PEER-TO-PEER COMMUNITIES:

ARCHITECTURE, INFORMATION AND TRUST MANAGEMENT

by

Mujtaba Khambatti

A Dissertation Presented in Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy

ARIZONA STATE UNIVERSITY

December 2003

PEER-TO-PEER COMMUNITIES:

ARCHITECTURE, INFORMATION AND TRUST MANAGEMENT

by

Mujtaba Khambatti

has been approved

December 2003

APPROVED:

_____, Co-Chair

_____, Co-Chair

_____

_____

Supervisory Committee

ACCEPTED:

_____

Department Chair

_____

Dean, Graduate College

ABSTRACT

Peer-to-peer systems have the ability to harness vast amounts of resources from a scalable collection of autonomous peers and especially emphasize on de-centralization and lack of a central authority. As a result these systems are particularly attractive to everyday home computer users, who seem empowered by the potential to independently select and change their own policies, roles, and responsibilities. By allowing peers to share a portion of the authority, these systems also possess other interesting technical characteristics such as self-organization and adaptation.

This dissertation introduces the notion of interest-based communities of peers and describes how these implicit structures can be used to naturally organize peer-to-peer systems for discriminative information dissemination, pruning the search space, and role-based trust. Communities are like interest groups, modeled after human communities. They are self-organizing, possibly overlapping structures involving peers that are actively engaged in the sharing, communication and promotion of common interests.

Initially, the behavior of randomly created communities is investigated and modeled. Then the community formation and discovery algorithms are presented and their complexity studied. The experiments illustrate that these algorithms do not require extensive computation or communication on the part of individual peers. Subsequently, a novel push-pull gossiping technique is provided that improves decentralized information dissemination by communicating only amongst peers within a specified community of interest. The technique enables the maintenance of dynamically changing communities and consists of a distributed discovery algorithm followed by repeatable push-pull gossiping. Experiments show that pushing gossip information to only a small number of

peers allows a large percentage of peer members of a community to obtain (pull) the information within just two hops.

Peer communities help in pruning the search space and in content-based searches within the peer-to-peer network. This dissertation describes a community-based search query propagation scheme which provides more efficient searching by targeting one or more communities, irrespective of the current membership of the searching peer. The experiments demonstrate how a community-based search can reduce the number of messages and improve the quality of the search results when compared to other known peer-to-peer search algorithms.

Finally, an approach for community-based trust management is discussed. An optimistic role-based model for trust amongst peers is employed and it is shown to be scalable, dynamic, revocable, secure and transitive. The solution permits asymmetric trust relationships that can be verified by any peer in the system through a simple, low-cost algorithm.

In the name of Allah, the Compassionate, the Merciful.

To my mother, whose dedication to my success I shall always remember;

to my father, whose ingenious edification methods armed me for success;

and to my loving wife, whose inspiration and encouragement drives me towards success.

ACKNOWLEDGMENTS

My journey during graduate school was challenging and educational. I would like to acknowledge the efforts of the individuals who provided me with valuable guidance, encouragement and tools during all or a part of the process.

Most beneficial to my doctoral research was the vision, direction and significant feedback from my advisor, Dr. Partha Dasgupta; and the guidance and committed concentration towards technical quality from my co-advisor Dr. Kyung Dong Ryu. Additionally, I also received useful comments from the members of my committee, Dr. Sandeep Gupta and Dr. Donald Miller, during my doctoral research and presentations.

I frequently engaged in discussions with my colleagues and this provided me with feedback about my work and also contributed towards my education process. Notable amongst them are Austin Godber, Mufaddal Khumri, Prashant Dewan and Shu Zhang, and all other members of the Distributed Operating Systems Group at Arizona State University.

I received indispensable encouragement from my mother, while she was alive, my father, my wife, and my sister, many times despite the long distances that separated us. Finally, I would also like to mention my appreciation for my special friends Jim and Ellie Sprout.

TABLE OF CONTENTS

CHAPTER Page

CHAPTER Page

LIST OF TABLES

LIST OF FIGURES

Figure Page

CHAPTER 1

INTRODUCTION

1.1 Overview

Most earlier multi-computer, distributed systems projects were built either using a client-server architecture or as a collection of independent computers that appeared to its users as a whole. The challenges faced by systems adhering to either paradigm were: transparency, scalability, fault tolerance, heterogeneity, and openness. Client-server applications generally adopted a network operating system design and therefore had the advantage of openness, heterogeneity and scalability. However, with dependence invested in the server/s, many client-server applications suffered due to low fault tolerance. On the other hand, middleware based operating systems offered high transparency, heterogeneity and openness. The downside in this case was that all computers had to conform to a standard interface and policies set by the designers of the middleware.

The emergence of decentralized and dynamic file-sharing applications, such as Napster in 1999 and Gnutella in 2000, provided the catalyst that drew a lot of attention to a new breed of distributed systems called peer-to-peer (P2P) systems. We define a peer-to-peer system as a distributed system in which network-addressable computing elements called *peers* have comparable roles and responsibilities, communicate information, share or consume services and resources between them. The ability of peer-to-peer systems to harness vast amounts of storage from a scalable collection of autonomous peers and its emphasis on de-centralization and lack of a central authority have made it an attractive

systems solution to everyday home computer users, who seem empowered by the ability to independently select and change their own policies, roles, and responsibilities. By allowing peers to share a portion of the authority, these systems also possess other interesting technical characteristics such as self-organization and adaptation.

Throughout this dissertation, the term "peer" is used interchangeably with "node", to refer to a network-addressable computing element, like a desktop personal computer, a laptop computer, a personal digital assistant, a networked printer, or any other electronic device that has the capability of computation and network connectivity. We assume that each peer has a static IP address that serves as the *peer identity*. While this assumption is not universally true, it can be facilitated through various techniques (dynamic DNS, IPv6, or firewall penetrating mechanisms) that are outside the scope of this dissertation.

Current peer-to-peer systems are often targeted for global information sharing, replicated file storage, and searching by using an end-to-end overlay network. The building block of these systems is the notion of a peer-group, or a number of peers that cooperate with each other for a common purpose. In our research, we investigate a generalization of the notion of peer group to a multiplicity of groups (possibly overlapping) called *peer communities*. While a group is a physical collection of objects, a community is a set of active members, who are involved in sharing, communicating and promoting a common interest.

Our concept of peer communities is loosely based on the idea of "interest groups", such as Yahoo Groups, Usenet Newsgroups, or web communities. The user of a peer in the system claims to have some interests and depending upon the claims of all the peers' users, communities are implicitly formed (made up of peers with the same or similar

interests). Note that communities are formed implicitly, i.e. they are self organizing. If a peer in New York declares an interest in wombats, and a peer in China also declares the same interest, then the two peers become part of an implicit, undiscovered community. A peer may belong to many different communities and communities may overlap.

With the exception of web communities that have been shown to be self-organized (Flake et al., 2002) and the alt.* Usenet groups; almost all other present-day groups are a result of *a priori* planning and implementation or at the very least they require some central control or a central authority through which advertisements can be made. In contrast to this, peer-to-peer systems are usually completely de-centralized and can also be dynamic.

We associate a peer-to-peer network with graph G (V, E), where the set of nodes, V, represents peers and the set of edges E consists of end-to-end overlay *links* between pairs of peers from V. We consider this network of peers as being analogous to social networks that are comprised of humans. In fact, much of our proposed modeling of peer-to-peer networks was motivated by our observations of similar patterns in social networks. For instance, the inclination of autonomous elements in a social system to form groups and associations led us to believe that a populated peer-to-peer system that is made up of peers and an end-to-end overlay network will also form similar groups (communities). Thus, peer-to-peer communities are a natural extension for arranging distributed peer-to-peer systems. Like their social network counterpart, communities also enhance the capabilities of each member.

1.2 Statement of the Problem

The recent popularity of peer-to-peer systems has fueled numerous research proposals and commercial ventures to organize peer-to-peer networks, efficiently search for files, secure information, and provide new applications. Of these approaches, some propose structured peer-to-peer networks, where there is a close coupling between the topology of the network and the location of data. This dissertation focuses entirely on unstructured peer-to-peer networks where the reverse is true, i.e. there is no coupling between network topology and the location of data. In fact the topology of the network is not controlled by any global scheme and is allowed to adapt, grow/shrink dynamically as new nodes join or leave the network corresponding to the actions of their users.

