A Study on Efficient Search Approaches in Unstructured Peer-to-Peer Systems

Yu Wu

Dissertation submitted to

Graduate School of Information Science and Technology

Osaka University

January 2009

List of Related Publications

Journal Papers

1. Yu Wu, Taisuke Izumi, Fukuhito Ooshita, Hirotsugu Kakugawa, and Toshimitsu Masuzawa, "A message-efficient Peer-to-peer search protocol based on adaptive index dissemination", *IEICE Transactions on Information and Systems*, Vol.E92-D, No.2, Feb. 2009.

2. Yu Wu, Fukuhito Ooshita, Hirotsugu Kakugawa, and Toshimitsu Masuzawa, "Distributed construction protocols of probabilistic degree-weighted peer-to-peer overlays", *IEICE Transactions on Information and Systems*, (Conditional acceptance).

Conference Papers

3. Yu Wu, Taisuke Izumi, Fukuhito Ooshita, Hirotsugu Kakugawa, and Toshimitsu Masuzawa, "An Adaptive Randomized Searching protocol in Peer-to-peer systems based on Probabilistic Weak Quorum System", *Proceedings of the Eighth International Symposium on Stabilization, Safety, and Security of Distributed Systems*, Nov. 2006. (Brief Announcement)

4. Yu Wu, Taisuke Izumi, Fukuhito Ooshita, Hirotsugu Kakugawa, and Toshimitsu Masuzawa, "An Adaptive Randomized Searching protocol in Peer-to-peer systems", *Proceedings of The 22nd Annual ACM Symposium on Applied Computing*, pp 533-537, Mar. 2007.

Technical Reports

5. Yu Wu, Taisuke Izumi, Fukuhito Ooshita, Hirotsugu Kakugawa, and Toshimitsu Masuzawa, "An adaptive searching protocol in Peer-to-peer systems based on Probabilistic Weak Quorum System", *Technical Report of IPSJ*, 2006-AL-105, Vol.2006, No.30, pp 41-88, Mar. 2006.

6. Yu Wu, Fukuhito Ooshita, Hirotsugu Kakugawa, and Toshimitsu Masuzawa, "Building dynamic random peer-to-peer overlays", *Technical Report of IPSJ*, 2008-DPS-134, Vol.2008, No.21, pp 25-30, Mar. 2008.

List of Unrelated Publications

Conference Papers

7. Rikiya Hasegawa, Yu Wu, Fukuhito Ooshita, Hirotsugu Kakugawa, and Toshimitsu Masuzawa, "A resource replication protocol based on node density for mobile ad hoc networks", *Proceedings of the Eighth International Conference on Applications and Principles of Information Science*, Jan. 2009, (To appear).

Technical Reports

8. Masayuki Kobayashi, Yu WU, Fukuhito Ooshita, Hirotsugu Kakugawa, and Toshimitsu Masuzawa, "Object search using lane structure in mobile P2P systems", *Technical Report of IPSJ*, 2008-DPS-134, Vol.2008, No.21, pp 219-224, Mar. 2008 (in Japanese).

Abstract

Peer-to-Peer(P2P) systems, with prominent advantages such as scalability, robustness and low development cost, have developed quickly in recent years. P2P systems are Internet-based applications which consist of a number of autonomic user clients called peers. Different from traditional Client-Server systems, P2P systems do not have central servers which manage system resources such as shared contents and services. Therefore, the resource search problem, that is to find out a peer having the desired resource, is a fundamental problem which must be solved at first in most of the P2P applications.

Unstructured search protocols are widely adopted in modern P2P applications for their flexibility and robustness. A distinct feature of such so-called unstructured systems is that they randomly route query messages to unspecified peers e.g., by flooding or random walk. Although the random query dissemination seems not so efficient in large-scale systems, it is very practicable in heterogeneous system environments and flexible to system dynamics. Moreover, in practice, the performance of unstructured search protocols can be greatly improved by optimizing their search process to the statistical biases of the system environment e.g., the biases in peers' capacities and users' interests, etc. The environment-optimized design of unstructured search protocols has become a major topic of the research area.

The architecture of unstructured P2P systems contain four basic functional components: the overlay construction protocol, the message routing protocol, the search protocol and the workload allocation strategy. The overlay construction protocol organizes peers in a connected network. Based on the network, the routing protocol enables peers to communicate with each other. A search protocol is in the highest layer of the architecture which manages the search process such as query dissemination etc. The basic function of the workload allocation strategy is to prevent peers from overload. It also greatly affects the performance of the other components and the robustness of the whole system. The search performance of an unstructured P2P system is not only decided by the search protocol but also strongly affected by the conformance among these functional components. This dissertation presents an integrated solution for the resource search problem in unstructured P2P systems. The solution contains three main contributions which are

summarized as follows.

First, we study the overlay construction problem. We propose distributed protocols for constructing degree-weighted overlay networks. In such a network, each peer's in-degree (i.e., the number of incoming links) is proportional to its weight which is a local parameter of the peer. The objective of the network is to enable upper-layer applications to control the in-degree distribution of peers by setting appropriate rules to determine the value of each peer's weight. Then, they can control the number of messages routed to each peer in the systems which adopt flooding or random walk as their routing methods. The proposed network has good connectivity and self-organization ability. It also has a compact structure (e.g., small network diameter) that is favorable for achieving efficient gossip-based message dissemination such as flooding and random walk.

Then, we investigate the search performance of unstructured P2P systems from the view point of workload allocation. We show that the search performance can be improved by concentrating system search workload to a part of peers. We propose an optimized workload allocation strategy which fully utilizes the most powerful peers' capacities to maximize the search performance. The proposed strategy and other popular workload allocation strategies in traditional P2P applications are realized by distributed workload management protocols based on the degree-weighted overlay network we proposed . Benefiting from the good conformance with the network construction protocol, these protocols incur very low additional communication overhead.

At last, we propose a message-efficient search protocol based on adaptive index dissemination. In the protocol, peers disseminate their resources' indices (i.e., location informations) to some other peers in advance so that the resources can be found easier. The protocol minimizes the number of messages used for both search and index dissemination by optimizing the number of indices disseminated for each shared resource based on its popularity i.e., the frequency it is searched. Moreover, it optimizes the scheme of disseminating a given number of indices under *Churn* (i.e., peers continually join and leave) to minimize the negative impact caused by the loss of indices when peers leave. The protocol works in a completely distributed manner without any global knowledges such as resources' popularities and the number of peers in the system. Moreover, it can adapt to the changes of the system environment.

Our solution optimizes the search approach based on the biases in peers' capacities and resources' popularities to improve the search performance. The effectiveness of the solution is theoretically guaranteed and proved by simulation. It also has excellent feasibility for heterogeneous P2P system environments and can be applied in most of the P2P applications including P2P file sharing systems, P2P voice applications, grid computing etc. We hope this work can progress the formulation of the design of unstructured P2P systems.

Contents

Chapter 1

Introduction

1.1 Background

Peer-to-Peer(P2P) systems, with prominent advantages such as scalability, robustness and low development cost, have developed quickly in recent years [1]. P2P systems are Internet-based applications which consist of a number of autonomic user clients called peers. Different from traditional Client-Server systems, P2P systems do not have central servers which manage system resources such as shared contents and services. Therefore, the resource search problem, that is to find out a peer having the desired resource, is a fundamental problem which must be solved at first in most of the P2P applications.

The P2P system environment usually has the following characteristic features.

- **Large scale.** A P2P system may contain up to millions of peers.
- **Heterogeneous.** The statistical features of P2P systems are usually highly biased. Large biases in peers' capacities, resources' popularities (i.e., the frequency to be searched), connect time etc. have been observed by previous measurement studies.
- **Highly dynamic.** The system environment (e.g., resources' popularities etc.) changes over time. A distinct feature of P2P systems is that they suffer *Churn* i.e., peers continually join and leave.

These features make the design of P2P search protocols very difficult. But on the other hand, they also provide possibilities to improve the search performance by optimizing the search approach based on them. The environment-optimized design of unstructured search protocols has become a major topic of the research area.

Some early P2P systems (e.g., Napster, BitTorrent etc.) adopt the centralized search method. They register all shared resources in some broker servers so that a peer can easily obtain the location of its desired resource by accessing those broker servers. However, this approach is not

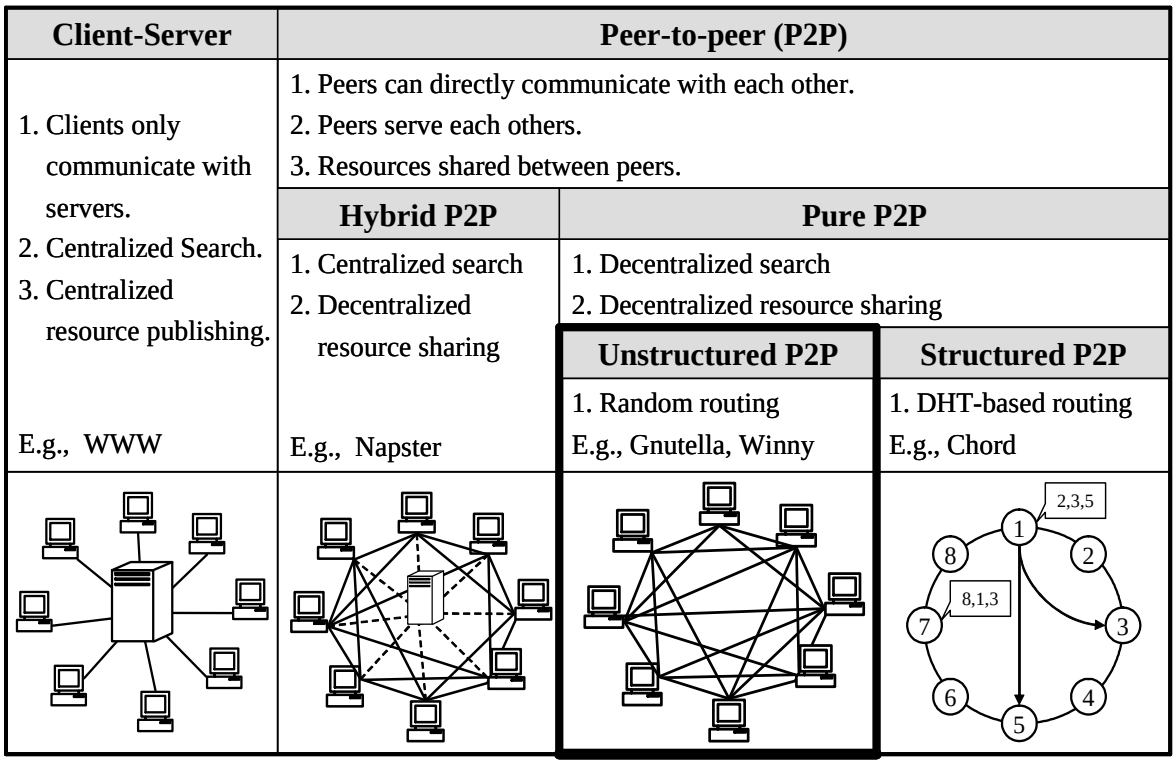


Figure 1.1: Summary of the features of Client-Server and Peer-to-peer systems.

scalable when the number of peers and shared resources increase. Such systems are called hybrid P2P systems because of their centralized search and decentralized resource sharing features. Pure P2P systems (e.g., Gnutella, Winny, Skype etc.) adopt the decentralized search method in which all peers are responsible to take part in the search process. They attract more attentions because of their scalability and low hardware investment cost.

The pure P2P systems can be classified into two distinct types according to their search approaches: the structured systems and unstructured systems [2][3].

Most of the structured systems are based on the distributed hash table (DHT) technique [4][5][6]. They organize peers in predefined topologies and place resources (or their location informations) in specified peers. Structured search protocols can efficiently route search queries to peers having the target resources within a few hops even in large-scale systems. However, they incur high overhead to maintain a tight network structure in dynamic P2P systems under Churn. Moreover, because their network topologies are decided in advance, they are lack of flexibility to the heterogeneity of the system environment. Furthermore, the DHT-based search has a critical disadvantage that they do not support pure text search i.e., resources are identified by hash values. The structured search protocols are hardly adopted in real P2P applications.

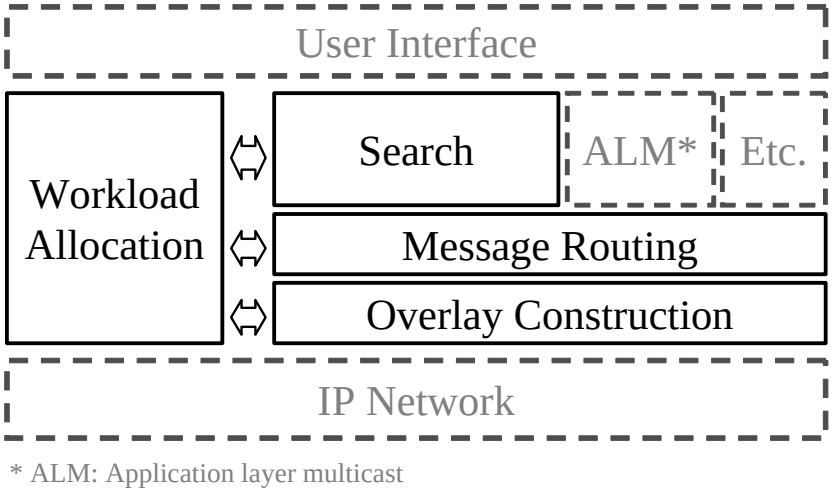


Figure 1.2: A basic architecture of unstructured P2P systems.

Unstructured systems are widely adopted in modern P2P applications including file sharing systems, voice applications etc., for their flexibility and robustness [7][8][9][10]. The unstructured search protocols usually work on any connected network topologies and do not need to relocate resources. Therefore, such systems incur very low maintenance cost when peers join and leave. A distinct feature of such search approaches is that they randomly route query messages to unspecified peers e.g., by flooding or random walk. Although the random query dissemination seems not so efficient in large-scale systems, it is very practicable in heterogeneous system environments and flexible to system dynamics. Moreover, in practice, the performance of unstructured search approaches can be greatly improved by optimizing them to the statistical biases of the system environment such as the biases in peers' capacities and resources' popularities, etc. Those optimized search protocols in many cases can achieve good search performance close to the structured ones.

1.2 System Architecture

This dissertation focuses on the search problem in unstructured P2P systems. The basic architecture of unstructured P2P systems is shown in Figure 1.2. It contains four functional components.

- **Overlay Construction.**
  A distributed overlay construction protocol is in the lowest layer of the architecture. It maintains an overlay network upon the IP-based Internet which keeps peers connected with each other. In such a network, a peer v_i can directly send messages to any other peer v_j

as long as v_i knows the IP address of v_j. Therefore, a P2P overlay is free to form any topologies regardless of the structure of the underlay physical networks. Considering the P2P systems are usually large-scale and highly dynamic, an efficient P2P overlay network is expected to have the following three properties. The first property is the *connectivity*. A network should be connected with a high probability to enable all-to-all information exchanging among peers. The second property is the *compactness*. A network should have a small diameter to achieve efficient gossip-based communications such as broadcasting, flooding and random walk in large-scale systems. The third property is the *self-organization ability*. A network should be able to converge to the expected topologies from any initially connected topologies. This property is important to keep the network topology under Churn.

- **Message Routing.**
  A message routing protocol enables peers to communicate with each other. A distinct feature of the unstructured P2P systems is that they adopt random message routing protocols such as flooding and random walk. Such routing methods do not incur additional control overhead for routing messages. They work on any connected network topologies so that they do not require the under-layer network construction protocols to provide specified topologies. This property makes the system be highly practicable in heterogeneous application environments and flexible to system dynamics.

- **Search.**
  A search protocol is in the highest layer of the architecture upon the overlay construction and message routing protocols. The basic function of a search protocol is to manage the query dissemination process when a peer is searching for some desired resources. Advanced search protocols also publicize resources by replicating them or disseminating their indices (i.e., location informations) to some other peers so that the resources can be found easier. An important statistical feature of the P2P resource sharing is the highly biased popularity distribution of shared resources i.e., a small part of resources are searched very frequently but most of the others are rarely searched. Efficient search protocols optimize the search and resource management schemes based on the popularity of each shared resource. They disseminate more indices (or replicas) for popular resources to reduce the total overhead related to those resources which includes the search cost and the publicizing cost. Another important factor which affects the search performance is the Churn. Since disseminated replicas and indices disappear when their holders leave the system, an efficient search protocol also needs to minimize the negative impacts caused by the loss of replicas or

indices.

- **Workload allocation.**
  The workload allocation strategy greatly affects the performance of the other functional components and the robustness of the whole system. In pure P2P systems, all peers are responsible to serve other peers. However, the amount of workload a peer can bear is limited by its physical capacity such as bandwidth, storage, CPU power etc. Therefore, reliable workload allocation strategies are necessary for preventing peers from overload. In addition, many measurement studies show that the distribution of peers' capacities is highly biased e.g., following power-law distributions. A reasonable workload allocation strategy is the capacity-aware allocation that allocates more workload to powerful peers than powerless ones. Moreover, an efficient workload allocation strategy can effectively utilize the powerful peers to improve the search performance of the system.

The search performance of unstructured P2P systems are not only decided by the search protocol but also strongly affected by the conformance among these functional components. Therefore, an integrated design is necessary for achieving high search performance in such systems.

1.3 Overview of this Dissertation

This dissertation aims to find an integrated solution for the resource search problem in unstructured P2P systems which adopt flooding or random walk as their routing methods. The subjects of this dissertation are related to the rest three functional components respectively. In this section, we give an overview of this dissertation.

1.3.1 Overlay construction

In Chapter 3, we study the overlay construction problem. We propose distributed protocols to construct degree-weighted networks in which the in-degree (i.e., the number of incoming links) of each peer is proportional to its *weight*, which is a local parameter of the peer. The objective of the network is to enable upper-layer applications to control the in-degree distribution of peers by setting appropriate rules to decide the value of each peer's weight. Then, they can control the number of messages routed to each peer in the systems which adopt flooding or random walk as their routing methods.

The basic network model is the out-regular directed network in which all peers have the same number of outgoing links. The model can satisfy our purpose because the number of those

randomly disseminated messages routed to a peer mainly depends on its in-degree but hardly affected by its out-degree (i.e., the number of outgoing links). In addition, compared with the traditional undirected network model, the out-regular directed network requires much less links for constructing networks with highly biased in-degree distributions.

The objective network is defined by the *Probabilistic Weighted d-out-regular Directed Network* (PWDN), where d is the predefined out-degree for all peers. The PWDN is a probabilistic space on the universe of the d-out-regular directed networks. A certain network randomly appears with the probability defined by the PWDN and the expected in-degree of each peer is proportional to its weight. The PWDN is constructed by periodically exchanging links between peers. A distributed framework, which includes 81candidate protocols by combination of link exchange rules, is presented and evaluated by simulation. The simulation result shows that two protocols can generate the networks having similar aspects with the PWDN.

1.3.2 Workload allocation

In Chapter 4, we investigate the search performance of unstructured P2P systems from the viewpoint of workload allocation.

We first explain why the search performance can be improved by biased workload allocation under a simple probabilistic search model. Then, we study the workload allocation strategies of traditional unstructured P2P systems and classify them into four distinct types: *Uniform*, *Capacity-Proportional*, *Fixedly-Layered* and *Adaptive-Layered*. Taking the advantages of both the Capacity-Proportional and the Adaptive-Layered types, we propose a novel strategy, the *Adaptively-Layered* & *Capacity-Proportional allocation* (ALCP), which has the following properties. (1) The basic network model is a layered (super peer) network in which a number of the most powerful peers are selected to be super peers which serve other peers. (2) The number of super peers is adjusted as less as possible based on the total workload of the system. (3) The workload allocated to each super peer is proportional to its capacity.

The ALCP is the most biased workload allocation under the restriction that peers do not overload. It requires the workload allocated to each super peer to be proportional to its capacity. We adopt the PWDN to achieve this requirement by setting each super peer's weight to its capacity. We also present a set of distributed workload management protocols to achieve the ALCP and other traditional workload allocation strategies. These protocols select the most powerful peers to be super peers and adaptively adjust the number of super peers based on the total workload of the system. They only use local workload situation informations of peers so that the additional communication cost incurred is very small. The simulation results show that ALCP has significantly higher search performance than traditional approaches. Moreover, it well

prevents peers from overload even in heavy-load system environments.

1.3.3 A Message-efficient search protocol

In Chapter 5, we propose a message-efficient search protocol. The protocol is an index-dissemination-based search protocol which disseminates resource's indices (i.e., location informations) to make them easier to find. The objective is to find out the optimal index dissemination scheme that can minimize the total message cost used for both index dissemination and searching.

The problem is firstly investigated by theoretical approaches. We analyze the system under an uniform-random access model that each peer disseminates messages to peers selected uniformly at random. We also introduce the Churn model that peers join and leave frequently. Since indices disappear when their holders leave the system, peers have to disseminate the indices of their resources periodically. The optimal index dissemination problem consists of two subproblems: The first one is to find the optimal scheduling for disseminating a given number of indices. We propose the *Stream Method* that is to averagely disseminate the same number of indices in each time unit. It minimizes the expected search cost with a given number of indices. The second one is how many indices of a resource should be disseminated. We show that the communication cost of a resource is minimized when its index dissemination cost equals to its search cost, what we called *Equal Rule* in this dissertation. The *Equal Rule* holds no matter how frequently a resource is searched. Based on the *Stream Method* and *Equal Rule*, we obtain the optimal index number for each resource and the lower bound of the total message cost related to it.

Then, we propose a distributed protocol to realize the index dissemination scheme in a self-adaptive manner. The protocol does not need any global knowledges such as the number of peers and resources' popularities. It yields almost no additional message cost to achieve the self-adaptive feature.

Chapter 2

Preliminaries

2.1 System Model and Notations

Throughout this dissertation, we adopt the discrete time model. Continuous time is divided into a series of discrete time intervals of the same length. Each time interval, which is called a time unit, is relatively-long (e.g., several minutes) so that the delay of delivering a short message can be ignored. Notice that the time model is only used to simplify the system description. We do not require peers to synchronize or be aware of the global clock.

A P2P network is a directed network $D(V, E)$ which consists of a set of independent peers $V = \{v_1, v_2, ..., v_n\}$, $n = |V|$, and a set of directed links $E \subseteq \{e_{i,j} \mid v_i, v_j \in V,\ i \neq j\}$ where $e_{i,j}$ is a directed link from v_i to v_j. We adopt the simple network model in which no self-loops or multiple links are contained. The P2P network is an overlay network in which a peer v_i can directly send messages to another peer v_j as long as v_i knows the network address (e.g., IP address) of v_j. This implies the entity of $e_{i,j}$ is an entry of v_j, denoted by et_j, stored in v_i. The entry of a peer contains its network address and some other information of it for upper-layer application usages.

If there exists a link $e_{i,j} \in E$, v_i is called the *in-neighbour* of v_j and v_j is called the *out-neighbour* of v_i. The *out-view*, denoted by $view_i^+$, of v_i is defined by the set of outgoing links of v_i. The *out-degree* of v_i is denoted by $\Delta_i^+ = |view_i^+|$. Similarly, we define the *in-view* of v_i, denoted by $view_i^-$, by the set of its incoming links. The *in-degree* of v_i is denoted by $\Delta_i^- = |view_i^-|$. Obviously, $|E| = \sum_{v_i \in V} \Delta_i^+ = \sum_{v_i \in V} \Delta_i^-$. Notice that since the in-view of a peer is a set of the peer's entries stored in other peers, the peer knows neither the in-view nor the in-degree of itself. We say a network is *strongly-connected* if for all pairs of peers, there exists a directed path in $D(V, E)$ and *weakly-connected* if there exists a path neglecting the direction of the links for all pairs of peers.

Table 2.1: Symbols of the system model.

$D(V,E)$	$=(V,E)$, a directed network.
V	$=\{v_1, v_2, ..., v_n\}$, a set of peers.
n	$=\|V\|$, the number of peers.
E	$\subseteq \{e_{i,j} \mid v_i, v_j \in V,\ i \neq j\}$, a set of links.
$e_{i,j}$	a directed link from v_i to v_j.
et_j	the entry of v_j, the entity of $e_{i,j}$.
$view_i^+$	the set of outgoing links of v_i .
$view_i^-$	the set of incoming links of v_i .
Δ_i^+	$=\|view_i^+\|$, the out-degree of v_i.
Δ_i^-	$=\|view_i^-\|$, the in-degree of v_i.

There are some shared resources in the system which can be some replicable data items such as documents, media files in file sharing systems [9][7][8] or some untransferable entities and services such as terminals (i.e., the user clients themselves) in IP telephone applications [11], computing capacities in grid computing systems [12] etc. For simple description, we call those shared resources by a unified name *objects* in the following of this dissertation. The *index* of an object is an entry of it which contains its identification information, the network address of its owner and some other informations for advanced functions. Since a P2P network is an overlay network, a peer can access an object as long as it has the index of the object.

2.2 A Search Model

Below, we introduce the index-dissemination-based search model which is used throughout this dissertation [13]. The model abstracts the implementation details but puts emphasis on the backbone principles of their probabilistic search methods.

In this model, the search process consists of two phases: the proactive *index phase* and the event-driven *search phase*. The detailed description is as follows.

- **Index Phase:**
  Each peer randomly disseminates the indices of their objects to some other peers by flooding or random walk. Each index has a predefined time-to-live (TTL) value which is initially set to T_I. The TTL value of each disseminated index decreases by one in every time unit. An index is deleted by its holder when its lifetime is expired. To keep the number of indices in the system, indices are periodically re-disseminated in a cycle of T_I time units.

- **Search Phase**:
  The search phase is activated when a search query is generated. In this phase, the searcher randomly disseminates the search query to some other peers. If a query hits a peer that has the index of the target object, the query succeeds.

