中央研究院
資訊科學研究所
Institute of Information Science, Academia Sinica • Taipei, Taiwan, ROC

TR-IIS-08-012

A Scalable Peer-to-Peer Presence Directory

Chi-Jen Wu, Jan-Ming Ho and Ming-Syan Chen

November 25, 2008 || Technical Report No. TR-IIS-08-012

http://www.iis.sinica.edu.tw/page/library/LIB/TechReport/tr2008/tr08.html

A Scalable Peer-to-Peer Presence Directory

Chi-Jen Wu*†, Jan-Ming Ho† and Ming-Syan Chen*
*Department of Electrical Engineering, National Taiwan University, Taiwan
†Institute of Information Science, Academia Sinica, Taiwan
{cjwu,hoho}@iis.sinica.edu.tw, mschen@cc.ee.ntu.edu.tw

***Abstract*—Instant Messaging (IM) has emerged as a popular communication service over the Internet. One of the themes of IM systems is to provide a presence directory that carries information on user's presence or absence to his/her friends. In this paper, we present new presence directory architecture and give a comparison of existing presence directories. We first introduce the distributed buddy-list search problem. We then present P2Dir, a distributed peer-to-peer presence directory protocol to address this problem. For each newly arriving user, the protocol is used to search for network presence of his/her buddies and also to notify them on his/her presence. P2Dir organizes directory servers into a 2-hop P2P overlay for efficient buddy searching. Moreover, P2Dir leverages the breadth-first search algorithm and a one-hop caching strategy to achieve small constant search latency on average. We measure the performance of our P2Dir system, in terms of search cost and search satisfaction, where search cost is defined as total number of messages incurred among the directories upon the arrival of a user, and search satisfaction is defined as the time it takes to search for the newly arriving user's buddy list and to notify presence of the newly arriving user to his/her buddies. We evaluate the performance of our P2Dir system in terms of search cost and search satisfaction through simulations, and compare it with a mesh-based presence protocol. The results show that our P2Dir achieves performance gains in search cost without sacrificing search satisfaction.**

I. Introduction

Social network applications, such as group communication and Instant Messaging (IM), have emerged as an attractive service over the Internet. In a social network application such as IM systems, e.g., AOL Instant Messenger (AIM), Microsoft MSN and Yahoo! Messenger, every user may easily form a social network to real-time communicate with friends. For example, a user may log into a system, and then start sending and receiving instant messages to and from other users. Nowadays, there are more than several millions of IM users over the Internet [1]. Based on its growth momentum, it is expected that the number of IM users to drastically increase in the future.

Presence directory service is an essential component of an IM system. It maintains an up-to-date list of presence information of all the users. It also brokers user's presence or absence status to his/her friends whenever appropriate. The presence directory service should include binding of a user's name to his/her current network location, and retrieving and subscribing to changes in the presence information of other users. In most IM systems, each user has a contact list, typically called the buddy list that associates with whom a user wants to communicate. The status of a user is advertised automatically to each online user on his/her buddy list whenever he/she transits from one status to another. For example, when a user logs in an IM system, the presence directory should search and alert everyone in the buddy list of that user. In order to maximize the search speed and to minimize the notification time of a presence directory service, most IM systems use server cluster technology [2], which allows an IM system to scale to millions of users. To date, little has been documented on presence directories used by existing IM systems.

In this paper, we give a brief discussion on the problems of existing presence directories. We first present a simple mathematical model of the communication cost in terms of the number of messages of a distributed presence directory. Then we introduce the buddy-list search problem in a distributed presence directory. The *buddy-list search problem* refers to the scalability problem that a presence directory in general may be deluged with torrential searching messages. Further details will be given in the Section III.

We then present the design of P2Dir, a distributed peer-to-peer (P2P) presence directory that can be used as a building block of Internet IM systems. The intent of designing P2Dir is to grip millions of users over thousands of directory servers distributed across the wide Internet. The importance of our method is that each directory server does not need to maintain global information such as the set of all users, and therefore, our protocol is adaptive for large scale IM systems. P2Dir organizes the DS nodes into a 2-hop P2P overlay for efficient buddy searching. Moreover, P2Dir leverages this 2-hop overlay and the breadth-first search algorithm to achieve small constant search latency in average, and resorts to an active caching strategy to dramatically reduce the number of messages generated by each search for a list of buddies. Through simulation, we evaluate the performance of our P2Dir system in terms of the number of messages and search satisfaction including search response time and search notification time. We also compared P2Dir with a mesh-based algorithm similar to Skype's presence protocol. The results show that our P2Dir achieves major performance gains in terms of the number of messages without sacrificing search satisfaction.

The rest of this paper is organized as follows. In the next section, we describe related works. In Section III, we present the analytic model to compute the number of messages generated in a buddy-search query on a distributed presence directory and briefly discuss the buddy-list search problem. In Section IV, we present the detailed design of the proposed P2Dir protocol. Complexity analysis of P2Dir and the mesh-based scheme are presented in Section V. In Section VI, we

start with introducing our performance evaluation methodology and present performance results on P2Dir system and the mesh-based scheme. In Sections VII and VIII, we discuss the further consideration and conclude this paper with a summery of the main research results from this study.

II. Related Work

In this section, we describe previous research on IM systems, and survey the directory services of existing systems. The concept of an IM system developed from the Internet Relay Chat Protocol (IRC) [3], and it is now a widely used in applications in like ICQ, AIM, Microsoft MSN, Yahoo! Messenger, and Skype. Because of their enormous popularity and user bases, most studies [1], [4], [5] of IM systems have focused on understanding the network traffic generated by IM applications. For example, in [5], which was an early study of IM systems, the authors developed a methodology for separating IM traffic from other Internet traffic. The analyses in [1] represent a comprehensive study of interactive traffic in the Microsoft MSN network. Similarly, the authors of [4] analyzed the traffic of two popular IM systems, AIM and Microsoft MSN. They found that most instant messaging traffics are due to presence/keep alive activities, hints or other extraneous traffic, not chat messages produced by users.