The main challenges that needed to be addressed were:

1. Distributed discovery and dynamic maintenance of communities
2. Distributed dissemination of information
3. Distributed searches of information
4. Trust management

An evaluation of our research will demonstrate the feasibility and efficiency of our algorithms with the support of peer communities.

The canonical application that we consider for our algorithms is a digital library built out of a collection of peers in which each peer owns a set of books that it is willing to share with other peers[1]. The subjects of the books owned by a peer form its set of interests. Peers are implicitly grouped into communities based on the common interests

[1] Assume these are non-copyrighted works.

they share. Because a peer could own books from a variety of subjects, we can imagine that a peer could be a member of multiple communities.

1.3 Potential Contributions and Limitations

This dissertation proposes a natural organization of autonomous peers into communities that can be uncovered using decentralized techniques. It provides a motivation for the study of peer-to-peer communities and illustrates some scenarios to define and discover peer communities. Using simulated models of communities, we have gained an insight to the architecture of randomly created communities. Our algorithm for the discovery of communities allows for the computation of *Link Weights*, a very important value that enables the working of all our subsequent algorithms. Link weights help determine the membership of a peer in a community. They are also used to rank peers in a community for the purposes of information dissemination and trust.

Next it presents a novel *push-pull* communication technique that utilizes communities for better information dissemination via a repeatable push-pull gossiping protocol. Prior approaches for information dissemination or information retrieval within a peer-to-peer network have tried to send out messages through all or selected peers and up to a certain depth. Unlike these approaches, our technique involves a distributed discovery phase to gather data on peers and identify highly popular peers (called *seers*) in a community. Unlike *supernodes* or *hubs*, which are peers that have a lot of links to other peers, seers are identified based on their links to peers within a community. It describes the distributed discovery algorithm and show that it is a low overhead, simple protocol that is also resilient to failures and delays in peers.

Thereafter, the push phase of peer-to-peer gossiping multicasts information to the seers identified in the distributed discovery phase. Whenever required, a peer can easily and quickly retrieve this information from a nearby seer via a pull phase. Our experiments show that pushing gossip information to only a small number of nodes allows a large percentage of peer members to obtain (pull) the information within just two hops.

Our proposed solution for pruning the search space and content-based searches within the peer-to-peer network also takes advantage of interest-based communities of peers. Earlier peer-to-peer search techniques, such as flooding, directed flooding, iterative deepening (Yang and Garcia-Molina, 2002), and local indices (Yang and Garcia-Molina, 2002), had a major drawback that information located farther away from a peer can be found only at a considerable search expense. We built a community-based search query propagation scheme that provides more efficient searching by targeting one or more communities, irrespective of the current membership of the searching peer. Our technique follows the innate method of searching that human beings use in the analogous social network, where queries for unknown items are asked to "those that know." The community-based search technique also allows search operations to be based on content rather than just filename searches employed by many existing peer-to-peer search techniques.

Finally, this dissertation proposes an approach for community-based trust management in peer-to-peer systems. It presents an optimistic role-based model for trust amongst peers and shows that it is scalable, dynamic, revocable, secure and transitive. Our proposed solution permits asymmetric trust relationships that can be verified by any

peer in the system through a simple, low-cost algorithm. This dissertation also introduces a metric known as *iComplex* that combines a peer's trust values for each of its roles into a single, relative, probabilistic guarantee of trust. At the end, it discusses how our trust model allows peers to revoke relationships with malicious peers, and the non-repudiation of peer relations.

In terms of limitations, the techniques that we developed can only be applied to specific applications, such as the digital library, where the set of interests is constrained, well defined and understood by almost all the peer members. Our proposed algorithms would place individual users into peer communities based on the common interests that they share with other peers. A generalized peer-to-peer system, where the set of interests includes the universe of all possible interests, might not contain a single peer that shares common interests with other peers. Therefore no communities would form.

Some of the other problems that are outside the scope of this dissertation are: specifying how static IP addresses can be used as peer identities, providing more than just probabilistic guarantees of trust, describing the channel or protocol of communication used by peers, defining the format for the interests of a peer, obtaining the interests from a peer, and fragmentation of the network after targeted denial-of-service attacks on seers.

1.4 Organization

This dissertation is organized as follows:

Chapter Two surveys literature related to the area. It describes related applications, such as existing peer-to-peer applications; related systems, such as complex,

self-organizing systems; and related techniques in information dissemination, distributed search and decentralized trust models.

Chapter Three discusses our modeling of a peer-to-peer network. The chapter introduces and defines some important terms, including Link Weights, which are associated with modeling peers. Next, it presents our method of forming peer-to-peer networks for use in our experiments.

Chapter Four presents our Attribute Escalation algorithm to aid peer community formation and subsequent discovery. Our experiments reveal that Attribute Escalation stabilizes after just one round of communication.

Chapter Five contains our technique for disseminating information through peer communities using our novel push-pull communication method. The chapter also describes the Distributed Discovery algorithm which is executed before gossiping can disseminate information amongst community members.

Chapter Six provides a discussion on our community-based search technique. It describes the experiments used to evaluate our method with existing peer-to-peer search algorithms. These experiments illustrate that community-based search results in higher quality results, usually with lower communication costs.

Chapter Seven delves into our trust management using peer communities. First, the trust model is introduced. Next, the chapter details how the trust model can be employed in a distributed system. Finally, the chapter discusses the revocation and non-repudiation of trust amongst peers in the network.

Chapter Eight provides our conclusions and recommendations for future directions.

CHAPTER 2

BACKGROUND LITERATURE

2.1 Related Applications

This section describes existing groupware, collaborative and peer-to-peer applications that are related or appear related to our work with peer communities.

2.1.1 Groupware and Collaborative Applications

Groupware are computer-based systems that support groups of people engaged in a common task (or goal) and provide an interface to a shared environment (Ellis, Gibbs and Rein, 1991). Some examples of groupware computer-based systems are message systems, group decision support systems, electronic conference rooms, multi-user editors, coordination systems, and intelligent agents.

Usually, groupware is not developed ‘‘from scratch’’, but with the help of a groupware toolkit. One particular example of a groupware application is Jive Forums (Jive Software). This is an open architecture Java based discussion forum application. Jive Forums is a flexible and reliable discussion forum software, and can handle the heavy traffic of major sites.

Argo (Gajewska et a., 1994) is another groupware application that allows medium-sized groups of users to collaborate remotely from their desktops, in a way that approaches as closely as possible the effectiveness of face-to-face meetings. It combines high quality multi-party digital video and full-duplex audio with *telepointers*, shared applications, and whiteboards in a uniform and familiar environment.

2.1.2 Peer-to-Peer Applications

PAST (Druschel and Rowstron, 2001) is a large-scale, peer-to-peer archival storage utility that provides scalability, availability, security and cooperative resource sharing. Files are immutable and can be shared at the discretion of their owner. PAST is built on top of Pastry (Rowstron and Druschel, 2001).

Another example of a peer-to-peer storage system is OceanStore (Kubiatowicz et al., 2000). This utility is a global persistent data store designed to scale to a large number of users. It provides a consistent, highly available, and durable storage utility on top of an infrastructure comprised of untrusted servers.

P-Grid (Aberer, 2001) is a peer-to-peer system based on a virtual distributed search tree. The tree exists in part within each peer and only the cooperation of all peers can provide a view of the overall tree. Every participating peer's position is determined by its path, that is, the binary bit string representing the subset of the tree's overall information that the peer is responsible for.

KaZaA is a peer-to-peer file sharing system that allows users to search and download files. It gives clients more power by transforming them into "*supernodes*" that can broker search requests of weaker clients.

On the similar lines, Farsite (Bolosky et al., 2000) is a serverless, distributed file system that does not assume mutual trust among the client computers on which it runs; and Publius (Waldman, Rubin and Cranor, 2000) is a Web publishing system that is highly resistant to censorship and provides publishers with a high degree of anonymity.

Some other peer-to-peer projects like (Stoica et al., 2001) aims to build scalable, robust distributed systems using peer-to-peer ideas. Their work is mainly based on the

Chord distributed hash lookup primitive. There have even been attempts such as Anthill (Babaoglu, Meling and Montresor, 2002), which is a framework aimed at supporting the design, development and analysis of peer-to-peer protocols and algorithms.

Also noteworthy is Kademlia (Maymounkov and Mazi`eres, 2002), a novel routing algorithm for peer-to-peer networks based on the XOR metric. The Kademlia project is a research effort to implement a full-featured peer-to-peer system based on the XOR metric routing.