Notice that although the limited lifetime of indices may cause some available indices be deleted, it is favorable for fault tolerance because bad indices (i.e., indices pointing to some disappeared objects that have been deleted or left with their owners) can only stay in the system before their lifetimes expired. Moreover, from the viewpoint of load balance, it also prevent old peers, which have joined the system for long time, storing too much indices.

The index-dissemination-based search has become the major search approach of most of the modern unstructured P2P applications, such as Winny, Skype, Gnutella etc.[14][15][13][16]. Some file sharing systems replicate objects themselves to improve the search performance [2][3]. Although the replication can speed up the download speed in P2P file sharing systems on the same time, the effect of disseminating a replica is almost the same as an index in searching but that incurs much higher communication and storage cost. Another restriction of the replication is that it can not be applied for untransferable objects. This dissertation focuses on the index-dissemination-based search approach.

Chapter 3

Overlay Construction

Unstructured overlay networks are widely adopted in large-scale and heterogeneous peer-to-peer(P2P) systems for their scalability and flexibility [7][8][9]. Many P2P systems aim to construct uniform-random networks in which peers have nearly uniform degrees [17][18][13][16][19]. Then they can uniformly distribute the messages routed by flooding or random walk over all peers, which is called random sampling [20]. The random sampling implies the search workload is uniformly allocated to each peer. However, it is not reasonable in case peers have different capacities [21][22][23][24].

The objective of this chapter is to build a *Probabilistic Weighted d-out-regular Directed Network* (PWDN) in which the expected in-degree of each peer is proportional to its *weight* which is a local parameter of the peer. By adjusting its weight, a peer can control the number of randomly disseminated messages routed to it. In addition, in order to bound the construction overhead for highly biased networks [25][26], we restrict all peers to have the same number of outgoing links. The objective network is constructed by local topology transformations that peers periodically exchange outgoing links with each other. We present a distributed framework which includes 81 different candidate protocols by combination of link exchange rules. The simulation result shows that two protocols can generate networks having similar aspects with the PWDN. They are also proved to be scalable and self-organizing that is very suitable for P2P system environments which are usually large-scale and highly dynamic.

This chapter is organized as follows. In Section 3.1, we give the definition of the objective network. In Section 3.2, we present a distributed framework to realize the objective network. In Section 3.3, we evaluate the framework by simulation. In Section 3.4, we discuss related works. Finally in Section 3.5, we give concluding remarks.

3.1 The Objective Network

In this section we define the objective network PWDN and introduce some properties of it. The PWDN is a probabilistic space on the universe of the d-out-regular directed networks. A certain network randomly appears with a probability defined by it.

The basic network model of the PWDN is an out-regular directed network in which each peer has the same number of outgoing links i.e., $\forall v_i \in V$, $\Delta_i^+ = d$ and the in-degree of each peers is proportional to its weight. The out-regular directed network model requires much less overhead to construct networks with highly biased in-degree distributions than traditional undirected network models [20][24]. That is because in undirected networks, low-weight peers also need to establish enough links in order to keep the network connected. Therefore, a high-weight peer has to establish many links in order to keep the same capacity-degree ratio as those powerless peers. For example, many studies show that in real P2P systems, the most powerful peer's capacity may be up to 10000 times higher than the most powerless peer's [25]. If one requires each peer's in-degree be proportional to its capacity, those powerful peers must maintain a huge number of links and the network construction overhead becomes terribly high. In out-regular directed networks, the expected in-degree of a powerless peer can be very low because it has enough outgoing links to keep the network weakly-connected. Therefore, powerful peers need much less incoming links to keep the capacity-degree ratio and thus highly-biased networks can be constructed with a reasonable overhead. Moreover, overlay networks are directed networks by its nature so that an undirected link is in fact two directed links in practice and the maintenance cost is double.

In addition, if an application must be executed in undirected networks, one can easily turn a directed network to an undirected one by letting each peer send entries to its out-neighbours. Of course, each peer should remember the original direction of each link so that the in-degree distribution can be maintained.

3.1.1 Definition

Below, we give the formal definition of the PWDN.

Definition 3.1.1 Weight vector.
The *weight vector* of V is defined by $\bar{w} = (w_1, w_2, ..., w_n)$ where w_i (≥ 0) is the *weight* of v_i that satisfies $w_i \ll \sum_{x=1}^{n} w_x$.

Definition 3.1.2 d-out-regular directed network.
The d-out-regular directed network $D_d^+(V, E)$ is defined by a set of peers $V = \{v_1, ..., v_n\}$ and a set of directed links $E \subseteq \{e_{i,j} \mid v_i, v_j \in V,\ i \neq j\}$ that for any $v_i \in V$, $\Delta_i^+ = d$.

Table 3.1: Symbols of the objective network.

$D(V,E)$	$=(V,E)$, a directed network.
V	$=\{v_1, v_2, ..., v_n\}$, a set of peers.
E	$\subseteq \{e_{i,j} \mid v_i, v_j \in V,\ i \neq j\}$, a set of links.
w_i	The weight of v_i.
$\bar{w}$	$=(w_1, w_2, ..., w_n)$, a weight vector of V.
$view_i^+$	The set of outgoing links of v_i .
$view_i^-$	The set of incoming links of v_i .
Δ_i^+	$=\lvert view_i^+ \rvert$, the out-degree of v_i.
Δ_i^-	$=\lvert view_i^- \rvert$, the in-degree of v_i.
δ_i^-	$=E[\Delta_i^-]$, the expectation of Δ_i^-.
$D_d^+(V,E)$	A d-out regular directed network.
d	The predefined out-degree of all peers.
$\mathcal{G}(V)$	$=\{g_1, ..., g_m\}$, the universe of $D_d^+(V,E)$.
m	$=\lvert\mathcal{G}(V)\rvert$, the size of $\mathcal{G}(V)$.
$\bar{\pi}$	$=(\pi_1, ..., \pi_m)$, a probability vector of $\mathcal{G}(V)$.
$\langle \mathcal{G}(V), \bar{\pi} \rangle$	A probability space of $D_d^+(V,E)$.
$p_{i,j}$	The probability $e_{i,j}$ appears in $\langle G(V), \bar{\pi} \rangle$.

The networks to be discussed in the following of this dissertation are based on this model. For short, we call the d-out-regular directed network simply by 'network'.

Definition 3.1.3 Universe of $D_d^+(V,E)$.
The universe $\mathcal{G}(V)$ of $D_d^+(V,E)$ is the set of all possible d-out-regular directed networks with the peer set V, that is $\mathcal{G}(V)=\{g_1, ..., g_m\}$ where $m=\lvert\mathcal{G}(V)\rvert$ and g_x $(1 \leq x \leq m)$ denotes each certain network.

Since a peer's out-view can be any d peers selected from the other $n-1$ peers, a combination of $\binom{n-1}{d}$ selections are available for each peer. Thus, the size of $\mathcal{G}(V)$ is $m=\binom{n-1}{d}^n$.

Definition 3.1.4 Probability vector.
A probability vector $\bar{\pi}$ of $\mathcal{G}(V)=\{g_1, ..., g_m\}$ is defined by $\bar{\pi}=(\pi_1, ..., \pi_m)$ such that $\sum_{x=1}^{m} \pi_x = 1$, where π_x $(1 \leq x \leq m)$ indicates the occurrence probability of a certain network g_x.

Lemma 3.1.1
The probability that a link $e_{i,j}$ appears is

$$p_{i,j} = \sum_{x=1}^{m} \pi_x f(e_{i,j}, g_x), \tag{3.1}$$

where

$$f(e_{i,j}, g_x) = \begin{cases} 1, & \text{if } e_{i,j} \text{ exists in } g_x \\ 0, & \text{otherwise.} \end{cases}$$

Proof: The lemma proves itself. □

Definition 3.1.5 PWDN.
A *Probabilistic Weighted d-out-regular Directed Network* with the weight vector $\bar{w}$ is a probability space $\langle \mathcal{G}(V), \bar{\pi} \rangle$ that satisfies, for each pair of peers v_i and v_j $(i \neq j)$,

$$p_{i,j} = \frac{dw_j}{\sum_{k=1}^{n} w_k - w_i}, \tag{3.2}$$

3.1.2 Properties

The PWDN has the following two properties.

Property 1
The expected in-degree of each peer $v_i \in V$ is $\delta_i^- \approx ndw_i / \sum_{x=1}^{n} w_x$.

Proof: By letting $\Delta_i^-(g_x)$ be the in-degree of v_i in g_x, we have

$$\begin{aligned} \delta_i^- &= \sum_{x=1}^{m} \pi_x \Delta_i^-(g_x) \\ &= \sum_{x=1}^{m} \pi_x \sum_{j=1}^{n} f(e_{j,i}, g_x) \\ &= \sum_{j=1}^{n} \sum_{x=1}^{m} \pi_x f(e_{j,i}, g_x) \\ &= \sum_{j=1}^{n} \frac{dw_i}{\sum_{k=1}^{n} w_k - w_j} \\ &\approx \frac{ndw_i}{\sum_{k=1}^{n} w_k}. \end{aligned}$$

□

Property 1 is the main purpose of this work. It implies for any two peers, the ratio of their in-degrees equals to the ratio of their weights i.e., $\forall v_i, v_j \in V, \delta_i^- / \delta_j^- \approx w_i / w_j$. Notice that the in-degree of a peer is automatically adjusted by the network adapting to all peers' weights, the peer dose not need to (cannot) decide its in-degree manually. This property is reasonable because the ratio of different peers' in-degrees is much more important than their exact in-degrees especially from the view point of workload allocation.

Property 2

For each peer v_i, other peers $v_j \in V, i \neq j$ have nearly the same probability to be its in-neighbour.

Proof: By Equality 3.2 and 3.1, we have

$$\begin{aligned} p_{j,i} &= \sum_{x=1}^{m} \pi_x f(e_{j,i}, g_x) \\ &= \frac{dw_i}{\sum_{k=1}^{n} w_k - w_j} \\ &\approx \frac{dw_i}{\sum_{k=1}^{n} w_k}, \end{aligned}$$

which does not depend on v_j. □

The Property 2 implies that the in-view of each peer holds randomness. That is, each peer is possible to be adjacent from other peers with a same probability unless its weight is 0. This property makes the network possible to achieve fair workload allocation even the amount of tasks generated by each peer is not uniform. It is also favorable for keeping the network topology compact [19].

3.2 A Distributed Framework for Overlay Construction

3.2.1 General description

The objective network is constructed by local topology transformations that peers periodically exchange outgoing links with each other. The entity of each link is the entry of the destination peer so that the link exchange in practice is that a peer sends replicas of the entries it holds to other peers. We divide the link exchange process to several independent operations and investigate the possible options of each operation. The proposed framework includes all of the possible combinations of those options. A wide range of traditional construction protocols, including both experimental and theoretical ones, can fit into this framework [27][20].

We adopt two simple rules to make high-weight peers obtain more incoming links and keep peers' in-degrees stable i.e., prevent the network from being a 'rich get richer' network such as the random growing networks [28]. First, a link incident to a peer having a higher weight is replicated with a higher probability during the link exchange process. Second, if the number of links incident to a peer increases, the probability of replicating such links is decreased. The second rule gives negative feedback to the first one so that the in-degree of each peer can stabilize. We introduce a new parameter, called heft, which indicates the priority a link to be replicated. The heft of an entry et_j of v_j (i.e., a link $e_{i,j}$ if v_i holds et_j) is denoted by $et_j.h$). Based on

the hefts of links (or say, entries), we propose distributed link exchange protocols without any global knowledges of peers' weights and in-degrees. In short, the initial heft of a newly created link et_j is set to the w_j and once a link is replicated and exchanged with other peers, its heft is decreased by a half.

3.2.2 Operation details

The framework is a set of protocols of how peers exchange links with each other. Each peer, denoted by v_i, periodically executes a 4-operation processes including (1) Target Selection, (2) Seed Planting, (3) View Merging, and (4) View Selection operations. Each operation has three options that can be selected independently. All peers in the system use a same combination of the options of each operation. Detailed descriptions of the operations from the viewpoint of peer v_i are as follows. For simple description, we assume that the links stored in each peer's out-view are sorted by the decreasing order of their hefts.

(1) Target selection: Peer v_i selects a link et_j from $view_i^+$. The peer v_j is decided to be the target peer to exchange links with. There are three different options of how to select a link from its out-view based on the heft of each link.

- **Random:** A link is selected from $view_i^+$ uniformly at random regardless of its heft.
- **Head:** The first link in $view_i^+$ (with the highest heft) is selected.
- **Tail:** The last link in $view_i^+$ (with the lowest heft) is selected.

(2) Seed Planting: A new link, called seed, of v_i (or v_j) itself is created and inserted to v_j's (or v_i's) out-view. The initial heft of the newly created link is set to the weight of its owner.

- **Push:** Peer v_i inserts its seed et_i, into $view_j^+$. The seed's heft $et_i.h$ is set to w_i.
- **Pull:** Peer v_j inserts its seed et_j, into $view_i^+$. The seed's heft $et_j.h$ is set to w_j.
- **Push&Pull:** Execute both Push and Pull.

(3) View Merging: A peer, either v_i or v_j, inserts a copy of its out-view into the other's out-view. The heft of the replicated links (both the original and the copy) are decreased by a half. If there are two links of the same heft, the newly inserted one is ordered behind the other.

- **Push:** Peer v_i decreases the heft of all links in $view_i^+$ by a half i.e., for all $et_x \in view_i^+$, $et_i.h/2 \rightarrow et_i.h$ where v_x is an out-neighbour of v_i. Then, v_i insert $view_i^+$ into $view_j^+$ i.e., $view_i^+ \cup view_j^+ \rightarrow view_j^+$.
- **Pull:** Peer v_j decreases the heft of all links in $view_j^+$ by a half i.e., for all $et_j \in view_j^+$, $et_j.h/2 \rightarrow et_j.h$ where v_x is an out-neighbour of v_j. Then, v_j insert $view_j^+$ into $view_i^+$ i.e., $view_j^+ \cup view_i^+ \rightarrow view_i^+$.
- **Push&Pull:** Executes both Push and Pull.

(4) View Selection: After the Seed planting and View merging operations, some peers' out-degrees may temporarily have more than d links and the network may temporarily have self-loops and multiple links. The View Selection operation is a mechanism for keeping the network be d-out-regular i.e., to select d links to be remain in a peer's out-view based on the heft of each link and deleted others. It also deletes self-loops and multiple links to keep the network to be simple.

- **Random:** Firstly, the self-loops are deleted. Then, for each group of multiple links, one link is selected uniformly at random from each group to remain and others are deleted. Finally, the peer selects a set of d links from its out-view uniformly at random to remain, other links are deleted.
- **Head:** Firstly, the self-loops are deleted. Then, for each group of multiple links, the link with the highest heft in each group remains and others are deleted. Finally, the first d outgoing links (with higher hefts) in the peer's out-view are selected to remain and other links are deleted.
- **Tail:** Firstly, the self-loops are deleted. Then, for each group of multiple links, the link with the lowest heft in each group remains and others are deleted. Finally, the last d outgoing links (with lower hefts) in the peer's out-view are selected to remain and other links are deleted.

The framework includes $3^4 = 81$ protocols which are the combinations of the 4 operations' options. We denote each protocol by a 4-tuple (*ts, sp, vm, vs*) where *ts, sp, vm, vs* indicates the options of the target selection, seed planting, view merging and view selection operations respectively. A wild-card is denoted by the symbol '$*$'. For example, (*Random, Push,* $*$*, Head*) indicates three different protocols that adopt the Random target selection, Push seed planting, any one of the three view merging options and Head view selection.

3.2.3 Peers' join and leave

When a peer v_i joins the system, we assume it can access at least one peer, called initiator, in the system. The initiator can be a login server or simply a peer which registers its entry in a published website [29]. From the initiator, the peer can obtain some links as the initial out-view. Then, by several link exchange processes, the out-view can be full filled. We need not any restrictions on the initial out-view of each peer, e.g., it can be only a link incident to the initiator or a set of links collected by a random walker starting from the initiator.

When a peer leaves the system, no additional procedures are required such as leaving announcing because its entires will be finally deleted during the exchanging operations and no fresh seeds will be planted. Therefore, the network has fault tolerance to peers' crash and disconnect, which are the most frequently occurred faults in P2P systems, because a crashed or disconnected peer can be considered as a peer normally left the system.

The join and leave procedures is so simple that the network can adapt to different system environments. However, such a design requires the network to be well self-organizing because some undesirable join and leave patterns may distort the network topology. Therefore, the self-organization property is an important evaluation criterion of the proposed protocols.

3.3 Simulation

3.3.1 Evaluation criterion

We evaluate the proposed protocols by simulation to find some protocols that can generate networks having similar properties with the PWDN. The evaluation criterion is decided from two standpoints. First, the network generated by a protocol should have similar properties with the PWDN. Second, a protocol should be applicable in P2P environments. Below, we introduce the details of the evaluation criterion.

We say a network is a good approximation of the PWDN if it satisfies the PWDN's Property 1 and Property 2. We say a network is applicable for P2P environments if it satisfies the following properties.

- Compact. A network should have small diameters to achieve efficient gossip-based communications in large-scale P2P systems.
- Self-organized. A network should be able to quickly converge to the expected topology from any initial weakly-connected topologies. The property is important for keeping the topology while peers frequently leave and join.
- Connected. A network should be at least weakly-connected with a high probability.

3.3.2 Simulation environment

The simulation includes 10000 peers, each peer has at most 30 outgoing links. Each peer has the same executing interval. In the first execution cycle, they execute the protocol in a random order.

All of the 81protocols are evaluated by a set of tests. For concise expression, we only show a part of essential results of them. In the following of this section, if a protocol clearly generates undesirable networks in any test, it will be excluded from further evaluation.

By preliminary extermination, the protocols ($*$, $*$, $*$, *Random/Tail*) are excluded from the candidate protocols because they are clearly not able to control each peer's in-degree proportion to its weight. From the principle of the weight-based degree control, we know that links incident to a peer of a high weight often has high hefts. In the case of the Random View Selection, links remain with the same probability regardless of its heft. And in the case of the Tail View Selection, links with lower hefts can remain earlier. Both of them are inconsistent to our purpose. In fact, such networks converge to star-like networks. Similar arguments can be found in M.Jelasity's work [20]. Blew, we focus on the rest 27 protocols ($*$, $*$, $*$, *Head*).

3.3.3 Uniform weight setting

We firstly evaluate the protocols under the uniform weight setting that each peer has the same weight. By the definition of the PWDN, if all of the peers have a same weight, any pairs of peers should be connected by a directed link with a same probability. That is the d-out-regular random network [19]. We verify if the protocols can generate networks having similar statistical and graph properties with that of the d-out-regular random network.

The protocols is started from two kinds of initial network topologies, the d-out-regular random network and the star-network. After 1000 cycles, we compute some statistical and graph parameters from the snapshots of the candidate networks. We also show same parameters of the d-out-regular random network generated by the centralized protocol that each peer selects d out-neighbours from the system uniformly at random. Notice that the simulation time is long enough for the networks to stabilize. Latter, it will be shown that most networks stabilize within 100 cycles.

The test items are as follows.

- **Variance Test** evaluate the necessary condition of the Property 1 that peers' in-degree should be nearly the same under the uniform weight setting. The test excludes some networks in which peers in-degrees vary widely.

- **Transformation Test** evaluate the necessary condition of the PWDN's Property 2 that

the network topology should continuously transform over time. The test excludes the networks in which peers have fixed views.

- **Self-organization Test** verifies if the protocols can generate similar networks starting from different initial topologies.
- **Scale Test** compares the diameter and the average path length of the protocols with the out-regular random network.

Notice that the above test items are not sufficient condition of the objective network. The network passed the test will be further evaluated by other tests. Below, we show the detailed evaluation contents of the test items. The simulation results are shown in Tables 3.2 and 3.3.

(1) Variance Test. The variance of the 10000 peers' in-degrees are shown by the item VAR in Table 3.2. In the d-out-regular random network, each peers' in-degrees follows a *Binomial distribution* that

$$\forall v_i \in V, \Pr[\Delta_i^- = k] = C_{n-1}^k p^k (1-p)^{n-1-k} \tag{3.3}$$

where p is the probability of any pair of peers being connected by a directed link. By Equality 3.1, we have $p_i = d/(n-1)$. Therefore, the variance of an out-regular random network is $(n-1)p(1-p) \approx 30$. Compare with it, 8 networks, generated by (*, *Pull*, *, *Head*) except for (*Head, Push, Push, Head*), have obviously higher variances. That implies peers' in-degrees are quite different. In addition, the variances of such protocols keep increasing during the whole simulation period while others' stabilize within 100 cycles. Therefore, these 8 protocols are excluded from further evaluation.

The undesired result is mainly caused by the Pull option in the Seed Planting operation that creates more seeds for popular peers (i.e., the peers having high in-degrees) than unpopular ones. Since more seeds a peer has, more links incident to the peer are replicated by the View Merging operation, the Pull option gives positive feedback to the in-degree of each peer that makes the in-degree distribution divergent. With the same reason, it can be seen from Table 3.2 that the protocols adopting the Push&Pull option in the Seed Planting operation also have higher variance than that of adopting the Push option. However, we remain them for the moment because the results are still acceptable in this test.

(2) Transformation Test. A necessary condition of the PWDN's Property 2 is that each peer must have a dynamic in-view in which all peers' entries may appear. For each peer v_i, a parameter called *sight* is defined by the number of the peers from which a link to v_i is generated

Table 3.2: Protocols (∗, ∗, ∗, *Head*), uniform weight setting,
$n = 10000$, $d = 30$, 1000 cycles executed, starting from uniform-random network.

Protocols	VAR	SIGHT	Connectedness	Diameter	Path Length
Uniform-random	30	-	Strong	4	2.97
Random, Push, Push	23	3652	Strong	4	3.06
Random, Push, Pull	44	4048	Strong	4	3.03
Random, Push, Push&Pull	45	6905	Strong	4	3.06
Random, Pull, Push	**991**	2219	**Weak**	-	-
Random, Pull, Pull	**38631**	514	**Weak**	-	-
Random, Pull, Push&Pull	**19028**	1890	**Weak**	-	-
Random, Push&Pull, Push	77	3528	Strong	4	3.09
Random, Push&Pull, Pull	76	3735	Strong	4	3.07
Random, Push&Pull, Push&Pull	129	6807	Strong	5	3.10
Head, Push, Push	28	**36**	Strong	4	3.00
Head, Push, Pull	41	**38**	Strong	5	2.98
Head, Push, Push&Pull	74	**46**	Strong	5	3.01
Head, Pull, Push	29	**30**	Strong	4	2.97
Head, Pull, Pull	**9748**	**64**	**Weak**	-	-
Head, Pull, Push&Pull	**7977**	**93**	**Weak**	-	-
Head, Push, Push	32	**33**	Strong	4	2.98
Head, Push, Pull	54	**43**	Strong	4	2.99
Head, Push, Push&Pull	76	**48**	Strong	5	3.02
Tail, Push, Push	19	3810	Strong	4	3.03
Tail, Push, Pull	42	4254	Strong	4	3.00
Tail, Push, Push&Pull	49	6887	Strong	4	3.06
Tail, Pull, Push	**951**	2337	**Weak**	-	-
Tail, Pull, Pull	**25308**	746	**Weak**	-	-
Tail, Pull, Push&Pull	**68258**	1655	**Weak**	-	-
Tail, Push&Pull, Push	76	3735	Strong	4	3.05
Tail, Push&Pull, Pull	73	4014	Strong	4	3.02
Tail, Push&Pull, Push&Pull	154	6926	Strong	5	3.07

Table 3.3: Protocols (∗, ∗, ∗, *Head*), uniform weight setting,
$n = 10000$, $d = 30$, 1000 cycles executed, starting from star network.

Protocols	VAR	SIGHT	Connectedness	Diameter	Path Length
Random, Push, Push	**5344**	2881	**Weak**	-	-
Random, Push, Pull	44	4035	Strong	4	3.03
Random, Push, Push&Pull	47	6886	Strong	4	3.06
Random, Push&Pull, Push	**5494**	339	**Weak**	-	-
Random, Push&Pull, Pull	76	3714	Strong	4	3.07
Random, Push&Pull, Push&Pull	132	6781	Strong	5	3.10
Tail, Push, Push	**4328**	508	**Weak**	-	-
Tail, Push, Pull	43	4260	Strong	4	3.00
Tail, Push, Push&Pull	48	6872	Strong	4	3.06
Tail, Push&Pull, Push	**4110**	541	**Weak**	-	-
Tail, Push&Pull, Pull	74	4016	Strong	4	3.02
Tail, Push&Pull, Push&Pull	150	6909	Strong	5	3.07

at least once during the simulation period i.e., the size of the union of the in-neighbours of v_i during 1000 cycles. In Table 3.2, the average sight of all peers are shown by the item SIGHT. From the results we can find that 9 protocols (*Head,* ∗, ∗, ∗) have very low sights. That implies the peers are only accessed by a small part of all peers that is clearly undesirable. Therefore, these 9 protocols (2 of them have already been excluded by the Variance Test) are excluded from the simulation.

By the Head option in the Target Selecting operation, peers select a link of the highest heft. That implies a peer often selects a target with which the latest link exchange is executed since a newly received seed often has the highest heft, As the result, each peer exchanges links with a fixed set of peers which depends on the initial network topology. In fact, such networks result in severe clustering [20].