Well known commercial IM systems leverage some form of centralized directory to provide a presence service. However, little is known about the technical aspects of the directory services used such systems [2], [6]. Jennings III *et al*. [2] presented a taxonomy of different features and functions supported by the three most popular IM systems, AIM, Microsoft MSN and Yahoo! Messenger. The authors also provided an overview of the system architectures and observed that the systems use client-server-based protocols. Baset and Schulzrinne [6] studied Skype, another popular application that was launched in 2003 to support both instant messaging and voice conferencing. Skype utilizes Global Index (GI) technology [7] to provide a directory service for users. The GI technology is an overlay network in which every node has full presence knowledge about all available users. Skype claims that GI technology is guaranteed to locate a user if he/she has used the network in the previous 72 hours. However, since Skype is not an open protocol, it is difficult to determine how GI technology is used.

Recently, there is an increase amount of interest in how to design a decentralized peer-to-peer SIP [8]. For example, the P2P-SIP [9]–[12] has been proposed to remove the centralized server, reduce maintenance costs, and prevent failures in server-based SIP deployment. Like the IM system, the most important feature of a SIP system is the registrar directory. To maintain presence information, P2PSIP users/clients are organized in a DHT system, rather than in a centralized server. For small-company applications, the self-organizing aspects of P2P make SIP systems easier to configure and manage. P2PSIP is also being considered to support ad-hoc communication environments or emergency responder networks.

Several IETF charters [13]–[15] have addressed closely related topics, and many RFC documents on instant messaging and presence services have been published, e.g., [16]–[18]. Jabber [19] is a well-known deployment of instant messaging technologies based on related RFC documents. It captures the distributed architecture of SMTP protocols so that any Jabber server can communicate with any other Jabber server that is accessible via the Internet. Since Jabber's architecture is distributed, the result is a flexible network of servers that can be scaled much higher than the monolithic, centralized services. However, the *buddy-list search problem* we defined earlier can also affect such systems. Two articles [20], [21] discuss related issues of the eXtensible Messaging and Presence Protocol (XMPP) [16] and the Session Initiation Protocol for Instant Messaging and Presence Leveraging Extensions (SIMPLE) [15], [17] protocols. Saint Andre [20] analyzes the traffic generated as a result of presence information between users of inter-domains that support the XMPP. Houri *et al*. [21] show that the amount of presence traffic in SIMPLE can be extremely heavy, and they analyze the effect of a large presence system on the memory and CPU loading. Currently, Professor Schulzrinne and his group [22] are studying related problems and developing an initial set of guidelines for optimizing inter-domain presence traffic.

Compared with the above efforts, this work makes the following contributions. First, we analysis scalability problems of the distributed directory protocols, and introduce a new problem called *buddy-list search problem*. Although, our mathematical model is simple, it is not hard to comprehend a scalability problem in here. Second, we design a P2Dir system that is the scalable protocol designing for the presence service of IM systems. Our P2Dir can be easily integrated with any open-source IM system, such as Jabber.

III. The Model and Problem Statement

In this section, we describe the buddy-list search problem, and define the system model. Distinct from traditional network services, an IM system usually notifies its users on the online status of one's buddies as the basis of a real-time chatting environment. Moreover, the IM service is characterized by the frequent login/logoff behavior of its users. Thus, we may expect to observe a large amount of messages generated to search for buddies in an IM system. The presence directory designed to deal search for buddies must be able to handle these search messages. We refer to this problem as the buddy-list search problem. A brief analysis on a primitive presence directory architecture is presented below to illustrate the amount of messages in such a system.

A presence directory may be regarded as an overlay network which is defined as a directed graph $G = (V, E)$, where V is the set of n nodes each representing a Directory Server (DS), and E is a collection of ordered pairs in V. An edge $(v_1, v_2) \in E$, where $v_1, v_2 \in V$, is called an outgoing edge of v_1 and an incoming edge of v_2. The overlay network enables nodes to communicate with one another by forwarding messages to and through other nodes in the overlay. In

addition, the users of an IM system comprise a set of processes $U = p_1, p_2, \ldots, p_m$. Hereafter, the terms *process*, user and IM client are used interchangeably. Then, we define the *buddy list* as follows.

Definition 1. buddy list, a buddy list g_i of process $p_i \in U$ is a subset of U, i.e., $gi \subseteq U$ where $1 \leq i \leq m$. We also define the buddy relation as a symmetric binary relation. That is, if $p_i \in g_j$ then $p_j \in g_i$.

For example, if a user, A is in the buddy list of a user, B, then the user, B will be also in A's buddy list.

A new process p_i randomly connects with a DS node and search for locations of other existing processes in its buddy list g_i, and to request for notification of locations of other processes in g_i on their arrival. Note that we refer to binding of user id and IP address of a process p_i as p_i's location. A process p_i can then communicate with a process p_j via p_j's IP address. Assume that each process and each DS node can join or leave the network arbitrarily at any time, and a DS node knows only those processes directly attached to it. We refer to this architecture as the basic model. Note that for a DS node to search for a process in the basic model, it has to send a query to every DS node. Thus, the underlying IM system will have to handle a large amount of messages in searching for buddies of new processes.

In the following, we will give an analysis of the expected rate of messages generated to search for buddies of newly arriving processes in a IM system. A newly arriving process p_i of the IM system sends a message containing its buddy-list g_i to the DS node which it directly attaches to. Let's denote μ as the average rate of processes arriving at the IM system. We assume the probability for a process to attach to a DS node to be uniform. In other words, $u = \frac{\mu}{n}$ is the average rate of new processes attached to a DS node. The probability for each process $p_j \in g_i$ to attach to the same DS node is denoted as h. And the probability for each p_j to attach to a specific DS node is $\frac{1}{n}$, thus, h equals to $n^{-|g_i|}$. The expected number of search messages generated by this DS node per unit time is then

$$(n-1) \times (1-h) \times u.$$

Considering the expected number M of messages generated by the n DS nodes per unit time, then we have

$$\begin{aligned} M &= n \times (n-1) \times (1-h) \times u \\ &\geq \frac{n \times (n-1) \times u}{2} \\ &\geq \frac{n^2 \times u}{4} = \frac{n \times \mu}{4}. \end{aligned}$$

Thus, the total communication cost and the total CPU processing overhead of the system increase linearly as the number of DS nodes increases. The above analysis shows that supporting newly arriving users to search for buddies in a distributed directory service system is rather expensive. In this paper, we are going to present a new distributed directory architecture which scales better than the previous basic model, and compare its performance with the mesh-base architecture which is used by some popular IM services.