2.2 Related Systems

We consider peer-to-peer networks and complex systems as being closely related. Complex systems have been defined by (Flake, 2000) as "a collection of many simple nonlinear units that operate in parallel and interact locally with each other so as to produce emergent behavior." Patterns in several complex systems have been found to be self-organizing (Axelrod and Cohen, 2000), often because of the autonomous creation of links by participating nodes (with some influence of a partial system view). Discussed below are some noteworthy techniques and properties of complex, self-organizing systems.

2.2.1 Link Structure Analysis

A considerable amount of research has focused on the analysis of link structure in collections of objects. Through these analyses, researchers had hoped to discover a process that could be implemented to effectively identify and discover specific patterns in the collection. Early attempts to analyze the collective properties of interacting agents have been found in social networks (Scott, 1991), where link structures like cliques,

centroids and diameters were studied. The field of citation analysis (Garfield, 1979) and bibliometrics (White and McCain, 1989) seek to identify patterns in collections of literature documents by using citation links. Stated below is one such notable definition of web communities that uses an analysis of the link structure of web pages to effectively identify communities.

"We define a community to be a set of web pages that link (in either direction) to more web pages in the community than to pages outside of the community." (Flake, Lawrence and Giles, 2000)

At first glance, the above definition seems just what is needed to identify peer-to-peer communities. However a closer look will indicate that if peers are placed in peer-to-peer communities based entirely on link analysis, we will not accomplish our goal of allowing peers to simultaneously be members of more than one peer-to-peer community.

A particularly remarkable solution was proposed by (Kleinberg, 1998) to discover patterns by analyzing links. It offers an approach called HITS that is related to spectral graph partitioning and methods used by the Google search engine (Page, 1997). Their recursive definition of hub and authority web pages work to rank results by a measure of importance and usefulness, thereby identifying key web sites related to some community and also the related websites that might be members of the same community (Gibson, Kleinberg and Raghavan, 1998). However their approach depends on the link topology; and therefore, it cannot effectively assist in the discovery of communities that are ring-based without any dominating members.

Complementary to the HITS algorithm, (Flake, Lawrence and Giles, 2000) requires a "seed" web site as the starting point to begin a focused crawl in order to

identify a community. Members of the community are discovered by using the maximum flow / minimum cut framework using the two sets called source and sink, initially composed of well-known web sites. When mapped to the peer-to-peer domain, this link-based technique has the drawback of not truthfully modeling real world peer-to-peer communities where peers can simultaneously be members of more than one community.

Perhaps a combination of graph theory and link analysis is needed to correctly identify patterns within collections of peers. For instance, if the links of a peer were classified, as outgoing and incoming links, and if there existed cycles in the directed graph formed by the peers and their links, we would have successfully identified peer-to-peer communities. These cycles could be discovered in a depth-first manner by exhaustively traversing the outgoing links to check if they visit a peer twice.

Except for the expected scalability problem in the above technique, it seemed to guarantee to uncover any pattern that might exist. Yet the procedure would fail for some of the most commonly occurring patterns: the star, where many peers are huddled around a single peer like in a client-server configuration, or the tree, where no cycles would exist. Even in a domain that is purely peer-to-peer, one cannot avoid the fact that often many peers will use a small subset of peers or even a single peer as a source of information or for collaborations.

In trying to empower peers to discover their community membership, we had to find an efficient solution that would be practical, unlike the NP-complete solutions offered by graph partitioning (Garey and Johnson, 1990).

2.2.2 Small-World Networks

Previously, self-organizing systems had been described by Erdös and Rényi (1961) who modeled them as random networks and studied their properties. In random networks, the topology of the network is the result of links between randomly selected nodes. Watts and Strogatz (1956) also studied the properties of large regularly connected graphs of nodes that contain a few random long-distance edges between nodes. They modeled this structure and demonstrated that the path-length between any two nodes of the graph is in fact surprisingly small. As a result, they called these semi-random systems *small-world networks*. These networks have low characteristic path lengths (as in random networks) and high clustering coefficients (as in regular networks). Subsequently, a number of papers have acknowledged the existence of small-world networks in the Internet topology (Bu and Towsley, 2002; and Albert, Jeong and Barabási, 1999); the power grid of the western United States; various social networks (Watts and Strogatz, 1956), such as the collaboration graph of film actors; Erdös numbered research scientists; and even in the neural network of the worm *Caenorhabditis elegans*. Further, Granovetter (1973) discusses the existence and shows the importance of weak social ties (links) between highly connected clusters of friends.

The similarity of peer-to-peer networks to social networks and the fact that humans direct peer links led us to believe that peer-to-peer networks would also exhibit small-world behavior. In fact, this has already been observed in existing peer-to-peer networks, such as Gnutella.

2.2.3 Scale-Free Networks

More recently, Strogatz (2001) and Amaral et al. (2000) observed that many networks demonstrated topological properties that were different from the predictions made by random network theory. In particular, the existence of some very well connected "hub" nodes dramatically influenced the behavior of these networks during random node failures and spreading of information. These networks are characterized by the uneven distribution of connections (links) in the nodes of the network.

Specifically in these networks, called "*scale-free networks*," the degree distribution of participating nodes was found to decay as a power law, very much unlike a random network that exhibits a Poisson distribution of node degrees. Hence, scale-free networks have sometimes been described as *power-law networks.*

It has been shown that various networks, such as the collaboration graphs of actors and scientists, were developed due the feature of *preferential attachment* (Jeong, Néda and Barabási, 2003; Newman, 2001; and Barabási and Albert, 1999). This feature describes the probability of a node acquiring new links as an increasing function of the links that it currently has.

In order to correctly model the peer-to-peer network, it is important that we also incorporate the scale-free property into the network topology.

2.2.4 Multicast Groups and Communities in Ad Hoc Networks

Multicast groups and ad hoc communities may seem similar to our work. A literature-survey with special emphasis on the differences between our approaches is provided below.

Caronni et al. (1998) support dynamic groups of arbitrary size where members can join and leave at random. Groups are created by a Group Manager who is responsible for receiving and processing "join" and "leave" requests from participating nodes. Additionally, group parameters are published using a directory service. In the distributed flat key management scheme, certain nodes in the group perform a regular *heartbeat* communication by sending out some messages using a multicast, broadcast, or anycast channel.

Keoh and Lupu (2003) call an ad hoc network of devices a *community*. This differs greatly from our definition of communities as interest-based groups of peers. The approach relies on a community specification, called *doctrine* that is created prior to the establishment of the community. The doctrine specifies various policies, roles and permissions that users need to abide by in order to participate in activities of the community.

Hong and Gerla (2002) introduce a dynamic group discovery and formation technique that aggregates ad hoc nodes that have *movement affinity*. Like in our work, these groups are dynamically formed, split or broken down and group members may join or leave at random. Their system requires the execution of Group Leader Election algorithms that rely on periodic broadcast of messages.

Meissner and Musunoori (2003) propose a group integrity management scheme and describe scenarios that seem similar to the goals of our research. Their concept of groups involves WiFi-enabled gadgets aggregated based on application-level or user-specified reasons, such as a "music" group or a "books" group. Members of a group

communicate using either unicast or multicast and each group requires a group manager that administers join or leave requests.

In contrast to these systems, peer communities implicitly form due to declared interest attributes of peers, they require no manager, peers join and leave a group implicitly, but can efficiently discovery their community memberships, and inter-group communication uses a push-pull technique.

2.3 Related Techniques

Although peer-to-peer systems have only recently become popular, distributed algorithms, such as gossiping, have been around for more than a decade. This section provides an overview of some existing algorithms and techniques that have similar goals as our peer-to-peer techniques for information dissemination, search and trust.