(3) Self-Organization Test. A network should be well self-organizing to keep the network topology in dynamic P2P environments that peers frequently leave and join. That is, it should be able to quickly converge to the desired topology from any initial network topologies. In this test, we start the simulation from a star network and test if the protocols can generate similar networks as in previous tests. The results of remaining 12 protocols are shown in Table 3.3. Clearly, 4 networks, generated by (*Random/Tail, Push/ Push&Pull, Push, Head*), have much higher variance than previous results. Other protocols' results are almost the same as those in

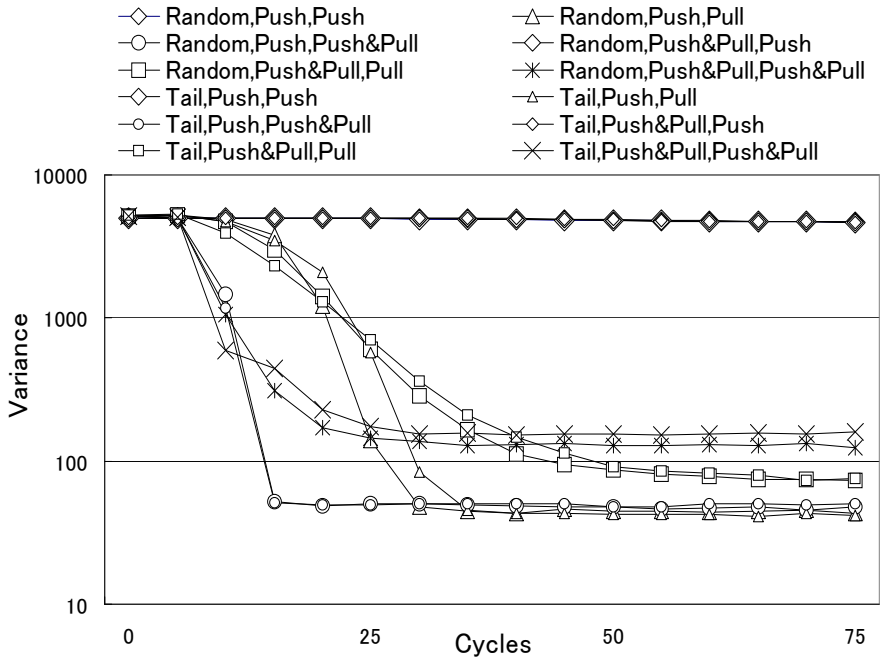


Figure 3.1: Changing of in-degree variance.

Table 3.2. Figure 3.1 shows the change of each network's variance during the simulation period. We can see that, except the above 4 networks, the variance of other networks quickly decrease and converge in a short time. Therefore, they are excluded for lack of the self-organization ability.

These 4 excluded protocols adopt the Push option in the View Merging operation. In such a network, a peer must passively wait for other peers push out-links to it. That implies an unpopular peer (i.e., a peer of a low in-degree) can hardly establish enough out-links. Therefore, such networks can not recover from high-biased topologies quickly. This is also a critical drawback that new peers can not quickly join the system.

(4) Scale Test. Up to now, 8 protocols, (*Random/Tail, Push/Push&Pull, Pull/Push&Pull, Head*), have passed all of the previous tests. We show some graph properties, such as the connectedness, the diameter and the average path length of them in Tables 3.2 and 3.3. It can be seen that these parameters of the 8 networks are similar to the out-regular random network. Therefore, all of the 8 protocols pass the Scale Test and will be evaluated by more strict tests for degree control.

Finally, we show degree distribution of the 8 remaining protocols under the uniform weight setting in Figures 3.2 and 3.3. The curve 'Ideal' is the in-degree distribution of a out-regular

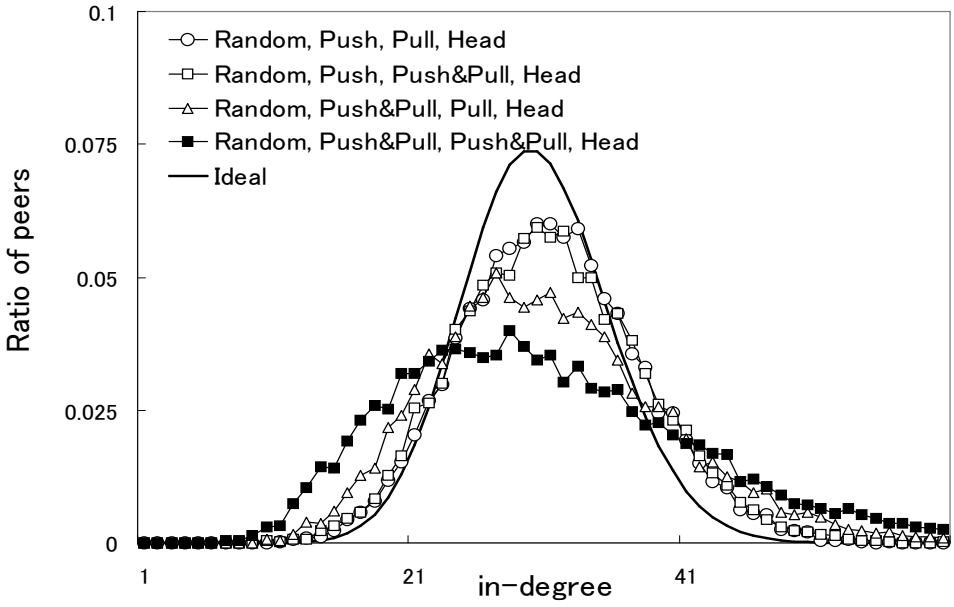


Figure 3.2: In-degree distribution, Random Target Selection.

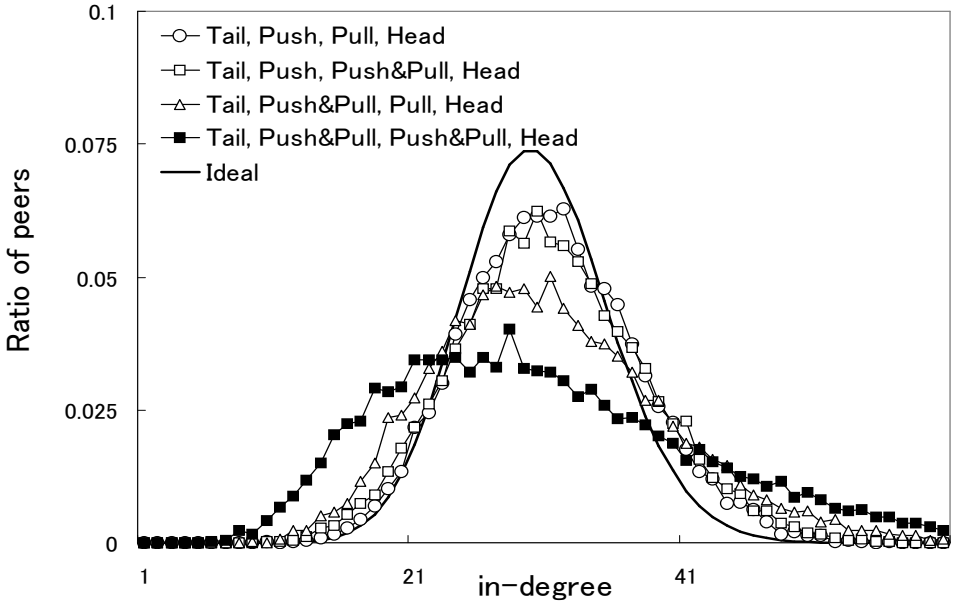


Figure 3.3: In-degree distribution, Tail Target Selection.

random network that follows a binomial distribution (Equality 3.3) as we mentioned in the Variance Test. Among these protocols, only 2 protocols (*Random/Tail, Push&Pull, Push&Pull, Head*) can also be generated by M.Jelasity's framework [20]. These results clearly show that ,under the uniform weight setting, our new protocols generate better uniform networks (i.e., peers' in-degrees have less random variance) than any protocols in M.Jelasity's framework.

3.3.4 Weighted setting

In this subsection, the 8 protocols that passed all tests under the uniform weight setting are evaluated for their performance of degree control.

Firstly we test the accuracy of the weight-based in-degree control under a simple weight setting. We divide peers into two groups of $V_A = \{v_1, ..., v_{9000}\}$ and $V_B = \{v_{9001}, ..., v_{10000}\}$, called group A and B respectively. The weight of all peers in group A, denoted by W_A, is fixed by 1. The weight of all peers in group B, denoted by W_B, is set to $2 \sim 128$ in each experiment respectively. Notice that the grouping plan is imaged by the layered networks in which usually 10% of the peers are selected to be super peers [30]. The average in-degree of the two groups, denoted by $\bar{\Delta}_A^-$ and $\bar{\Delta}_B^-$ respectively, are computed after 1000 cycles stating from a d-out-regular random network. The ratio $R_{B,A} = \bar{\Delta}_B^-/\bar{\Delta}_A^-$ of the protocols is shown in Figures 3.4 and 3.5. By Property 1, the ideal ratio is $R_{B,A} = W_B/W_A$ which is shown by the curve 'Ideal'.

From the results, it can be clearly seen that the protocols (*Random/Tail, Push&Pull, Pull /Push&Pull, Head*) and (*Random/Tail, Push, Pull, Head*) can not control the peers' in-degree correctly. The results of protocols (*Random/Tail, Push, Push&Pull, Head*) are accurate while W_B is lower than 32 but slightly higher than the expected value while W_B is higher than 64. Although the result implies they may have error when generate highly biased networks, they are still good approximations of the PWDN.

Then we test the two protocols under a power-law weight setting. For peer $v_i, 1 \leq i \leq n$, it's weight is set to $w_i = 1 + i^2/10000$. The simulation started from a d-out-regular random network and executed for 1000 cycles. The in-degree of each peer are shown in Figures 3.6 and 3.7 for each protocol respectively. In these figures, the curve 'd1000' is the in-degree of each peer taken from the snap shot of the network at the end of the 1000-th cycle. By PWDN's Property 1, we know the expected in-degree of a peer is

$$\delta_i^- = \frac{ndw_i}{\sum_{j=1}^{n} w_j} = \frac{30000 + 3i^2}{\sum_{j=1}^{10000} 1 + j^2/10000},$$

where $n = 10000, d = 30$. The ideal in-degree of each peer is shown by the curve 'Ideal'. Clearly, both of the two protocols can generate networks having expected in-degree distributions. Notice in this test the network is highly biased that $w_1/w_{10000} = 5000$. The expected in-degrees of some

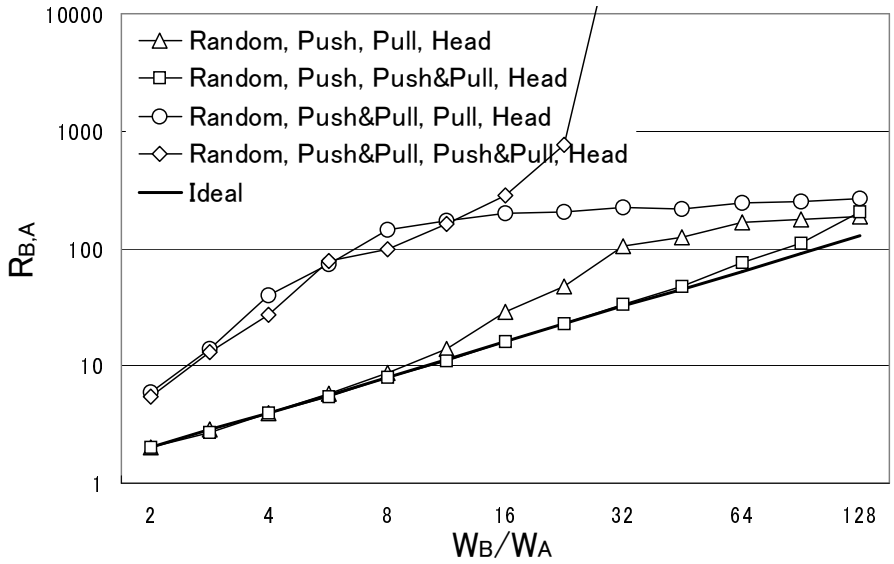


Figure 3.4: Degree control results, Random Target Selection.

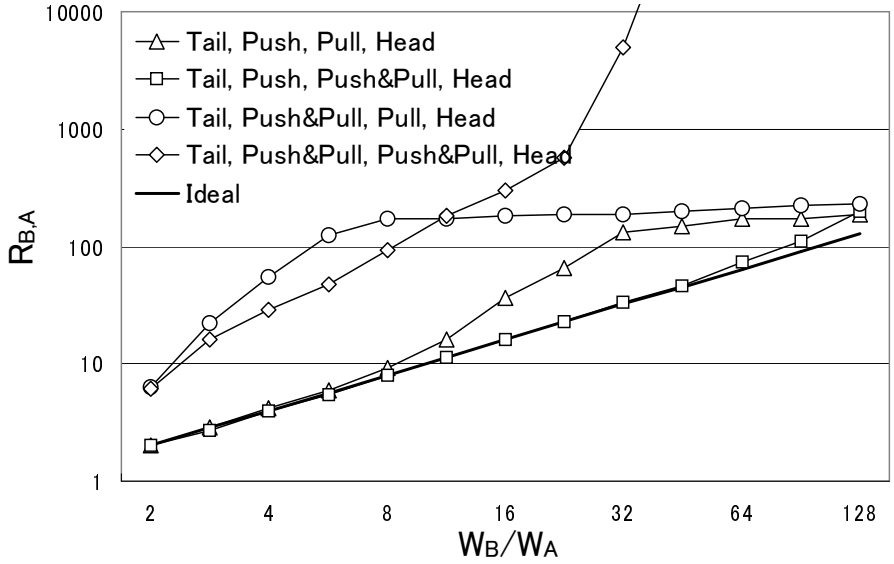


Figure 3.5: Degree control results, Tail Target Selection.

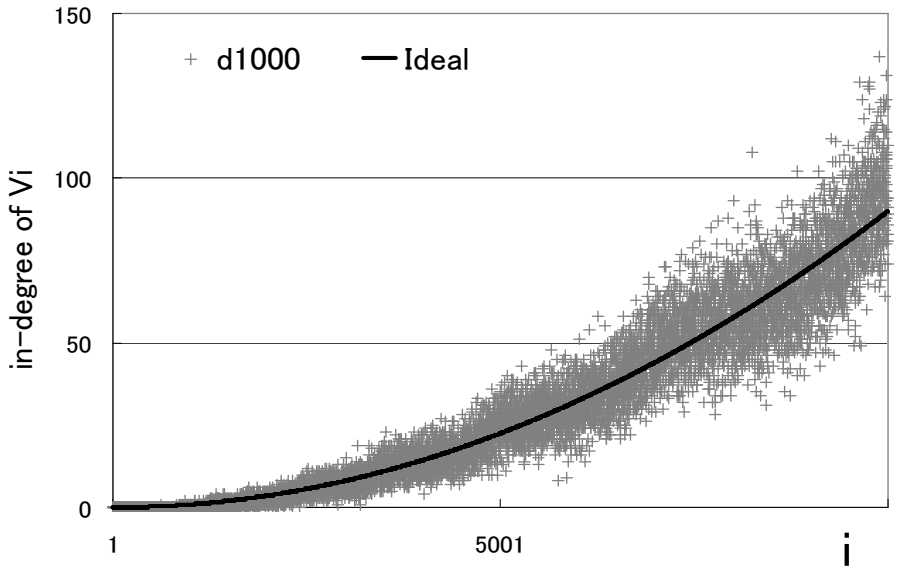


Figure 3.6: Peers' in-degree under power-law weight setting, (*Random, Push, Push&Pull, Head*).

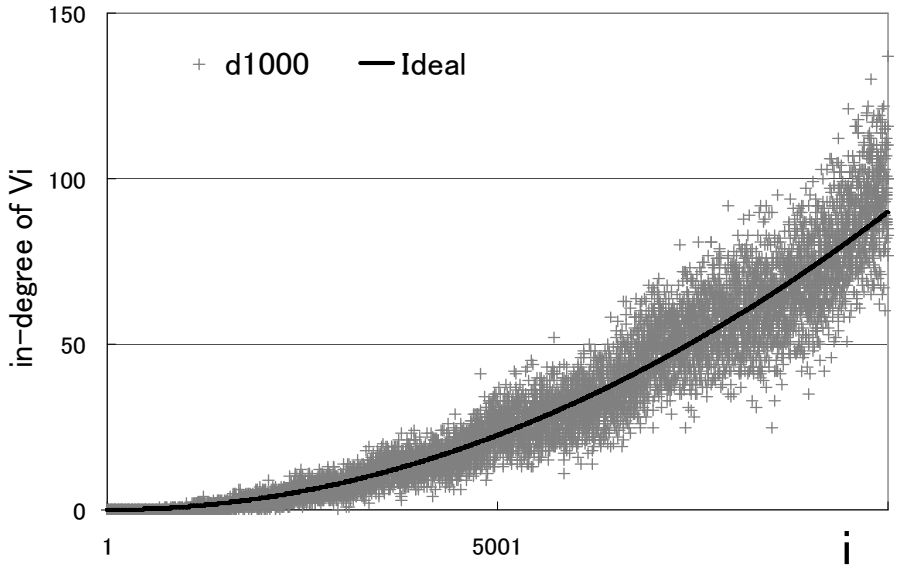


Figure 3.7: Peers' in-degree under power-law weight setting, (*Tail, Push, Push&Pull, Head*).

peers having low weights are much less than 1. This is the reason why we can construct the network with only 30 links per peer. Clearly, such a highly biased network can not be realized with reasonable overheads using undirected networks.

3.3.5 Randomness

In this subsection, the two protocols (*Random/Tail, Push, Push&Pull, Head*) are verified for the PWDN's Property 2. This test adopts the uniform-weight setting. We trace a randomly selected peer for a long time. At the end of each cycle, the entries in the in-view of the peer (the peer's in-neighbours) are recorded. The appearance probability of each peer's entry is computed from the accumulated records at the end of the 10000-th, 50000-th and 100000-th cycles. Figures 3.8 and3.9 show the distribution of the appearance probabilities of each peer's entry for the two protocols respectively. Clearly, the expected appearance probability of each peer's entry should be $d/(n-1) \approx 3‰$. From both of the protocols' results we can see that if the simulation time is long enough, the appearance probability of each peer converges to the expected probability.

The term 'Randomness' also means a peer's view (either in-view or out-view) should be unpredictable. For the protocol (*Random, Push, Push&Pull, Head*), the condition is clearly fulfilled because of the Random option in the Target Selection operation. For the protocol (*Tail, Push, Push&Pull, Head*), if the executing order and the initial out-view (or in-view) of each peer is known, the network can be predicted. However, because peers randomly join and leaves, the network topology is unpredictable in practice.

3.3.6 Connectivity

During all of the above simulations, the two protocols always generate strongly-connected networks. It is mainly because the out-degree ($d = 30$) is large enough for a 10000 peers' network. B.Yang, et al. show that a 2-out-regular random directed graph is weakly-connected with high probability [31]. It has also been proved that a random-graph requires $O(\log n)$ average degree to be strongly-connected with a high probability [19]. We are interested in how much out-links are required for our protocols to generate weakly-connected networks as well as strongly-connected ones.

We execute the two protocols for 1000 cycles starting from the d-out-regular random network. The network size is set to $n = 1000$ or $n = 10000$. The out-degree d is a variable in this simulation. After each cycle, we take a snapshot of the network and check for the connectedness. The percentage of strongly-connected snapshots are shown in Table 3.4. The symbol ' - ' indicates the protocol can not keep the network connected in 1000 cycles. We also show the results of 1000 d-out-regular random networks generated by the centralized protocol.

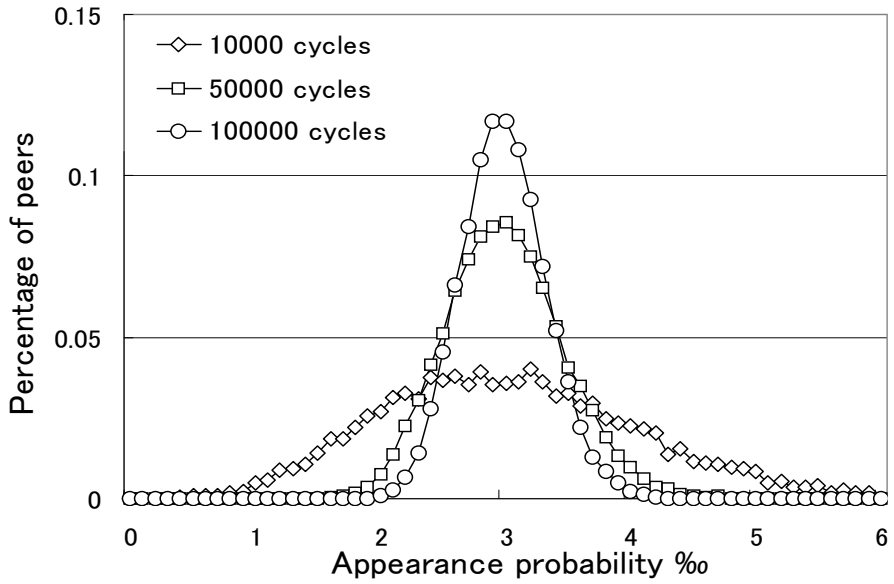


Figure 3.8: Appearance probability distribution, (*Random, Push, Push&Pull, Head*).

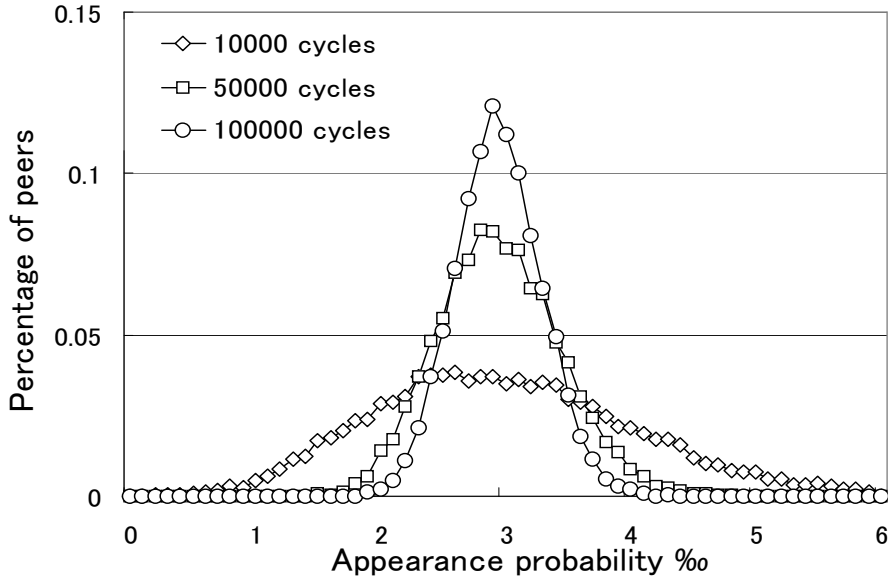


Figure 3.9: Appearance probability distribution, (*Tail, Push, Push&Pull, Head*).

Table 3.4: Percentage of strongly-connected networks, (∗, *Push, Push&Pull, Head*)

n=1000:	d=6	d=7	d=8	d=9	d=10
Random, ...	-	-	97%	99%	99%
Tail, ...	-	71%	98%	99%	99%
random network	1%	9%	100%	100%	100%
n=10000:	d=6	d=7	d=8	d=9	d=10
Random, ...	-	-	-	94%	96%
Tail, ...	-	-	91%	95%	99%
random network	0	0	3%	10%	65%

From the Table 3.4 we can see that our protocols require more out-degree to generate weakly-connected networks than that of out-regular random networks. However the required out-degree only increased by 1 from a 1000 peers' network to a 10000 peers' one. The result shows the proposed protocols have good scalability so that even when the system becomes large e.g., consists of up to millions of peers, the network construction cost is still low.

Interestingly, in some cases our protocols have higher probabilities to generate strongly-connected networks than the out-regular random network. It is unclear whether this feature is an advantage of better connectivity or just implies higher cluster coefficients.

3.4 Related Works

3.4.1 Degree-weighted networks

The distributed construction of degree-weighted networks is an important design component for modern P2P systems. Many works aim to construct capacity-aware networks with non-uniform degree distributions but few of them can adjust peers' degrees proportionally to their capacities (weights) [25][24]. That is possibly because, as we mentioned in Section 2, the traditional undirected network model cannot construct networks with highly-biased degree distributions with reasonable construction overhead.

Vishnumurthy, et al. proposed random-sampling-based protocol to construct directed networks in which a peer's in-degree are proportional to its capacity. Their purpose is similar to ours but the approach is quite different [23]. In their networks, a peer must establish the same number of outgoing links as their expected in-degrees so that both the out-degree and the in-degree are proportional to the peer's capacity. Therefore, the problem of high construction overhead for highly biased networks still remains in their approach, as that in undirected net-

works. Moreover, their protocol requires that all peers know the capacity and in-degree of the most powerless peer in order to decide the capacity-degree ratio in advance. Such an approach is difficult to be implemented in distributed environments and lack of flexibility.

3.4.2 Gossip-based overlay construction.

Our protocol is a kind of gossip-based overlay construction protocols [20][27]. A distinct feature of such protocols is the proactive link maintenance approach that incurs no additional overhead when peers join and leave. The contrary approach is the reactive link maintenance approach that the network topology is maintained only when peers join and leave [23][24]. Since in real P2P systems peers frequently join and leave, it is considered there is no obvious difference in the construction overhead between the reactive approaches and the proactive approaches. In addition, reactively constructed networks usually do not have the 'randomness' property.

Jelasity et al. propose a gossip-based framework, which consists of 27 candidate protocols, to generate uniform-random networks under the out-regular directed network model. So their objective network is a special case of the PWDN [20]. By setting all peers' weights to the same value, our framework can be regarded as an extension of theirs. A major difference of the two frameworks is that the Seed Planting and the View Merging operations are bounded in one operation in Jelasity's framework. So our framework has many new protocols including the two eligible protocols of the PWDN. The simulation results also show these two protocols can generate better uniform networks than any protocols in their framework.

3.5 Concluding Remarks

In this chapter, we studied a fundamental problem in P2P overlay construction: given a set of peers with respective weights, adjust each peers' in-degree proportionally to its weight. In order to bound the construction overhead in biased networks, we restrict all peers to have the same number of out-links. To the best of our knowledge, our work is the first solution to this problem under the out-regular directed network model.