IV. Design of P2Dir

The performance of distributed presence directory systems are affected by the large amount of messages in searching for buddies of newly arriving users, which gets worse as the size of the directory network increases. Our motivation for designing P2Dir is twofold: 1) to develop a P2P-based distributed directory system that can provide a fault tolerant presence service for general IM systems; and 2) to reduce buddy list search latency, and achieve high scalability. In the following, we begin by explaining the rationale behind our system design. We then present a overview of the P2Dir protocol, including details of the 2-hop DS overlay construction, buddy list searching, and caching operations.

A. Design Rationale

Locating/searching for objects in distributed networks is not a new problem, especially in P2P networks. Two recently developed systems, Gnutella and Distributed Hash Tables (DHT) [23], are designed to improve Internet-scale object searching. In recent years, there has been a great deal of research activity in this field, and many protocols and algorithms have been proposed. Existing algorithms address different aspects of the object search problem in distributed systems. Compared to file-sharing, presence information is more mutable; however, the above systems do not consider the buddy-list search problem when designing protocols for directory services.

Gnutella searches P2P file-sharing systems to locate files that match all the keywords in a search query. In Gnutella, the number of search recalls is the most important criterion. It tries to find more desired files efficiently, rather than reduce the response time for users. For the *buddy-list search problem* we consider, Gnutella may take a long time to conduct searches. Moreover, Gnutella's search algorithm does not reach all nodes, so it can not guarantee returning the required buddy list. Although, several Gnutella-like protocols have been proposed to improve the original Gnutella's performance, they focus on the scaling problems and search recall issues. In summary, Gnutella is not suitable for designing presence directories.

DHT systems are another class of distributed networks designed to locate objects. Most DHT systems provide efficient lookup functions that operate in $O(logN)$ overlay hops by only maintaining $O(logN)$ routing table entries. Generally, DHTs are well-suited to large-scale distributed applications, but they are less adept at buddy-list search. When using DHT in directory systems, each peer is required to perform $O(logN)$ registering operations after login, and also conduct $O(logN)$ lookup operations for each buddy. Moreover, node failure can cause churn in DHT systems, and most systems need $O(logN)$ repair operations after each failure to preserve the correct and efficient lookup operations. Therefore, in DHT systems, replicating lost data and handling churn increases both the workload and the time complexity. Even though some

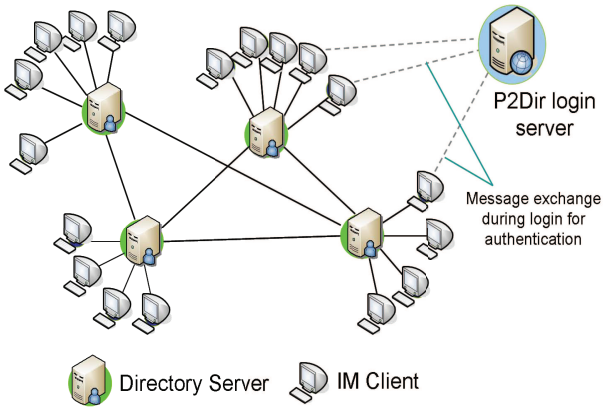


Fig. 1. An overview of a P2Dir system

DHT systems can address these problems, a search operation that must visit a logarithmic number of nodes to reach the buddy lists of users could be very slow. This is because each hop involves sending a message to a host that may be on the other side of the world, and some hosts may be heavily loaded, or have slow connections. Thus, for latency-sensitive applications, DHT systems may be unsuitable for presence directory design due to their high lookup costs [24].

The P2Dir protocol is used to construct and maintain a distributed directory and can be used to efficiently query the directory for buddy list searches. The protocol consists of three component protocols that are run on a set of directory servers. The design of P2Dir refines the concept of P2P systems to meet the particular needs of presence services. The three key components of our design are summarized below:

- A **2-hop DS overlay construction algorithm** that organizes Directory Servers in a fully distributed way, such that the resulting DS overlay network has a balanced load and a 2-hop diameter overlay with $O(\sqrt{n})$ node degree, where n is the number of nodes.
- A **one-hop caching algorithm** that is used to reduce the number of transmission messages and accelerate query speeds. All directory servers maintain caches of the buddy lists provided by their immediate neighbors.
- A **buddy searching protocol** that is based on the breadth-first search (BFS) algorithm. Since the 2-hop overlay ensures a low-TTL search, it achieves a small constant search latency on average.

B. P2Dir Overview

The P2Dir protocol is used to construct a distributed P2P-based directory for presence services, and to efficiently search desired buddy lists in the distributed directory. Figure 1 presents an overview of the P2Dir system. After a IM client logs in with an authentication server (the P2Dir login server in Figure 1), the client is randomly directed to one of the Directory Servers in the DS overlay. Alternatively, it can find the nearest DS node by using the sever selection technique [25]. The client opens a TCP connection to the DS node for control message transmission, particularly for the presence information. After establishing the control channel, the IM client sends a request for a buddy list search to the connected DS node. P2Dir then implements an efficient search operation and returns the desired buddy list to the IM client. During the search operation, the client's buddies will be notified about its presence. If the current DS node fails, the client can connect to another one. We assume that, in practice, the instant messages generated by users are transmitted by a direct TCP connection between IM clients. The P2Dir system deals primarily with the control and signal messages sent between DS nodes and IM clients. Next, we discuss the three protocol components in detail.