2.3.1 Information Dissemination in Distributed Systems

Listed below are some well-known techniques that are being used for propagating information in distributed systems. Some of these algorithms are also applicable for searching peer-to-peer networks.

a. Flooding and Swamping

The flooding algorithm allows for directed communication amongst an a priori fixed set of neighboring machines (Harchol-Balter, Leighton and Lewin, 1999). Internet routers use this method today (Moy, 1998]. The swamping algorithm is similar to the flooding algorithm, except that, it facilitates communication between all the neighbors of a particular machine, not just its initial neighbors (Harchol-Balter, Leighton and Lewin, 1999).

b. Random pointer jump

The disadvantage of the swamping algorithm is that it is highly communication intensive. Instead of communicating with all its neighboring machines, if only one randomly chosen neighbor was selected during each round, it would reduce the communication complexity. However, this method can only be applied to strongly connected networks, where there is a path between every pair of machines (Harchol-Balter, Leighton and Lewin, 1999).

c. Gossiping and Rumor Spreading

The class of gossip algorithms has long been used in distributed systems (Agrawal, Abbadi, Steinke, 1997; Demers et al., 1987; Hayden and Birman, 1998; Oppen and Dalal, 1983; Pelc, 1996; and Renesse, Minsky and Hayden, 1998). Some examples of how it is being used include efforts to maintain consistency in a distributed replicated database (Agrawal, Abbadi, Steinke, 1997; and Demers et al., 1987), or to gather information about failures in a network of machines (Renesse, Minsky and Hayden, 1998). Randomized rumor spreading (Karp et al., 2000) is a similar epidemic algorithm that can be used for the lazy transmission of updates to distributed copies of a database. The algorithms use a simple randomized communication mechanism to ensure robustness. However, most gossiping algorithms assume knowledge of all the machines that exist on the network, and that information from one or more of the machines needs to be broadcast to the others.

d. Name-Dropper

The Name-Dropper (Harchol-Balter, Leighton and Lewin, 1999) algorithm is similar to the Random Pointer Jump algorithm, but it has a lower communication cost and

also requires fewer rounds to terminate. The Name-Dropper algorithm has been implemented at MIT to solve a resource discovery problem. Currently, it is also being used by Akamai Technologies. In contrast to gossiping and rumor spreading, the Name-Dropper algorithm attempts to broadcast in networks where the machines might not be aware of each other's existence. Their results show that allowing machines to learn about the existence of other machines during the gossip process will make gossiping efficient even when starting from a weakly connected graph.

2.3.2 Peer-to-Peer Searching Techniques

In the client-server model, a single server or a group of servers that has access to a data repository execute the search algorithms on behalf of a requesting client machine. The absence of servers in the peer-to-peer model makes searching for information in these networks a potentially costly operation. Peer-to-peer searching involves cooperatively passing a query message until a peer that published the desired information is found. Current peer-to-peer applications employ search algorithms that attempt to bring down the cost of searching in terms of number of hops/messages while still covering the largest possible number of peers in the system.

Search techniques in unstructured peer-to-peer systems have loose guarantees. Existing techniques include the naïve flooding; propagating search queries to all neighboring peers who in turn forward the query until the query has been forward a pre-defined number of times (Gnutella); forwarding a search query with a pre-defined hop limit to only one neighboring peer at a time (Clarke et al., 2002); employing highly connected peers or "superpeers" to propagate or broker the search query (Adamic et al.,

2001; and KaZaA); peer gossiping to maintain accurate local copies of membership directories and summaries of shared content (Cuenca-Acuna et al., 2003); and so on.

Often structured approaches employ Distributed Hash Tables (DHT) to organize peer-to-peer networks. DHT based systems require participating peers to store either entire files or file locations when the identity of the peer corresponds to the hash of the filename published by another peer. Therefore, DHT based systems can be used to perform efficient filename searches because they guarantee the location of the data if it exists within the system at the cost of a data insertion overhead (i.e. the process of updating tables at the peer whose identity matches the hash of the filename). Example algorithms include (Stoica et al., 2001), which uses a distributed hash lookup primitive to build a scalable and robust peer-to-peer system; and (Rowstron and Druschel, 2001), which is a large-scale, peer-to-peer archival storage utility that provides scalability, availability, security and cooperative resource sharing.

Additionally, (Yang and Garcia-Molina, 2002) has proposed techniques that reduce communication and increase the probability of arriving at a solution to the search query. Some examples of these proposed techniques are iterative deepening, which iteratively increases the depth unto which to flood; directed BFS, which selects a subset of immediate neighbors heuristically and entrusts them with further propagation of the search query using directed BFS again; and local indices, which require each peer to maintain a local index of the contents of peers that exist within a pre-determined range.

2.3.3 Trust Models

Perhaps one of the earliest formalizations of trust in computing systems was done by Marsh (1994). He attempted to integrate the various facets of trust from the disciplines of economics, psychology, philosophy and sociology. Rahman and Hailes (2000) proposed a trust model based on the work done by Marsh but specifically for online virtual communities where every agent maintained a large data structure representing a version of global knowledge about the entire network. Gil and Ratnakar (2002) describe a feedback mechanism that assigns credibility and reliability values to sources based on averages of feedback received from individual users.

More along the lines of trust and social networks, Golbeck, Hendler and Parsia (2003) presented an approach to integrate social network analysis and the semantic web. Yu and Singh (2000) introduced a referral graph comprising agents as weighted nodes and referrals as weighted edges between participating agents. The graph topology can be changed over time, for instance after bad experiences agents can change their list of neighbors and also propagate information about the "bad" agent within the network. Yolum and Singh (2003) propose a similar approach that enables the study of the emergence of authorities in self-organizing referral networks. Pujol et al. (2002) associate reputation of an agent with its degree in a social network graph. Similar to PageRank in Google, an agent gets a high reputation if it is pointed to by other agents that also have high reputation. Aberer and Despotovic (2001) analyze earlier transactions of agents and derive from that the reputation of an agent. Reputation provides a value that indicates the probability that the agent will cheat. They also presented a design for trust management using their proprietary decentralized storage method.

2.4 Summary and Conclusions

This chapter presented a few existing groupware, collaborative and peer-to-peer applications that are related to the community-based organization of peers described in this dissertation. Next it discussed some of the characteristics of complex, self-organizing systems which are related to our peer-to-peer network model, specifically the properties of small-world networks and scale-free networks. Finally, the chapter ended with an overview of some algorithms and techniques that have the related goals of distributed information dissemination, distributed searching and trust modeling.

Our work introduces a novel community-based organization of peer-to-peer networks that retains the flexibility of unstructured peer-to-peer networks and assists in searching, gossiping and trust. Our gossiping algorithm employs a simple push-pull operation that exploits community-based link structures. Our technique for searching also makes use of peer communities to prune the search space, by following the innate method of searching that humans use in the analogous social network, where queries for unknown items are asked to "those that know." Peer-to-peer systems used for information exchange have either protected peers' anonymity, or required transacting peers to trust each other implicitly. Both these approaches are vulnerable to attacks by malicious peers who could abuse the peer-to-peer system to spread viruses, incorrect, or damaging information. Finally, we associate trust values with Link Weights instead of links.

CHAPTER 3

MODELING A PEER-TO-PEER NETWORK

3.1 Modeling Peers

The formation and discovery of peer communities depend significantly on how peers declare and use their common "interests". In this section, attributes are first described as a method of declaring interests. A thorough treatment of the subject of peer links is then provided later in the section.

3.1.1 Interest Attributes

In our model, interests are represented by attributes, which are used to determine the peer communities in which a particular peer would participate. Attributes can be either explicitly provided by a peer or implicitly discovered from past queries. For example, a housewife can express that she is interested in French wines and house decoration. Such expressions are personal declarations. Also, her repeated web search queries to find "K-12 education in Arizona" can be used to provide implicit information about her interests. For the implicit discovery of interest attributes, we assume the presence of a semantic analyzer that can access browser history, sent-mail and other files available on a peer's storage to intelligently deduce the list of tasks for which the peer is being regularly used. This list forms the list of implicitly derived interests. Examples of semantic analyzers that could be incorporated into our system with some minor changes are (Hayes et al., 1986; and Matrouf et al., 1990). Our system design, however, does not depend on the existence of a semantic analyzer. Therefore, the list of interests can even be entirely obtained from the owner(s) in an explicit manner (possibly through a number

of iterations). There are of course privacy and security concerns in using such information, so we divide interests into two classes – *personal* and *claimed*.

The full set of attributes for a peer is called *personal attributes*. However, for privacy and/or security reasons, all these attributes may not be used to determine community membership. A client may also not want to reveal some of her personal attributes because she might not consider them relevant amongst the peers that she knows. Hence, a peer explicitly makes only a subset of these attributes public, which are called *claimed attributes*.

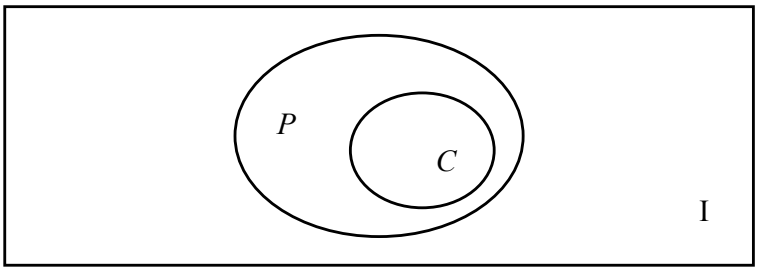


Figure 1. Venn diagram of interest attribute sets for a peer

In Fig. 1, *I* is the universe of all attributes; *P* is the set of personal attributes; and *C* is the set of claimed attributes. Below, we formally define a peer-to-peer community and its signature based on the attributes of each peer.