The objective network is defined by the *Probabilistic Weighted d-out regular Directed Network (PWDN)*. By simulation, two protocols, the (*Random/Tail, Push, Push&Pull, Head*), are proved to be feasible for constructing the PWDN in P2P environments. The simulation result also shows that they can construct highly-biased networks with a reasonable number of total links. The result implies that we have overcome the problem of the high construction overhead for highly biased networks which has been considered impossible to be solved by traditional approaches based on the undirected network model.

The PWDN is a simple but powerful middle-ware for constructing heterogeneous overlay networks. It has only one interface parameter called 'weight'. By giving appropriate rules to decide the value of weight, one can apply the PWDN for different types of applications. The most representative application of the PWDN is to achieve the capacity-proportional workload assignment by setting each peer's weight to its capacity. The PWDN can also be applied to solve other gossip-based problems such as distributed search and election. Its applications are worthy of further studies.

Chapter 4

Workload Allocation

The basic function of a workload allocation strategy is to prevent peers from overload. It also greatly affects the search performance of unstructured P2P systems. In this chapter we investigate the search performance of unstructured peer-to-peer(P2P) systems from the viewpoint of workload allocation (WA).

A large number of distributed search algorithms have been proposed for unstructured P2P systems. By investigating these algorithms from the view point of their WA strategies, we can find an interesting rule, that is, a system adopting more biased WA seems to have better search performance. The Gnutella is the first pure P2P file sharing system that adopts a distributed search algorithm. In earlier versions of the Gnutella, all peers are responsible to process search queries which are randomly disseminated by flooding [17][13][16]. It tried to equally distribute the search workload among all peers (but failed to do so). In such systems, one must disseminate search queries to a large number of peers to find the target object (i.e. shared data items such as documents and music files). Some improved approaches disseminate search queries with high probability to popular peers which store more objects or indices (i.e. the location informations of objects) [32][33]. In such systems, popular peers take on more workload (e.g. process more queries) than unpopular ones. They can effectively decrease the number of query messages disseminated for search. Modern P2P systems usually adopt the layered networks (or say, super-peer networks) [14][22][15]. In such systems, a small number of powerful peers are selected to be *super peers* (SPs). SPs work like index servers and process all of the search queries. Other peers, called *leaf peers* (LPs), register their objects' indices in neighbouring SPs. Therefore, one can find the target object by searching only a few SPs. We can also consider the centralized search algorithms are extreme cases of such networks in which all indices and search queries are concentrated to the index server [34].

In this chapter, we first classify the traditional WA strategies of, which include that of the

above systems, into four distinct types: *Uniform (UN)* [17][13], *Capacity-Proportional (CP)* [32][33], *Fixedly-Layered (FL)* [14][15][35] and *Adaptive-Layered (AL)* [30]. Then, taking the advantages of both the capacity-proportional and the adaptive-layered types, we introduce a novel strategy, the *Adaptively-Layered & Capacity-Proportional allocation (ALCP)*, which has the following properties.

- The basic network model is a super-peer network in which a number of the most powerful peers are selected to be SPs which serve other peers.
- The number of SPs is adjusted as less as possible adapting to the total workload of the system.
- The workload allocated to each SP is proportional to its capacity.

Under the restriction that peers do not overload, the ALCP has the most biased workload distribution so that it can maximizes the search performance. Although its principle is intuitive and simple, it has never been realized by any previous works to the best of our knowledge. Finally, we present a framework which consists of a set of workload management protocols to realize the ALCP and other traditional workload allocation strategies faithfully. The simulation results show that the ALCP achieves obviously higher search performance than existing approaches UN, CP, FL and AL.

This chapter is organized as follows. In Section 4.1, we introduce a capacity model. In Section 4.2, we study the relationship between WA and search performance and propose the ALCP. In Section 4.3, we propose a distributed framework to realize the ALCP and other WA strategies. In Section 4.4, we evaluate the framework by simulation. In Section 4.5, we discuss related works. Finally in Section 4.6, we give concluding remarks.

4.1 A Capacity Model

The amount of workload a peer can bear is limited by its physical capacity such as bandwidth, CPU power, etc. [25][30]. Since a peer often have some other tasks running in parallel with P2P applications, it cannot always contribute all its physical capacity for system use. We consider the available capacity of a peer is a part of its physical capacity which is specially contributed for the system search usage. The actual amount of that can be manually decided by the user or automatically allocated by the client application. Below, we give the formal definition of the the capacity model.

A distributed P2P system consists of a set of peers $V = \{v_1, ..., v_n\}, |V| = n$. The capacity of peer v_i, denoted by c_i (≥ 0), is defined by the amount of tasks it can process in each time unit.

Table 4.1: Symbols of the capacity model.

V	$= \{v_1, v_2, ..., v_n\}$, a set of peers.
n	$= \|V\|$, the number of peers.
c_i	The capacity of v_i.
$l_i(t)$	The workload of v_i in time unit t.
$\bar{l_i}$	The average workload of v_i.
$r_i(t)$	$= l_i(t)/c_i$, the load rate of v_i in time unit t.
$\bar{r_i}$	$= \bar{l_i}/c_i$, the average load rate of v_i.

Without loss of generality, V is sorted by peer's capacity in decreasing order i.e., if $i > j$, $c_i \geq c_j$. We assume that the capacity distribution is approximatively continuous i.e., $c_i/c_{i-1} \geq 1 - \varepsilon$, $0 < \varepsilon \ll 1$. One may consider the distribution of peers' physical capacity (e.g., bandwidth) is discrete. However, because each peer contributes a different part of its physical capacity for system use, the distribution of peers' capacity can be considered approximatively continuous if the number of peers is large enough. The workload of v_i in time unit t, denoted by $l_i(t)$, is defined by the amount of tasks it receives in time unit t. The average workload of v_i in each time unit is denoted by $\bar{l_i}$. The load rate $r_i(t)$ and average load rate $\bar{r_i}$ of v_i are defined by $l_i(t)/c_i$ and $\bar{l_i}/c_i$ respectively. If $\bar{r_i} > 1$, we say v_i is constantly overloaded. Clearly, a robust system must not have constantly overloaded peers. If $r_i(t) > 1$, we say v_i is transiently overloaded in time unit t. Although a peer can buffer some tasks, which cannot be processed on time, in the task queue, the responding time of those tasks becomes long. Therefore, frequent transient overload should also be avoided.

4.2 Workload Allocation vs. Search Performance

In this section, we investigate the relationship between WA and search performance. We first explain why the search performance can be improved by biased WA. Then, we study some different types of WA strategies.

4.2.1 A probabilistic analysis of the index-dissemination-based search

In the index-dissemination-based search model, peers randomly disseminate messages i.e., indices and search queries, to others. We define the *access strategy* of V by a vector $A = (a_1, ..., a_n)$, $\sum_{i=1}^{n} a_i = 1$, where $a_i (\ll 1)$ is the probability that v_i receives a randomly disseminated message. The value is a statistical expectation which is independent of the sender. Clearly, a_i also implies

the ratio of the total system workload allocated to v_i. If an object has q indices disseminated to other peers, the probability that v_i holds an index of the object is $1-(1-a_i)^q \approx qa_i$. Then, because each peer $v_i \in V$ receives a search query with the probability a_i, each search query can find the object with the probability

$$\varphi(1) = \sum_{i=1}^{n} qa_i^2. \tag{4.1}$$

If the search query is disseminated to k peers, the success rate becomes $\varphi(k) = 1-(1-\varphi(1))^k$, which is a generalized expression of result presented by Miura's work [13]. By Equality (4.1), we obtain

$$\begin{aligned} \varphi(1) &= \sum_{i=1}^{n} q(a_i - \frac{1}{n} + \frac{1}{n})^2 \\ &= \sum_{i=1}^{n} q(a_i - \frac{1}{n})^2 + \frac{2}{n}(1-1) + \frac{q}{n} \\ &= q\mathrm{VAR}[A] + \frac{q}{n}, \end{aligned} \tag{4.2}$$

where VAR$[A]$ is the variance of A. It implies the bias of peer's workload distribution. Therefore, Equality (4.2) shows that more biased WA achieves better search performance.

4.2.2 Workload allocation strategies

We classify the WA strategies into the following five types.

Uniform allocation (UN)

All peers share the same amount of workload. Its access strategy A_{UN} is given by: for $1 \le i \le n$, $a_i = 1/n$. UN is the most original WA strategy of the P2P system [17][13][16][3]. However, it must search a lot of peers to find the target object because VAR$[A_{UN}] = 0$.

Capacity-Proportional allocation (CP)

The workload allocated to each peer is proportional to its capacity. Its access strategy A_{CP} is given by: for $1 \le i \le n$, $a_i = c_i/C$, where $C = \sum_{i=1}^{n} c_i$. CP is a typical type of the capacity-aware WA in which powerful peers take on more workload [32][33]. It achieves better search performance than UN because VAR$[A_{CP}] \geq$VAR$[A_{UN}]$ always holds. The actual value of VAR$[A_{CP}]$, and thus the search performance, is decided by the capacity distribution.

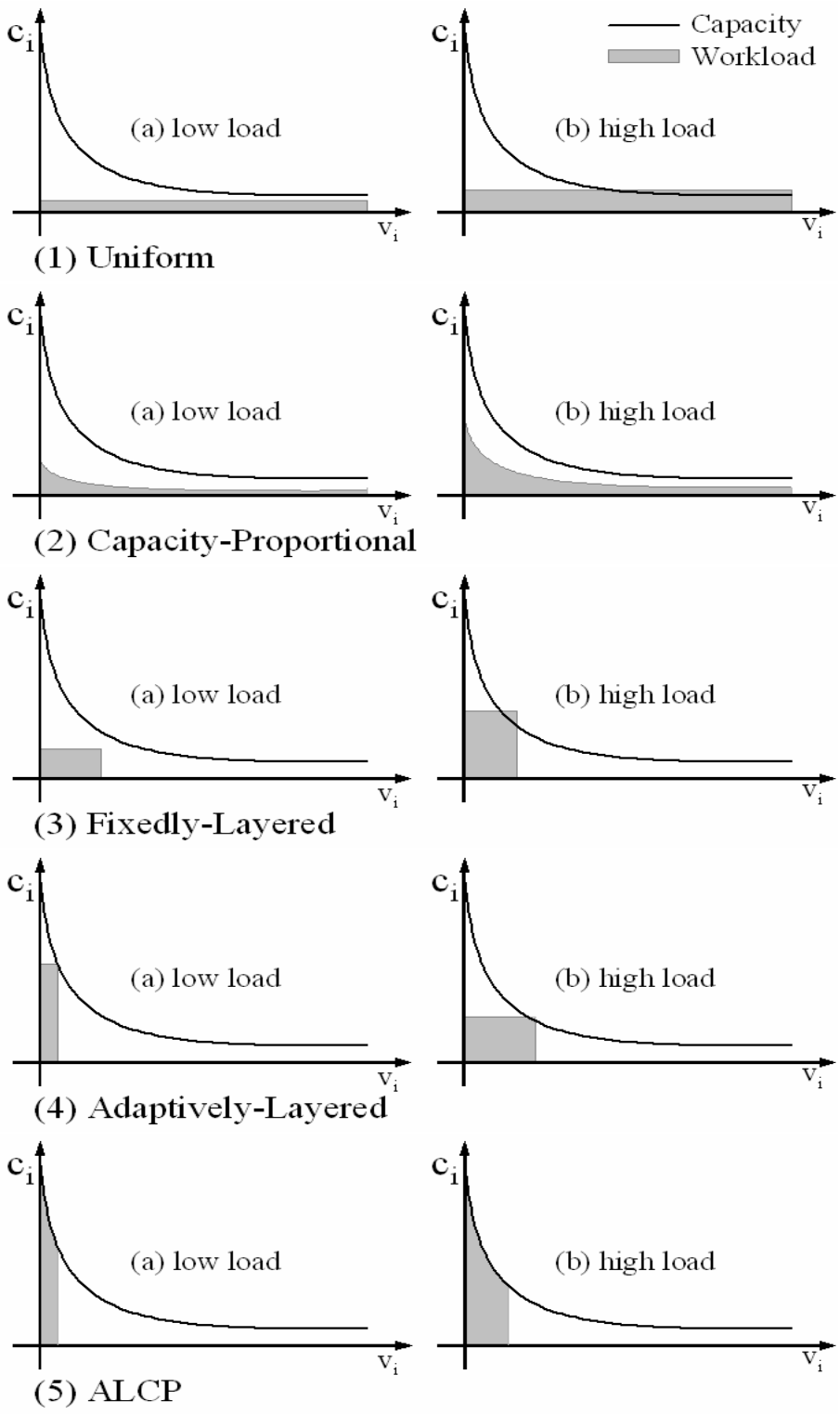


Figure 4.1: Workload distributions in low/high load systems.
x-axis: peer v_i (sorted by capacity), y-axis: the amount of tasks.

Fixedly-Layered allocation (FL)

A fixed ratio (or number) of peers are selected to be super-peers (SPs) and the workload allocated to each SP is the same [14][15][35]. An ideal definition of its access strategy A_{FL} is given by:

$$a_i = \begin{cases} 1/n_{sp}, & 1 \le i \le n_{sp} \\ 0, & n_{sp} < i \le n, \end{cases}$$

where n_{sp} is the number of SPs. We know in such systems the workload allocated to each SP is not always the same. But the most familiar design idea is to balance the system workload among all SPs. We classify all system which roughly adopt a fixed SP ratio (i.e., n_{sp}/n) into FL. In addition, the term 'ideal' means the most powerful n_{sp} peers are selected to be SPs. Because the ratio of SPs is usually small, the inequality VAR$[A_{FL}]$ >VAR$[A_{CP}]$ holds in most cases. Therefore, FL usually achieves better search performance than CP. Since FL is very easy to be realized in a distributed manner, it has become the most popular WA strategies now.

Adaptively-Layered allocation (AL)

AL is an improved WA strategy of FL [30]. Its ideal access strategy A_{AL} is given by:

$$a_i = \begin{cases} 1/n_{sp}, & 1 \le i \le n_{sp} \\ 0, & n_{sp} < i \le n. \end{cases}$$

In AL, n_{sp} is the minimum number of SPs that satisfies $(n_{sp}-1)\cdot c_{n_{sp}-1} < W \le n_{sp}\cdot c_{n_{sp}}$, where W is the total workload of the system. When the system workload becomes low, AL decreases n_{sp} to achieve higher search performance. When the system workload becomes high, AL increases n_{sp} to prevent SPs from overload. Clearly, AL always achieves better search performance than FL because it keeps n_{sp} be the minimum.

Adaptively-layered & Capacity-Proportional allocation (ALCP)

ALCP takes the advantages of both AL and CP. Based on AL, it adjust the workload of each SP proportionally to its capacity. Its ideal access strategy A_{ALCP} is given by:

$$a_i = \begin{cases} c_i / \sum_{i=1}^{n_{sp}} c_i, & 1 \le i \le n_{sp} \\ 0, & n_{sp} < i \le n. \end{cases}$$

where n_{sp} satisfies $\sum_{i=1}^{n_{sp}-1} c_i < W \le \sum_{i=1}^{n_{sp}} c_i$. Clearly, A_{ALCP} has the maximum variance and thus it achieves the best search performance.

Besides the search performance, the maximum workload a system can bear, denoted by C', is also an important property of a WA strategy. From Figure 4.1 we know that CP and ALCP can

fully utilize all peers' capacities i.e., $C' = \sum_{i=1}^{n} c_i$. For UN, the most powerless peer becomes the bottleneck and thus the system capacity is $C' = nc_n$. And for FL and AL, the most powerless SP becomes the bottleneck. The system capacity of FL is $C' = n_{sp}c_{n_{sp}}$. The system capacity of AL is affected by the distribution of peers' capacities. That is, $C' = n'_{sp}c_{n'_{sp}} = \max_{n_{sp}=1}^{n} n_{sp}c_{n_{sp}}$ where n'_{sp} is a threshold of n_{sp} that maximizes $n_{sp}c_{n_{sp}}$. If a system already has more than n'_{sp} SPs, by employing more SPs, the workload it can bear decreases on the contrary.

4.3 A Distributed Framework for Workload Allocation

In this section, we present a distributed framework to realize the five WA strategies. The framework is applicable for unstructured search protocols which adopt blind routing algorithms such as flooding and random walk. It consists of two layers. The lower layer is the PWDN we presented in Chapter 3. It adjusts each peer's in-degree and thus the workload proportionally to its weight. The upper layer includes five distributed workload management protocols which decide each peer's weight according to the five WA strategies respectively.

4.3.1 An improved construction protocol for PWDN

We adopt the protocol (*Random, Push, Push&Pull, Head*) to construct the PWDN with some minor improvements. Notice the protocol (*Tail, Push, Push&Pull, Head*) can also be adopt in this work. For simple description, we do not introduce it in this chapter. The protocol is divided into an active thread and a passive thread which are shown by Protocols 1 and 2 respectively.

In the protocol, each peer v_i periodically executes four operations per time unit.

- **Target Selection** (Active thread, Line 3)
  Peer v_i selects an entry et_j from $view_i^+$ uniformly at random. The peer v_j becomes the target with which peer v_i exchanges their outgoing links.

- **Seed Planting** (Active thread, Lines 4, 6)
  The peer v_i sends an entry et_i of itself to v_j. We call it the *seed* of v_i. The initial heft of the seed is set to the weight of v_i.

- **View Merging** (Active thread, Lines $5 \sim 8$ and Passive thread, Lines $3 \sim 5$)
  Both v_i and v_j send a copy of their first $K, (1 \leq K \leq d)$ entries (with higher hefts) to each other. The heft of the shared entries, including both the original and the copy, are decreased by a half.

Protocol 1: Construction protocol of PWDN, Active thread.

Input: d, $K(1 \leq K \leq d)$

Data:

v_i: this peer

v_j: the peer to exchange view with

$view_i^+ = \{et_{x_1}, ..., et_{x_d}\}$, the out-view of v_i

$view_j^+ = \{et_{y_1}, ..., et_{y_d}\}$, the out-view of v_j

```
1  while true do
2      wait for 1 time unit;
3      select an entry et_j from view_i^+ randomly at uniform;
4      create et_i, et_i.h := w_i;
5      for 1 ≤ k ≤ K, et_{x_k}.h := et_{x_k}.h/2;
6      send {et_i, et_{x_1}, ..., et_{x_K}} to v_j;
7      receive {et_{y_1}, ..., et_{y_K}} from v_j;
8      view_i^+ := view_i^+ ∪ {et_{y_1}, ..., et_{y_K}};
9      ViewSelection(view_i^+);
10 end
```

Protocol 2: Construction protocol of PWDN, Passive thread.

Input: d, $K(1 \leq K \leq d)$

Data:

v_i: this peer

v_j: the peer which request for view exchange

$view_i^+ = \{et_{x_1}, ..., et_{x_d}\}$, the out-view of v_i

$view_j^+ = \{et_{y_1}, ..., et_{y_d}\}$, the out-view of v_j

```
1  while true do
2      if receive {et_j, et_{y_1}, ..., et_{y_K}} from v_j then
3          for 1 ≤ k ≤ K, et_{x_k}.h := et_{x_k}.h/2;
4          send {et_{x_1}, ..., et_{x_K}} to v_j;
5          view_i^+ := view_i^+ ∪ {et_j, et_{y_1}, ..., et_{y_K}};
6          ViewSelection(view_i^+);
7      end
8  end
```

- **View Selection** (Active thread, Line 9 and Passive thread, Line 6)
  After the Seed Planting and View Merging operations, a peer v_i may temporarily have more than d outgoing links. It may also have some self-loops and multiple links. The View Selection operation keeps the network d-out-regular and simple. The detailed operation of *ViewSelection*($view_i^+$) is as follows.

  1. Sort $view_i^+$ by the decreasing order of entries' hefts.
  2. Delete all entries of v_i itself.
  3. For each group of entries of the same peer, remain the entry of the highest heft and delete others.
  4. Remain the first d entries (with higher hefts) in $view_i^+$ and delete others.

The join and leave procedures are the same as what we presented in Chapter 3. When a peer joins the system, it accesses an initiator in the system to obtain some links as its initial out-view. Then, by several link exchange processes, it can fill its out-view. When a peer leaves the system, no additional procedures are required such as leaving announcing.

The main difference between the improved protocol and the original one we proposed in Chapter 3 is that a peer only replicate a part of K entries in its out-view each times it exchanges links with other peers. In the original protocol, once a peer receives d entries from another peer, in average only a half of them which have higher hefts can remain in its out-view. Therefore, this change saves wasteful communication cost in the link exchanging process. Moreover, it decreases the probability of replicating entries of low hefts so that the network can achieve lower variances in peers' in-degrees.

Below, we explain the principle of the weight-proportional in-degree control. When an entry is replicated by the View Merging, both the original entry's and the copy's hefts are the half. Therefore, the sum of the hefts of those entries is a constant during the View Merging. The Seed Planting is the only way for a peer to increase the total heft of its entries. In our protocol, a peer can create one seed per time unit. The initial heft of the seed, that is the peer's weight, decides how many times the seed can be replicated before being deleted by the View Selection. Therefore, if the network is fair that entries of the same heft are replicated with the same probability, the number of entries of each peer is proportional to its weight.

The protocol adopts a metabolic mechanism to maintain links [20]. A link is given an initial heft by the peer it incident to and the heft decreases during the exchange process. Finally, the link and its replicas are deleted by the View Selection operation. Therefore, a peer does not need to check if its outgoing links are pointing to exist peers because only newly created links can remain in the system. This is why no additional procedures are required when a peer leaves. It

also implies that the protocol has good failure tolerance to peers' crash and disconnect (i.e., the most frequently occurred failures in P2P systems) because a crashed or disconnected peer can be considered as a peer normally leaving from the system.

Finally, let us see the communication cost of the protocol. Each peer sends $K+1$ entries and receives K entries from another peer per time unit. The protocol does not incur additional cost when peers join and leave. It also does not need to check if peers are crashed or disconnected. Thus, the maintenance cost of the network is $n(2K+1) \leq n(2d+1) = O(|E|)$ per time unit. Clearly, it is the essential maintenance cost for a dynamic network of $|E|$ links.

4.3.2 Workload management protocols

The upper layer of the framework consists of five distributed workload management protocols which decide the weight of each peer according to the five WA strategies respectively.

The protocols for the three basic WA strategies are very simple and incur no additional communication cost.

Protocol UN:

Each peer set its weight to 1.

Protocol CP:

Each peer set its weight to its capacity.

Protocol FL:

If a peer's capacity is larger than a given threshold value c_t, it behaves as an SP and sets its weight 1. Otherwise, it behaves as an LP and sets its weight to 0. Since the capacity distribution of a large-scale system is almost static, the system in fact keeps a fixed ratio SPs. This approach is widely adopted by real P2P systems for its simplicity [15][10]. By surveying the capacity distribution in advance, one can also set an appropriate value of c_t to employ a predefined ratio of peers as SPs. We skip the details because FL is not the main interest of this paper.

The protocols for the two adaptive WA strategies are more complex. Adapting to change of the system workload [15], they satisfy the following three requirements. (1) The number of SPs is adjusted as less as possible. (2) The most powerful peers are selected to be SPs. (3) All SPs are prevented from frequently transient overload. Three operations are introduced for the three requirements respectively [30]:

- **Demotion:**
  If SPs' average load rates are low, some powerless SPs are demoted to be LPs.

- **Substitution:**
  If an LP is more powerful than an SP, they change their status i.e., an SP or an LP.

- **Promotion:**
  If some SPs' transient load rates are too high that they are likely to transiently overload, some powerful LPs are newly promoted to be SPs.

Each peer periodically checks if it satisfies the execution conditions of the above operations. The judgement is made according to the workload situations of the peer itself and its out-neighbours. Notice that a peer's out-neighbours are almost SPs because only SPs have in-neighbours. The only exception is that some LPs, which are newly demoted from SPs, may keep some in-neighbours for a short period of time. In addition, since the PWDN is constructed by periodical link exchange, the set of out-neighbours of each peer changes over time. In the following protocols, a peer needs to know the capacity of its out-neighbours. The capacity of each peer is included in its entry so that no additional communication is needed for a peer to know its out-neighbours' capacities.

Protocol AL:

The protocol AL is shown by Protocol 3. In AL, the weights of selected SPs are set to 1 and those of LPs are set to 0. Each peer executes the protocol every T time units. Since the system workload changes relatively slow (e.g., in cycles of several hours [15]), the protocol does not need to be executed frequently. Moreover, after a peer changed its status (e.g., be promoted to be an SP), it takes several time units for the network construction protocol to adjust its in-degree. The interval is favorable for such peers to make the correct judgement.

If v_i is an SP (i.e., w_i=1), it checks the conditions for Demotion (Line $3 \sim 6$):

- v_i is the most powerless peer in $\{v_i\} \cup view_i^+$.
- The load rate of v_i is low: $\bar{r_i} = \bar{l_i}/c_i < \eta$.

The system parameter η, $0 < \eta < 1$, is a predefined threshold value which represents the meaning of 'low load rate'. It decides the strength of the Demotion condition. A large value of η decreases the number of SPs so that the search performance becomes high. However, the overload rate of SPs becomes high. A small value of η has the contrary effect. Since the transient workload of an SP has large variance, its average workload must be low so that frequent transient overload can be prevented [15]. Therefore, η should be set to a small value e.g., smaller than the expected average load rate.

If v_i is an LP (i.e., w_i=0), it first checks the conditions for Substitution (Line $11 \sim 14$):

Protocol 3: Workload management protocol AL.

