representation; (b) a term-based profile representation

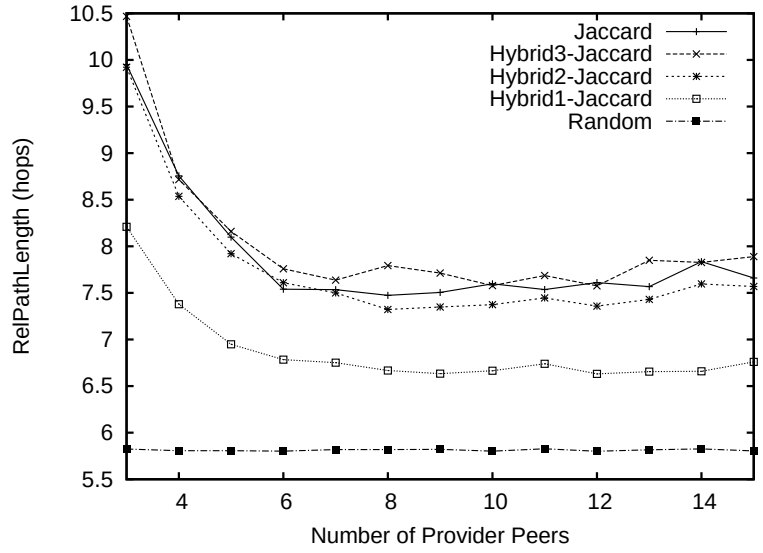


(a)

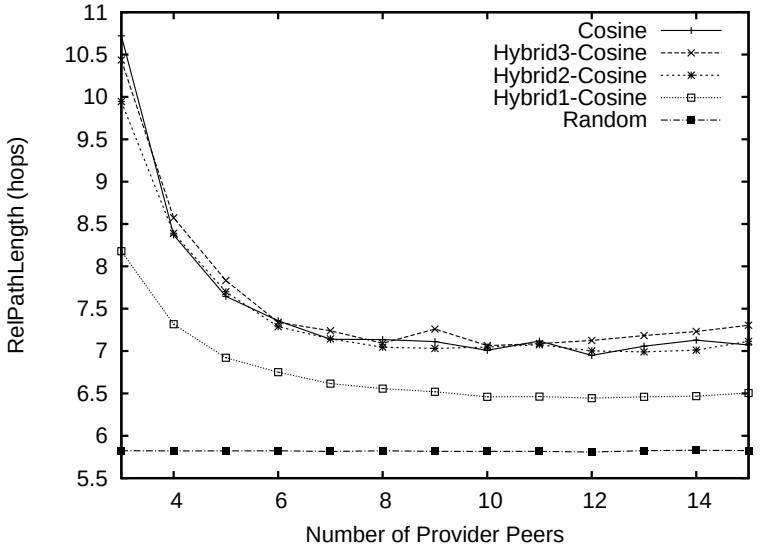


(b)

Figure A.4: The relevant pull delay as the number of provider peers increases using different provider peer selection strategies with unlimited TTL: (a) an item-based profile representation; (b) a term-based profile representation

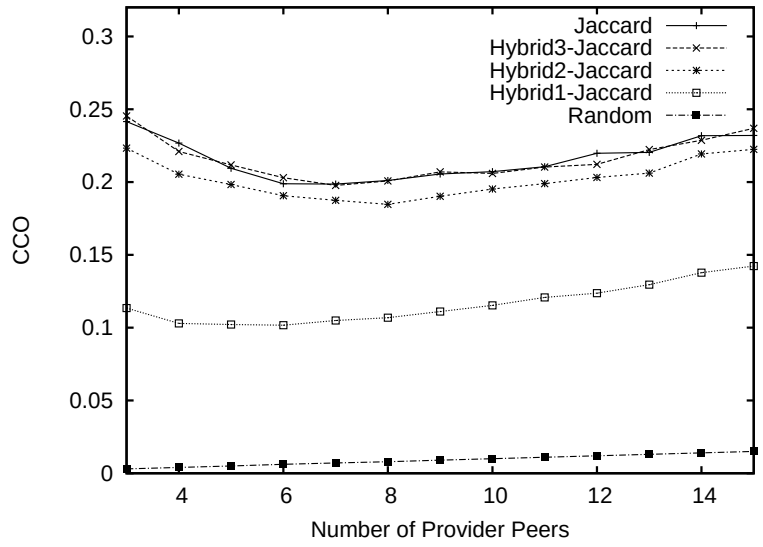


(a)

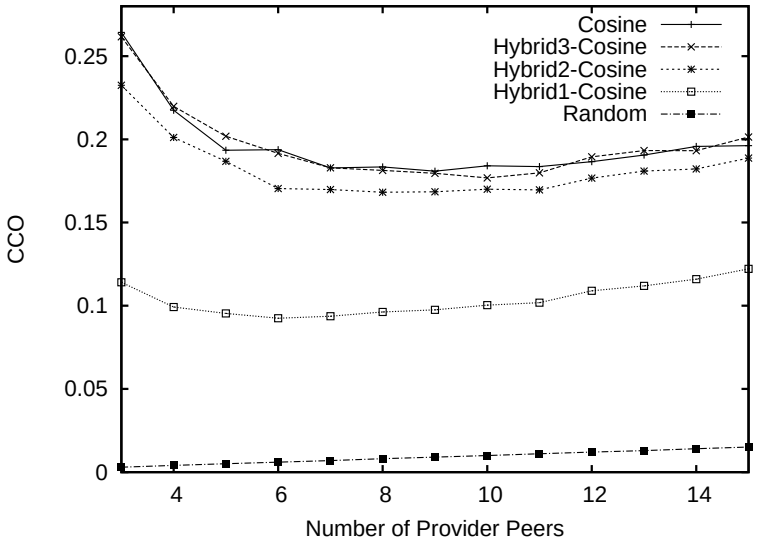


(b)

Figure A.5: The relevant path length as the number of provider peers increases using different provider peer selection strategies with unlimited TTL: (a) an item-based profile representation; (b) a term-based profile representation

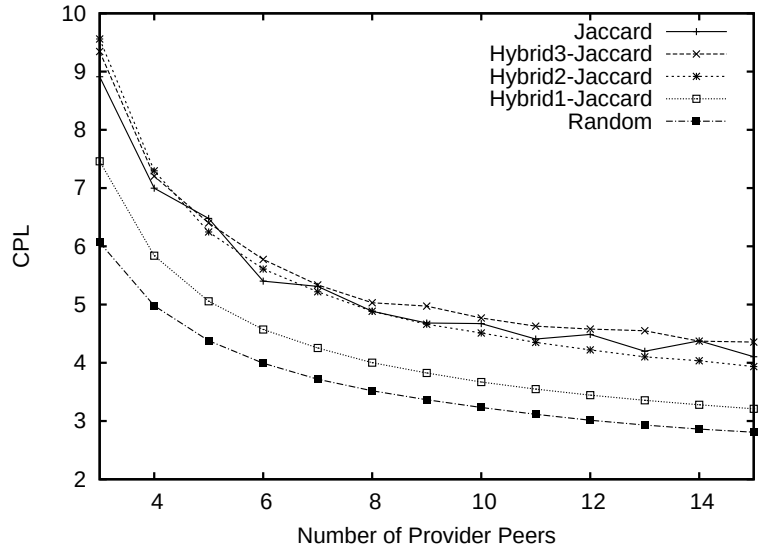


(a)

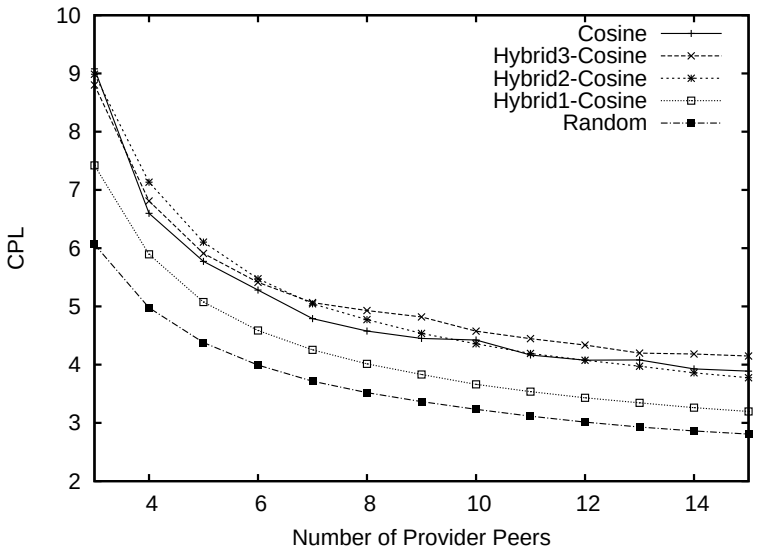


(b)