Definition 1

Peer-to-Peer Community

The non-empty set N of nodes is a peer-to-peer community iff N has a non-null signature.

Definition 2

Signature of a Set of Peers

Let i be a node and C_i be a set that contains attributes claimed by i. Consider a non-empty set N of nodes. Then the set resulting from the intersection of C_k, for all $k \in N$ is called a signature of the set N.

With these definitions, given any collection of peers, we would be able to tell whether the collection is a peer-to-peer community or not.

3.1.2 Peer Links

We observe in projects like HITS (Kleinberg, 1998) and Web Communities (Flake, Lawrence and Giles, 2000) the concept of self-organized communities that form implicitly based on hypertext links between web pages. The human creators of the web page explicitly place these links typically in order to point towards web pages with similar content. This is one of the factors that Internet search engines have exploited to enhance their search operation.

We draw an analogy from the above research to understand the behavior of peer-to-peer systems. We find that peers also regularly link to other peers, in the form of relationships (being present in their address book), or direct connections (being on the same network), when their human owners share something in common. We assume that these links are bi-directional communication channels that can be established on an as-needed basis. In a social network, this is similar to getting in touch with people you know when you need something. We refer to these end-to-end overlay communication channels as *peer links*.

3.1.2.1 The Need for Peer Links. Links are not necessary to form and manage peer-to-peer communities. However, they are needed to feasibly run low-cost algorithms that are used for formation and discovery of communities, as it is conceptually and algorithmically simpler to use the notion of a set of *neighbors*, which are directly (1-hop) linked peers, when communicating with other peers.

When node *X* is born, it needs to have one or more logical neighbors. If it has three neighbors, *A*, *B*, and *C*, then we say that it has three links, $X \rightarrow A$, $X \rightarrow B$, and $X \rightarrow C$. Unlike overlay networks used by Distributed Hash Table (DHT) based peer-to-peer systems, our link based network is not contingent to node names, but to user selected neighbors. Like in social networks, the more links a node acquires the more successful it will be in receiving information and searching the peer-to-peer network. It is the responsibility of each peer to acquire as many neighbors as possible.

First, let us explain a case where links are essential. Suppose a node, belonging to domain abc.com claims the attribute "baseball". This node is essentially isolated, unless it *a priori* knows about the other members of the baseball community or the other members of the abc.com community. There is a need for a "seed" to start the community formation and information search needs.

3.1.2.2 Creating Peer Links. Flooding and querying a central server are two solutions to the isolation problems; however, the first is expensive and the second violates the self-configuring tenet of the peer-to-peer model.

A new node *X* has the following options that solve the isolation problem:

1. Connect to a special bootstrapping node present within each network domain

2. Connect to a peer known to *X* – a friend / colleague.

For a novice/new node, the first option may be the most appropriate. As *X* ages, it finds other nodes and adds these links to improve search speed and information access. The linkages are bi-directional and similar to friendships in real life, or to http links in the Web. They are directed by humans.

3.1.3 Link Weights

Peer links are used to compute *Link Weights* at each peer in the network. Below, a definition for Link Weights is provided followed by an illustrative example in Fig. 2.

Definition 3

Link Weight

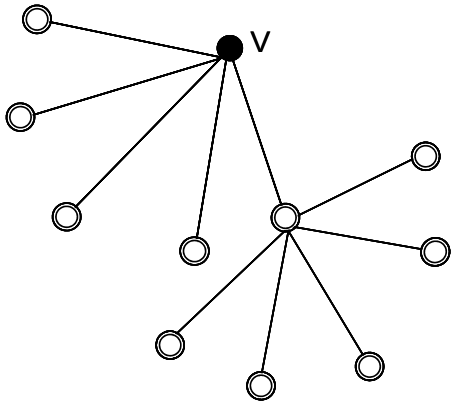
This is the weight calculated for each claimed attribute of a peer V based on the number of links from V that can reach, after at most one indirection, other peers that claim the same attribute.

The constraint of at most one indirection is necessary to restrict the maximum depth up to which peers will be examined since more than two levels deep resulted in an unacceptably high number of communication messages. See section 4.4.3 for the average number of peers that are reachable from a peer.

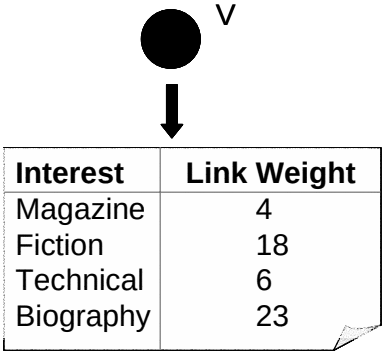
The Link Weight values of a peer can change over time because of the value's tight coupling with peer links. All peers are not equal in terms of the number of their neighbors. Older peers, i.e. peers that have been part of the network longer than others, are likely to have higher Link Weights because they have more time to accumulate more neighbors. This is observed in our experiments described in section 5.3.4. In addition to

favoring older peers, higher Link Weights are observed in peers that have popular/ interesting content or useful resources. As peers in the network search for and discover these content or resources at the peer that owns them, their human users might create a link to the content/resource owner to avoid subsequent searching. Finally, since peer links are directed by humans, social aspects, such as the popularity of the peer's owner, will contribute to the increase in Link Weight values at a peer.

Link Weight values can conversely be reduced by the removal of links through a process we call revocation (see section 7.6.1). The removal of links might be a result of the waning interest in content or resources at a peer, malicious behavior of a peer during transactions, or simply due to social aspects, such as death, separation, and so on, of the peer owner. Due to the dynamic nature of peer links, Link Weight values are calculated at regular intervals decided autonomously by each peer.



a. Comparing with 1-hop / 2-hop neighbors



b. Link Weights for each attribute

Figure 2. Peer V compares its claimed attributes with neighboring peers and then calculates Link Weight values

As mentioned earlier, this value is very important to help determine the membership of a peer in a community and rank peers in a community for the purposes of information dissemination and trust. Although later chapters explain how this value is used in our algorithms, a summary of the importance of Link Weights is provided in the following paragraph.

After the community formation algorithm described in chapter 4, each peer calculates its Link Weights by analyzing the claimed attribute sets that it receives from neighboring peers. The subsequent community discovery algorithm uses the definition of community membership, which relies on Link Weight values. A critical part of our information dissemination algorithm is the selection of special peers to carry out push-pull gossiping. These peers are selected by virtue of having high Link Weights. During community-based search query propagation, higher Link Weights of peers responding to the query indicate that the peers are more likely to have the requested resource. Our trust model uses Link Weights as an indication of role-based trust. Finally the single *iComplex* trust value described in chapter 7 is calculated using Link Weights.

3.2 Peer-to-Peer Network Formation

In this section, two approaches are explained for generating a realistic peer-to-peer network topology that we used to simulate our algorithms.

3.2.1 Using the Internet Topology

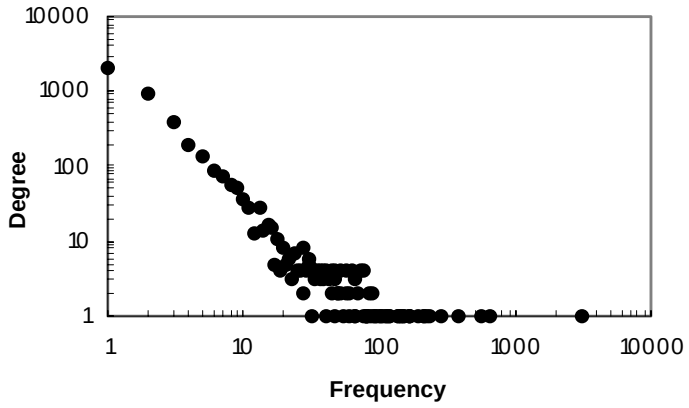
Due to the manual nature of setting up links, we believed that the resulting network topology would be similar to that of the Internet. In fact, the pre-cursor to the Internet (Arpanet) was one of the first peer-to-peer networks, with its ability to connect

computers as peers having equal rights in sending and receiving packets. We therefore wrote a spider program that crawled a subset of the Internet (a pre-specified domain, such as “asu.edu”) and created a topological map that could be used to calculate various parameters such as clustering coefficient and power-law exponents. The spider program started at a user-specified website, requested its HTML content and parsed the HTML code to extract all the linked websites. It recorded the websites that lay within a pre-specified domain, such as “asu.edu” and discarded the rest. Thereafter, the spider recursively visited each website from its recorded list repeating the same steps. By programming the spider to travel the Internet domain using an Eulerian path, we could create a map of the web topology where each node was a website and edges represented a link from one site to another.