Input: η, ε, T

Data:

t: current time

v_i: this peer

$v_\perp$: the most powerless out-neighbour of v_i

```
1  while true do
2      wait for T time units;
3      if w_i = 1 then                                   // This is an SP.
4          if (c_i < c_⊥) ∧ (r̄_i < η) then
5              w_i := 0;                                  // Demotion!
6          end
7      else                                              // This is an LP.
8          if c_i/c_⊥ < 1 − ε then
9              continue;                                  // Goto the start of loop.
10         end
11         request w_⊥ and r_⊥(t) from v_⊥;
12         if (c_i > c_⊥) ∧ (w_⊥ = 1) then
13             w_i := 1;                                  // Substitution!
14             order v_⊥ to Demotion;                     // w_⊥ := 0
15         else if r_⊥(t) > 1 − ε then
16             w_i := 1;                                  // Promotion!
17         end
18     end
19 end
```

- $v_\perp$ is an SP.
- v_i is more powerful than $v_\perp$: $c_i > c_\perp$.

The condition of Substitution is simple. If the peer finds an SP which is less powerful than it, they exchange status. It does not need to check other peers in its out-view because $v_\perp$ is most likely to be such powerless SPs. If the network has good randomness (PWDN's property 2), powerless SPs will eventually be substituted because they have non-zero probabilities to have some powerful LPs be their in-neighbours.

If the Substitution conditions are not satisfied, the peer then checks the conditions for Promotion (Line $8, 14 \sim 16$):

- $v_\perp$ is likely to overload: $r_\perp(t) = l_\perp(t)/c_\perp > 1 - \varepsilon$.
- v_i is more powerful than most of the other LPs: $c_i/c_\perp \geq 1 - \varepsilon$.

If the first condition (Line 15) is satisfied, more SPs are needed to share the system workload. In AL, since all SPs take on the same amount of workload, $v_\perp$ has the highest load rate so that it is the most likely to overload among v_i's out-neighbours. The second condition implies only powerful LPs are qualified to be SPs (Line 8). The judgement can be made only based on $c_\perp$ because it is usually close to the capacity of the most powerless SP, or say, the most powerful LP. The predefined parameter ε, $0 < \varepsilon \ll 1$, decides the strength of the Promotion condition. A large value of ε achieves a low overload rate but employs many SPs so that the search performance becomes low. A small value of ε has the contrary effect. Since we say a peer is likely to overload if its transient load rate is larger than $1 - \varepsilon$, it has little sense to set ε to a larger value.

Protocol ALCP

The Protocol ALCP is shown by protocol 4. In ALCP, the weights of SPs are set to their capacities and those of LPs are set to 0. Its principle is similar to the Protocol AL. However, when a peer checks for the conditions for Promotion, besides $v_\perp$, it also checks the load rate of a randomly selected out-neighbour v_R (Line $16 \sim 18$). In ALCP, all SPs have almost the same load rate so that one can obtain SPs' average load rate by sampling some randomly selected SPs. Moreover, since the $v_\perp$ is the most powerless SP in $view_i^+$, it may be a newly promoted SPs or a newly demoted LP of which the load rate are less than other SPs. Therefore, one cannot correctly keep track of the workload situations of all SPs by only sampling $v_\perp$ in ALCP.

Finally, let us see the communication cost of the two adaptive protocols. In protocol AL and ALCP, only a few peers of which the capacities are close to that of the most powerless SP may

Protocol 4: Workload management protocol ALCP.

Input: η, ε, T

Data:

t: current time

v_i: this peer

$v_\perp$: the most powerless out-neighbour of v_i

```
1  while true do
2      wait for T time units;
3      if w_i > 0 then                                   // This is an SP.
4          if (c_i < c_⊥) ∧ (r̄_i < η) then
5              w_i := 0;                                  // Demotion!
6          end
7      else                                              // This is an LP.
8          if c_i/c_⊥ < 1 − ε then
9              continue;                                 // Goto the start of loop.
10         end
11         Request w_⊥ and l_⊥(t) from v_⊥;
12         if (c_i > c_⊥) ∧ (w_⊥ = 1) then
13             w_i := c_i;                               // Substitution!
14             order v_⊥ to Demotion;                    // w_⊥ := 0
15         else
16             Select a peer v_R from view_i^+ randomly;
17             Request r_R(t) from v_R;
18             if (r_⊥(t) > 1 − ε) ∨ (r_R(t) > 1 − ε) then
19                 w_i := c_i;                           // Promotion!
20             end
21         end
22     end
23 end
```

request workload informations from others in every T time units. Therefore, the communication cost of them are vanishingly small compared with the construction cost of the PWDN.

4.4 Simulation

This section evaluates the performance of our framework. We first show some new evaluation results of the network construction protocol of the PWDN. Then we evaluate the workload management protocols by comparing their search performance, overload rate, etc.

In order to show the nature performance of our framework, we do not run the simulation under churn models. Otherwise, the evaluation result will be greatly affected by the maintenance of the system e.g., the negative impact of churn can be weaken by shortening the execution interval i.e., the length of a time unit. The availability of the framework under churn is guaranteed by the self-organizing property of the network construction protocol and the self-adaptive property of the workload management protocols.

4.4.1 Performance of the network construction protocol

The evaluation criteria of the network construction protocol are similar to what we adopted in Chapter 3.

- **Accuracy of the degree control:**
  A network should satisfies PWDN's Property 1.

- **Self-organization:**
  A network should be able converge to the expected topology from any initial topologies as long as they are weakly-connected.

- **Connectivity:**
  A network should be weakly-connected with a high probability.

- **Compactness:**
  A network should keep a compact structure in large-scale systems.

The basic simulation setting adopts 10000 peers, each peer has at most 10 outgoing links ($d = 10$). In the View Merging operation, peers share 5 links to each other ($K = 5$). Peers execute the protocol in a randomly decided order in the first time unit and keep the execution order in the followings. The simulation is started from two kind of initial networks, the d-out-regular random network (latter, simply called 'random network') in which each peer has d

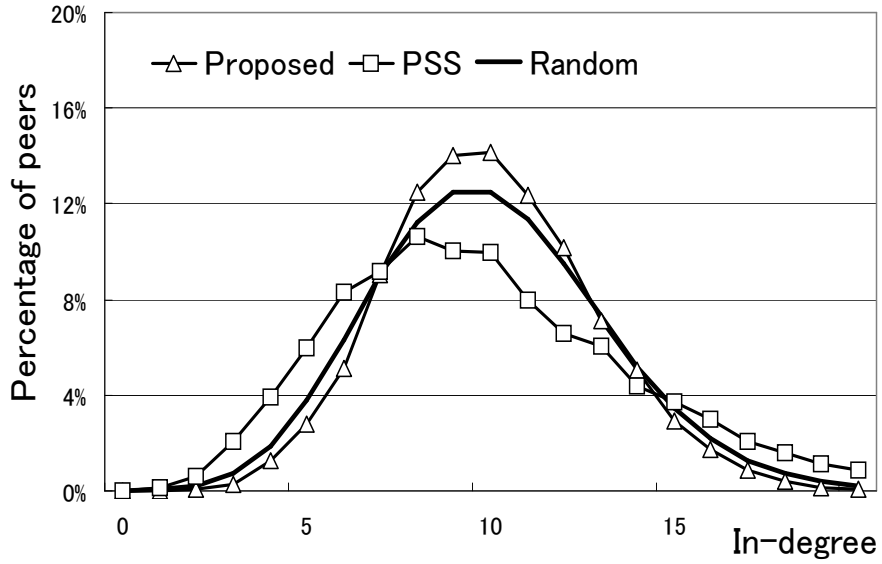


Figure 4.2: In-degree distribution under uniform weight setting.

randomly selected out-neighbours and the star-network in which all peers have the same out-neighbour which is randomly selected. Two kind of weight setting are adopted, the uniform weight setting given by $\forall v_i \in V, w_i = 1$ and a power-law weight setting given by $w_i = i^3$.

Accuracy of degree control (PWDN's Property 1)

First we evaluate the protocols under the uniform weight setting. As we mentioned in Chapter 3, the expected network is the d-out regular random network. In such a network, peers' in-degrees follow a *Binomial distribution* [19]:

$$\forall v_i \in V, \Pr[\Delta_i^- = k] = C_{n-1}^k p^k (1-p)^{n-1-k}$$

where p is the probability of any pair of peers being connected by a directed link. By Equality (3.1), we have $p = d/(n-1) \approx 0.001$. In Figure 4.2, we show the in-degree distribution of the network generated by our protocol (curve 'Proposed'). The data is taken from the snapshot of the network after 1000 time units are executed. In this test and most of the following tests, the results of starting from different initial networks are almost the same. We do not show them respectively in such cases. The result is compared with that of the random network (curve 'Random') and the best network generated by Jelasity's framework (curve 'PSS') [20]. Clearly, our network has less variance in peers' in-degrees. And compare the figure with what show in Figure 3.2, we can find that when $K < d$ the protocol generates networks having less variance

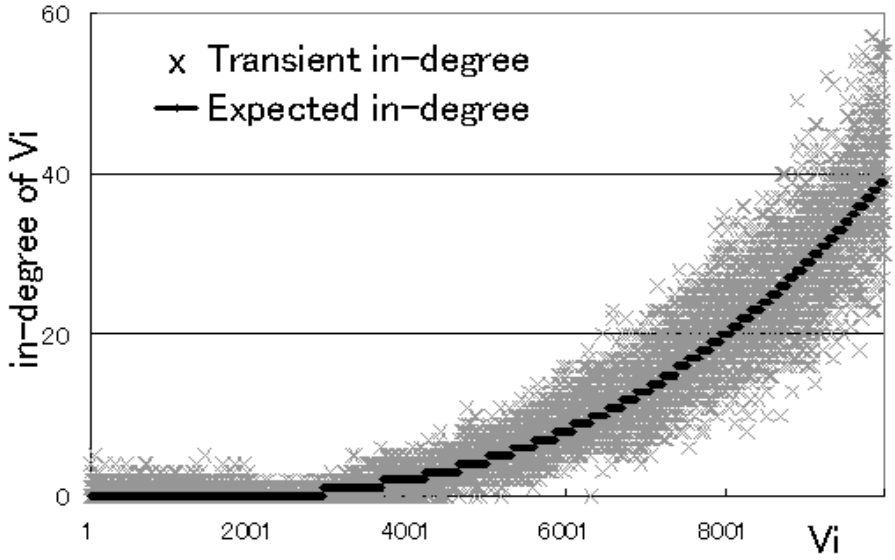


Figure 4.3: Transient in-degree, power-law weight setting.

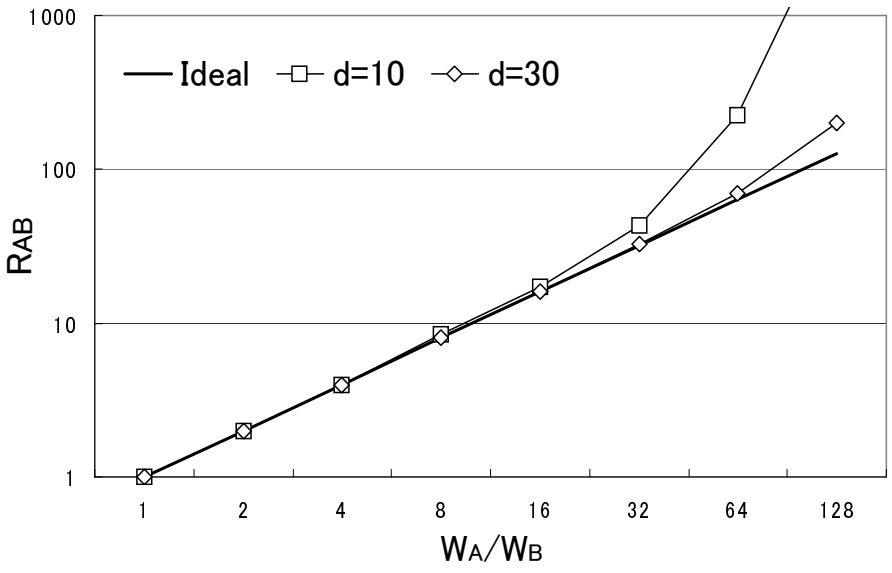


Figure 4.4: Accuracy of degree control.

than the original protocol in which $K = d$. That is favorable for achieving stable WA with small random variance in peers' workloads.

Next, we evaluate the protocol under non-uniform weight settings. In Figure 4.3, we show peers' transient in-degrees under the power-law weight setting. We can find that peers' in-degrees (item 'Proposed') are close to the expected in-degree (curve 'Ideal') given by PWDN's Property 1.

For further evaluation, we adopt a simple weight setting to test the accuracy of the weighted degree control. Peers are divided into two groups: $V_A = \{v_1, ..., v_{1000}\}$ and $V_B = \{v_{1001}, ..., v_{10000}\}$. The weights of peers in V_A and V_B are set to W_A and W_B respectively. The value of W_A is set to $2 \sim 128$ in each experiment respectively and W_B is fixed to 1. We compute the ratio $R_{A,B} = \bar{\Delta}^-_A / \bar{\Delta}^-_B$ where $\bar{\Delta}^-_A$ and $\bar{\Delta}^-_B$ are the average in-degree of the peers in two groups respectively. The test is executed in networks of d=10, K=5 and d=30, K=5 respectively. In Figure 4.4, R_{AB} of those networks are compared with the expected ratio W_A/W_B (curve 'Ideal'). Clearly, our protocol can accurately control peers' in-degree ratio while W_A/W_B is low but fails while the ratio is high. By comparing the curves 'd=10' and 'd=30', it can be found that if we adopt more links, the protocol can construct more biased networks. From the experiment data, we found that in those failed networks, peers of very low weights cannot obtain enough in-links as expected. Although the absolute error between their average in-degree and the expected in-degree is very small, the error in $R_{A,B}$ is greatly zoomed in highly biased weight settings because the expected in-degrees of those low-weight peers are very low. Therefore, even those failed networks do not have critical drawbacks in practical use because we do not mind very powerless peers receive less workload than expected.

Self-organization

In previous tests we find that the protocol can generate networks of expected topologies from both the star network and the random network. This test evaluates how much time is required for the network topology to be stable. We compute the variance of the network at the end of each time unit starting from the random network and the star network. Figure 4.5 shows the results of under the uniform weight setting and the power-law weight setting respectively. For reference, we also show the in-degree variance of the random network (curve 'REF'), that equals to $(n-1)p(1-p) \approx 10$. We can see that curves represent the same weight setting quickly converge to the same within 20 time units value and stabilize, no matter what kind of initial topology is adopted. The results show that network can converge very quickly because it cost only 20 time units to transform the topology form the star network to the out-regular uniform random network (i.e., the expected network of the uniform weight setting) which are the two

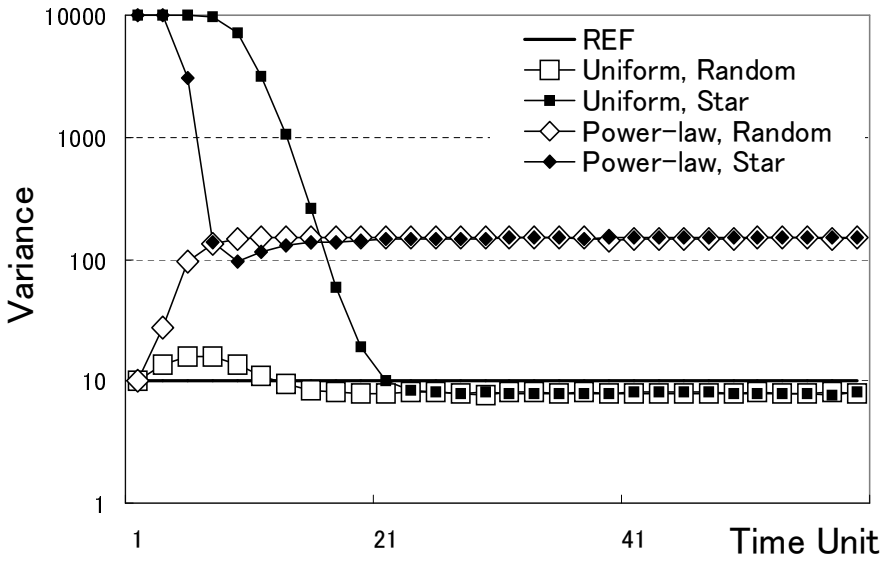


Figure 4.5: Change of in-degree variance (self-organization).

n=1000:	d=5	d=6	d=7	d=8	d=9	d=10
Power-law	×	○	○	○	○	○
Uniform	×	×	×	○	○	○
n=10000:	d=5	d=6	d=7	d=8	d=9	d=10
Power-law	×	×	○	○	○	○
Uniform	×	×	×	×	○	○

Table 4.2: Weak-connectivity

extremes of the degree-weighted networks. The short converge time is favorable for keeping the network topology when peers frequently join and leave. The simulation setting also covers most of the possible topology transformation scenarios in unstructured P2P networks. Therefore, the network has sufficient self-organization ability for unstructured P2P systems.

Connectivity

In previous tests, the network always keeps weakly-connected. That is because each peer has enough outgoing links. It is known that a 2-out-regular random graph is weakly-connected with a high probability if the number of nodes is large enough [31]. We are interested in how many outgoing links are required for our network to be weakly-connected with a high probability. We execute the protocol for 10000 time units using different values of d, under different weight

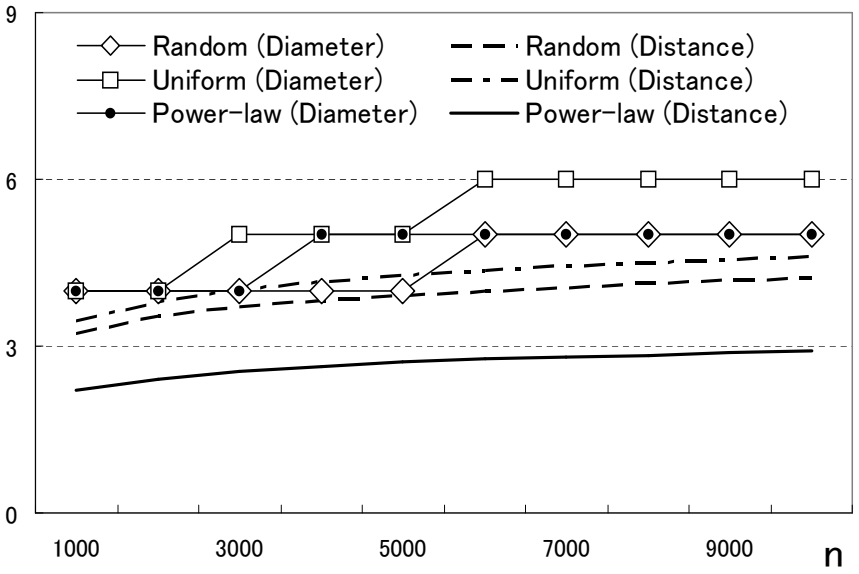


Figure 4.6: Diameter and average distance between peers (compactness).

settings. If a network keeps weakly-connected in the test, we mark the setting '◯', otherwise '×'. The results are shown in Table 4.2. We can see that our protocols require more outgoing links to keep the network weakly-connected than out-regular random networks. However it requires only 1 more links for a 10000 peers' network than a 1000 peers' one. Therefore, the protocol is scalable.

Compactness

Under different weight settings, we compute the network diameter and the average distance between peers to see if the proposed protocol can generate compact networks. Since our protocol do not always generated strongly-connected networks, in this test we ignore the directions of links [20]. In Figure 4.6, we show the results and compare them with the random network which is known have very compact structure i.e., both the diameter and the average path length of a random network can be bounded by $O(\log n)$ [19]. We can find that the results under the uniform weight setting are close to that of the random network. The networks with the power-law weight setting seem to have more compact topologies.

Moreover, the diameters and the average distances of the networks under both of the weight settings increased by at most 2 while the network size increased by 10 times. Therefore, the results show good scalability of the proposed network which guarantees the network topology to be compact even the number of peers becomes very large.

4.4.2 Evaluation of the workload allocation

In this subsection, we evaluate the performance of the whole system. A simple 'one hop flooding' search protocol is adopted in this simulation [13].

- Each peer periodically disseminates the indices of its objects to all of its out-neighbours every T_I time units.
- Each index has an initial lifetime set to T_I. The lifetime decreases by 1 per time unit. An index is deleted when its lifetime is decreased to 0.
- When a search query is generated, the searcher disseminates query messages to all of its out-neighbours. If any one of its out-neighbours has the replica or the index of the target object, the search success.

The basic simulation setting is as follows: $n = 10000$, $T = 10$, $d = 10$, $K = 5$, $T_I = 20$ and $n_{sp} = 500$ for FL. Peers' capacity distribution follows a power-law distribution given by

$$\forall v_i \in V,\ c_i = \frac{10^5}{\sqrt{i} + 5} - 940$$

which is approximatively consistent to the bandwidth distribution in real P2P systems [25][26]. The workload of each peer is estimated by the number of the messages it receives in each time unit which includes search queries, indices, link exchange requests and control packages of the workload management protocols. There are 10000 different objects in the system. Each object, denoted by o_x, is searched for $f_x = L \cdot 10/x$ times (i.e., following a Zipf-distribution) by randomly selected peers in each time unit, where $x, 1 \leq x \leq 10000$ is the popularity rank of the object and L is a parameter which adjusts the total workload of the system. Each object o_x has $\lceil 100/x \rceil$ replications which are stored in randomly selected peers. Notice that our framework is independent of the search protocol and the system environment. The above setting aims to evaluate the performance of our framework in the system environment close to real P2P systems.

In the following tests, the initial network topology adopts the 10-out-regular random network. AL and ALCP initially have 500 SPs as same as FL. Considering the system workload changes by time, the average workload of each peer is estimated by the simple moving average. In the following tests, the average load rate of a peer is the average of the transient load rate in the last 10 time units.

Search performance

Table 4.3 compares the hit rate (i.e., ratio of succeeded queries), the number of SPs and the average overload rate (i.e., the ratio of transient overloaded SPs) of the of the five WA strategies.

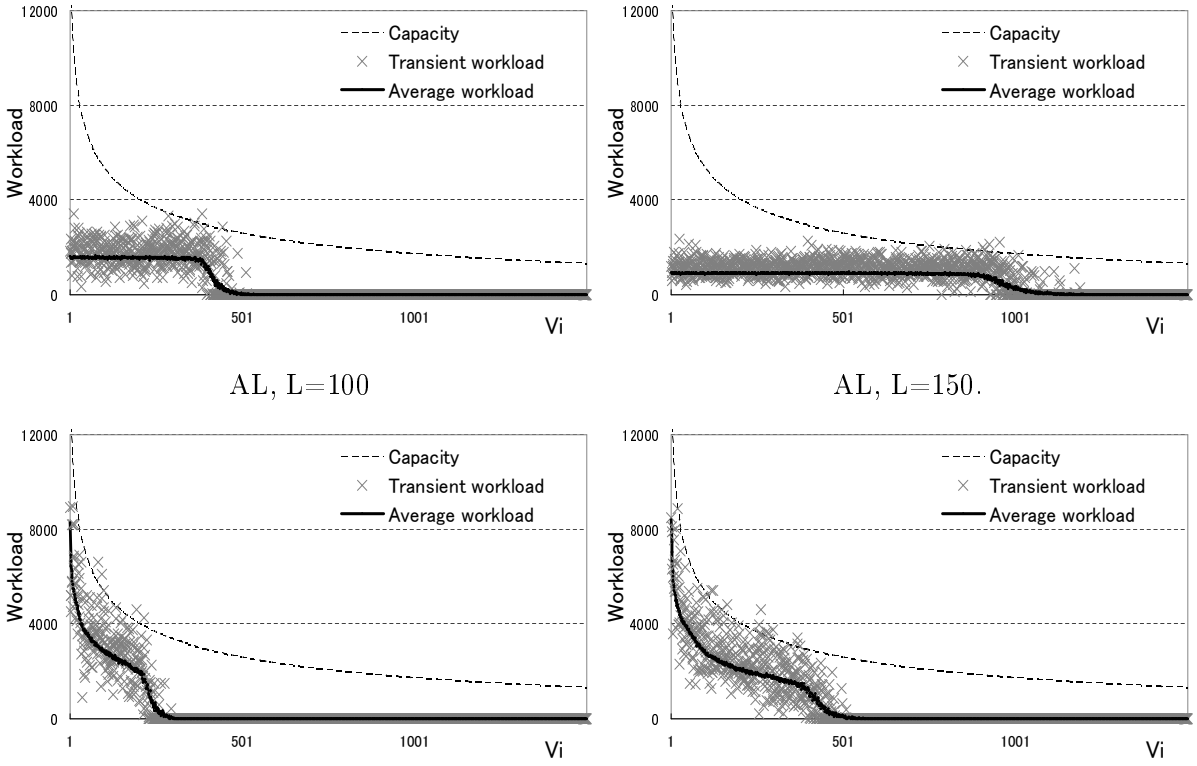
Table 4.3: Performance comparison (T=10, η=0.5, ε=0.1).

UN:	L=50	L=100	L=150	L=200
Hit Rate	16.5%	N/A	N/A	N/A
Overload Rate	7.6%	14.7%*	20.6%*	31.2%*
CP:	L=50	L=100	L=150	L=200
Hit Rate	30.1%	30.1%	30.1%	30.1%
Overload Rate	0%	0%	0.1%	0.6%
FL:	L=50	L=100	L=150	L=200
Hit Rate	60.3%	60.3%	60.3%	N/A
# SP	500	500	500	500
Overload Rate	0%	0.2%	6.8%	26.8%*
AL:	L=50	L=100	L=150	L=200
Hit Rate	82.6%	63.2%	49.6%	N/A
# SP	128	417	968	10000
Overload Rate	0%	0.47%	0.7%	31.2%*
ALCP:	L=50	L=100	L=150	L=200
Hit Rate	89.4%	73.9%	64.6%	58.2%
# SP	87	233	433	655
Overload Rate	1.2%	1.7%	2.2%	3.2%

The symbol '*' in the overload rate indicates some peers constantly overloaded so that the results are not available (N/A) in such cases. Clearly ALCP has the highest hit rate and the least number of SPs in all cases. It has better hit rate than AL even adopting the same number of SPs, e.g., the hit rate of ALCP in the case $L = 150$ is higher than that of AL in the case $L = 100$.