Figure A.6: The network clustering coefficient as the number of provider peers increases using different provider peer selection strategies with unlimited TTL: (a) an item-based profile representation; (b) a term-based profile representation

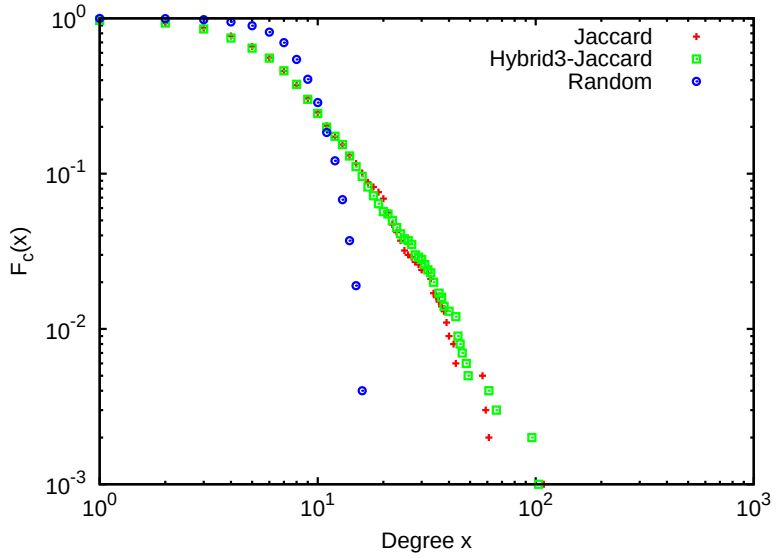


(a)

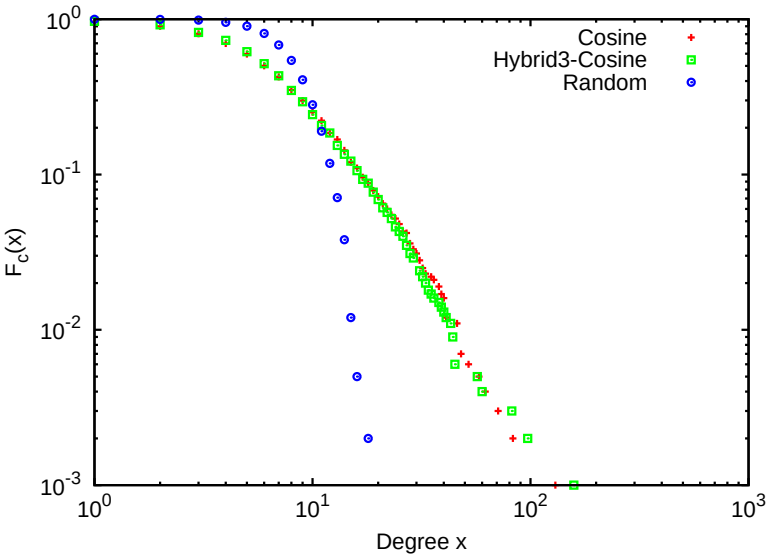


(b)

Figure A.7: The network characteristic path length as the number of provider peers increases using different provider peer selection strategies with unlimited TTL: (a) an item-based profile representation; (b) a term-based profile representation



(a)



(b)

Figure A.8: Example of the in-degree complementary cumulative distribution function of the networks under different provider peer selection strategies with unlimited TTL; (a) an item-based profile representation; (b) a term-based profile representation

Appendix B

Statistic Results of ANOVA Tests on Self-Organizing Networks with Limited TTL

This appendix presents a complete statistic resuls of ANOVA tests on the self-organzing Shrack networks. We did full factorial ANOVA to test statistical difference of the experiment results. Note that "Algo" refers to the provider peer selection strategy and "Profile" refers to the profile representation.

Table B.1: Tests of Between-Subjects Effects; Dependent Variable: Precision

Source	**Type III SSQ**	**df**	**MS**	**F**	**Sig.**	**Partial η^2**
Corrected Model	76.671a	59	1.300	2594.229	.000	.963
Intercept	2080.699	1	2080.699	4153753.087	.000	.999
Algo	53.854	4	13.464	26877.675	.000	.948
N	11.240	5	2.248	4487.771	.000	.791
Profile	4.172	1	4.172	8327.753	.000	.584
Algo*N	4.332	20	.217	432.427	.000	.593
Algo*Profile	2.982	4	.746	1488.400	.000	.501
N*Profile	.051	5	.010	20.260	.000	.017
Algo*N*Profile	.039	20	.002	3.937	.000	.013
Error	2.975	5940	.001			
Total	2160.345	6000				

a. R Squared = .889 (Adjusted R Squared = .888)
b. Computed using alpha = .05

Table B.2: Tests of Between-Subjects Effects; Dependent Variable: Recall

Source	**Type III SSQ**	**df**	**MS**	**F**	**Sig.**	**Partial η^2**
Corrected Model	89.123a	59	1.511	1901.172	.000	.950
Intercept	4873.691	1	4873.691	6133954.656	.000	.999
Algo	35.372	4	8.843	11129.651	.000	.882
N	36.275	5	7.255	9131.156	.000	.885
Profile	.183	1	.183	230.429	.000	.037
Algo*N	17.023	20	.851	1071.242	.000	.783
Algo*Profile	.112	4	.028	35.196	.000	.023
N*Profile	.094	5	.019	23.782	.000	.020
Algo*N*Profile	.063	20	.003	3.991	.000	.013
Error	4.720	5940	.001			
Total	4967.534	6000				
Corrected Total	93.843	5999				

a. R Squared = .950 (Adjusted R Squared = .949)
b. Computed using alpha = .05

Table B.3: Tests of Between-Subjects Effects; Dependent Variable: F-score

Source	**Type III SSQ**	**df**	**MS**	**F**	**Sig.**	**Partial η^2**
Corrected Model	18.414a	59	.312	806.388	.000	.889
Intercept	2660.118	1	2660.118	6872983.827	.000	.999
Algo	11.185	4	2.796	7224.927	.000	.830
N	1.079	5	.216	557.651	.000	.319
Profile	2.418	1	2.418	6247.657	.000	.513
Algo*N	2.194	20	.110	283.381	.000	.488
Algo*Profile	1.479	4	.370	955.222	.000	.391
N*Profile	.011	5	.002	5.856	.000	.005
Algo*N*Profile	.048	20	.002	6.174	.000	.020
Error	2.299	5940	.000			
Total	2680.831	6000				
Corrected Total	20.713	5999				

a. R Squared = .889 (Adjusted R Squared = .888)
b. Computed using alpha = .05

Table B.4: Tests of Between-Subjects Effects; Dependent Variable: Relevant Pull Delay

Source	**Type III SSQ**	**df**	**MS**	**F**	**Sig.**	**Partial η^2**
Corrected Model	9.440E11	59	1.600E10	4573.820	.000	.978
Intercept	6.602E12	1	6.602E12	1887173.450	.000	.997
Algo	9.484E10	4	2.371E10	6778.093	.000	.820
N	7.925E11	5	1.585E11	45308.172	.000	.974
Profile	5.241E8	1	5.241E8	149.810	.000	.025
Algo*N	5.455E10	20	2.728E9	779.719	.000	.724
Algo*Profile	8.799E8	4	2.200E8	62.883	.000	.041
N*Profile	2.507E8	5	5.014E7	14.334	.000	.012
Algo*N*Profile	4.714E8	20	2.357E7	6.739	.000	.022
Error	2.078E10	5940	3498110.537			
Total	7.566E12	6000				
Corrected Total	9.648E11	5999				

a. R Squared = .978 (Adjusted R Squared = .978)
b. Computed using alpha = .05

Table B.5: Tests of Between-Subjects Effects; Dependent Variable: Relevant Path length