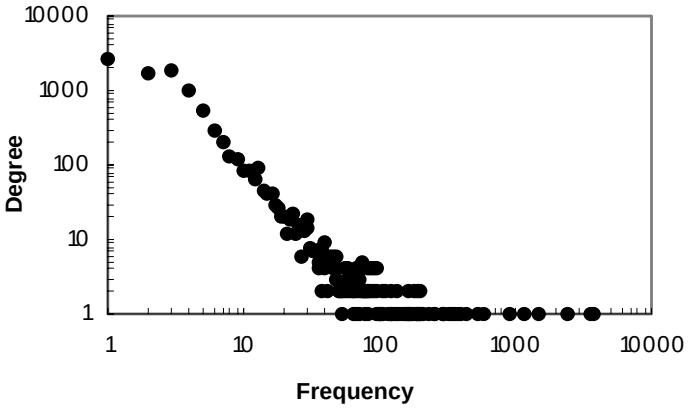
Apart from a few changes, like converting the graph from a directed graph to an undirected graph, the web topology graph that we generated closely resembled a peer-to-peer network. Websites are analogous to peers; and http links are equivalent to peer links. The other reason for the close resemblance was the similarity between website content and a peer’s interest attributes.

Fig. 3 demonstrates the power-law distribution of the frequency (logarithmic X-axis) Vs the degree (logarithmic Y-axis) in the two Internet topology maps. However, our calculations for small-world behavior revealed low characteristic path lengths (2.46 and 2.32) and low clustering coefficients (0.28 and 0.32) for asu.edu and umich.edu, respectively. We attribute this phenomenon to the popular use of increasingly efficient search engines on the Internet. While a few years ago, a website owner had to place http links to frequently visited websites on her website so that they could be easily accessible,

contemporary search engines efficiently locate websites so that many website owners do not even have to link to the websites of their colleagues and friends any more. The resulting topology graph therefore showed fewer regular links, and calculations for the clustering coefficient hence revealed low values.



a. “asu.edu” domain



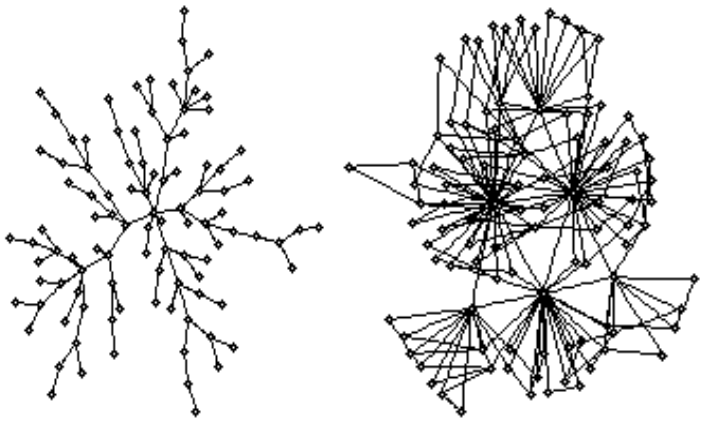
b. “umich.edu” domain

Figure 3. Power-law distribution of the frequency Vs degree plot for the web topology graphs

3.2.2 Creating Our Own Peer-to-Peer Network

A peer-to-peer network comprising of peers linking to known peers is analogous to social networks, and therefore should have a high clustering coefficient to represent the interconnected social links amongst circles of friends. We needed to provide a mechanism to guarantee that our peer-to-peer network topology exhibits the properties of a small-world network and also shows a power-law distribution for frequency versus degree.

Our next approach involved enforcing certain rules (similar to preferential attachment) on new peers that wanted to join the peer-to-peer system. We were inspired by the work of Steyvers and Tenenbaum on semantic networks, and extended the domain of their model to a peer-to-peer network that involved peers and links.



a. Random Attachment b. Enforced rules

Figure 4. Topology of networks grown up to 100 nodes

Fig. 4 (a) indicates what happens if *Random Attachment* is used to create the peer-to-peer network, i.e. new nodes join the network by linking to a randomly picked

existing node. The resulting network resembles a tree instead of a graph. Fig. 4 (b) shows a peer-to-peer network that has been formed using our enforced rules. The rule-based network is closer to a realistic topology. A new peer, *X*, therefore has to follow a two-step procedure (described below) in order to join a peer-to-peer system.

Let N = {set of peers known to X} *(See section 3.1.2)*, the symbol $\xrightarrow{\wp}$ means "connects (links) to", with probability $\wp$, and $\wp_A \propto$ degree of node A, which is denoted as k_A. The two-step procedure is:

$X \xrightarrow{\wp} A, \quad A \in N$ **Step (1) ,** *X selects a node A and connects to it*

$\forall\ B = m'_A \in M'_A,\ X \xrightarrow{1} B$ **Step (2) ,** *X selects d nodes from M'$_A$ to link to*

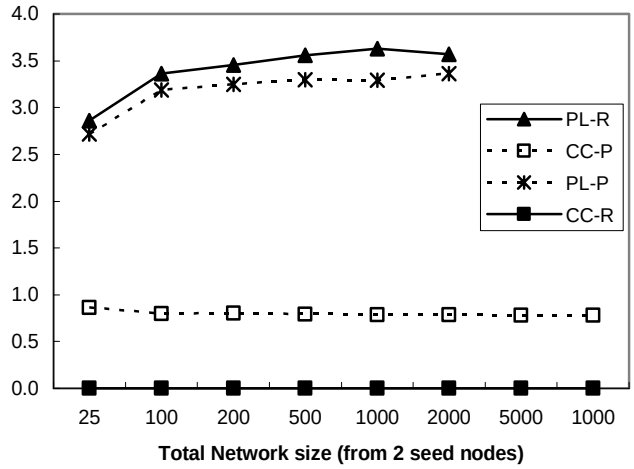
where,

M_A = {set of neighbors of A}, $|M_A| = k_A$

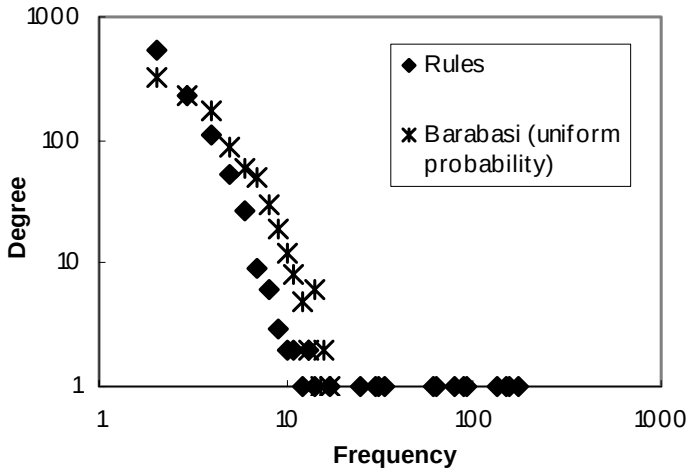
$M'_A \subseteq M_A$ such that $|M'_A| = d$ (globally defined)

$i \in M'_A$ with probability $\wp_i \propto k_i$ *iff* $i \in M_A$

Fig. 5 (a) shows the clustering coefficient (*CC*) and the characteristic path length (*PL*) for various network sizes generated with an initial seed of 2 nodes. The suffixes *P* and *R* indicate peer-to-peer and random networks respectively. Fig. 5 (b) shows the power-law distribution in a frequency (logarithmic X-axis) Vs degree (logarithmic Y-axis) plot of our rules mechanism compared with the well-known "Barabasi" technique. The network size was 1000 nodes grown from 2 seed nodes.



a. Small-world properties of resulting peer-to-peer network



b. Scale-free property of resulting peer-to-peer network

Figure 5. Properties of the resulting peer-to-peer network created by enforcing rules

3.3 Summary

Formation and discovery of peer communities are significantly dependent on how peers declare and use their common "interests". This chapter begins by defining attributes as a method of declaring interests, and then it describes a bidirectional, end-to-

end communication channel called peer links. More importantly, this chapter provides a set of rules for peers to join a peer-to-peer network such that the network will always exhibit certain properties like small world behavior and power-law distribution. We use these rules to build a peer-to-peer network simulator so that our simulations mirror real peers as closely as possible.