C. DS Overlay Construction

The DS overlay construction algorithm organizes the DS nodes into a 2-hop P2P overlay. P2Dir uses Kelips [26] to form a 2-hop DS overlay, the core component of the system, and leverages it to maintain a cooperative buddy cache efficiently. Kelips is designed for dynamic P2P networks in which nodes can join and leave at any time. It provides a good low-diameter overlay property. The low-diameter property ensures that a node only needs 2 hops to reach any other nodes. For more details about the join/leave properties of the Kelips system, readers may refer to [26]. Here, we only introduce the core design of the system. Kelips organizes nodes into n virtual affinity groups, numbered 0 to $(\sqrt{n}-1)$, as shown as Figure 2. In the Kelips system, each node maintains a list of peers of size $O(\sqrt{n})$, where n is the number of nodes in the DHT. When a Kelips node joins the system, it attaches to an affinity group determined by using a consistent hash function, such as SHA-1, to map the node's IP address to a integer interval between $[0, \sqrt{n}-1]$. Using the SHA-1 hash function [27] ensures that each affinity group will contain close to $\frac{n}{\sqrt{n}}$ nodes with high probability. The routing table of a node is comprised of two lists: an *Affinity Group View*, which is a list of other nodes in the same affinity group; and a *Contacts Group*, which is a list of the other affinity groups in the system, i.e., a set ($\sqrt{n}-1$ sized) of nodes lying in the foreign affinity groups. Figure 2 illustrates the Kelips system, which clearly has the 2-hop diameter property.

Consequently, P2Dir has the 2-hop diameter property based on the Kelips system, and DS nodes can join or leave P2Dir freely. However, a new DS node needs to establish connections with existing DS nodes when joining. When a DS node leaves, the remaining DS nodes must establish new connections. Thus, P2Dir contains a central element, called a *root server*, which maintains a cache of DS nodes at all times. The root server is reachable by all DS nodes at all times. When a new DS node joins, it first contacts the root server, which gives it k random nodes from the cache to connect to. The k value is determined by the root server. We assume that a DS node knows when any of its neighbors leaves the system. The root server is contacted whenever a DS node needs to reconnect to the network, and when a new DS node joins the network. The advantages of our algorithm are that it is simple to implement, it is naturally

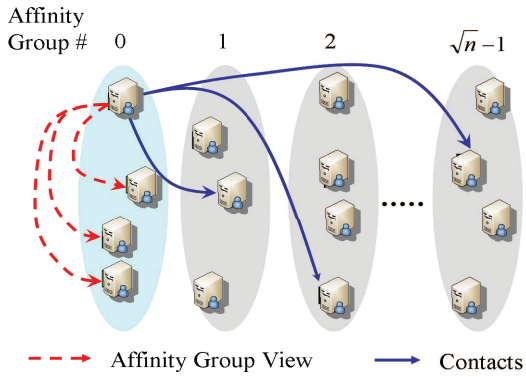


Fig. 2. A perspective of the Kelips system

robust to failures, and it has the 2-hop diameter property.

D. One-hop Caching

To improve the efficiency of the search operation, P2Dir requires that the caching strategy can replicate the presence information of users. To adapt to changes in the presence of users, the caching strategy should be asynchronous and not require expensive mechanisms for distribution. In P2Dir, each DS node maintains a user list of presence information of the current users, and it is responsible for caching the user list of each of its neighbors; in other words, a DS node only replicates the user list of nodes at most one hop away from itself. A DS node updates the cache when neighbors establish connections with it, and periodically updates its neighbors with the cache. Therefore, when a DS node receives a query, it can respond with matches from its own user list, and can also provide matches from its cache of user lists provided by all of its neighbors.

Our caching strategy does not incur a large overhead for the presence consistency among the DS nodes. When a user changes its presence information, either because it leaves the IM system or the IM application's failure, the responding DS node can disseminate its new presence to neighboring DS nodes, so that they can update the caches quickly. This one-hop caching strategy ensures that the user's presence information remains up-to-date and consistent throughout the session time of the users.

More specifically, each DS node creates roughly $2\sqrt{n} \times u$ replicas of buddy information, since each DS node replicates the *user lists* of nodes at most one hop away from itself. Recall that u denotes the average number of processes (IM clients) attached to one DS node. Based on this one-hop cache mechanism, a one-hop search operation can be conducted with very high probability. By maintaining $2\sqrt{n} \times u$ replicas of buddy information at each DS node and the simple 2-hop overlay design, P2Dir has sufficient redundancy to maintain an efficient buddy search service. Furthermore, the caching mechanism significantly reduces the communication costs of the searching. In the next section, we explain why the one-hop cache mechanism reduces the cost of buddy searches in P2Dir.

E. Buddy List Searching

Minimizing the search response time is important to the presence service of IM systems. Therefore, we combine P2Dir's buddy list search algorithm with the 2-hop DS overlay and one-hop caching strategy to ensure that P2Dir can provide swift responses for a large number of IM users. First, by organizing DS nodes into a 2-hop overlay network, we can use a smaller TTL value (i.e., 1) for queries and thereby reduce the network traffic, without having a significant impact on the search results. Second, by capitalizing on the one-hop caching mechanism, which maintains the user lists of its neighbors, we improve the response time by increasing the chances of finding buddies. As mentioned previously, P2Dir does not require a complex or specialized search algorithm. Instead, it adopts the TTL (Time-To-Live)-limited flooding technique used in Gnutella-like P2P file-sharing systems, and still improves the search efficiency.

Next, we describe the P2Dir Buddy List Search algorithm in detail. When a process (an IM client) logs into an IM system, P2Dir searches for the client's buddy list by performing a Buddy List Search operation. The search message contains all of the client's buddy information and a TTL field set to a constant value of 1. The DS nodes process the query by searching their local user lists and the cached buddies. If a DS node can respond to a buddy in the query, it returns the response to the buddy and removes the buddy from the query and decrements the TTL field by 1. If the resulting value is greater than zero, it forwards the message; otherwise, the message is not forwarded. Consequently, the buddy list search algorithm combined with the above two mechanisms can reduce the number of search messages sent by the flooding algorithm used in Gnutella-like P2P file-sharing systems.