Source	**Type III SSQ**	**df**	**MS**	**F**	**Sig.**	**Partial η^2**
Corrected Model	347.527a	59	5.890	1047.313	.000	.912
Intercept	190702.302	1	190702.302	3.391E7	.000	1.000
Algo	133.795	4	33.449	5947.293	.000	.800
N	163.921	5	32.784	5829.126	.000	.831
Profile	5.346	1	5.346	950.535	.000	.138
Algo*N	42.496	20	2.125	377.800	.000	.560
Algo*Profile	1.486	4	.371	66.044	.000	.043
N*Profile	.211	5	.042	7.494	.000	.006
Algo*N*Profile	.273	20	.014	2.425	.000	.008
Error	33.408	5940	.006			
Total	191083.237	6000				
Corrected Total	380.935	5999				

a. R Squared = .912 (Adjusted R Squared = .911)
b. Computed using alpha = .05

Table B.6: Tests of Between-Subjects Effects; Dependent Variable: CCO

Source	**Type III SSQ**	**df**	**MS**	**F**	**Sig.**	**Partial η^2**
Corrected Model	21.047a	59	.357	3596.735	.000	.973
Intercept	76.545	1	76.545	771761.367	.000	.992
Algo	20.559	4	5.140	51822.718	.000	.972
N	.244	5	.049	491.230	.000	.293
Profile	.003	1	.003	35.071	.000	.006
Algo*N	.149	20	.007	75.300	.000	.202
Algo*Profile	.007	4	.002	18.748	.000	.012
N*Profile	.062	5	.012	125.347	.000	.095
Algo*N*Profile	.022	20	.001	10.877	.000	.035
Error	.589	5940	9.918E-5			
Total	98.181	6000				
Corrected Total	21.636	5999				

a. R Squared = .973 (Adjusted R Squared = .973)
b. Computed using alpha = .05

Table B.7: Tests of Between-Subjects Effects; Dependent Variable: CPL

Source	**Type III SSQ**	**df**	**MS**	**F**	**Sig.**	**Partial η^2**
Corrected Model	5904.216a	59	100.071	1885.077	.000	.949
Intercept	113106.007	1	113106.007	2130612.530	.000	.997
Algo	1173.208	4	293.302	5525.020	.000	.788
N	4640.618	5	928.124	17483.347	.000	.936
Profile	9.543	1	9.543	179.766	.000	.029
Algo*N	56.622	20	2.831	53.331	.000	.152
Algo*Profile	17.246	4	4.311	81.215	.000	.052
N*Profile	2.663	5	.533	10.033	.000	.008
Algo*N*Profile	4.317	20	.216	4.066	.000	.014
Error	315.332	5940	.053			
Total	119325.555	6000				
Corrected Total	6219.548	5999				

a. R Squared = .949 (Adjusted R Squared = .949)
b. Computed using alpha = .05

Appendix C

Statistic Results of ANOVA Tests on the Effect of TTL

This appendix presents a complete statistic resuls of ANOVA tests on the effect of TTL on Self-Organizing Shrack Networks. We did ANOVA test on the system using the Common-Interest Strategy and the Hybrid Strategy with $\beta = 10^{-3}$. Note that "Algo" refers to the provider peer selection strategy and "Profile" refers to the profile representation.

Table C.1: Tests of Between-Subjects Effects; Dependent Variable: Precision

Source	**Type III SSQ**	**df**	**MS**	**F**	**Sig.**	**Partial η^2**
Corrected Model	40.606a	47	.864	1101.660	.000	.916
Intercept	2542.897	1	2542.897	3242527.794	.000	.999
Algo	.947	1	.947	1207.500	.000	.203
N	13.677	5	2.735	3487.872	.000	.786
Profile	18.180	1	18.180	23181.412	.000	.830
TTL	6.293	1	6.293	8023.976	.000	.628
Algo*N	.073	5	.015	18.655	.000	.019
Algo*Profile	.005	1	.005	6.071	.014	.001
Algo*TTL	.102	1	.102	129.561	.000	.027
N*Profile	.020	5	.004	5.097	.000	.005
N*TTL	.648	5	.130	165.300	.000	.148
Profile*TTL	.530	1	.530	675.775	.000	.125
Algo*N*Profile	.014	5	.003	3.474	.004	.004
Algo*N*TTL	.003	5	.001	.827	.530	.001
Algo*Profile*TTL	.001	1	.001	1.537	.215	.000
N*Profile*TTL	.110	5	.022	28.097	.000	.029
Algo*N* Profile*TTL	.004	5	.001	1.113	.351	.001
Error	3.727	4752	.001			
Total	2587.229	4800				
Corrected Total	44.333	4799				

a. R Squared = .916 (Adjusted R Squared = .915)
b. Computed using alpha = .05

Table C.2: Tests of Between-Subjects Effects; Dependent Variable: Recall

Source	**Type III SSQ**	**df**	**MS**	**F**	**Sig.**	**Partial η^2**
Corrected Model	57.967a	47	1.233	878.023	.000	.897
Intercept	3312.097	1	3312.097	2.358E6	.000	.998
Algo	1.757	1	1.757	1251.054	.000	.208
N	51.027	5	10.205	7265.367	.000	.884
Profile	.139	1	.139	99.308	.000	.020
TTL	.531	1	.531	377.738	.000	.074
Algo*N	.140	5	.028	19.898	.000	.021
Algo*Profile	.219	1	.219	155.826	.000	.032
Algo*TTL	.469	1	.469	333.540	.000	.066
N*Profile	.167	5	.033	23.836	.000	.024
N*TTL	2.556	5	.511	363.897	.000	.277
Profile*TTL	.788	1	.788	560.995	.000	.106
Algo*N*Profile	.022	5	.004	3.130	.008	.003
Algo*N*TTL	.008	5	.002	1.150	.331	.001
Algo*Profile*TTL	.132	1	.132	93.836	.000	.019
N*Profile*TTL	.010	5	.002	1.490	.189	.002
Algo*N* Profile*TTL	.001	5	.000	.189	.967	.000
Error	6.675	4752	.001			
Total	3376.738	4800				
Corrected Total	64.642	4799				

a. R Squared = .897 (Adjusted R Squared = .896)
b. Computed using alpha = .05

Table C.3: Tests of Between-Subjects Effects; Dependent Variable: F-score

Source	**Type III SSQ**	**df**	**MS**	**F**	**Sig.**	**Partial η^2**
Corrected Model	13.075a	47	.278	431.237	.000	.810
Intercept	2596.268	1	2596.268	4.025E6	.000	.999
Algo	.091	1	.091	140.757	.000	.029
N	3.169	5	.634	982.575	.000	.508
Distance	5.607	1	5.607	8692.144	.000	.647
TTL	3.338	1	3.338	5175.071	.000	.521
Algo*N	.010	5	.002	2.970	.011	.003
Algo*Distance	.185	1	.185	286.012	.000	.057
Algo*TTL	.115	1	.115	178.908	.000	.036
N*Distance	.087	5	.017	26.978	.000	.028
N*TTL	.305	5	.061	94.688	.000	.091
Distance*TTL	.011	1	.011	16.778	.000	.004
Algo*N*Distance	.012	5	.002	3.837	.002	.004
Algo*N*TTL	.006	5	.001	1.962	.081	.002
Algo*Distance*TTL	.107	1	.107	165.248	.000	.034
N*Distance*TTL	.030	5	.006	9.194	.000	.010
Algo*N* Distance*TTL	.001	5	.000	.437	.823	.000
Error	3.065	4752	.001			
Total	2612.408	4800				
Corrected Total	16.140	4799				

a. R Squared = .810 (Adjusted R Squared = .808)
b. Computed using alpha = .05

Table C.4: Tests of Between-Subjects Effects; Dependent Variable: Relevant Pull Delay

Source	**Type III SSQ**	**df**	**MS**	**F**	**Sig.**	**Partial η^2**
Corrected Model	4.385E11	47	9.331E9	2410.051	.000	.960
Intercept	4.861E12	1	4.861E12	1.256E6	.000	.996
Algo	4.601E9	1	4.601E9	1188.350	.000	.200
N	4.012E11	5	8.025E10	20727.289	.000	.956
Profile	3.444E9	1	3.444E9	889.482	.000	.158
TTL	2.201E10	1	2.201E10	5685.433	.000	.545
Algo*N	2.281E7	5	4.562E6	1.178	.317	.001
Algo*Profile	5.241E6	1	5.241E6	1.354	.245	.000
Algo*TTL	3.974E8	1	3.974E8	102.642	.000	.021
N*Profile	7.029E7	5	1.406E7	3.631	.003	.004
N*TTL	3.265E9	5	6.530E8	168.671	.000	.151
Profile*TTL	3.186E9	1	3.186E9	823.017	.000	.148
Algo*N*Profile	7.690E7	5	1.538E7	3.972	.001	.004
Algo*N*TTL	1.035E7	5	2.071E6	.535	.750	.001
Algo*Profile*TTL	3.015E6	1	3.015E6	.779	.378	.000
N*Profile*TTL	6.118E7	5	1.224E7	3.161	.007	.003
Algo*N* Profile*TTL	1.516E8	5	3.033E7	7.834	.000	.008
Error	1.840E10	4752	3871611.103			
Total	5.318E12	4800				
Corrected Total	4.569E11	4799				

a. R Squared = .960 (Adjusted R Squared = .959)
b. Computed using alpha = .05

Table C.5: Tests of Between-Subjects Effects; Dependent Variable: Relevant Path Length