AL also has good performance in low-load environments. However, as we mentioned in Section 4, it cannot fully utilize all peers' capacities so that the system itself overloads in high-load environments (the case $L = 200$). In this case, the system adds SPs to share the workload of current overloaded SPs but those newly added powerless SPs overload instead. Finally, all peers are promoted to be SPs and thus the workload distribution becomes the same as UN.

Notice that the overload rate of ALCP is essentially different from that of AL, FL and UN. In ALCP, when the number of SPs increases, more powerless peers become SPs. Those powerless peers are easy to be transiently overloaded because their load rates are greatly affected by the



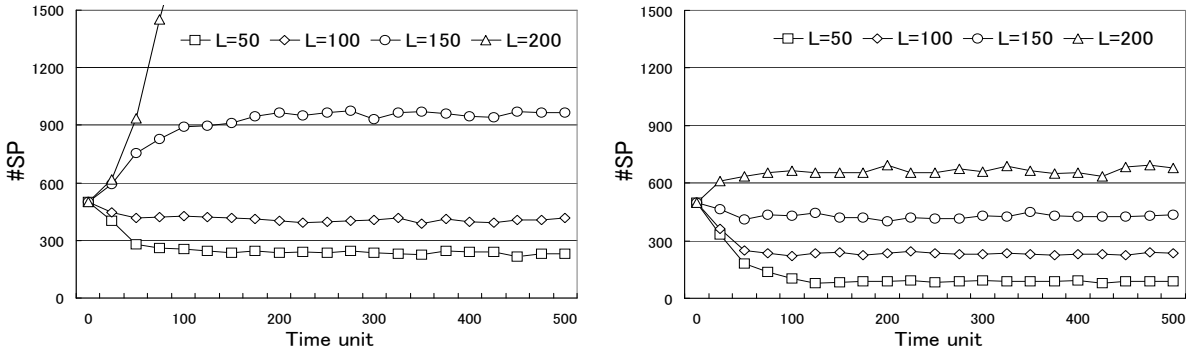
AL, L=100 AL, L=150.

ALCP, L=100. ALCP, L=150.

Figure 4.7: Transient workload, zoomed in $v_1 \sim v_{1500}$.

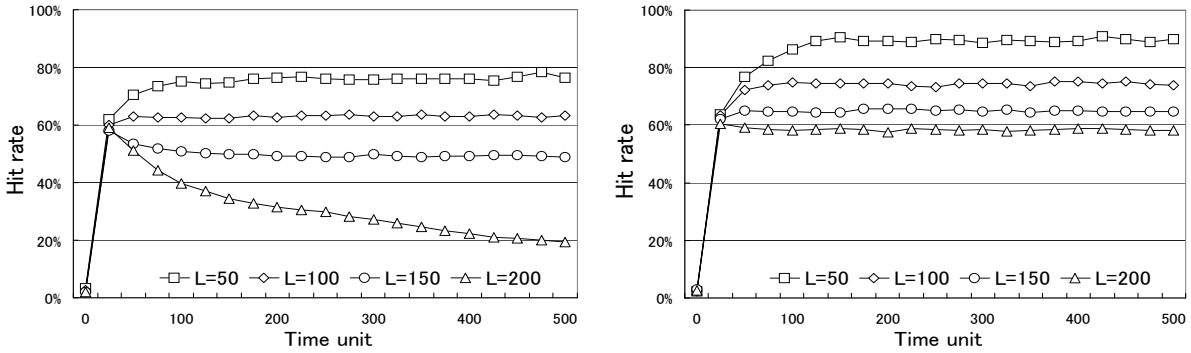
random variance of the workload allocated to them. Therefore, in ALCP, SPs hardly constantly overload even the transient overload rate is high. However in AL, FL and UN, all SPs take on the same amount of workload so that powerless peers have higher load rates than powerful ones. Therefore, some powerless SPs may constantly overload even the transient overload rate is low.

In Figure 4.7, we show the average workload and the transient workload of AL and ALCP in low-load and high-load environments respectively. The tests are executed for 1000 time units starting from different initial network topologies. The average workload is taken from the last 500 time units and the transient workload is taken from the snapshots at the end of the 1000-th time unit. Clearly in all figures we can find that around $v_{n_{sp}}$ the average workload of rapidly decreases to 0. That implies our protocols select the most powerful peers to be SPs. We can also see that the workload distributions faithfully obey the WA strategies i.e., all SPs have the same workload in AL and workload of SPs in ALCP is proportional to their capacities. The results also show the peers' workload have large random variance (which is ineluctable in unstructured P2P systems) but only a few peers transiently overload. Notice that, different from the variance of



AL. ALCP.

Figure 4.8: Number of SP



AL. ALCP.

Figure 4.9: Hit rate.

the access strategy VAR[A], the random variance does not improve search performance because even if a peer transiently has a high in-degree and thus a high probability to receive the index of an object, the random variance does not guarantee it has a high in-degree when the query of the object is disseminated.

Self-adaptability

Starting with 500 SPs, AL and ALCP adjust the number of SPs adapting to the total workload of the system. We show in Figs.4.8 and 4.9 the change of the number of SPs and the hit rates of them during the simulation period. We can see that the protocols can stabilize in a short time except for the case of AL when $L = 200$. Of course, if we adopt a smaller T, the stabilization time can be furtherer shortened. The good adaptability guarantees the performance of the system when peers frequently join and leave. It also enables peers to control the workload allocated to them by adjusting their capacities.

Table 4.4: Impact of η. (T=10, L=100 ε=0.1).

AL:	η=0.3	η=0.4	η=0.5	η=0.6	η=0.7
Hit Rate	58.6%	60.8%	63.2%	63.7%	64.2%
# SP	543	467	417	381	367
Average Load	33.6%	35.9%	38.1%	39.3%	39.8%
Overload Rate	0.18%	0.43%	0.47%	0.51%	0.54%
ALCP:	η=0.3	η=0.4	η=0.5	η=0.6	η=0.7
Hit Rate	72.4%	72.9%	73.9%	75.4%	76.3%
# SP	269	249	233	218	214
Average Load	51.8%	53.4%	55.5%	58.1%	59.2%
Overload Rate	1.48%	1.61%	1.73%	2.32%	3.21%

Table 4.5: Impact of ε (T=10, L=100, η=0.5).

AL:	ε=0.06	ε=0.08	ε=0.1	ε=0.12	ε=0.14
Hit Rate	67.3%	64.6%	63.2%	62.2%	60.2%
# SP	313	375	417	430	480
Average Load	45.1%	40.2%	38.1%	36.7%	34.5%
Overload Rate	2.87%	0.79%	0.47%	0.23%	0.20%
ALCP:	ε=0.06	ε=0.08	ε=0.1	ε=0.12	ε=0.14
Hit Rate	78.6%	76.9%	73.9%	73.6%	72.3%
# SP	180	201	233	241	268
Average Load	66.0%	59.8%	55.5%	53.9%	49.1%
Overload Rate	5.86%	2.48%	1.73%	1.26%	1.03%

Impact of η and ε

Tables 4.4 and 4.5 show the impact of the parameters η and ε. Both of the two parameters affect the number of SPs as our design. It seems that ε has larger impact than η. That is because the number of SPs is usually small and only powerless SPs execute the Demotion operation so that the number of Demotion events happen in each time unit is quite limited but much more Promotion events can happen unless ε is too small. Therefore, even in some cases the 'low load rate' condition is always satisfied (i.e. in the cases that η is larger than the average load rate in ALCP) the system can keep a low overload rate because many LPs are promoted against the frequent Demotion events. That also implies the number of SPs stabilizes in a dynamic state that the same number of Promotion and Demotion events happen in each time unit.

From the results, the trade-off between the search performance and overload rate can be clearly seen. A higher average load rate achieves larger variance in workload distribution and

thus higher search performance. However, it also incurs higher overload rate because of the random variance in the amount of workload allocated to each peer. In real P2P systems, the distribution of the transient workload is difficult to estimate in advance because it depends on the variance of each peer's transient in-degree, users' behaviours and the type of the application. Moreover, the acceptable transient overload rate also depends on applications. Therefore, it is hard to say how much is the optimal average load rate and the optimal value of η and ε. We test a wide range of values of those parameters and all of the results seem acceptable. Therefore, we can initially set up a system without difficulties by setting those parameters to some intuitive values.

4.5 Related Works

The performance of unstructured search algorithms can be improved by optimizing the network topology to the heterogeneity of peers. The well-known heterogeneous features of peers in real P2P systems include the peers' capacities, peers' interests and the distance between peers in physical networks. Among them, the most biased feature should be the peers' capacity which is proved to follow power-law distributions.

Our work focuses on the heterogeneity of peers' capacities and maximizes the search performance by concentrating search workload to powerful peers. The WA strategies in our work are realized by constructing degree-weighted networks. The heterogeneity in peers' interests and locations can also be utilized for improving the search performance. For example, some works form cluster of peers with similar interests so that a peer can easily find its target objects by disseminating search queries to neighbouring peers [36][37]. Some other works connect peers of which the locations are close in the physical network [38][39]. Such networks incur less overhead in physical networks than random networks even the communication cost in the overlay network is the same. They can also decrease the economic liability of ISPs [40]. Unfortunately, the design purposes of the above three kinds of networks are not consistent e.g., a peer may not be able to find a powerful peer close to it. Although the networks of modern P2P applications are usually combinations of these networks, they are in fact the trade-off among those design purposes [15][14][9]. Therefore, it is unclear if those combined networks are more efficient than the ALCP which draws out the best search performance of the degree-weighted network.

4.6 Concluding Remarks

We showed that the search performance of unstructured P2P systems can be improved by concentrating both object informations and search queries, and thus the search workload, to a part

of peers. This approach is contrary to traditional design ideas of balancing the search workload among all peers. Considering the large biases in peers' capacities, the capacity-aware WA is clearly more reasonable than the uniform WA. Under the restriction that all peers do not overload, we present the ALCP which has the most biased workload distribution. It fully utilize the available capacity of the most powerful peers to maximize the search performance. Notice we define the available capacity of a peer as a part of its physical capacity which is contributed by the peer for system use. That implies we allow peers to decide the maximum workload to take on but not forcibly expropriate their hardware resource. We hope this design can urge powerful peers to stay in the system longer and contribute their hardware resource use as best they can. It is also favorable for introducing incentive mechanisms [41].

A distributed framework is presented to realize the ALCP and other traditional WA strategies. The lower layer of the framework constructs a dynamic network in which each peer's in-degree, and thus the workload, is proportional to its weight. The upper layer includes five workload management protocols which decide peers' weights according to the five WA strategy respectively. Those protocols use only a few local workload situation informations so that their communication cost is negligibly small.

By simulation, we proved that both the network construction protocol and the workload management protocols perform as our design objective. The simulation results also show the ACLP has obviously higher hit rate and larger system capacity than traditional approaches.

Chapter 5

A Message-efficient Search Protocol

Unstructured search approaches are widely used because of their flexibility and robustness. However, such approaches incur high communication cost. The index-dissemination-based search is a kind of efficient unstructured search approach. In this chapter we study such approaches with respect to decrease the communication cost incurred by the index-dissemination-based search protocols. Under the Churn model that peers continuously join and leave, we solve two subproblems. One is how to efficiently disseminate and maintain a given number of indices. For this subproblem we present the *Stream method* which averagely disseminates the same number of indices in each time unit. It can minimize the negative impact of the loss of indices when their holders leave the system. Another one is to determine the optimal number of indices for each object of a given popularity. For this subproblem we present the *Equal Rule* which shows that the total communication cost related to each object can be minimized by adjusting the index dissemination cost equally to the query dissemination cost. We propose a distributed protocol to realize the optimal index dissemination scheme in a self-adaptive manner. A remarkable advantage is that the protocol yields almost no additional communication cost to achieve the self-adaptive feature.

This Chapter is organized as follows. In Section 5.1, we introduce the system model which includes a general Churn model. In Section 5.2, we study the optimal index dissemination scheme by theoretical approaches. In Section 5.3, we propose a distributed protocol to realize the optimal index dissemination. In Section 5.4, we evaluate the protocol by simulation. In Section 5.5, we discuss some supplemental issues. Finally in Section 5.6, we give concluding remarks.

5.1 Preliminaries

5.1.1 System model

A P2P system is defined by a dynamic set of peers in which peers join and leave continually. In time unit t, $m(t)$ peers join the system. When a peer joins the system, it is assigned a random lifetime L drawn from some distribution $l(\tau, t)$ i.e., in time unit t, $\Pr[L = \tau] = l(\tau, t)$, $l(\tau, t) \geq 0$ for any τ and $\sum_{\tau=0}^{\infty} l(\tau, t) = 1$ for any t [42][43]. The lifetime distribution can be arbitrary as long as the expectation $E[L] = \sum_{\tau=1}^{\infty} \tau \cdot l(\tau, t)$ is finite. The lifetime of each peer decreases by 1 per time unit. After the lifetime decreased to 0, the peer leaves the system. Re-joined peers are regarded as newly-joining peers i.e., if a peer leaves the system, its historical information is vanished.

There are some objects $\{a, b, c...\}$ in the system. Each object is independent i.e., the copies of the same data item are regarded as the same object. The popularity $f_x(t)(\geq 0)$ of object x is defined by the the total number of times that x (including all copies of x) is searched during time unit t. The popularity of each object is independent of the others.

We assume an ideal random sampling service that a peer can send messages to peers selected from the system uniformly at random i.e., each peer is selected with the probability $1/n$ where n is the number of peers. That implies the system adopts the uniform workload allocation. The service can be realized by adopting random walk on a uniform-random network or on the PWDN with the uniform weight setting. The communication cost for the sampling service is the network construction cost which is fixed for a given number of peers so that each sampling incurs a unit cost. The random peer sampling assumption is only necessary for theoretical analysis. We will show latter that our protocol works well with non-ideal sampling services.

For simple presentation, we measure the communication cost by the number of transferred messages. The term 'message cost' is used instead of the term 'communication cost'. This metric is reasonable because the sizes of messages used in index-dissemination-based search protocols are almost equal regardless of their types i.e., search queries or indices. Notice the message cost is the logical communication cost on an overlay network. It does not represent physical distance between peers. One can consider that the message cost is the average physical communication cost for delivering a message between any two peers in the network.

Finally, we introduce some notions which will appear in the following of this chapter. A peer which currently attend the system is called an *active* peer. If an index is stored in an active peer and points to an active owner of the object, we say the index is *available*. The variables (functions) $m(t), l(\tau, t)$ and $f_x(t)$ are called *environment parameters*. The environment parameters are not known by any peers.

5.1.2 Index-dissemination-based search

The index-dissemination-based search under the random sampling service model is described as follows.

- Index dissemination: Each peer disseminates some indices of its objects to some other peers selected uniformly at random from the system.
- Search process: The searcher sends a query messages to a peer selected uniformly at random. If the query message is received by the peer which holds an index of the target object (or the object itself), the search process succeeds. Otherwise, the searcher sends the query to another peer. This process is repeated until the target object is found.
- Index maintenance: Each index is given a initial TTL value T_I. An index is deleted when its lifetime is expired. Indices are maintained by periodical re-dissemination by the owner of the object.

We show some mathematical results of this model below:

Lemma 5.1.1
Let n, q and p respectively be the number of peers, the number of available indices of an object in the system and the number of query messages used to search for the object. The success probability that the searcher find the target object is at least $1 - e^{-qp/n}$.

Proof: Since query messages are sent to the peers selected uniformly at random, each query finds the target object' index with a probability q/n. Thus the object can be found with a probability

$$\begin{aligned} \rho &= 1 - (1 - q/n)^p \\ &\geq 1 - e^{-qp/n}. \end{aligned}$$

□

Lemma 5.1.2
Let n, q and p respectively be the number of peers, the number of available indices of an object in the system and the query messages used until the index of the target object is found. The expectation of p is $\mathrm{E}[p] = n/q$.

Proof: The probability that the first index of the object is found by the exactly the k th probing is

$$\Pr[p = k] = (1 - q/n)^{(k-1)} \cdot q/n.$$

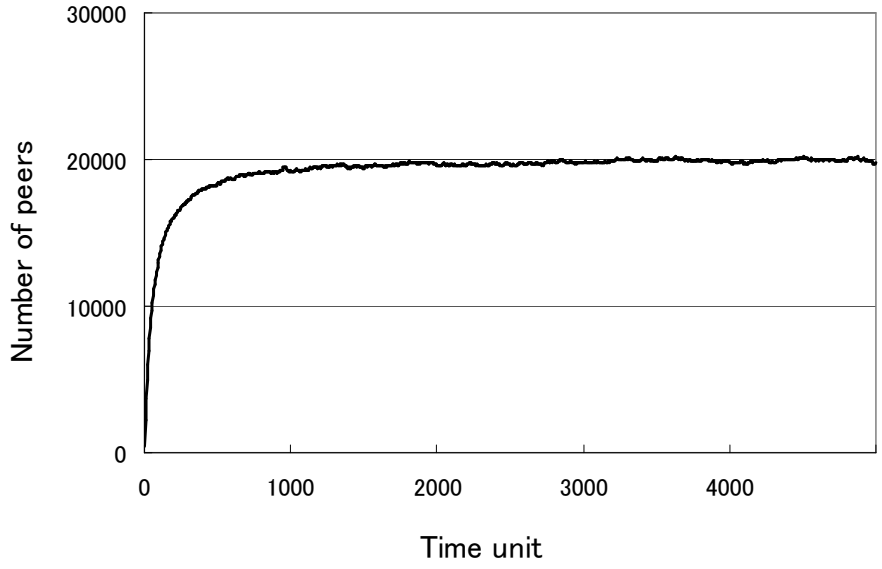


Figure 5.1: The number of peers in a newly created system. $m = 400$; Pareto distribution for peers' lifetime: $\text{Prob}[L \leq \tau] = 1 - (1 + \tau/50)^{-2}$ that implies $E[L] = 50$, $n = 20000$.

That implies there must be $k-1$ failed probes followed by the successful one. Thus, the random variable p follows a *geometric distribution* that $\text{E}[p] = n/q$ [44]. □

5.2 Optimization of Index Dissemination

In this section, we study the optimal index dissemination scheme in a stable system environment. We say a system environment is stable if $m(t) = m, m > 0$; $l(\tau, t) = l(\tau), l(\tau) \geq 0$; $f_x(t) = f_x$, $f_x > 0$. In a stable system environment, if t is large enough, the expected number of the peers join in time unit $t - i$ is $m \sum_{\tau=i}^{\infty} l(\tau)$. So the number of peers in a stable system converges to $n = m \sum_{i=1}^{\infty} \sum_{\tau=i}^{\infty} l(\tau)$. Then, we have

$$n = m \sum_{i=1}^{\infty} \sum_{\tau=i}^{\infty} l(\tau) = m \sum_{\tau=1}^{\infty} \sum_{i=1}^{\tau} l(\tau) = mE[L]. \tag{5.1}$$

As shown in Figure 5.1, if m and t are large enough, we can approximately consider the number of peers is fixed to n.

5.2.1 Formulation of the system message cost

We consider the system message cost consists of the search cost and the index maintenance cost of all objects in the system. Notice the cost for the random peer sampling service is not

omissible. However, as we mentioned in Section 5.1, those algorithms work pro-actively and that their cost is fixed for each peer [20][13]. Therefore, the sampling cost does not affect the trade-off between index maintenance cost and search cost. For simple presentation, we do not count it in the following of this chapter.

Due to independence of the message cost related to each object, the system message cost is minimized iff the message cost related to each object is minimized. Therefore, in the following of the chapter, we focus on how to minimize the total message cost related to an single object x.

Definition 5.2.1 *(Search size).*
The search size, denoted by $p_{x,s}(t)$, is the number of search queries each searcher s uses to find object x in time unit t. The search size $p_{x,s}(t)$ is a random variable.

Definition 5.2.2 *(Search cost).*
The search cost, denoted by $s_x(t)$, is the total number of query messages used to find object x during time unit t. That is, $s_x(t) = \sum_{\forall s} p_{x,s}(t)$. Since $p_{x,s}(t)$ is a random variable, $s_x(t)$ is also a random variable.

Definition 5.2.3 *(Index maintenance cost).*
The index maintenance cost, denoted by $q_x(t)$, is the number of the indices for object x disseminated during time unit t.

Definition 5.2.4 *(Message Cost).*
The message cost, denoted by $m_x(t)$, is the sum of the index maintenance cost $q_x(t)$ and the search cost $s_x(t)$ of object x during time unit t. That is, $m_x(t) = q_x(t) + s_x(t)$. Since $s_x(t)$ is a random variable, $m_x(t)$ is also a random variable.

By the definitions, we obtain the message cost of object x is $m_x(t) = q_x(t) + \sum_{\forall s} p_{x,s}(t)$. Letting $M_x(t)$, $S_x(t)$ and $P_x(t)$ be the expectations of $m_x(t)$, $s_x(t)$ and $p_{x,s}(t)$ respectively, we obtain

$$M_x(t) = q_x(t) + S_x(t) = q_x(t) + f_x \cdot P_x(t). \tag{5.2}$$

Notice that no matter which peer is the searcher, the expected search size is the same because each searcher sends query messages to randomly selected peers in the system.

Table 5.1: Symbols' definition, time unit t.

n	The number of peers in the system.
m	The number of peers join the system in each time unit.
T_I	The initial lifetime of an index.
$l(\tau)$	The lifetime distribution of peers.
$d(\tau)$	The probability of peers leave after τ time units.
f_x	The popularity (search frequency) of object x.
$p_{x,s}(t)$	(A random variable). The search size for searcher s to find object x.
$P_x(t)$	The expectation of $p_{x,s}(t)$.
$s_x(t)$	(A random variable). The search cost of object x, $s_x(t) = \sum_{\forall s} p_{x,s}(t)$.
$S_x(t)$	The expectation of $s_x(t)$, $S_x(t) = f_x \cdot P_x(t)$.
$q_x(t)$	The index maintenance cost of object x.
$m_x(t)$	(A random variable). The message cost of object x, $m_x(t) = q_x(t) + s_x(t)$.
$M_x(t)$	The expectation of $m_x(t)$, $M_x(t) = q_x(t) + S_x(t) = q_x(t) + f_x P_x(t)$.

5.2.2 Index dissemination method

A disseminated index may disappear in two cases. One case is that the index's TTL value is expired. Anther case is that the peer which stores the index leaves the system. Therefore, the number of available indices for each object is decided by not only how many but also when those indices were disseminated.

The leave of peers can be described as follows. In time unit t, the expected number of peers with lifetime τ is $\eta(\tau) = m \sum_{i=0}^{\infty} l(\tau + i)$ where $ml(\tau + i)$ is the expected number of peers joined in time unit $t - i$. Those $\eta(\tau)$ peers will leave the system in time unit $t + \tau$. We define a damping function by $d(\tau) = \sum_{i=1}^{\tau} \eta(i)/n$ which indicates the probability that peers in the current system leave after τ time units. Clearly, $d(\tau)$ is monotonically increasing and $0 \leq d(\tau) \leq 1$ for any τ. Notice $\eta(\tau)$ and $d(\tau)$ are independent of t.

For case study, we analyze two index dissemination methods: the *burst method* which is used in the quorum-based search protocol [13], and the *stream method* we newly proposed. In the burst method, the owner of an object x disseminates Q_x indices once per T_I time units (called a TTL cycle). In this case, the number of available indices decreases during each TTL cycle. In the τ-th time unit of each TTL cycle, the expected number of available indices $q_x^B(\tau)$ is

$$q_x^B(\tau) \quad = \quad Q_x \cdot (1 - d(\tau)). \tag{5.3}$$

In contrast, in the stream method, the owner disseminates Q_x/T_I indices in each time unit. In

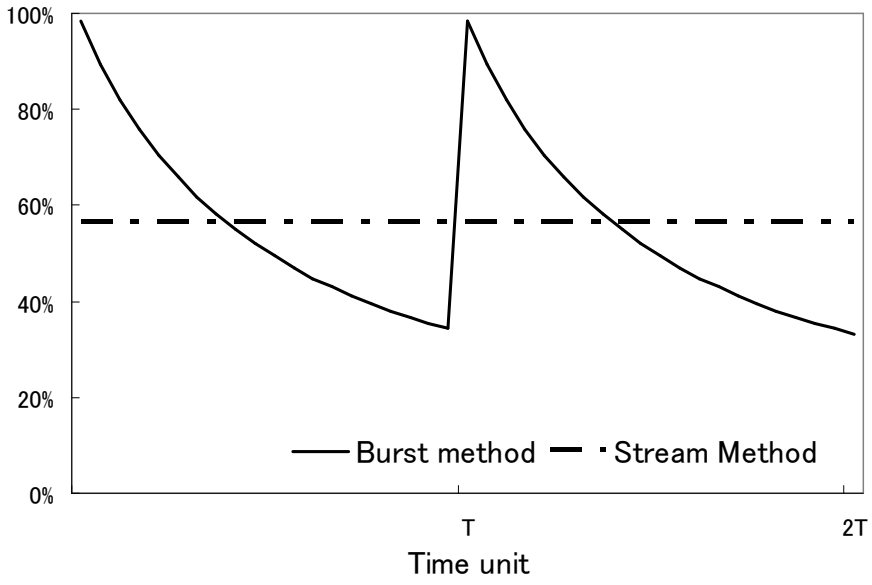


Figure 5.2: The percentage of the number of available indices number, $T_I = 50$, $m = 400$; Pareto distribution for peers' lifetime: $\text{Prob}[L \leq \tau] = 1 - (1 + \tau/50)^{-2}$.

this case, the expected number of available indices $q_x^S(t)$ is fixed that

$$q_x^S(t) = \sum_{\tau=1}^{T_I} (1 - d(\tau)) \cdot Q_x / T_I. \tag{5.4}$$

Figure 5.2 shows the percentage of the number of available indices adopting two index dissemination methods. The same number of indices, marked by 100%, are disseminated in each TTL cycle.