Note that buddy searches can be performed in a locality aware manner. In the DS overlay construction, a joining DS node, d, requires a list of existing DS nodes in the P2Dir system. Nodes on the list are chosen randomly by the root server without considering their localities. However, a joining node can employ the well-known Proximity Neighbor Selection scheme in the P2P routing systems [28] to improve and maintain the network locality. The computation of a buddy search is performed locally because each search operation involves only the $\sqrt{n}-1$ closest DS nodes on the *Contacts list*; the DS nodes in the *Affinity Group View* are not involved with the network locality. This results in at most $2 \times \sqrt{n}$ messages in a buddy list search operation.

V. Cost Analysis

In this section, we provide a complexity analysis of the communication cost of P2Dir in terms of the number of messages required to retrieve the buddy information of **a user**. The *buddy-list searching problem* can be solved by a brute-force search algorithm, which simply searches all the DS nodes. In a mesh-based system, the algorithm replicates the all user information at each DS node; hence its search cost, denote by S^m_{cost}, is only one message. In other words, the system needs $n-1$ messages to replicate a user's presence

information to all DS nodes, where n is the number of DS nodes. The communication cost of retrieving buddies and replicating presence information can be formulated as $M_{cost} = S^m_{cost} + R^m_{cost}$, where R^m_{cost} is the cost of replicating presence information to all DS nodes. Accordingly, we have $M_{cost} = O(n)$.

In the analysis of our P2Dir system, we assume that the IM clients are distributed equally among all the DS nodes, which is the worst case for improving the performance of the P2Dir system. Here, the search cost of P2Dir is denoted by S^p_{cost}, which is only $2\times\sqrt{n}$ messages for searching buddy lists and replicating presence information. This is because we can combine the search message and replica message of presence information into one message. Moreover, each message may have a reply message for cache hitting, so we should double the cost of each DS node. It is straightforward to know that the communication cost of retrieving buddies and replicating presence information in a P2Dir system is $P_{cost} = 2 \times S^p_{cost}$. Thus, we have $P_{cost} = O(4\sqrt{n})$.

However, in a P2Dir system, a DS node not only searches a buddy list and replicates presence information, but also notifies users of the buddy list about the new presence event. Let b be the maximum number of buddies of an IM system user. Thus, the worst case is when none of the buddies are registered with the DS nodes reached by the search messages and each user on the buddy list is located on a different DS node. Since P2Dir must notify every user on the buddy list individually, it is clear that extra b messages must be transmitted in the worst case. When all users are distributed equally among the DS nodes, which is considered to be the worst case, the P_{cost} is $O(4\sqrt{n} + b)$. Consequently, we have the following lemma.

lemma 1: In a buddy searching operation of P2Dir system, the maximum communication cost of retrieving buddies and replicating presence information is $O(4\sqrt{n} + b)$.

Example. The following simple example illustrates the efficiency of the P2Dir system. Assume there are 1,000 DS nodes in the P2Dir system and the maximum number of buddies is 20. When a user joins, the expected value of the number of messages that a DS node sends is less than 148 (4×32+20). This means that our P2Dir system saves 85% (148/999) of the communication cost of the mesh-based approach.

Next, we discuss the search complexity of the DHT-based presence directory. We make the following assumptions to simplify analysis: 1) user presence information is only stored in one DS node (i.e. no replication); and 2) all users are uniformly distributed in all DS nodes. Note that some replica algorithms [29] have been proposed for DHT systems, but they increase the complexity of DHT. Although our analysis is based on the Chord [30] DHT, it can be extended to other DHTs.

Let n be the total number of nodes in a Chord network, in which a node can be either an IM Client or a DS node. Chord nodes are mapped on a denominational cyclic identifier space $[0, \ldots, 2^m]$, and a node with an identifier in the cycle of n nodes, maintains $logn$ neighbors, *i.e.* fingers, to provide a $O(logn)$ lookup operations. However, the lookup operation in DHT systems is based on exact-matching, so it has difficulty supporting complex queries like buddy list searches. Since the buddies, b, must be searched one by one, the total search complexity of DHT is equal to $D_{cost} = b \times \log n + 2b$. The $2b$ messages consist of the reply messages and the notification messages.

TABLE I
PRESENCE DIRECTORY COMPARISON

	Mesh	P2Dir	DHT-based
Search	$O(n)$	$O(4\sqrt{n}+b)$	$O(b \times \log n + 2b)$
Replicas	$O(\|U\|)$	$O(2\sqrt{n} \times u)$	$O(u)$
Latency	one hop	2 hops	$\log n$ hops

We summarize the comparison of different schemes in Table I. The columns show the different schemes, while the rows show different desired features. The "Search" label means the maximum number of messages sent by a DS node when a user joins (including search and cache); the "Replicas" Label means the maximum number of buddy replicas in a DS node; and the "Latency" label means the buddy search latency, we quantify this metric by the diameter of the overlay. This is reasonable because, in general, the search latency is dominated by the diameter of the overlay.

None of the schemes is a clear winner. The mesh-based system achieves good search latency at the expense of the other metrics. Our P2Dir approach yields a low communication cost in a medium-size presence directory system $(n < 10,000)$ and small search latency. Meanwhile, the DHT-based method provides good features for low communication cost and low replica load at the expense of increased search latency.

VI. PERFORMANCE EVALUATION

In contrast to studies that use high-level complexity analysis to compare different presence directories, we demonstrate the important properties of P2Dir through simulations. Our implementation of the network simulator with the Mesh-based scheme and P2Dir, is written in Java. The experiments were preformed on an Intel 2.8GHz Pentium PC with a 4G RAM. We describe our simulation setup in Section VI-A, and discuss the three important criteria used in the evaluation in Section VI-B. We conclude the section with a report on the performance results of the two protocols.