Source	**Type III SSQ**	**df**	**MS**	**F**	**Sig.**	**Partial η^2**
Corrected Model	5029.270a	47	107.006	1436.097	.000	.934
Intercept	212871.088	1	212871.088	2.857E6	.000	.998
Algo	2.643	1	2.643	35.475	.000	.007
N	512.529	5	102.506	1375.704	.000	.591
Profile	94.019	1	94.019	1261.799	.000	.210
TTL	4259.182	1	4259.182	57161.402	.000	.923
Algo*N	.533	5	.107	1.430	.210	.002
Algo*Profile	.013	1	.013	.176	.675	.000
Algo*TTL	1.008	1	1.008	13.526	.000	.003
N*Profile	6.871	5	1.374	18.442	.000	.019
N*TTL	93.689	5	18.738	251.475	.000	.209
Profile*TTL	45.163	1	45.163	606.119	.000	.113
Algo*N*Profile	3.600	5	.720	9.664	.000	.010
Algo*N*TTL	.498	5	.100	1.338	.245	.001
Algo*Profile*TTL	3.931E-5	1	3.931E-5	.001	.982	.000
N*Profile*TTL	5.368	5	1.074	14.408	.000	.015
Algo*N* Profile*TTL	4.155	5	.831	11.152	.000	.012
Error	354.079	4752	.075			
Total	218254.437	4800				
Corrected Total	5383.349	4799				

a. R Squared = .934 (Adjusted R Squared = .934)
b. Computed using alpha = .05

Table C.6: Tests of Between-Subjects Effects; Dependent Variable: CCO

Source	**Type III SSQ**	**df**	**MS**	**F**	**Sig.**	**Partial η^2**
Corrected Model	2.976a	47	.063	312.699	.000	.756
Intercept	158.414	1	158.414	782351.067	.000	.994
Algo	.006	1	.006	27.163	.000	.006
N	.404	5	.081	398.732	.000	.296
Profile	.123	1	.123	609.508	.000	.114
TTL	2.168	1	2.168	10709.111	.000	.693
Algo*N	.002	5	.000	2.421	.034	.003
Algo*Profile	6.525E-5	1	6.525E-5	.322	.570	.000
Algo*TTL	7.039E-5	1	7.039E-5	.348	.555	.000
N*Profile	.108	5	.022	107.070	.000	.101
N*TTL	.020	5	.004	19.763	.000	.020
Profile*TTL	.124	1	.124	613.931	.000	.114
Algo*N*Profile	.003	5	.001	2.814	.015	.003
Algo*N*TTL	.004	5	.001	3.728	.002	.004
Algo*Profile*TTL	.001	1	.001	2.566	.109	.001
N*Profile*TTL	.006	5	.001	5.639	.000	.006
Algo*N* Profile*TTL	.007	5	.001	6.611	.000	.007
Error	.962	4752	.000			
Total	162.352	4800				
Corrected Total	3.938	4799				

a. R Squared = .756 (Adjusted R Squared = .753)
b. Computed using alpha = .05

Table C.7: Tests of Between-Subjects Effects; Dependent Variable: CPL

Source	**Type III SSQ**	**df**	**MS**	**F**	**Sig.**	**Partial η^2**
Corrected Model	4340.723a	47	92.356	655.175	.000	.866
Intercept	114468.363	1	114468.363	812041.982	.000	.994
Algo	33.345	1	33.345	236.551	.000	.047
N	4046.384	5	809.277	5741.033	.000	.858
Profile	70.886	1	70.886	502.866	.000	.096
TTL	167.091	1	167.091	1185.345	.000	.200
Algo*N	2.830	5	.566	4.015	.001	.004
Algo*Profile	2.338	1	2.338	16.584	.000	.003
Algo*TTL	.622	1	.622	4.415	.036	.001
N*Profile	5.022	5	1.004	7.125	.000	.007
N*TTL	.815	5	.163	1.156	.328	.001
Profile*TTL	2.121	1	2.121	15.049	.000	.003
Algo*N*Profile	3.001	5	.600	4.258	.001	.004
Algo*N*TTL	.976	5	.195	1.385	.227	.001
Algo*Profile*TTL	.046	1	.046	.326	.568	.000
N*Profile*TTL	2.996	5	.599	4.251	.001	.004
Algo*N* Profile*TTL	2.251	5	.450	3.193	.007	.003
Error	669.859	4752	.141			
Total	119478.945	4800				
Corrected Total	5010.582	4799				

a. R Squared = .866 (Adjusted R Squared = .865)
b. Computed using alpha = .05

Table C.8: Tests of Between-Subjects Effects; Dependent Variable: Pull Load

Source	**Type III SSQ**	**df**	**MS**	**F**	**Sig.**	**Partial η^2**
Corrected Model	259325.498a	47	5517.564	98.524	.000	.494
Intercept	1.054E6	1	1.054E6	18827.639	.000	.798
Algo	180.826	1	180.826	3.229	.072	.001
N	191147.627	5	38229.525	682.640	.000	.418
Profile	47045.337	1	47045.337	840.058	.000	.150
TTL	8748.485	1	8748.485	156.216	.000	.032
Algo*N	40.660	5	8.132	.145	.981	.000
Algo*Profile	24.605	1	24.605	.439	.507	.000
Algo*TTL	29.836	1	29.836	.533	.465	.000
N*Profile	11756.640	5	2351.328	41.986	.000	.042
N*TTL	267.122	5	53.424	.954	.445	.001
Profile*TTL	56.308	1	56.308	1.005	.316	.000
Algo*N*Profile	4.978	5	.996	.018	1.000	.000
Profile						
Algo*N*TTL	3.782	5	.756	.014	1.000	.000
Algo*Profile*TTL	9.654	1	9.654	.172	.678	.000
N*Profile*TTL	7.309	5	1.462	.026	1.000	.000
Algo*N*	2.330	5	.466	.008	1.000	.000
Profile*TTL						
Error	266123.874	4752	56.002			
Total	1579844.227	4800				
Corrected Total	525449.372	4799				

a. R Squared = .494 (Adjusted R Squared = .489)
b. Computed using alpha = .05

Appendix D

ShrackSim: A Shrack Simulator

This chapter provides details of implementation of ShrackSim and describe how to use ShrackSim. We first present the main components of ShrackSim in Section D.1. After that, we explain the configuration file in Section D.2. Finally, we explain how to run and evaluate experiments in Section D.3.

D.1 Main Components of ShrackSim

This section explains main components of ShrackSim. First, we describe `ShrackNode` and its related components. After that, we explain `Control` classes, and then, describe interface and abstract classes.

D.1.1 `ShrackNode` and Related Components

Figure D.1 shows the main components that model a Shrack peer and their related components and protocols. Classes with gray background colour are classes that exist in PeerSim. We have already explained `Node`, `Protocol`, `EDProtocol`, and `Linkable` classes in Section 5.3.2 (page 61). Following is the description of `GeneralNode`, `ShrackNode`, `UserModel`, `ShrackProtocol`, `ShrackNeighbourhood`, `PullRequestGenerator`, and `PublishProtocol` classes.

`GeneralNode`: A default `Node` class in PeerSim that implements `Node` interface, which is used to compose an overlay network.