Let M_x^B and M_x^S be the average message cost of the burst method and the stream method respectively. According to lemma 5.1.2 and equality 5.2,we obtain:

$$\begin{aligned} M_x^B &= \frac{Q_x}{T_I} + \frac{1}{T_I} \sum_{\tau=1}^{T_I} \frac{f_x n}{Q_x \cdot (1 - d(\tau))} \\ &= \frac{Q_x}{T_I} + \frac{f_x n}{Q_x T_I} \sum_{\tau=1}^{T_I} \frac{1}{(1 - d(\tau))}. \end{aligned} \tag{5.5}$$

$$\begin{aligned} M_x^S &= \frac{Q_x}{T_I} + \frac{f_x n}{\sum_{\tau=1}^{T_I} (1 - d(\tau)) \cdot Q_x / T_I} \\ &= \frac{Q_x}{T_I} + \frac{f_x n T_I}{Q_x} \frac{1}{\sum_{\tau=1}^{T_I} (1 - d(\tau))}. \end{aligned} \tag{5.6}$$

$$\begin{aligned} M_x^B &= \frac{Q_x}{T_I} + \frac{f_x n}{T_I Q_x} \sum_{\tau=1}^{T_I} \frac{1}{(1-d(\tau))} \\ &\geq \frac{Q_x}{T_I} + \frac{f_x n}{Q_x} \frac{1}{\sqrt[T_I]{\prod_{\tau=1}^{T_I}(1-d(\tau))}} \\ &\geq \frac{Q_x}{T_I} + \frac{f_x n T_I}{Q_x} \frac{1}{\sum_{\tau=1}^{T_I}(1-d(\tau))} \\ &= M_x^S. \end{aligned}$$

Therefore, the stream method achieves lower search cost than the burst method even the index maintenance cost is the same. Actually, we can show that the stream method is the best index dissemination scheme in the sense that it minimizes the expected search cost.

Theorem 5.2.1
The stream method is the optimal index dissemination method that minimizes the expected search cost under a fixed index maintenance cost.

Proof: We assume the number of indices being disseminated in each time unit follows a periodic function $g(\tau), 1 \leq \tau \leq \Gamma$, where Γ is the cycle of $g(\tau)$. Without loss of generality, we assume $\Gamma > T_I$. (By combining several consecutive short cycles, we can regard $g(t)$ as a function of a long cycle.) The expected number of available indices in the τ-th time unit, denoted by $a(\tau)$, is

$$a(\tau) = \sum_{t=\tau-T_I}^{\tau-1} g(t)(1-d(\tau-t)), \tag{5.7}$$

where a non-positive time label t indicates the $(\Gamma - t)$th time unit of the previous cycle. Clearly, for any τ, $a(\tau) \geq 0$.

Letting q be the fixed average number of the indices being disseminated in each time unit, we obtain

$$\sum_{\tau=1}^{\Gamma} g(\tau) = \Gamma \cdot q. \tag{5.8}$$

By Equality 5.7 and 5.8, we obtain

$$\sum_{\tau=1}^{\Gamma} a(\tau) = q\Gamma \sum_{t=1}^{T_I} (1-d(t)).$$

Letting S_x be the sum of the expected search cost in each cycle, we obtain:

$$\begin{cases} S_x = \sum_{\tau=1}^{\Gamma} f_x \cdot n/a(\tau) \\ \sum_{\tau=1}^{\Gamma} a(\tau) = q\Gamma \sum_{t=1}^{T_I}(1-d(t)). \end{cases} \tag{5.9}$$

Notice $\sum_{\tau=1}^{\Gamma} a(\tau)$ is a finite constant which is independent of both t and $g(t)$. Therefore, by basic inequalities, we can obtain

$$S_x \geq f_x \cdot n \cdot \Gamma / q \sum_{t=1}^{T_I} (1 - d(t)).$$

Equality holds when

$$\forall \tau (1 \leq \tau \leq \Gamma),\ a(\tau) = q \sum_{t=1}^{T_I} (1 - d(t)).$$

Therefore, the search cost is minimized when the number of available indices is uniform in each time unit. Clearly, it can only be achieved by the stream method. Notice the theorem holds even if we consider non-periodical dissemination methods because the same argument is possible if Γ is infinitely long. □

5.2.3 Optimal index number

Next, we investigate the minimum message cost when adopting the stream method. By Equality 5.6 and the basic inequality $x + C/x \geq 2\sqrt{C}$, we obtain the minimum message cost $\min[M_x]$ of object x that

$$\min[M_x] = 2\sqrt{\frac{f_x n}{\sum_{\tau=1}^{T_I} (1 - d(\tau))}}. \tag{5.10}$$

Then, the optimal number of indices, denoted by $\hat{q_x}$, to be disseminated in each time unit is

$$\hat{q_x} = \sqrt{\frac{f_x n}{\sum_{\tau=1}^{T_I} (1 - d(\tau))}}. \tag{5.11}$$

Equality 5.10 shows the theoretical lower bound of the total message cost. The result indicates that there can not be any implementations of the random sampling service or any optimizing strategy of index dissemination can solve the search problem with less cost, as long as the system accord with the uniform-random sampling model.

5.3 A Self-adaptive Protocol

In Section 5.2, we obtained the optimal index number $\hat{q_x}$. However it can not be directly computed from Equality 5.11 because m, $l(\tau)$ and f_x are not known by any peer. In this section, we propose a self-adaptive protocol that implements the optimal index-dissemination without those global parameters.

5.3.1 The Equal Rule

Theorem 5.3.1 *(The Equal Rule).*
The message cost of an object is minimized when its index maintenance cost equals to its search cost.

Proof: Let $\hat{s_x}(t)$ be the expected search cost of x when $q_x(t) = \hat{q_x}$. According to Equality 5.10 and 5.11, we obtain

$$\hat{s_x}(t) = \min[M_x] - \hat{q_x} = \hat{q_x}. \tag{5.12}$$

□

The Equal Rule indicates that, if we disseminate the same number of indices as the number of the search queries disseminated in each time unit, the total message cost is minimized. By the Equal Rule, we obtain the skeleton of our protocol below:

- Search: When searching for an object, the searcher repeatedly sends query messages to randomly selected peers until the object is found. During the search, the searcher counts the number of query messages used.
- Index maintenance: After the search succeeds, the searcher disseminates the same number of indices to some randomly selected peers. Each index has a lifetime counter which is increased per time unit. An index will be deleted when its lifetime counter exceeds the predefined TTL value.

Notice that we allow the searchers to disseminate the indices. Since the searcher usually downloads the target object after finding it, the indices disseminated by the searcher can include anyone of two object locations; the searcher or the original owner. Such flexibility is favorable in terms of load balancing.

5.3.2 Index dissemination schemes

The following factors should be considered when each peer disseminates indices. First, as we show in Theorem 5.2.1, the search cost is minimized when the number of available indices is stable. However, it may make the number of available indices instable to disseminate straightforwardly the same number of indices at each time unit because of the fluctuation of search cost by the random noise effect. Second, the system environment parameters are usually not static, even change rapidly at sometimes. For example, when an object becomes a hot spot, its popularity,

together with the search cost, drastically increases in a short period of time [45]. The number of indices should adapt to such changes.

By referring the statistical estimation methods, we propose three approaches for deciding the number of indices to be disseminated.

- RT (Real-Time) mode:

$$q_x(t) = s_x(t-1).$$

- SMA (Simple Moving Average) mode:

$$q_x(t) = \sum_{\tau=1}^{T_I} s_x(t-\tau)/T_I.$$

- EMA (Exponential Moving Average) mode:

$$q_x(t) = \sum_{\tau=1}^{T_I} s_x(t-\tau) \cdot 2^{-\tau}.$$

The above descriptions indicate how we use the historical information of search cost to decide the number of disseminated indices in the global view. In the followings, the index dissemination schemes from the viewpoint of each searcher are described. After the searcher s completes the search process for object x by $p_{x,s}(t)$ query messages in time unit t, it disseminates some indices, denoted by $q_{x,s}(t+\tau), 0 \leq \tau \leq T_I - 1$, in the following T_I time units:

- RT mode:

$$\begin{cases} q_{x,s}(t+\tau) = p_{x,s}(t), & \tau = 0 \\ q_{x,s}(t+\tau) = 0, & \tau > 0. \end{cases}$$

- SMA mode:

$$q_{x,s}(t+\tau) = p_{x,s}(t)/T_I, \quad 0 \leq t \leq T_I - 1.$$

- EMA mode:

$$q_{x,s}(t+\tau) = p_{x,s}(t) \cdot 2^{-1-\tau}, \quad 0 \leq t \leq T_I - 1.$$

Obviously, there is a trade-off between the stability and adaptability. The RT mode has the fastest adaptation speed when the system environment parameters change. However it works in the stream method only when the object is frequently searched. In the contrast, the SMA mode can stabilizes the system well because the indices are disseminated averagely during the following T_I time units after each search event. However it may not be able to adapt to a highly dynamic system environment. The EMA mode is a middle approach between RT and SMA modes.

5.4 Simulation

In Section 3, we have the lower bound of the index-dissemination based search under the random peer sampling model. And in Section 4, we proposed a distributed protocol to achieve it. In this section, we compare the message cost of our protocol with the theoretical minimum message cost to justify its effectiveness. Unfortunately, we can not find any related works for comparative evaluation. For example, as we mentioned in Section 5.1, the quorum-based search and the square-root replication principle are not optimized for the total message cost, so fair comparison with them are impossible.

This section is divided to two parts. Subsection 5.4.1 justifies the adaptability of the protocol and compare the performance of the three index dissemination schemes. Subsection 5.4.2 justifies the practical impact of the protocol under realistic system environment settings.

5.4.1 Adaptability

According to the theoretical analysis, we know that the *Stream Method* and *Equal Rule* are the necessary conditions of the optimal index dissemination. However the theoretical results are obtained in a stable system environment. It is unclear how our protocol performs in unstable system environments because the *Equal Rule* is difficult to achieve in those cases. Moreover, because our protocol disseminate indices after each search, the search frequency decides the index dissemination timing that affects the *Stream Method*. This subsection evaluate the protocol with dynamic environment parameters and compare the performance of the three index dissemination schemes with some special simulation settings.

The environment parameters include $m(t), l(\tau, t)$ and $f_x(t)$. We mainly evaluate the protocol under dynamic settings of $f_x(t)$ because $m(t)$ and $l(\tau, t)$ do not vary quickly (often change in cycles of one day) in large-scale systems [7] and they do not affect the index dissemination timing. The simulation environments are as follows. In each time unit, 400 peers join the system (i.e., $m = 400$). The lifetime distribution of each peer is drawn from the Pareto distribution which is proved to be the peers' lifetime distributions in many real P2P systems [42]. The cumulative distribution function (CDF) of the distribution is $l_C(\tau) = \text{Prob}[L \leq \tau] = 1 - (1 + \tau/50)^{-2}$ which implies $E[L] = 50$. We execute the protocol under the ideal random sampling service to estimate the best performance of the proposed protocol. The initial lifetime of indices is set to $T_I = 50$.

This time we evaluate only one object x to show the difference of the three index dissemination schemes clearly. Notice our protocol minimize the message cost related to each object respectively, the distribution of objects' popularities do not affect the evaluation result. In time unit t, $f_x(t)$ searchers are selected randomly from the system. To have a stable result against

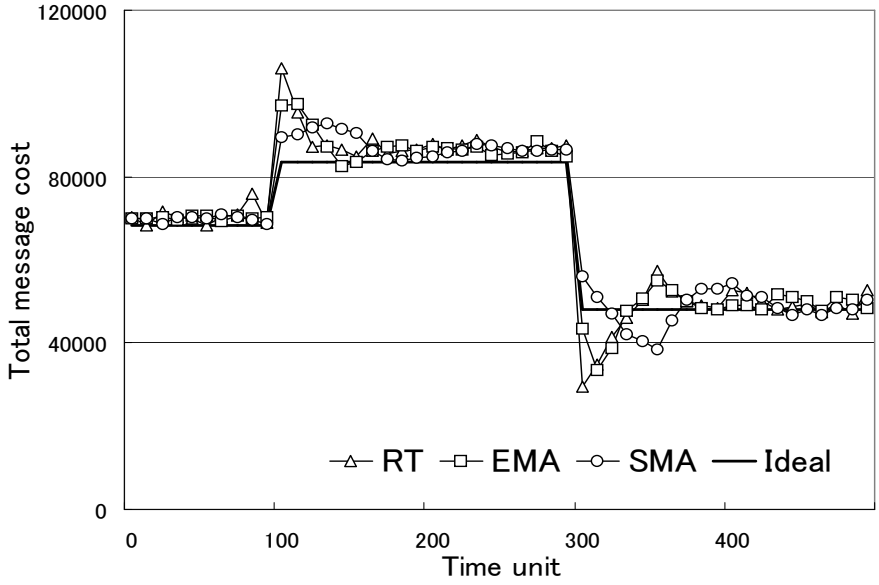


Figure 5.3: The total message cost (the sum of 1000 executions), $f_x(t) = 2$ for $t \in [0, 100)$; $f_x(t) = 3$ for $t \in [100, 300)$; $f_x(t) = 1$ for $t \in [300, 500)$.

the randomness of the protocol, we repeat the simulation for 1000 times and show the sum of the message cost in each independent execution.

The simulation results are shown by the total message cost in each time unit. In our protocol, the search process continues until the target object is found i.e., the success rate is always 1. Notice the result consists of both the maintenance cost (i.e., index dissemination cost) and search cost (i.e., query dissemination cost). Since the protocol is designed according to the *Equal Rule*, The maintenance and search cost occur exactly 50% of the total cost. Then from the results and $f_x(t)$, one can easily obtain the average search size and the number of indices disseminated. Such data will not be shown respectively for concise expression. For comparing, we show the theoretical minimum message cost by the curve 'Ideal'.

Figure 5.3 shows the results for discontinuous change of search frequency. All of the three schemes can converge to the theoretical minimum message cost and stabilize within $2T_I$ time units. The RT mode quickly responses but has the highest peak traffic. In Figures 5.4 and 5.5, the search frequency changes continuously. Figure 5.4 shows the message cost under a slowly changing $f_x(t)$. In this case, all the three methods work as expected. However, when the $f_x(t)$ changes rapidly (Figure 5.5), we can see all the three curves depart from the curve 'Ideal' and incur higher message cost. Especially the SMA mode consumes more messages than others. As Figure 5.6 shows, in each time unit, the number of index and query messages are quite different

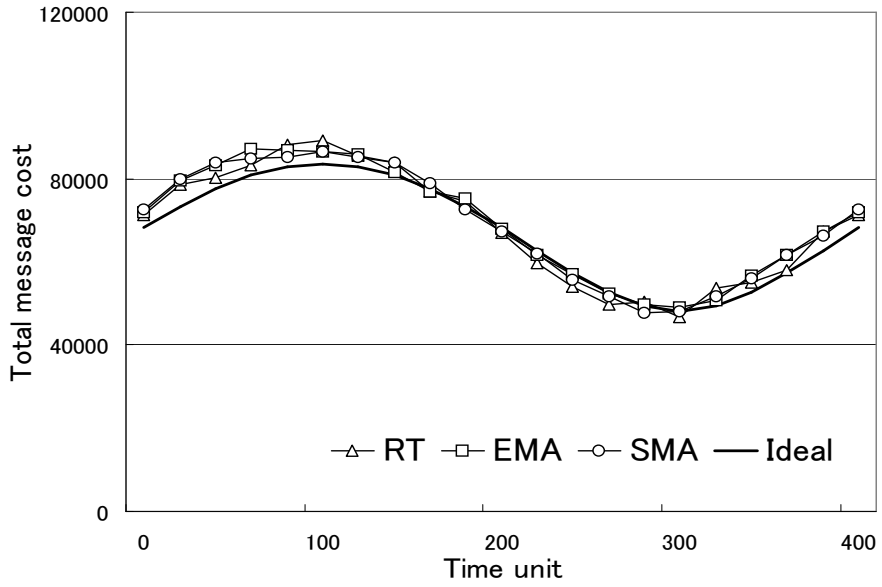


Figure 5.4: The total message cost (the sum of 1000 executions), $f_x(t) = 2 + \sin(2\pi t/400)$.

in the SMA mode although the total numbers are the same during the simulation. That implies the *Equal Rule* is not well achieved by the SMA mode in highly dynamic environments. As the result, the SMA mode cost more messages than the RT mode which achieves the *Equal Rule* much better.

Figure 5.7 shows the result when the object is rarely searched. We can see, when the f_x is lower than 0.2, both the RT mode and the EMA mode incurs much higher message cost than the theoretical result. Because the index dissemination process is executed after each search, the RT mode can not achieve the *Stream Method* well when the search frequency is low. The simulation result also implies that the *Stream Method* is much efficient than the *Burst Method*.

The above results show the trade-off between adaptability and stability of the three index dissemination schemes. The RT mode and the EMA mode have good adaptability while the SMA mode has good stability. However, in many P2P file sharing systems, the popularity of objects follow long-tail distributions that most of the objects are rarely searched. Therefore, the SMA mode seems to be more suitable for those systems.

At last, Figure 5.8 shows that the protocol can also adapt to the change of $m(t)$. We can find that there is no obvious difference among the three index dissemination schemes' performance. Although it seems that the protocol requires more time to converge, the long converge time does not imply the protocol has bad adaptability in this case. That is because the system itself takes time to stable as shown in Figure 5.9. Similar results can be seen by adopting dynamic lifetime

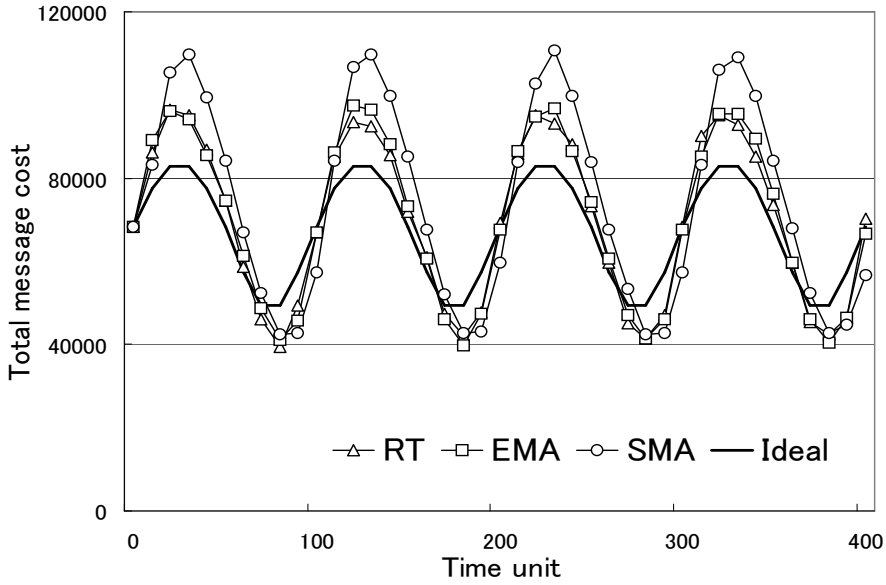


Figure 5.5: The total message cost (the sum of 1000 executions), $f_x(t) = 2 + \sin(2\pi t/100)$.

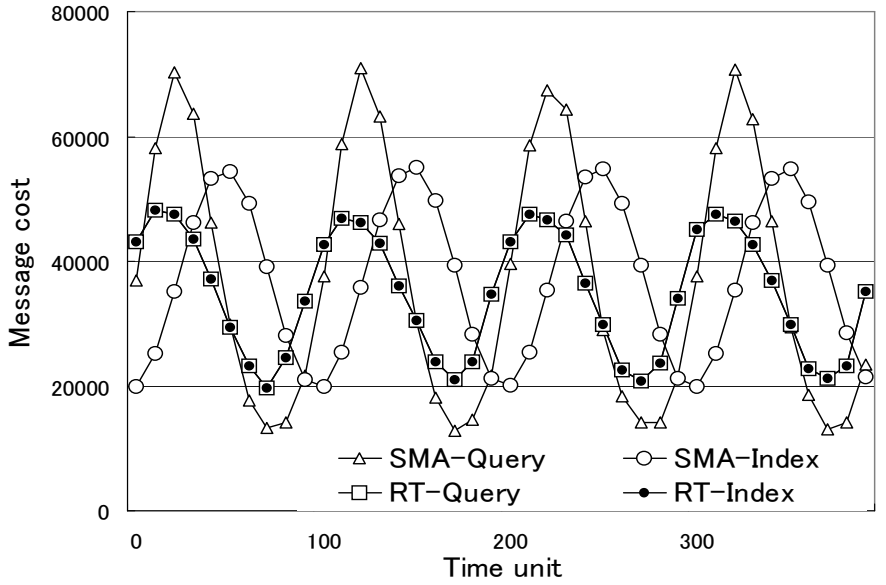


Figure 5.6: The message cost of index and query dissemination in RT and SMA mode respectively (the sum of 1000 executions); $f_x(t) = 2 + \sin(2\pi t/100)$.

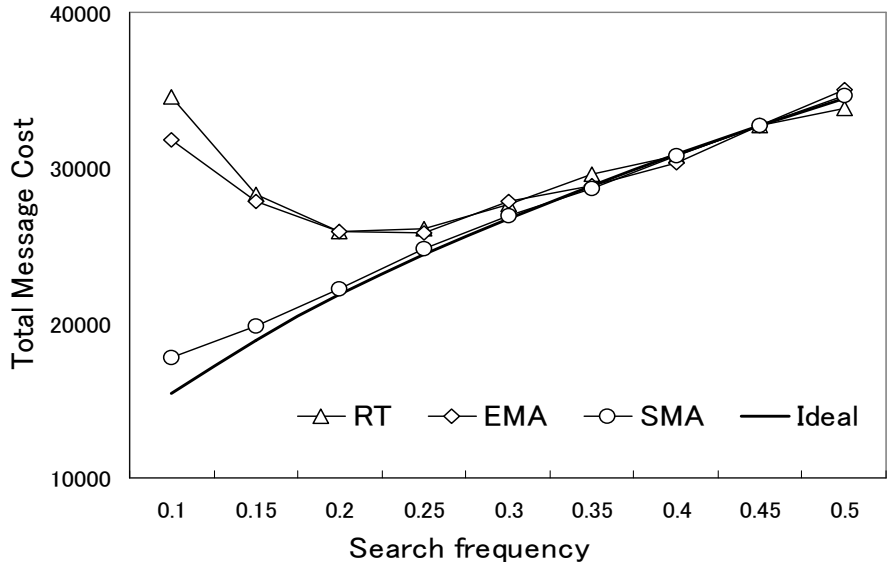


Figure 5.7: The total message cost (the sum of 1000 executions), $0.1 \leq f_x \leq 0.5$.

distributions of peers.

5.4.2 Feasibility

Up to now, we discuss the problem under the assumption of the ideal random sampling service. However, to implement the ideal random sampling is very costly in distributed systems e.g., each peer may have to know the whole peer set. Therefore, it is necessary to evaluate the protocol with non-ideal but cheap implementations. In this subsection, we show the performance of our protocol in realistic environments with feasible implementations of the random sampling service. We also compare the performance of our protocol with the protocols which adopt a fixed number of indices for all objects to show the advantage of the popularity-based index dissemination.

We adopt random walk to disseminate messages. A message i.e., an index or a query, is carried by a random walker such that the peers on the trace of the random walker receive the message. Three kinds of overlay networks are adopted in the simulation. All of those networks are 30-out-regular directed networks. The first type is the out-regular random network which is still an ideal model but is much easier to approach than the ideal random sampling [20]. In each time unit, the network is re-built in order to delete bad links pointing to the peers which have left the system. The second one is the PWDN with the uniform weigh setting which can is constructed by the protocols we proposed in Chapter 3 and 4. The third one is the random growing network [28]. The network converges to a power-law network in which the in-degree

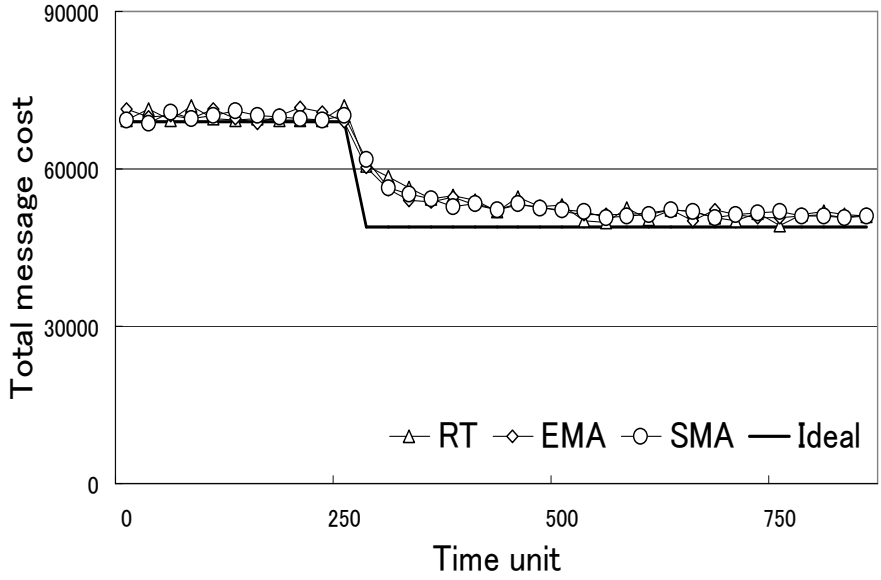


Figure 5.8: The total message cost (the sum of 1000 executions), $f_x = 2$; $m(t) = 400$ for $t \in [0, 250)$, $m(t) = 200$ for $t > 250$.