CHAPTER 4

FORMATION AND DISCOVERY OF PEER COMMUNITIES

4.1 Motivation

Psychologists have long shown that people have an affiliation motive (Murray, 1938) and a need for information about the world around us (Festinger, 1950). These are some of the instincts that have driven the formation of human groups. This tradition of group forming is not alien to computer science and we witness examples of it in Usenet groups, web communities (Flake, Lawrence and Giles, 2000), Yahoo Groups, chat rooms and so on.

Almost all present-day groups require some central control or a central authority through which advertisements can be made. In contrast to this, peer-to-peer systems are defined to be completely de-centralized and can also be dynamic. The popularity of schemes to form communities and associations on the web leads to the use of the peer-to-peer system structure for realization of underlying community structures. Thus peer-to-peer communities are not only a natural extension for arranging distributed systems, but also to enhance the capabilities of each member.

To enable regular searching as well as community-based searching, we must be able to efficiently discover communities, that is, we need searches for community members. This is somewhat complicated by the fact that communities are implicit, self-organizing, dynamic and constantly changing—forming, or breaking down due to changes in the peers.

This chapter is targeted to discovery of communities on the fly to enable efficient intra-community information dissemination as well as inter-community searching. To this end we have studied the community formation characteristics using simulations of large communities and found characteristics that can be exploited to perform discovery and searching. We show that this discovery does not require extensive computation or communication on the part of a peer.

Note that searching and information dissemination is not the focus of this chapter, they are the next step after community discovery algorithms are implemented.

4.2 Challenges for Formation and Discovery

In traditional centralized approaches, a server would be charged with maintaining a record of the interest attributes of each of its clients. New clients would merely register their interests at the server, which would periodically execute a matching algorithm to group peers that share common interests to form communities. This list of communities could be easily obtained from the server at any time. In this way, communities would be formed and clients would know their community membership in deterministic time. Existing examples of such systems are matchmaking servers and dating websites. As a bonus of a centralized approach, clients could also ask the server for a list of all the available communities without incurring any extra computational cost for server-side processing. Alternatively, a server would act as a central directory that lists all the available communities. New clients would then have to browse or search the directory for communities to join; or they could form a new community. A few other examples of these systems can be found in Internet chat sites (Yahoo Chat; MSN Chat; and ICQ).

In a distributed environment, such as a peer-to-peer network that lacks a central authority, this simple formation and discovery problem becomes much more complex. In order to compensate for the server, the cost of communication will be higher. Peers will need to exchange their interest attributes with each other for the purpose of forming communities and deriving their community membership. In addition, it is unlikely that any bonus information that was previously available for free due to the centralized approach will be similarly obtained without executing another round of communication.

We propose a community formation and discovery algorithm that works without any central authority and optimizes the cost of communication. Our algorithms are autonomous procedures that are executed asynchronously by each peer. Through a simple exchange of interest attributes, we demonstrate how communities can be formed. Further, without any additional communication, we show how peers can also discover exactly to which communities they belong. The algorithm for discovering communities in which a peer is not a member is left for chapter 6.

4.3 Forming Communities

The community formation algorithm assists in the establishment of community relationships amongst peers that share common interest attributes.

Normally, a network of peers can exchange or advertise their interest attributes with their neighbors in order to find peers with whom community relationships can be formed. There are two mechanisms here: an *active* approach and a *passive* approach. In the active approach, a peer actively polls peers in its neighborhood[2] for their claimed

[2] The neighborhood of a peer includes 1-hop and 2-hop neighbors.

interest attributes (*receive phase*) while simultaneously sending its own claimed attributes to them (*send phase*). On the other hand, if a passive approach is adopted, a peer merely advertises its claimed interest attributes on its website and relies on the active approaches of other peers for the exchange. The number of messages required during both send and receive phases for the active approach is $O(n^2)$, while for the passive approach, the number of messages during the send phase is $\emptyset$ and during the receive phase it is $\leq O(n^2)$. We use both these approaches in our community formation algorithm. The active approach occurs as an initial, periodically (daily, or weekly, etc.) repeatable exchange; and the passive approach exists continually between active exchanges. Over time the communities formed will become out dated as peers change their interest attributes. Therefore, we propose that the formation algorithm be repeated periodically. In between these repetitions, we propose that a peer advertises its interest attributes using its website, shared network drive, or a web service. Due to the asynchronous nature of our algorithm, in a large network of peers there will always be some peers in an active mode, at any given time, which can take advantage of these advertisements.

The active exchange of interest attributes is not required in order to form communities. In fact, if all peers only operated in passive mode and tried to form communities based on their claimed attributes, we could significantly reduce the cost of communication. However, it is possible that a peer-to-peer community with signature S might exist. As a result, a peer that has $S \subseteq$ personal attribute set and $S \not\subset$ claimed attribute set, will not be able to join this community and avail of the benefits until it claims all the attributes of set S.

Peers, therefore, need to expose (escalate from the personal list to the claimed list) as many attributes as possible to join the maximum number communities. This process is called *attribute escalation*, and can only be achieved by establishing active communication with other peers to obtain their claimed attribute sets. When no more attributes are being "escalated" by the peers, we say that the communication has achieved its goal and the collection of peers is *stable*.

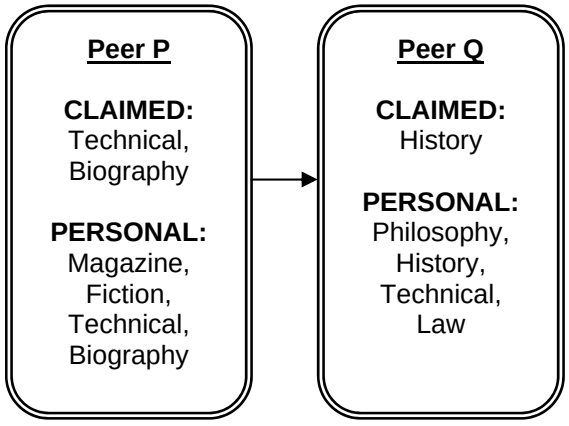
Two peers that are linked (shown by arrow) are illustrated in Fig. 6. Note the change in the claimed attributes of Peer *Q* it received claimed attributes from Peer *P*.

4.3.1 Attribute Escalation Algorithm

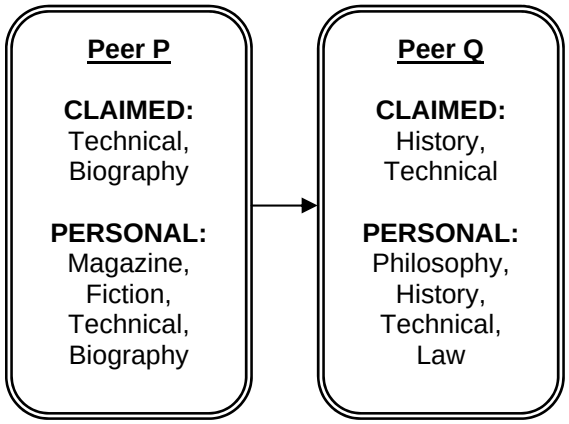
Provided below is an algorithm for attribute escalations:

1. Each peer *P* communicates with all of peers in its neighborhood. The communication involves a transmission of the claimed attribute set by *P*.
2. Each peer *Q* receiving the claimed attribute set from its links will find the intersection of the set with its own personal attribute set.
3. If an element is present in the intersection set and absent from the claimed attribute set of *Q*, then it is added to the claimed attribute set of *Q*.

The reason that transmission occurs only within the neighborhood is explained in section 4.4.3. Further, our simulations have shown that by using only one level of indirection, peers could participate in an average of eight communities. Probing to two levels of indirection is unnecessary, as it did not increase the average number of communities in which a peer could participate.



a. Before any communication and adjustment



b. After communication messages and adjustment

Figure 6. Procedure to escalate attributes ("Technical" in this case)

Note that the attribute escalation algorithm automatically escalates attributes from the personal set to the claimed set. This process may not be desired in practice and a human may be consulted before this escalation is performed. For this dissertation, automatic escalation is assumed.

The above procedure makes a change to each peer and therefore changes the communities identified. Since these changes might occur frequently with peers being

created, destroyed, or modified, we were concerned about the stability of the attribute escalation technique.