`ShrackNode`: An extension of `GeneralNode` that represents a Shrack peer. Each `ShrackNode` is associated with a predefined artificial user, which is represented by `UserModel`. `ShrackNode` also keeps status and local statistic of each peer. Each

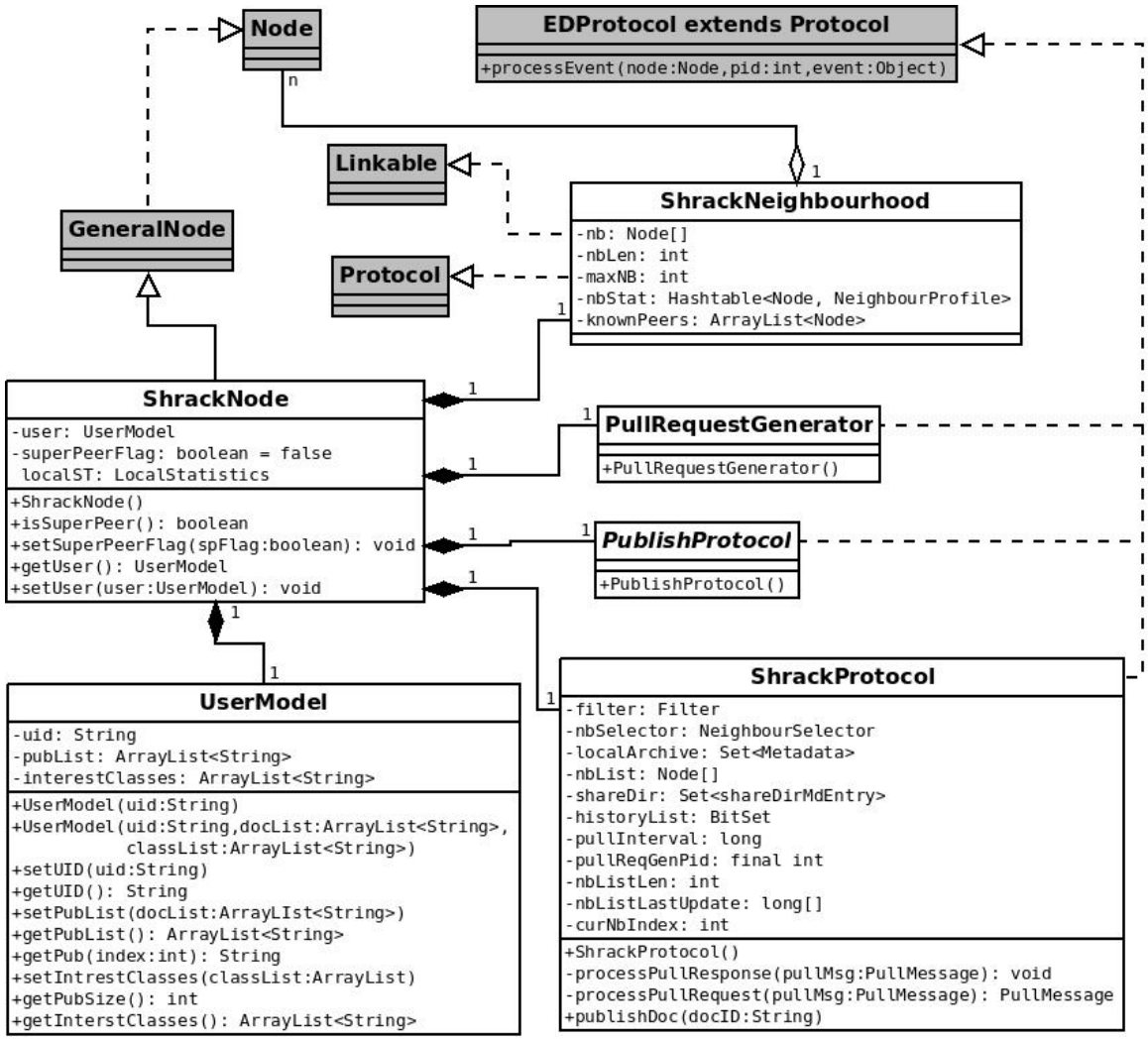


Figure D.1: ShrackSim main components

`ShrackNode` runs four protocols, namely `ShrackProtocol`, `ShrackNeighbourhood`, `PullRequestGenerator`, and `PublishProtocol`.

`UserModel:` `UserModel` represents an artificial user that is associated with a Shrack peer (`ShrackNode`). A `UserModel` consists of a user identifier, a list of publications that the associated peer will publish during the simulation, and a list of interest (publication) groups of which the peer should keep track.

`ShrackProtocol:` The main event-driven protocol that implements the Shrack dissemination protocol, which processes pull requests and pull responses.
The `ShrackProtocol` class implements the Pull Procedure represented in Section 4.1.2 (page 36). It contains `ShrackNeighbourhood` and `PullRequestGenerator`. The `ShrackNeighbourhood` provides the contact of provider peers and `PullRequestGenerator` issues pull requests to provider peers. The `ShrackProtocol` class also contains other three main interfaces that can be dynamically loaded from the configuration file including `Filter`, `NeighbourSelector`, and `LocalProfile`. We describe these three interfaces later in Section D.1.3.

`ShrackNeighbourhood:` A protocol that implements `Linkable`, creating the simulation overlay network that maintains a list of provider peers, a list of known peers, and profiles of known peers. The `ShrackNeighbourhood` class associates with a `NeighbourProfile` interface, described in Section D.1.3, that can be dynamically loaded from the configuration file.

`PullRequestGenerator:` An event-driven protocol that generates pull requests to provider peers.

`PublishProtocol:` An event-driven protocol that runs publish events. A publish event is an event that a peer publishes or injects a publication into the network.

The `ShrackProtocol`, `PullRequestGenerator`, and `PublishProtocol` classes are an extension (implementation) of `EDProtocol`.

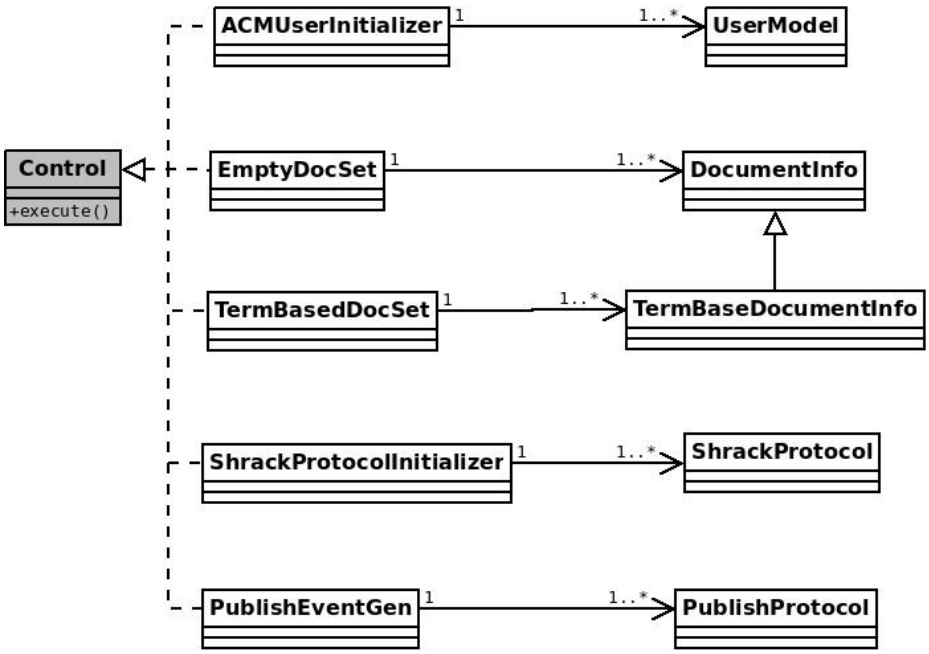


Figure D.2: ShrackSim initializer classes

D.1.2 `Control` Classes

There are two kinds of `Control` classes: (1) `Control` classes that are used for setting up the initial states of each protocol and the event queue, namely *initializer* classes, and (2) `Control` classes that are used for observing the Shrack behaviours during the simulation, namely *observer* classes. Both initializer and observer classes implement the PeerSim `Control` interface.

Initializer Classes

There are five main initializer classes for initializing ShrackSim components, called `ACMUserInitializer`, `EmptyDocSetInitializer`, `TermBasedDocSetInitializer`, `ShrackProtocolInitializer`, and `PublishEventGen`, as shown in Figure D.2. The initializer classes are called at the start of the simulation. Next, we provide the description of each class.

`ACMUserInitializer:` An initializer that initializes an artificial user (`UserModel`) for each peer. The `ACMUserInitializer` initializes `UserModel` from predefined input

files and is assigned to each peer (`ShrackNode`) in the network.

`EmptyDocSetInitializer:` An initializer that initializes a set of documents to be used during the simulation. The `EmptyDocSetInitializer` only keeps information of an identifier, a list of authors, and a list of publication groups of each publication. This information is sufficient for creating item-based peer profiles. The simulator does not need information of the document content.

`TermBasedDocSetInitializer:` A document set initializer that initializes a set of documents similar to `EmptyDocSetInitializer` with an additional information of term-based vector representation of documents. The `TermBasedDocSetInitializer` class is an extension of `EmptyDocSetInitializer` and is suitable for creating term-based peer profiles.

`ShrackProtocolInitializer:` An initializer that schedules to start up `ShrackProtocol` for each peer. The `ShrackProtocolInitializer` controls when each peer starts pulling Shrack messages from, or issuing a pull request to, its provider peers. In the current implementation, `ShrackProtocolInitializer` randomly schedules a peer to start issuing a pull request within a predefined cycle.