A. Simulation Setup

The simulator allows us to perform tests on up to 10,000 IM clients and 1,000 DS nodes, after which the simulation data no longer fits the RAM, so it is difficult to conduct the experiments. Therefore, we set the number of IM clients at 10,000, unless otherwise specified. The simulator first goes through a warm-up phase to reach the network size (both DS nodes and IM clients), and the simulator starts the 3-hour test

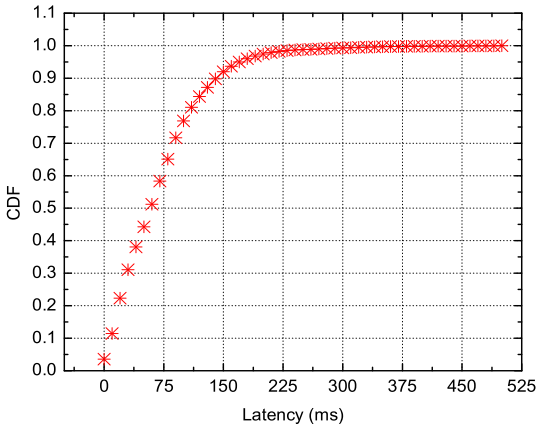


Fig. 3. Round-trip latency distribution of King data set.

after the measurement protocol has stabilized (the stabilized time is based on the network size). In each experiment, the mean session time of IM clients is 30 minutes, which means that a user stays in the system roughly 30 minutes. After a session, the user departs and waits approximately 30 minutes before rejoining the system. Note that the online sessions of IM users are important parts of user behavior in an IM system; however,, but we simplify this behavior in our experiments because the performance of the presence directory is not dominated by the online sessions of the IM user. The online sessions of MSN and AIM users fit the Weibull distribution approximately [4], so we will adapt our simulator for real IM systems in the future.

The simulated topology places every DS node in a position on the King data set [31]; the positions are chosen uniformly at random. The King data set delay matrix is derived from Internet measurements using techniques described by Gummadi *et al*. [32]. Note that since our simulations involve networks of less than 2,048 DS nodes, we use a pairwise latency matrix derived by measuring the inter-DS node latencies. In addition, since each IM client is uniformly attached to a random DS node, the propagation delay between the IM client and the DS node is randomly assigned in the range [1,20] (ms). In Figure 3, we show the CDF of the King data set's RTT. The average delay is 77.4 milliseconds. In addition, we assume that the DS nodes in the experiments do not fail. In this paper, we focus on the presence directory's performance metrics, which we discuss in the next. The failure of DS nodes will be addressed in a future work.

B. Performance Metrics

Within the context of the model, we measure the performance of the presence directory using the three metrics:

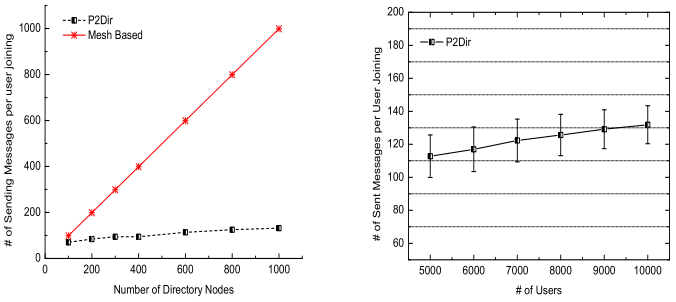
- ***1) Buddy Searching Messages:*** This metric represents the total number of messages transmitted between the query initiator and the other DS nodes. More specifically, a buddy search message includes the search/reply buddy list, the replicating user's presence information, and notification for buddies about the presence messages. This is a fundamental metric in our experiments, since it is widely regarded as critical in the presence directory system we discussed both in Section III and Section V. This metric is also a critical metric for measuring the scalability of a presence directory.
- ***2) Buddy Searching Latency***: This represents the maximum buddy search time of a joining user. We define the maximum buddy searching time as follows. The notation $t(p)$ indicates the searching time for a buddy, p.
  $\forall\ p \in g_i$ and p is online,
  Buddy Searching latency =
  $$\max\{t(p_1), t(p_2), \ldots, t(p_n)\},$$
  where $n \leq$ the maximum number of buddies and g_i is the buddy list of an enquirer user, q_i. Note that the status of p should be online. We ignore the offline searching time of p. This metric is a critical metric for measuring the search satisfaction of a presence directory.
- ***3) Buddy Notification Latency***: This represents that elapsed time for notifying the buddy. This metric, which is dominated by the diameter of the DS overlay, is also important for measuring the search satisfaction of a presence directory.

In our simulations, we compare the performance of P2Dir and a mesh-based presence directory in terms of buddy search messages, buddy search latency and buddy notification latency. For each simulation, we perform 20 tests.

C. Performance Results

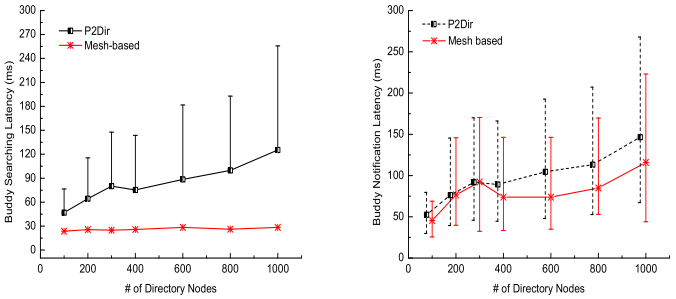
We first evaluated and compared the two protocols side by side by considering the buddy search messages metric. We instantiated a network of 10,000 users in our simulator, and ran a number of experiments to investigate the effect of the scalability of DS nodes on the involved search messages. More precisely, we varied the number of DS nodes from 100 to 1,000 to explore the relations between the number of DS nodes and the buddy search messages. In this test, the maximum number of buddies is set to 20. We list the experiment results in Figure 4.