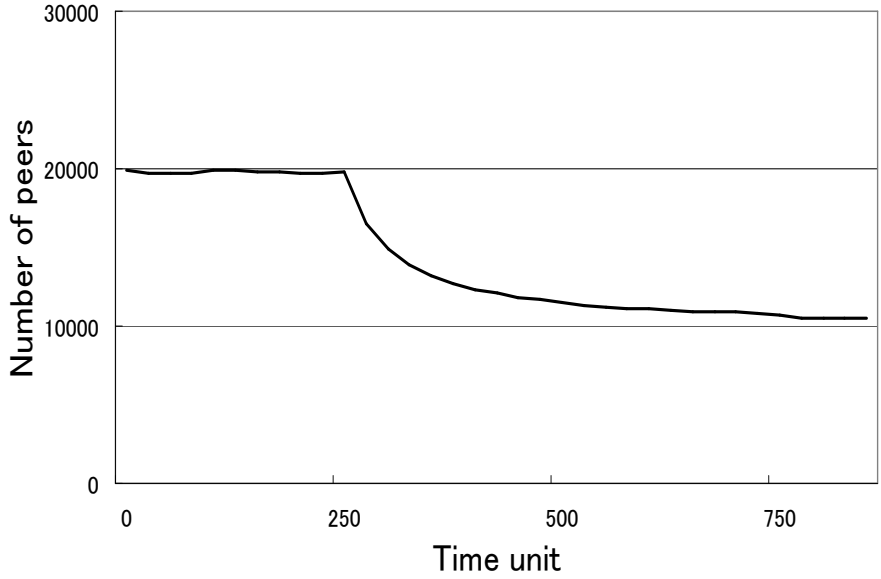
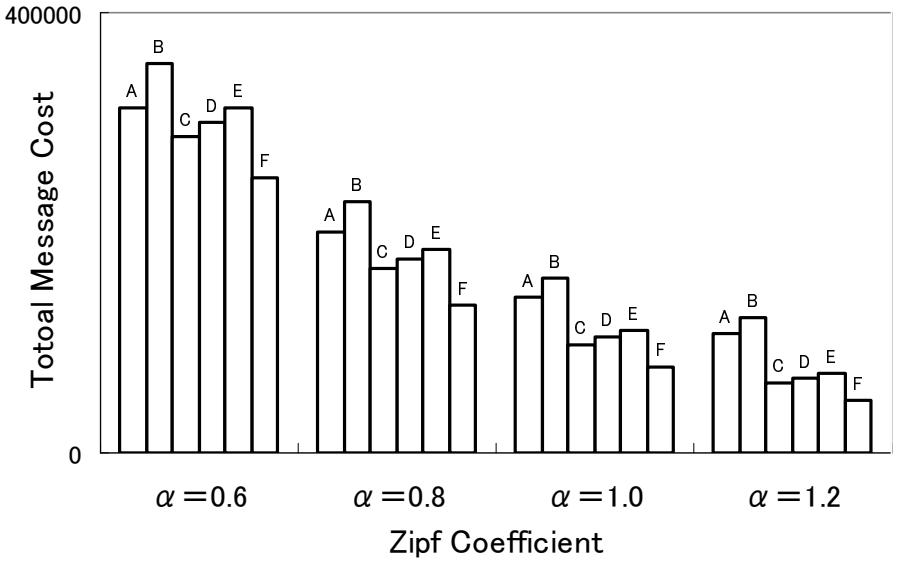


Figure 5.9: The number of peers. $m(t) = 400$ for $t \in [0, 250)$, $m(t) = 200$ for $t > 250$.



A: Fixed index number, theoretical minimum.

B: Fixed index number, PWDN (uniform weight setting).

C: Adaptive index dissemination, theoretical minimum.

D: Adaptive index dissemination, out-regular uniform-random network.

E: Adaptive index dissemination, PWDN (uniform weight setting).

F: Adaptive index dissemination, random-growing network.

Figure 5.10: Simulation results in implementations of the random sampling service.

distribution of peers is biased.

The simulation parameters are almost the same as which used in Section 5.4.1. The only difference is that we use 1000 objects and each of them has a different search frequency. Each object x has a unique popularity rank, denoted by $r_x, 1 \leq r_x \leq 1000$. The search frequency f_x of object x is given by $f_x = 1000/r_x^{\alpha}, 0.6 \leq \alpha \leq 1.2$. That implies the objects' popularities follow the Zipf-distribution where α is the Zipf coefficient. The Zipf distribution and the scope of the Zipf coefficient are proved to be consistent to the objects' popularity distributions in P2P file sharing systems [46].

The simulation results are shown in Figure 5.10. The three index dissemination schemes have almost the same result in this simulation because the objects' popularities are fixed and the most unpopular object is searched at least $1000/1000^{1.2} > 0.25$ times each time unit. The item A is the theoretical minimum message cost while each object has the same number of indices. Item B is the minimum cost of adopting fixed index number and random walk on the PWDN. Item C is the theoretical minimum message cost of our protocol which adopts adaptive index number.

The items D, E, F are the results of our protocol in the random network, the PWDN and the random-growing network respectively.

Comparing A with C (or B with D, E), we can see that the popularity-based index dissemination can effectively decrease the system message cost, especially when the popularity distribution are highly skewed (the cases $\alpha = 1.0$ and $\alpha = 1.2$). Comparing C with items D, E, it can be found that the message cost of adopting non-ideal random sampling implementations are about 10% higher than that of the ideal system model. That is because in those networks a random walker may visit a same peer more often. From the results we can see that the performance of our protocol is nearly optimal even adopting non-ideal sampling service so that it is considered the protocol is practicable in real system environments

The item F is an exceptional case that is under a non-uniform sampling model i.e., non-uniform workload allocation. Clearly, the results of F are much lower than those of C, D, E in all settings of α. The result is consistent with the arguments in Chapter 4. By comparing F in different settings of α, the popularity-based index disseminating is also applicable under non-uniform workload allocation models.

5.5 Supplemental Remarks

5.5.1 Message size

We assumed that both the dissemination of an index and a query message equally cost one message. This assumption can be easily removed. Letting I_x and P_x be the communication cost of disseminating a index and a query message respectively, we obtain

$$M_x(t) = \frac{Q_x}{T_I} \cdot I_x + \frac{f_x n T_I}{Q_x \cdot \sum_{\tau=1}^{T_I} (1 - d(\tau))} \cdot P_x.$$

Thus, the optimal index number is

$$\hat{q_x} = \sqrt{\frac{f_x n P_x}{\sum_{\tau=1}^{T_I} (1 - d(\tau)) I_x}},$$

and in the case the search cost is

$$s_x = \sqrt{\frac{f_x n I_x}{\sum_{\tau=1}^{T_I} (1 - d(\tau)) P_x}}.$$

Therefore, the Equal Rule still holds because the communication cost of an object is minimized when $q_x I_x = s_x P_x$, where $q_x I_x$ and $s_x P_x$ are the object's index maintenance cost and search cost respectively.

5.5.2 Queries for inexistent objects

When a peer search for some inexistent objects, the search process can not terminate because it continues searching until the object is found. To prevent the infinite search, a upper bound of search size, denoted by H, should be set. However, the bounded search size yields another problem that some rarely searched objects are difficult to find because they have almost no indices. To increase the success rate of searching for those objects, it is effective to let each peer disseminate a predefined minimum number of indices, denoted by L, in each time unit. By Lemma 5.1.1 and Equality 5.4, any object can be find with a minimum success rate ρ given by

$$\rho \geq 1 - e^{-HL\sum_{\tau=1}^{T_I}(1-d(\tau))/n}.$$

If L is small enough, the system message cost is still approximately the minimum because the additional index maintenance cost for disseminating those predefined indices is very small.

5.6 Related works

5.6.1 Quorum-based Search

The quorum-based search protocol formulated the index-dissemination-based search approach [13]. Under the random sampling model, the work presents a quantitative analysis of the hit rate with given number of indices and search size. However, the search protocol is not optimized. Based on the same search principle, we optimize the index dissemination scheme and minimize the system total communication cost. Our work can be regarded as a completed version of the quorum-based search.

5.6.2 Square-Root Replication

The Square-Root Replication(SRR) is a optimized storage assignment principle for replica-dissemination-based search protocols [2][3]. With a similar purpose, the SRR adopts popularity-based replication to minimize system search cost. Different from our approach in which the size of an index is so small that can be ignored, the total number of replicas can be disseminated is limited by the system storage capacity i.e., the sum of all peers' storage capacities. The SRR shows that the search cost for all objects can be minimized when the number of each object's replicas is proportional to the square root of the object's popularity. However, if the system storage capacity is small, it is still difficult to find objects because each object can have only a small amount of replicas. On the other hand, since the SRR does not consider the cost for

disseminating those replications, the communication cost may be huge in systems which have large storages. Modern P2P systems often adopt index-dissemination search protocols and adopt the replication for improve the performance of contents download.

5.7 Concluding remarks

In this chapter, we investigated the index-dissemination-based search approaches under a general churn model that peers' lifetime distribution can be arbitrary. The objective is to minimize the total communication cost of the system which includes both the index maintenance cost and search cost. The theoretical contribution consists of a tight lower bound of the total system communication cost and two principles for optimal index dissemination under the uniform-random sampling assumption. The first principle is the *Stream Method* that shows the best index dissemination method is to incrementally disseminate the same number of indices at each time unit. The method stabilizes the number of available indices in the system and minimizes the expected search cost against the loss of indices when peers leave the system. The second principle is the *Equal Rule* that shows the optimal balance point of the trade-off between the search cost and index maintenance cost is to assign the same communication on the query and index dissemination.

According to the two principles, we proposed a distributed search protocol to achieve the optimal index dissemination adapting to the system environment. A remarkable advantage of the protocol is that it yields almost no additional communication cost to achieve the self-adaptive feature. The effectiveness of the proposed protocol is proved by simulation in both dynamic and realistic system environments.

Chapter 6

Conclusion

6.1 Summary of Contributions

This dissertation presented an integrated solution for the resource search problem in unstructured P2P systems which adopt flooding or random walk as their routing methods. As shown in Figure 6.1, the contributions of this dissertation are related to three functional components i.e., the overlay construction, the workload allocation and the search problems, in the architecture of unstructured P2P systems respectively. A brief summary of them is as follows.

- **Overlay Construction:**
  In Chapter 3, we defined the *Probabilistic Weighted d-out regular Directed Network (PWDN)*. The PWDN's construction protocol is a simple but powerful middleware for realizing degree-controllable overlay networks. In such networks, each peer's out-degree is fixed but the in-degree is proportional to its weight. The weight is the only interface parameter for upper-layer applications to control the network topology. We proposed distributed protocols to realized the PWDN. By simulation, the networks generated by proposed protocols are proved to have similar aspects as the PWDN and have some other good properties such as good connectivity and compact structures which are suitable for P2P system environments.

- **Workload Allocation:**
  In Chapter 4, we showed that the search performance of unstructured P2P systems can be improved by concentrating both objects' indices and search queries, and thus the search workload, to a part of peers. We categorized traditional workload allocation strategies to four types: Uniform (UN), Capacity-Proportional (CP), Fixed-Layered (FL) and Adaptive-Layered (AL). Then we proposed a new strategy called Adapt-Layered & Capacity-Proportional

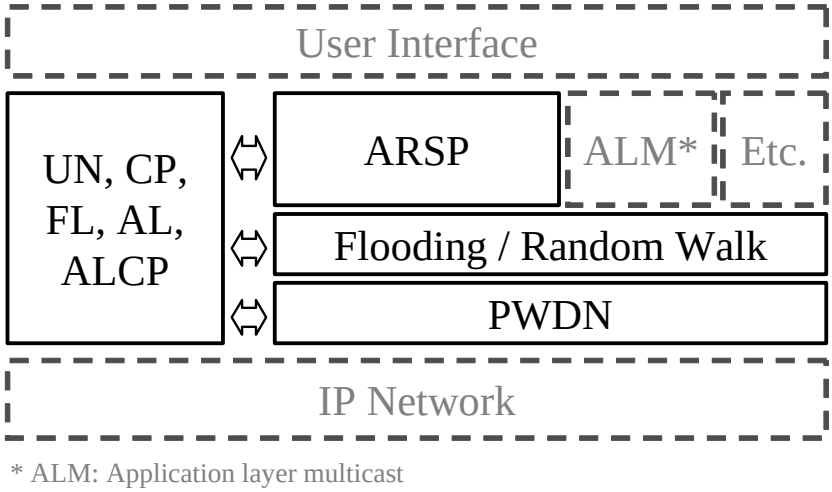


Figure 6.1: The proposed architecture of unstructured P2P systems.

(ALCP) which takes the advantages of both AL and CP. The ALCP fully utilizes the available capacity of the most powerful peers to maximize the search performance. A distributed framework is presented to realize the ALCP and other traditional workload allocation strategies. The lower layer of the framework is the PWDN which adjusts the workload (i.e., the number of disseminated indices and search queries) allocated to each peer proportionally to its weight. The upper layer includes five workload management protocols which decide peers' weights according to the five workload allocation strategies respectively. The simulation results show that the ALCP achieves significantly higher search performance than traditional approaches.

- **Search:**
  In Chapter 5, we proposed an adaptive search protocol called ARSP based on index dissemination. The work optimizes the index dissemination scheme of each object according to its popularity. We presented theoretical proofs to show the protocol can minimize total communication cost, which including both the index dissemination cost and search cost, related to each peer under the random sampling access model and a general Churn model in which peers' lifetime distribution can be arbitrary. The random sampling service can be achieved by adopting flooding or random walk upon the PWDN with the uniform weight setting i.e., the uniform workload allocation. The simulation results show that the message cost of ARSP is close to the theoretical minimum under the uniform workload allocation and it can also cost much less messages under non-uniform workload allocations than traditional approaches.

Our solution optimizes the probabilistic search approaches of unstructured P2P systems based on the statistical biases in peers' capacities and resources' popularities. The effectiveness of the solution is theoretically guaranteed and proved by simulation. It also has excellent feasibility for heterogeneous P2P system environments and can be applied in most of the P2P applications including P2P file sharing systems, P2P voice applications, grid computing etc. We hope this work can progress the formulation of the design of unstructured P2P systems.

6.2 Future Directions

In this work we tried to improve the search performance of unstructured P2P systems by optimizing their search approaches based on the biases in peers' capacities and objects' popularities. As we mentioned in Chapter 4, the search performance can also be improved by optimizing the system according to some other heterogeneous features in P2P system environments such as the distances between peers in the physical network and the interests of peers. However, the design purpose of the capacity-based workload allocation is not consistent with them e.g., a peer may not be able to find powerful peers, or peers having similar interests, physically close to it. That implies one must find a compromise solution when combines those optimizing ideas. It is hard to say how much improvement can be achieved by the combination of them in reality and the achievement seems to depend on application types. Nevertheless, the search approaches which combine those optimizing ideas are very worthy of study for better understanding and utilizing of the unstructured P2P systems.

The ARSP is an optimized search protocol under the uniform random sampling access model which applies the uniform workload allocation. Without any doubt that the ARSP can be applied under the ALCP and other workload allocation strategies we discussed in Chapter 4. However we cannot guarantee that the ARSP is optimized under those non-uniform workload allocation strategies because the performance is affected by the lifetime of each peer. When a peer leaves the system, the indices stored in it disappear at the same time. Therefore, a peer which is likely to stay in the system for a relatively long time e.g., longer than the maximum lifetime of each index, is suitable to take on more responsibility in search. However, peers' lifetimes can not be known in advance and the relationship between their capacities and lifetimes is still unclear. In some cases, it is efficient to select a peer which has stayed in the system for a long time to be an SP because peers' lifetimes usually follow long-tail distributions [30][42][43]. This mechanism can be easily achieved by adding those conditions into the Promotion conditions in our workload management protocols but sometimes we prefer to employ some short-life but powerful SPs in order to improve search performance even temporarily. The optimization of the

workload allocation under Churn is an interesting future problem.

Acknowledgements

I especially express gratitude to my supervisor Professor Toshimitsu Masuzawa for his guidance and encouragement. I also express gratitude to Professor Kenichi Hagihara, Professor Shinji Kusumoto and Associate Professor Hirotsugu Kakugawa of Graduate School of Information Science and Technology, Osaka University for their precious advice and valuable comments on this dissertation.

I appliceate Dr. Fukuhito Ooshita at Graduate School of Information Science and Technology, Osaka University, Dr. Tadashi Araragi at NTT Communication Science Laboratories, and Dr. Taisuke Izumi at Graduate School of Engineering, Nagoya Institute of Technology for their valuable comment on my work. I also also appreciate Dr. Yoshihiko Nakaminami at Panasonic Corporation, Mrs. Tomoko Arakawa, Ms. Megumi Maki, Ms. Fusami Nagae and students of Algorithm Engineering Laboratory, the Graduate School of Information Science and Technology, Osaka University for their kindness.

In particular, I appreciate NTT DOCOMO, INC. and Japan Student Service Organization (JASSO) for their economic support for my education in Japan.

Finally, I appreciate my parents XunE Wu, JiMin Xu and all of my families for their support and kindness in my study and life.

Bibliography

[1] Ralf Steinmetz and Klaus Wehrle. *Peer-to-Peer Systems and Applications.* Springer, 2005.

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

[3] Edith Cohen and Scott Shenker. Replication strategies in unstructured peer-to-peer networks. *SIGCOMM Comput. Commun. Rev.*, 32(4):177–190, 2002.

[4] Miguel Castro, Manuel Costa, and Antony Rowstron. Performance and dependability of structured peer-to-peer overlays. In *Proceedings of the International Conference on Dependable Systems and Networks (DSN'04)*, page 9, 2004.

[5] Sylvia Ratnasamy, Paul Francis, Mark Handley, M. Frans Karp, and Scott Schenker. A scalable content-addressable network. In *Proceedings of the international conference of the Special Interest Group on Data Communications (SIGCOMM'01)*, pages 161–172, 2001.

[6] Ion Stoica, Robert Morris, and David Karger. Chord: A scalable peer-to-peer lookup service for internet application. In *Proceedings of the 2001 international conference of the Special Interest Group on Data Communications (SIGCOMM'01)*, pages 149–160, 2001.

[7] Gnutella website: `http://gnutella.wego.com`.

[8] KaZaA website: `http://www.kazaa.com`.

[9] Winny info.: `http://en.wikipedia.org/wiki/Winny`.

[10] Saikat Guha. An experimental study of the skype peer-to-peer voip system. In *Proceedings of the 25th Conference on Computer Communications (IEEE INFOCOM'06)*, 2006.

[11] Skype website: `http://www.skype.com/intl/en/`.

[12] SETI@home website: `http://setiathome.berkeley.edu/`.

[13] K. Miura, T. Tagawa, and H. Kakugawa. A quorum-based protocol for searching objects in peer-to-peer networks. *IEEE Transactions on Parallel and Distributed Systems*, 17:25–37, 2006.

[14] M. Repeanu. Peer-to-peer architecture case study: Gnutella network. In *Proceedings of the 1st International Conference on Peer-to-Peer Computing (P2P'01)*, page 99, 2001.

[15] J. Liang, R. Kumar, and K. Ross. Understanding KaZaA, 2004.

[16] Yu Wu, Taisuke Izumi, Fukuhito Ooshita, Hirotsugu Kakugawa, and Toshimitsu Masuzawa. An adaptive randomized search protocol in peer-to-peer systems. In *Proceedings of the 2007 ACM symposium on Applied computing (SAC '07)*, pages 533–537. ACM Press, 2007.

[17] The Gnutella protocol specification v0.4.

[18] `http://www.sun.com/software/jxta/`.

[19] BÉLA BOLLOBÁS. *Random Graphs*. CAMBERIDGE, 2001.

[20] Márk Jelasity, Rachid Guerraoui, Anne-Marie Kermarrec, and Maarten van Steen. The peer sampling service: experimental evaluation of unstructured gossip-based implementations. In *Proceedings of the ACM/IFIP/USENIX 5th International Middleware Conference (Middleware'04)*, 2004.

[21] Daniel Stutzbach, Reza Rejaie, and Subhabrata Sen. Characterizing unstructured overlay topologies in modern P2P file-sharing systems. In *Proceedings of the 2005 Internet Measurement Conference (IMC'05)*, 2005.

[22] Shanyu Zhao, Daniel Stutzbach, and Reza Rejaie. Characterizing files in the modern gnutella network: a measurement study. In *Multimedia Computing and Networking 2006 (MMCN'06)*, volume 6071, NO.1, page 60710M. SPIE press, 2006.

[23] V. Vishnumurthy and P. Francis. On heterogeneous overlay construction and random node selection in unstructured P2P networks. In *Proceedings of the 25th Conference on Computer Communications (IEEE INFOCOM'06)*, 2006.

[24] KW. Kwong and HK. Tsang. Building heterogeneous peer-to-peer networks: protocol and analysis. *IEEE/ACM Transactions on Networking*, 2008.

[25] Yatin Chawathe, Sylvia Ratnasamy, Lee Breslau, Nick Lanham, and Scott Shenker. Making gnutella-like P2P systems scalable. In *Proceedings of the 2003 international conference of*

the Special Interest Group on Data Communications (SIGCOMM'03), pages 407–418. ACM Press, 2003.

[26] Stefan Saroiu, P. Krishna Gummadi, and Steven D. Gribble. Measurement study of peer-to-peer file sharing systems. *Proceedings of the 2002 international Multimedia Computing and Networking conference (MMCN'02)*, 4673:156–170, 2001.

[27] Peter Mahlmann and Christian Schindelhauer. Peer-to-peer networks based on random transformations of connected regular undirected graphs. In *Proceedings of the 17th Annual ACM Symposium on Parallelism in Algorithms and Architectures (SPAA'05)*, pages 155–164, 2005.

[28] Duncan S. Callaway, John E. Hopcroft, Jon M. Kleinberg, M. E. J. Newman, and Steven H. Strogatz. Are randomly grown graphs really random? *Phys. Rev. E*, 64(4):041902, 2001.

[29] Winny initiating website: `http://winny.cool.ne.jp/index.html`.

[30] Li Xiao, Zhenyun Zhuang, and Yunhao Liu. Dynamic layer management in superpeer architectures. *IEEE Transactions on Parallel and Distributed Systems*, 16(11):1078–1091, 2005.

[31] T. I. Fenner and A. M. Frieze. On the connectivity of random m-orientable graphs and digraphs. *AMS subject classification (1980) 05 C 40 - 60 C 05*, 1981.

[32] Ming Zhong and Kai Shen. Popularity-biased random walks for peer-to-peer search under the square-root principle. In *Proceedings of the 5th International Workshop on Peer-to-Peer Systems (IPTPS'06)*, 2006.

[33] B. Cooper. Quickly routing searches without having to move content. In *Proceedings of the 4th International Workshop on Peer-to-Peer Systems (IPTPS'05)*, 2005.

[34] Napster website: `http://www.napster.com`.

[35] Saeyoung Han, Jaeeui Sohn, and Sungyong Park. An efficient search scheme using self-organizing hierarchical ring in unstructured peer-to-peer systems. In *Proceedings of On the Move to Meaningful Internet Systems 2006 (OTM 2006's Workshops)*. Springer Berlin / Heidelberg, 2006.

[36] Taisuke Izumi and Toshimitsu Masuzawa. An interest-based peer clustering algorithm using ant paradigm. In *Proceedings of the 2nd International Workshop on Biologically Inspired Approaches to Advanced Information (BioADIT'06)*, 2006.

[37] F. Le Fessant, AM. Kermarrec, and L. Massoulié. Clustering in peer-to-peer file sharing workloads. In *Proceedings of the 3th International Workshop on Peer-to-Peer Systems (IPTPS'04)*, 2004.

[38] Ming Zhong, Kai Shen, and Joel Seiferas. Non-uniform random membership management in peer-to-peer networks. In *Proceedings of the 24th Conference on Computer Communications (IEEE INFOCOM'05)*, 2005.

[39] Y. Liu, X. Liu, L. Xiao, LM. Ni, and X. Zhang. Location-aware topology matching in P2P systems. In *Proceedings of the 23th Conference on Computer Communications (IEEE INFOCOM'04)*, 2004.

[40] Vinay Aggarwal, Anja Feldmann, and Christian Scheideler. Can isps and P2P users cooperate for improved performance? *SIGCOMM Comput. Commun. Rev.*, 37(3):29–40, 2007.

[41] Bridge Q. Zhao, John C.S. Lui, and Dah-Ming Chiu. Mathematical modeling of incentive policies in P2P systems. In *Proceedings of the 2008 Workshop on the Economics of Networks, Systems and Computation(NetEcon '08)*, pages 97–102. ACM, 2008.

[42] Derek Leonard, Vivek Rai, and Dmitri Loguinov. On lifetime-based node failure and stochastic resilience of decentralized peer-to-peer networks. In *Proceedings of SIGMETRICS '05*, pages 26–37. ACM, 2005.

[43] Zhongmei Yao, Xiaoming Wang, D. Leonard, and D. Loguinov. On node isolation under churn in unstructured P2P networks with heavy-tailed lifetimes. In *Proceedings of the 26th Conference on Computer Communications (IEEE INFOCOM'07)*, pages 2126–2134, May 2007.

[44] Michael Mitzenmacher and Eli Upfal. *Probability and Computing*. CAMBERIDGE, 2005.

[45] Ismail Ari, Bo Hong, Ethan L. Miller, Scott A. Brandt, and Darrell D. E. Long. Managing flash crowds on the internet. In *Proceedings of the 11th International Workshop on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS'03)*, 2003.

[46] Kunwadee Sripanidkulchai. The popularity of gnutella queries and its implications on scalability. URL `http://www.cs.cmu.edu/~kunwadee/research/p2p/paper.html`, 2001.