`PublishEventGen:` An initializer that schedules `PublishProtocol`. The `PublishEventGen` initializer schedules which peer publishes which documents and at which simulation cycle. In the current implementation, `PublishEventGen` schedules that documents are published in the system by the peer associated with the first author. The publication time follows a Poisson distribution with a predefined average publishing rate.

Observer Classes

The observer classes are an implementation of `Control` that is used to observe the status of `Protocol` and components in the network. The observer classes are run periodically at a predefined step. There are several observer classes provided by PeerSim, as well as several new observer classes introduced in ShrackSim. The four

commonly used observers are; `SlidingSlotRecvDocQuality`, `SCCDist`, `Clustering`, and `InDegreeObserver`. Next, we give the description of each observer class.

`SlidingSlotRecvDocQuality:` The main observer class that evaluates Shrack messages that peers receive. Five main evaluation metrics are used, namely *precision*, *recall*, *f-score*, *relevant pull delay* and *relevant path length*. The `SlidingSlotRecvDocQuality` observer is used to evaluate provider peer selection strategies in Shrack. It evaluates messages that peers receive that are published in the network during each observed time slot.

`SCCDist:` An observer class that measures the shortest path length (or distance) between two nodes in the largest strongly connected component of the network. The average distance between two nodes defines the characteristic path length of the network.

`Clustering:` An observer class that measures the clustering coefficient of nodes in the network.

`InDegreeObserver:` An observer class that measures the in-degree statistics of nodes in the network.

D.1.3 ShrackSim Interface and Abstract Classes

ShrackSim provides three main interfaces and one abstract class for users to develop and test different modules in Shrack, namely `KnownPeerProfile`, `LocalProfile`, `NeighbourSelector`, and `Filter`. At present, ShrackSim provides classes that support the prototype system as described in Chapter 4 (page 33). Figure D.3 shows the ShrackSim interfaces and abstract class, and their current implementation classes. The interfaces and the abstract class are highlighted with dark-gray text colour. The details of each class are described next.

KnownPeerProfile: An abstract class that manages statistics of each known peer and provides a method to update the profile of a known peer. Each known peer has

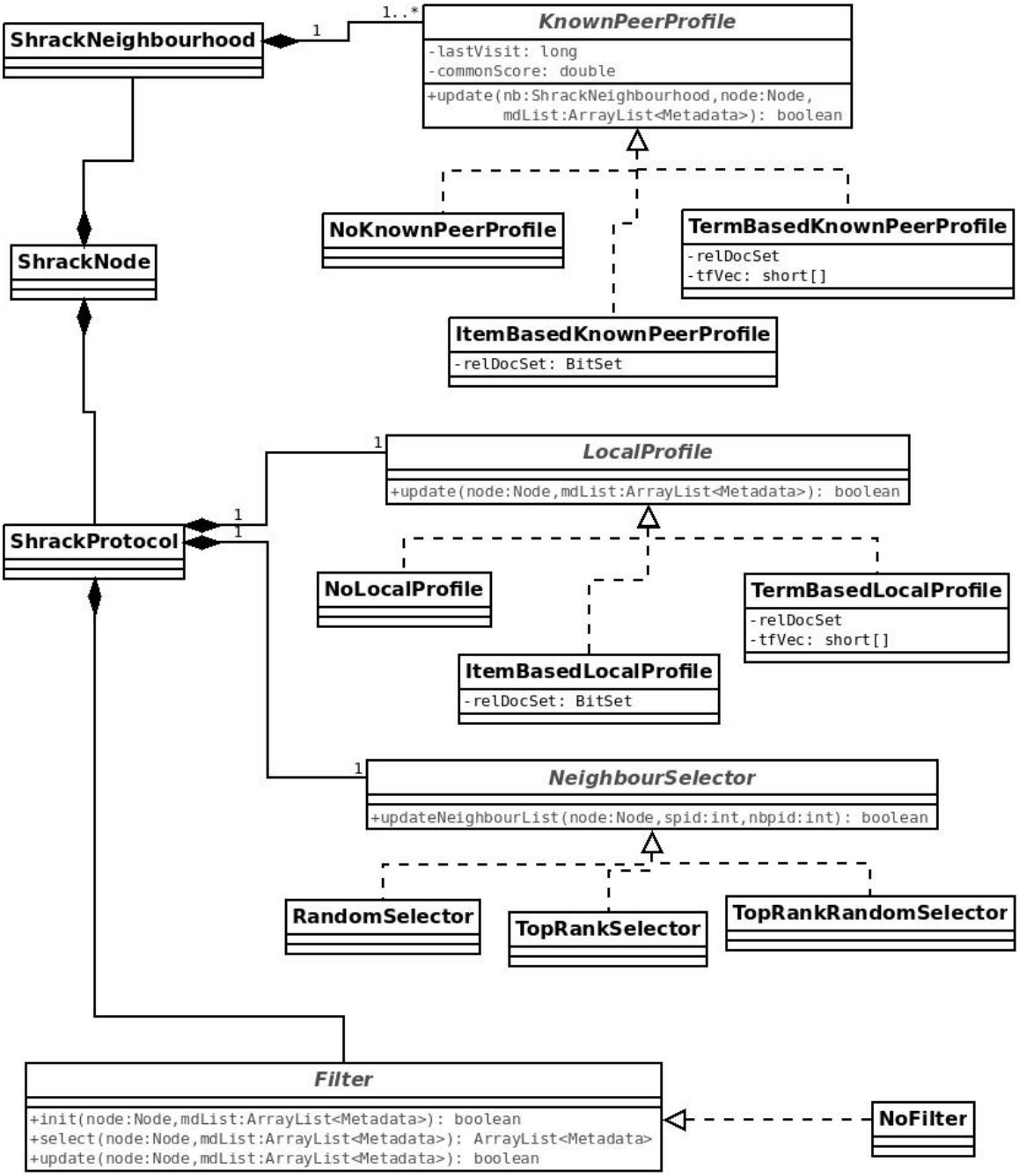


Figure D.3: ShrackSim interface and abstract classes

its own KnownPeerProfile object. Each ShrackNeighbourhood instance maintains a list of peers of which each local peers are aware, hence each ShrackNeighbourhood consists of several KnownPeerProfile instances. ShrackSim has an abstract method for a peer to update profiles of known peers according to a given set of Shrack messages. At present, ShrackSim provides three implementations of KnownPeerProfile: NoKnownPeerProfile, ItemBasedKnownPeerProfile, and TermBasedKnownPeerProfile. The NoKnownPeerProfile class does nothing. It serves as a empty container implementation for a configuration that does not require known peer profiles, i.e., when a peer randomly selects its provider peers. The ItemBasedKnownPeerProfile builds a known peer using information of relevant publication identifiers only. The TermBasedKnownPeerProfile builds a known peer using a term-weight vector representation. The ItemBasedKnownPeerProfile and TermBasedKnownPeerProfile implement an item-based known peer profile and a term-based known peer profile, respectively, as discussed in Section 4.2.5(page 45).

LocalProfile: An interface to manage a node's local profile. The LocalProfile interface provides a method to update a node's local profile. Classes implementing this interface are cloned to every nodes in the network, as part of ShrackProtocol. ShrackSim provides three implementations of LocalProfile named: NoLocalProfile, ItemBasedLocalProfile, and TermBasedLocalProfile. Similar to the implementations of KnownPeerProfile, NoLocalProfile is an empty container implementation and does nothing. The ItemBasedLocalProfile builds a local peer using information of relevant publication identifiers only, and the TermBasedLocalProfile builds a local peer using a term-weight vector representation.
The ItemBasedLocalProfile and TermBasedLocalProfile implement an item-based local profile and a term-based local profile, respectively, as discussed in Section 4.2.4 (page 43).

NeighbourSelector: An interface that provides a method for a peer to update its provider peers. Each peer has its own NeighbourSelector as part of ShrackProtocol. ShrackSim provides three implementations of NeighbourSelector named: TopRankSelector, RandomSelector, and TopRankRandomSelector implementing the

three provider peer selection strategies as discussed in Section 4.3 (page 49). The `TopRankSelector` implements the common interest strategy, the `RandomSelector` implements the random strategy, and the `TopRankRandomSelector` implements the hybrid strategy.

Filter: An interface that provides methods to test a filter module. The `Filter` provides three methods to initialize the filter, to filter messages, and to update the filter. At present, ShrackSim does not have an implementation of a real filter and only provides an empty container implementation called `NoFilter`.