Figure 4(a) depicts the average number of buddies searching messages per user joining. Figure 4(a) demonstrates two different schemes, P2Dir and mesh-based, respectively. For a given number of DS nodes, the average number of buddy search messages increases as the number of DS nodes grows, as shown in Figure 4(a). Moreover, for a given number of DS nodes in the P2Dir system, increasing the number of DS nodes moderately increases the average number of buddy search messages, suggesting a good scalability with the number of DS nodes in our P2Dir system. We also investigated how the average number of buddy search messages grows with the number of DS nodes in a mesh-based system. The search complexity of buddy search messages in mesh-based systems isO (n), which fits our analysis in Section V. The scalability problem of mesh-based systems may prevent a system scaling



(a) (b)

Fig. 4. Expected total transmissions during searching a buddy list



(a) (b)

Fig. 5. Expected searching latency during searching a buddy list

a network with thousands of DS nodes; hence, compared to P2Dir, a mesh-based system may not scalably support a very large number of DS nodes.

To study the scalability of P2Dir's overlay to the number of users (IM clients), we ran experiments in which the number of DS nodes was fixed at 1,000 and the maximum number of buddies was set to 20. In these experiments, we increased the number of users from 5,000 to 10,000. Figure 4(b) depicts the average number of buddy search messages per joining user for various numbers of online users. In the figure, the upper and lower bars represent, respectively, the maximum and minimum number of buddy search messages in the test. Increasing the number of users results in a moderate increase in the average number of buddy search messages, as shown in Figure 4(b). This result suggests P2Dir achieves good scalability with the number of users. Recall that the search complexity of the P2Dir system is $O(4\sqrt{n}+b)$. Based on the analysis in Section V, we can calculate that the maximum number of buddy search messages in this case is 148, which does not exceed the analysis bound. Hence, the experiment results verify our analysis. Figure 4(b) also shows that most of the DS nodes transmit roughly the same number of search messages when a user joins.

Next, we investigate the search satisfaction of P2Dir. We used our simulator to study the buddy search latency of P2Dir while varying the number of DS nodes. We ran experiments in which the number of users was fixed at 10,000 and the maximum number of buddies was set to 20. Figure 5(a) shows the buddy search latency as the number of DS nodes is increased to 1,000. The upper bar in the figure represents the maximum buddy search latency in the test, $\max t(p)$, and the point is denoted as the average buddy search latency, $\sum_{p\in g} t(p_i)/|g|$. In the P2Dir system, the buddy search latency grows slowly with the number of DS nodes. However, the buddy search latency of the mesh-based protocol is significantly better than that of P2Dir. The reason is that, by using the mesh-based approach, every DS node can retrieve all the desired buddy information in its current replica and send the information to the user in a one-hop RTT. Note that the one-hop RTT should be quite small in our assumption. Compared to the P2Dir protocol, the mesh-based protocol can achieve a faster buddy search time and a higher replica hit ratio, but it increases the communication cost.

Although the buddy search latency is a critical metric for measuring the search satisfaction of a presence directory, to the best of our knowledge, there are no studies of buddy search latency in presence directories of IM systems. In our literature survey, we found that the average DNS lookup latency was 255.9 ms, as reported by Ramasubramanian *et al*. [33]. The results were estimated in a large-scale DNS in Planet Lab. The report could become basic reference material for user satisfaction study. Compared to the DNS lookup results in the article, the buddy search latency of P2Dir is tolerable.

The third metric is the buddy notification latency, which is also an important criterion for search satisfaction. We ran experiments in which the number of users was fixed at 10,000 and the maximum number of buddies was set at 20. Figure 5(b) illustrates the average buddy notification latency as the number of DS nodes is increased from 100 to 1,000. The upper and lower bars represent, respectively, the maximum and minimum buddy notification latency in the test. In both P2Dir and the mesh-based system, the buddy notification latency grows moderately with the number of DS nodes. However, the latency of the mesh-based protocol is slightly better than that of P2Dir. The reason is that, by using the mesh-based approach, every DS node can notify all desired buddies in one hop overlay routing, while a DS node in P2Dir needs at least two hops to reach the DS nodes, which impacts on the buddy notification latency. Clearly, there is a tradeoff. The experiment results show that the mesh-based protocol performs faster buddy searching and buddy notification latency is smaller; however, communication cost is higher. In contrast, P2Dir reduces the communication cost significantly without sacrificing search satisfaction.

VII. Discussion

A number of issues require further consideration. Here, we address security issues among the DS nodes, i.e, communication security and authentication. Here, we discuss possible solutions to these problems. The distributed P2P directory may make the IM system more prone to communication security

problems, such as malicious attacks and invasions of privacy. Several approaches have been developed to address communication security issues. For example, the Skype protocol offers private key mechanisms for end-to-end encryption. In P2Dir, the TCP connection between a DS node and users, or another DS node, could be established over SSL to prevent user impersonation and man-in-the-middle attacks. This end-to-end encryption approach is also used in the XMPP/SIMPLE protocol.

The directory authentication problem is another security problem in distributed P2Dir systems. In centralized presence directories, there is no directory authentication problem, since IM clients only connect to an authenticated presence directory. P2Dir, however, is a distributed protocol that assumes there is no trust between DS nodes; thus, a P2Dir system may contain malicious DS nodes. To address this authentication problem, a simple approach is to apply a centralized authentication server. Every DS node needs to register an authentication server, so P2Dir could certify a DS node every time it joins to the P2Dir system. An alternative solution is the PGP web in the trust model, which is a decentralized approach. In this model, a DS node wishing to join the system creates a certifying authority and asks any existing DS node to validate the new DS node's certificate. However, such a certificate is only valid to another DS node if the replying party recognizes the verifier as a trusted introducer in the system. In principle, these two mechanisms can address the directory authentication problem.