D.2 The Configuration File

ShrackSim uses the PeerSim configuration file as an input parameter for running a simulation. The configuration file is a plain ASCII text file, composed of key-value pairs representing Java, java.util.Properties. Some of the key names refer to global properties and others refer to single component instances. Lines starting with the '#' character are treated as comments and are ignored. Each component has a name. In the case of protocols, each protocol's name is mapped to a numeric index called protocol ID, by the PeerSim engine. This index does not appear in the configuration file, but it is necessary for accessing protocols during a simulation. A component such as a protocol or a control is declared by the following syntax:

```
<protocol|init|control>.string_id [full_path_]classname
```

The full class path is optional. PeerSim can search in Java's `classpath` environment variable in order to find a class. If multiple classes share the same name (in distinct packages), then the full path is needed. The component parameters are defined by the following scheme:

```
<protocol|init|control>.string_id.parameter_name parameter_value
```

We can divide the configuration file into five sections:

1. Declaring constant variables that will be later used in the configuration file.

2. Setting up the simulation time and network.

3. Setting up protocols.

4. Initializing protocols and components.

5. Monitoring the simulation.

Next, we provide an example of a configuration file that configures ShrackSim to run `ShrackProtocol`, the Shrack information dissemination protocol, with the hybrid provider peer selection strategy using item-based peer profiles. The configuration file is divided into five sections as mention earlier.

Section 1: Declaring constant variables

```
 1 # shracksim test shrack config
 2
 3 # parameters of execution time
 4 # CYCLE defines the length of a cycle
 5 # CYCLES defines number of CYCLE to run the simulation
 6 # the simulation length = CYCLES*CYCLE+1
 7 CYCLE 1000
 8 CYCLES 6500
 9 PULL_INTV 20*CYCLE
10 OBS_STEP 200*CYCLE
11
12 # parameters of message transfer (cycles)
13 # this configuration ignore delay
14 MINDELAY 0
15 MAXDELAY 0
16
17 # drop is a probabiliy, 0 <= DROP <=1
18 # this configuration no message will be dropped
19 DROP 0
20
21 MAXNB 5
22 MAXHOP 8
```

Figure D.4: An example of a configuration file section 1: constant variable declaration

An example of a declaration part is shown in Figure D.4. Users can simply declare a variable and its value as a key-value pair, e.g., line 7 declares `CYCLE` to be equal to 1000. The configuration file can handle many types of expressions including common

```
24 random.seed 123456789
25
26 simulation.logtime OBS_STEP
27 simulation.endtime CYCLES*CYCLE+1
28
29 network.size 1000
30 network.node shracksim.core.ShrackNode
```

Figure D.5: An example of a configuration file section 2: setting up the simulation time and network

mathematical functions that are recognized by the Java Expression Parser (Jep) [1]. Expressions can be used anywhere instead of numeric values, for example, line 9 declares `PULL_INTV` to be equal to `20*CYCLE` which is 20,000. In this configuration, we defines that one cycle has 1,000 simulation clock units.

Section 2: Setting up the simulation time and network.

Figure D.5 defines the global simulation and network parameters. The `random.seed` defines the random seed of the simulation. The parameter `simulation.logtime` defines the interval of time at which the simulation prints the progress of simulation time on the standard error. In this configuration, we set simulation log time to be every 200 cycles. The parameter `simulation.endtime` tells the simulator when to stop. The parameter `network.size` defines the number of nodes in the network. The parameter `network.node` defines the type of nodes in the network, here defined as `ShrackNode`.

Section 3: Setting up protocols.

There are six protocols that needs to be configured to run ShrackSim. These protocols are: `ShrackNeighbourhood`, `PullRequestGenerator`, `ShrackProtocol`, `PublishProtocol`, `UniformRandomTransport`, and `UnreliableTransport` protocols. The last two protocols are required because `PullRequestGenerator` is built on top of them. Figure D.6 shows how to define protocols and their parameters. For example, we define `ShrackNeighbourhood` as a string ID "nb" (line 32). Then we use this

[1] http://www.singularsys.com/jep/index.html

```
protocol.nb ShrackNeighbourhood
protocol.nb.max MAXNB
protocol.nb.nbProfile ItemBasedKnownPeerProfile
protocol.nb.nbProfile.scoreType jaccard

protocol.reqGen PullRequestGenerator
protocol.reqGen.transport tr

protocol.shrack ShrackProtocol
protocol.shrack.localProfile ItemBasedLocalProfile
protocol.shrack.filter NoFilter
protocol.shrack.nbSelector TopRankRandomSelector
protocol.shrack.nbSelector.beta 0.01
protocol.shrack.nbSelector.seed random.seed+111
protocol.shrack.pullIntv PULL_INTV
protocol.shrack.maxHop MAXHOP
protocol.shrack.maxUpdate MAXHOP*PULL_INTV
protocol.shrack.neighbours nb
protocol.shrack.reqGen reqGen
protocol.shrack.transport tr

protocol.pub PublishProtocol
protocol.pub.shrack shrack

#no delay at this time MINDELAY = MAXDELAY = 0
protocol.urt UniformRandomTransport
protocol.urt.mindelay (MINDELAY*CYCLE)/100
protocol.urt.maxdelay (MAXDELAY*CYCLE)/100

#use unreliable Transport protocol but right now set DROP = 0,
#which means now the protocal is reliable no message DROP
protocol.tr UnreliableTransport
protocol.tr.transport urt
protocol.tr.drop DROP
```

Figure D.6: An example of a configuration file section 3: setting up protocols

```
67 init.um AcmUserInitializer
68 init.um.authorDocProfile acm_abstracts/top1000_authorDocIndex.txt
69 init.um.authorGroupProfile acm_abstracts/top1000_authorGroupIndex.txt
70
71 init.docSet EmptyDocSet
72 init.docSet.indexfile acm_abstracts/2007_2008a_2000_2008d_H33_f_Index.txt
73
74 init.rnd WireKOut
75 init.rnd.protocol nb
76 init.rnd.k MAXNB
77
78 init.startShrack ShrackProtocolInitializer
79 init.startShrack.bootTime PULL_INTV
80 init.startShrack.shrackProt shrack
81
82 init.pubGen PublishEventGen
83 init.pubGen.pubRate (1.0/(4*CYCLE))
84 init.pubGen.pubProt pub
85 init.pubGen.rSeed random.seed+777
86
87 include.init um docSet rnd startShrack pubGen
```

Figure D.7: An example of a configuration file section 4: initializing protocols and components

string ID to define its parameters, namely `MAXNB` (line 33), which defines the size of neighbourhood, `nbProfile` (line 34), which defines type of known peer profile to be `ItemBasedKnownPeerProfile`, and `nbProfile.scoreType` (line 35), which is a parameter of `ItemBasedKnownPeerProfile` defining a type of common-interest score to be used. Other protocols are defined with a similar manner. Details of which parameters are required to create each protocols are defined in the ShrackSim Java Documentation (JavaDoc).

Section 4: Initializing protocols and components.

Figure D.7 shows how to initialize protocols and components in ShrackSim. First, we need to define initializer classes and their parameters in the same way as defining protocols. Then, we need to explicitly specify which initializer classes we want to include in the simulation using the parameter `include.init` with a list of classes

```
89 control.netCluster Clustering
90 control.netCluster.protocol nb
91 control.netCluster.step OBS_STEP
92
93 control.slotObserver SlidingSlotRecvDocQuality
94 control.slotObserver.slotSize OBS_STEP
95 control.slotObserver.step PULL_INTV
96
97 control.sccDistance SCCDist
98 control.sccDistance.protocol nb
99 control.sccDistance.step OBS_STEP
```

Figure D.8: An example of a configuration file section 5: monitoring the simulation.

that we want to be included, as shown in line 87. The simulation will call the initializer classes in the same order as they are listed in the `include.init`. There are five initializer classes that we need to include. The first two classes are used to set up the user model and document set, which are defined in line 67-72. The last three classes are used to initialize three main protocols, namely `ShrackNeighbourhood` (string_id "nb"), `ShrackProtocol` (string_id "shrack"), and `PublishProtocol` (string_id "pub"). Line 67-85 presents how to initialize the three protocols using their associated initializer classes. Note that, `WireKOut` (line 74) is provided by PeerSim for initializing classes implementing the `Linkable` interface, which is a super class of `ShrackNeighbourhood`. The `WireKOut` class randomly connects each node with a predefined size of neighbourhood, defined by a parameter 'k' on line 76.

Section 5: Monitoring the simulation.

Finally, in section 5 of the configuration file, we define which observer classes we want to use to monitor the simulation. In this example, we include three observer classes. A special parameter "step" needs to be defined to specify the interval of cycles at which an observer is called. For example, line 91 defines that the `Clustering` observer is called every `OBS_STEP` (200) cycles, and line 95 defines that `SlidingSlotRecvDocQuality` observer will be called every `PULL_INTV` (20) cycles.

```
 1 Simulator: loading configuration
 2 ConfigProperties: File 20090808-example-r0-b001-jaccard-nb5.con loaded.
 3 Simulator: starting experiment 0 invoking peersim.edsim.EDSimulator
 4 Random seed: 123456789
 5 EDSimulator: resetting
 6 EDSimulator: running initializers
 7 - Running initializer init.um: class shracksim.initializer.AcmUserInitializer
 8 - Running initializer init.docSet: class shracksim.initializer.EmptyDocSet
 9 - Running initializer init.rnd: class peersim.dynamics.WireKOut
10 - Running initializer init.startShrack: class \\
           shracksim.initializer.ShrackProtocolInitializer
11 - Running initializer init.pubGen: class \\
           shracksim.initializer.PublishEventGen
12 EDSimulator: loaded controls [control.netCluster, control.sccDistance, \\
           control.slotObserver]
13 Current time: 0
14 Current time: 200000
15 Current time: 400000
16
17 .
18 .
19 .
20
21 Current time: 6000000
22 Current time: 6200000
23 Current time: 6400000
24 EDSimulator: queue is empty, quitting at time 6500000
```

Figure D.9: An example of a simulation status printed on the standard error.

D.3 Running and Evaluating Experiments

Users can run ShrackSim by invoking the ShrackSim class with a configuration file as a command line as following:

```
java shracksim.ShrackSim <configuration_file>
```

After running the simulation, the simulation status is printed on the standard error and the simulation results are printed on the standard output. The simulation output from running ShrackSim with the configuration file that is previously discussed (Figure D.4- D.8) is shown is Figure D.9 and D.10.

Figure D.9 shows the simulation status on the standard error. Lines 1-12 display status of the simulator when loading the configuration file, invoking the simulation engine, running initializers and loading controls (observers). Lines 13-23 show progress of the simulation time. Finally, line 24 shows the end of simulation.

An example of the simulation results is shown in Figure D.10. Lines 1 and 2 shows an initial statistic (at time 0) of the clustering coefficient of nodes and the shortest path length between nodes in the network, respectively. In general, the result of each observers is presented in the following format:

```
observer_id: min max #sample avg var #min #max
```

where

`observer_id`	is the observer identifier that is defined in the configuration file
`min`	is the minimum observed value
`max`	is the maximum observed value
`#sample`	is the number of the sample
`avg`	is the average of the observed value
`var`	is the variance of the observed value
`#min`	is the number of sample that are minimum values
`#max`	is the number of sample that are maximum values

For example, from Figure D.10 line 1, we can interpret that, initially, from 1,000 nodes, there are 891 nodes that have the minimum clustering coefficient of 0.0 and 3 nodes that have the maximum clustering coefficient of 0.1. On average, the network clustering coefficient is 0.01 with variance equal to 0.0. Note that, we show the statistic values in Figure D.10 with two decimal places only for the presentation.

Compared with other observers, the output of the `SlidingSlotRecvDocQuality` observer is presented differently indicating an observation time (Time), a publication

```
 1 control.netCluster: 0.0 0.1 1000 0.01 0.0 891 3
 2 control.sccDistance: 1.0 8.0 981090 4.38 0.83 4955 131 1
 3 Time: 20000 Slot Time: 200000 Slot No: 1 PubSize 3
 4 Time: 20000 Slot Time: 200000 Slot No: 1 Precision 0.0 1.0 20 0.35 0.24 13 7
 5 Time: 20000 Slot Time: 200000 Slot No: 1 Recall 0.0 1.0 827 0.01 0.01 820 3
 6 Time: 20000 Slot Time: 200000 Slot No: 1 F-score 0.0 1.0 14 0.381 0.181 7 3
 7
 8 Time: 40000 Slot Time: 200000 Slot No: 1 PubSize 7
 9 Time: 40000 Slot Time: 200000 Slot No: 1 Precision 0.0 1.0 121 0.47 0.25 63 56
10 Time: 40000 Slot Time: 200000 Slot No: 1 Recall 0.0 1.0 999 0.02 0.01 941 2
11 Time: 40000 Slot Time: 200000 Slot No: 1 F-score 0.0 1.0 121 0.24 0.07 63 1
12 .
13 .
14 .
15 Time: 180000 Slot Time: 200000 Slot No: 1 PubSize 50
16 Time: 180000 Slot Time: 200000 Slot No: 1 Precision 0.0 1.0 1000 0.51 0.06 21 21
17 Time: 180000 Slot Time: 200000 Slot No: 1 Recall 0.0 1.0 1000 0.50 0.02 21 8
18 Time: 180000 Slot Time: 200000 Slot No: 1 F-score 0.0 0.79 1000 0.48 0.02 21 1
19
20 control.netCluster: 0.0 0.5 1000 0.06 0.01 452 5
21 control.sccDistance: 1.0 11.0 888306 4.91 1.46 4715 14 1
22 Time: 200000 Slot Time: 200000 Slot No: 1 PubSize 56
23 .
24 .
25 .
26 control.netCluster: 0.0 0.95 1000 0.10 0.02 353 1
27 control.sccDistance: 1.0 14.0 871422 5.30 2.29 4670 12 1
28 Time: 400000 Slot Time: 200000 Slot No: 1 PubSize 108
29 Time: 400000 Slot Time: 200000 Slot No: 1 Precision 0.04 1.0 1000 0.56 0.05 1 5
30 Time: 400000 Slot Time: 200000 Slot No: 1 Recall 0.5 1.0 1000 0.84 0.01 5 17
31 Time: 400000 Slot Time: 200000 Slot No: 1 F-score 0.06 0.93 1000 0.65 0.03 1 1
32
33 Time: 400000 Slot Time: 400000 Slot No: 2 PubSize 52
34 Time: 400000 Slot Time: 400000 Slot No: 2 Precision 0.0 1.0 1000 0.68 0.05 2 54
35 Time: 400000 Slot Time: 400000 Slot No: 2 Recall 0.0 1.0 1000 0.74 0.02 2 17
36 Time: 400000 Slot Time: 400000 Slot No: 2 F-score 0.0 0.97 1000 0.66 0.03 2 1
37
38 Time: 420000 Slot Time: 200000 Slot No: 1 PubSize 108
39 .
40 .
41 .
42 control.netCluster: 0.0 0.85 1000 0.14 0.02 246 1
43 control.sccDistance: 1.0 14.0 878906 5.66 3.09 4690 168 1
44 Time: 6500000 Slot Time: 200000 Slot No: 1 PubSize 108
45 Time: 6500000 Slot Time: 200000 Slot No: 1 Precision 0.05 1.0 1000 0.56 0.04 1 4
46 Time: 6500000 Slot Time: 200000 Slot No: 1 Recall 0.59 1.0 1000 0.92 0.01 1 259
47 Time: 6500000 Slot Time: 200000 Slot No: 1 F-score 0.1 0.96 1000 0.67 0.03 1 1
48
49 Time: 6500000 Slot Time: 400000 Slot No: 2 PubSize 109
50 Time: 6500000 Slot Time: 400000 Slot No: 2 Precision 0.10 1.0 1000 0.66 0.05 1 31
51 Time: 6500000 Slot Time: 400000 Slot No: 2 Recall 0.44 1.0 1000 0.86 0.02 1 240
52 Time: 6500000 Slot Time: 400000 Slot No: 2 F-score 0.18 1.0 1000 0.72 0.03 1 1
53 .
54 .
55 .
56 Time: 6500000 Slot Time: 6600000 Slot No: 33 PubSize 22
57 Time: 6500000 Slot Time: 6600000 Slot No: 33 Precision 0.0 1.0 981 0.80 0.04 2 378
58 Time: 6500000 Slot Time: 6600000 Slot No: 33 Recall 0.0 1.0 1000 0.45 0.04 21 47
59 Time: 6500000 Slot Time: 6600000 Slot No: 33 F-score 0.0 1.0 981 0.56 0.03 2 33
```

Figure D.10: An example of a simulation result printed on the standard output.

slot time (Slot Time), a slot number (Slot No) followed by an observed statistics. Each observed statistics, except `PubSize`, is presented in the same format as previously defined. The `PubSize` presents the number of documents that are published in the observed publication time slot. The simulation periodically evaluates and presents the observed value as defined in the configuration file.

This chapter provides a general information of ShrackSim. We recommend those who want to use ShrackSim to read the details Java documentation. In addition, for those who want to modify ShrakSim, we recommend to study PeerSim documentation as well.