VIII. Conclusion

In this paper, we have presented P2Dir, a P2P design for a scalable directory system in support of presence service for IM systems and have shown that it is feasible to use P2P systems in a cooperative low search latency and high performance presence directory. We discussed the scalability problem of existing presence directories entirely and we introduced the *buddy-list search problem* that is a scalability problem in a general distributed presence directory. Using a simple mathematical model, we showed that the number of total buddy searching messages fatefully grows with the number of users and the number of directories. Hence, we present the design of P2Dir, a scalable P2P presence directory that leverages a 2-hop overlay to achieve small buddy search latency and resorts to an active one-hop caching strategy to reduce the search messages significantly. We quantified the performance of our P2Dir system through simulations, the experiment results show that our P2Dir achieves major performance gains, in terms of search cost and search satisfaction. Overall, P2Dir achieves high search performance by decoupling communication cost from the size of the system, it can be used as a building block for implementing customized presence director for Internet IM systems.

References

[1] J. Leskovec and E. Horvitz, "Planetary-scale views on a large instant-messaging network," *Proc. of WWW*, 2008.

[2] R. B. Jennings, E. M. Nahum, D. P. Olshefski, D. Saha, Z.-Y. Shae, and C. Waters, "A study of internet instant messaging and chat protocols," *IEEE Network*, 2006.

[3] J. Oikarinen and D. Reed, "Internet relay chat protocol," *RFC 1459*, 1993.

[4] Z. Xiao, L. Guo, and J. Tracey, "Understanding instant messaging traffic characteristics," *Proc. of IEEE ICDCS*, 2007.

[5] C. Dewes, A. Wichmann, and A. Feldmann, "An analysis of internet chat systems," *Proc. of ACM IMC*, 2003.

[6] S. A. Baset and H. Schulzrinne, "An analysis of the skype peer-to-peer internet telephony protocol," *Proc. of IEEE Infocom*, 2006.

[7] "http://www.skype.com/skype_p2pexplained.html."

[8] J. Rosenberg, H. Schulzrinne, G. Camarillo, A. Johnston, J. Peterson, R. Sparks, M. Handley, and E. Schooler, "Sip: Session initiation protocol," *RFC 3261*, 2002.

[9] "Peer-to-peer session initiation protocol ietf working group. http://www.ietf.org/html.charters/p2psip-charter.html."

[10] K. Singh and H. Schulzrinne, "Peer-to-peer internet telephony using sip," *Proc. of NOSSDVA*, 2005.

[11] D. A. Bryan, B. B. Lowekamp, and C. Jennings, "Sosimple: A serverless, standards-based, p2p sip communication system," *Proc. of AAA-IDEA*, 2005.

[12] A. Johnston, "Sip, p2p, and internet communications," *RFC Internet-Draft*, 2005.

[13] "Instant messaging and presence protocol ietf working group. http://www.ietf.org/html.charters/impp-charter.html."

[14] "Extensible messaging and presence protocol ietf working group. http://www.ietf.org/html.charters/xmpp-charter.html."

[15] "Sip for instant messaging and presence leveraging extensions ietf working group. http://www.ietf.org/html.charters/simple-charter.html."

[16] P. Saint-Andre., "Extensible messaging and presence protocol (xmpp): Instant messaging and presence describes instant messaging (im), the most common application of xmpp," *RFC 3921*, 2004.

[17] B. Campbell, J. Rosenberg, H. Schulzrinne, C. Huitema, and D. Gurle, "Session initiation protocol (sip) extension for instant messaging," *RFC 3428*, 2002.

[18] M. Day, S. Aggarwal, G. Mohr, and J. Vincent, "Instant messaging/presence protocol requirements," *RFC 2779*, 2000.

[19] "http://www.jabber.org/."

[20] P. Saint-Andre, "Interdomain presence scaling analysis for the extensible messaging and presence protocol (xmpp)," *RFC Internet Draft*, 2007.

[21] A. Houri, T. Rang, E. Aoki, V. Singh, and H. Schulzrinne, "Problem statement for sip/simple," *RFC Internet-Draft*, 2007.

[22] A. Houri, S. Parameswar, E. Aoki, V. Singh, and H. Schulzrinne, "Scaling requirements for presence in sip/simple," *RFC Internet-Draft*, 2007.

[23] H. Balakrishnan, M. F. Kaashoek, D. Karger, R. Morris, and I. Stoica, "Looking up data in p2p systems," *Communications of the ACM*, 2003.

[24] R. Cox, A. Muthitacharoen, and R. T. Morris, "Serving dns using a peer-to-peer lookup service," *Proc. of IPTPS*, 2002.

[25] A. Shaikh, R. Tewari, and M. Agrawal, "On the effectiveness of dns-based server selection," *Proc. of IEEE INFOCOM*, 2001.

[26] I. Gupta, K. Birman, P. Linga, A. Demers, and R. van Renesse, "Kelips: Building an efficient and stable p2p dht through increased memory and background overhead," *Proc. of IPTPS*, 2003.

[27] D. Eastlake and P. Jones, "Us secure hash algorithm 1 (sha1)," *RFC 3174*, 2001.

[28] A. Rowstron and P. Druschel, "Pastry: Scalable, decentralized object location and routing for large-scale peer-to-peer systems," *Proc. of Middleware*, 2001.

[29] X. Chen, S. Ren, H. Wang, and X. Zhang, "Scope: scalable consistency maintenance in structured p2p systems," *Proc. of IEEE INFOCOM*, 2005.

[30] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan, "Chord: A scalable peer-to-peer lookup service for internet," *IEEE/ACM Transactions on Networking*, February 2003.

[31] "http://pdos.csail.mit.edu/p2psim/kingdata/."

[32] K. P. Gummadi, S. Saroiu, and S. D. Gribble., "King: Estimating latency between arbitrary internet end hosts," *Proc. of ACM IMW*, 2002.

[33] V. Ramasubramanian and E. G. Sirer, "Beehive: 0(1) lookup performance for power-law query distributions in peer-to-peer overlays," *Proc. of USENIX NSDI*, 2004.