4.3.2 Experimental Evaluation

We conducted simulations to measure how many communication steps our algorithm requires to identify all the possible communities. In this simulation, we generated a peer-to-peer network using our generation method described in the previous section. The network consisted of 10,000 peers. Each peer was randomly assigned a set of personal attributes of random size but with at least one element. To simplify the process, the attributes were implemented as different letters of the alphabet. Thus we had 26 possible attributes (A-Z). Next, from the personal attribute set of each, we randomly selected some attributes and made these the claimed attributes of that node.

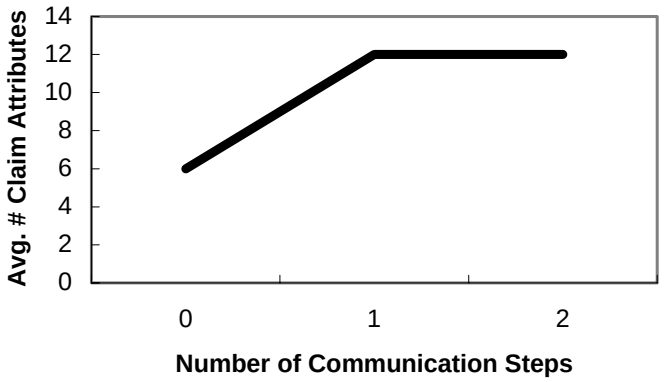


Figure 7. Stabilization of Attribute Escalations

On the average, we found each simulated peer contained 13 personal attributes and 6 claimed attributes; and was connected to 66 peers. We found that the average number of claimed attributes per peer stabilized *after one iteration* of the attribute

escalation algorithm (see Fig 7). This means that the number of times peers need to communicate before the average number of claimed attributes for all peers becomes constant is just once. Further, the computation at each peer is a trivial set algebra. Hence, each new arrangement of a collection of peers is expected to stabilize quickly and with little overhead.

4.4 Discovery of Community Membership

Discovery in the context of peer-to-peer communities usually indicates one of the following: discovery of the communities in which the peer is automatically a member, or discovery of remote communities that a peer might want to join or use. As indicated at the beginning of this chapter, the focus is on the former.

After the escalation of attributes, a peer has only facilitated the formation of communities. It still is not aware of its community memberships. At this stage, a peer knows two things about its interest attributes: the list of its claimed attributes, and the subset of the claimed attributes that were escalated. A peer can assume to be a member of communities whose signatures are equal to the attributes that have been recently escalated. This assumption makes sense because the only reason that the attributes were escalated was due to communication with a known peer (directly or indirectly known) that shares the same interest. However, the attributes of a peer that had already been claimed before the escalation process might or might not have been common with other known peers. Therefore, the peer cannot assume to be a member of a community that is based on these un-escalated claimed attributes. After analyzing the received claimed interest attribute sets, a peer will be able to discover two kinds of communities:

1. The communities that peers are a part of by virtue of their common claimed attributes
2. The communities in which peers become members by virtue of the claimed attributes that were escalated

The analysis of the claimed attribute sets received allows a peer to compute its Link Weights (*see section 3.1.3 for definition*). As stated earlier, link weights help determine the membership of a peer in a community.

Definition 4

Peer-to-Peer Community Membership

A node i, with claimed attribute set C_i, is a member of a peer-to-peer community, N, with signature set S if every claimed attribute of the intersection set $I = C_i \cap S$ has a link weight that is greater than the globally predetermined threshold T.

By including threshold *T* in the definition of membership, we can have various degrees of peer-to-peer communities. For instance, if the threshold is kept very low, then a community will be formed even with a small number of peers that share a common interest. If instead, it is desired that peer communities only form when there are many peers that share a common interest, the threshold can be set to a higher value. Although not required to be pre-set, the threshold value would give the network of peers a uniform behavior if all peers agreed upon its value.

4.4.1 Community Discovery Algorithm

The procedural pseudo-code for our membership discovery is shown in Table 1.

Table 1

Community Discovery Algorithm

Line	Pseudo-Code
1.	`MEMBERSHIP-DISCOVERY (peer)`
2.	`/* executes at each peer simultaneously */`
3.	`  foreach claimed attributes`
4.	`    compute link weight`
5.	`  end-for`
6.	`  foreach subset S of claimed attributes set`
7.	`    foreach attribute in S`
8.	`      if link weight > T then`
9.	`        check next attribute`
10.	`      else`
11.	`        break-for`
12.	`        /* try another subset */`
13.	`    end-for`
14.	`    if no break-for was executed`
15.	`      peer is member of community with signature S`
16.	`  end-for`

4.4.2 Experimental Evaluation

We measure the effectiveness of our algorithm by the difference in the number of peer-to-peer communities, in which a peer discovers membership.

We ran simulations on peer networks of two different sizes: one with 1,000 peers: and the other with 10,000 peers. The maximum number of personal attributes we allowed for each peer was 20, and a peer was explicitly placed into only one group, as described earlier. For both simulations, we set the link threshold to 40%.

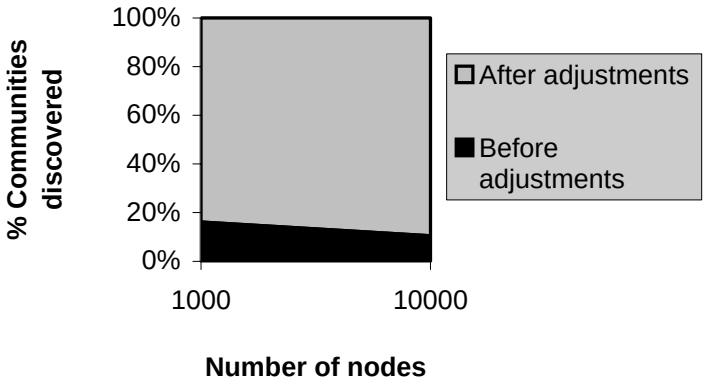


Figure 8. Percentage of communities discovered (average over all peers)

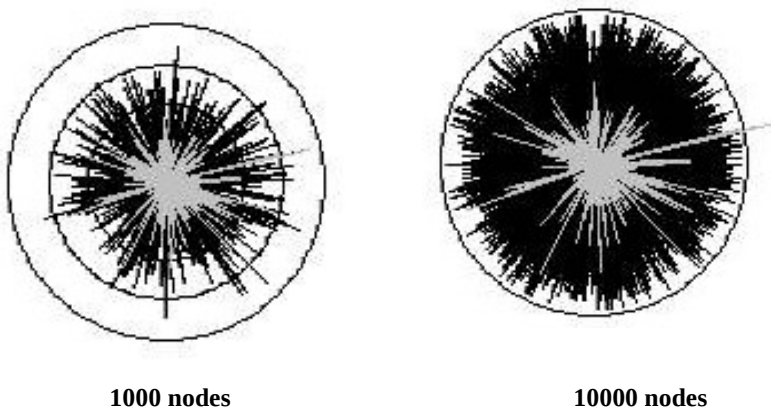


Figure 9. Number of communities discovered by each peer

The graphs shown in Fig. 8 and 9 are the results of our simulations. They demonstrate the performance of using the claimed attribute set and the link threshold for

the discovery of peer-to-peer communities. The graph in Fig. 8 shows the average number of communities discovered by all peers before and after executing the Community Discovery algorithm. Fig. 8 illustrates that the majority of peers have discovered communities, other than the ones, in which they were explicitly placed. The graphs in Fig. 9 show the number of communities discovered by the peers. The inner gray area is *before* and the outer black area is *after* the Community Discovery algorithm. The circles have values 0, 5, 10, and 15 from inside to outside. Our simulations have shown that before any communication, the average number of communities known to a peer was 0.5. This number increases to an average of 8.5 after the execution of the membership procedure at each peer.

4.4.3 Average Number of Peers Reachable from a Peer

We conducted experiments with 1,000 peers and 10,000 peers to traverse the outgoing links from each peer. We then counted the number of peers that could be reached after going to the direct neighbors (Level-1), in-direct neighbors after one indirection (Level-2), and in-direct neighbors after two indirections (Level-3). Table 2 provides the values obtained by our experiment.

Table 2

Number of peers reached in various levels (For 10,000 peers)

Max. Level traversed	Number peers reached
Level-1	65.8141
Level-2	4380.7874
Level-3	18929389.64

The graph in Fig. 9 was plotted using a logarithmic Y-axis. It shows how the number of peers that can be reached increases with greater depth.

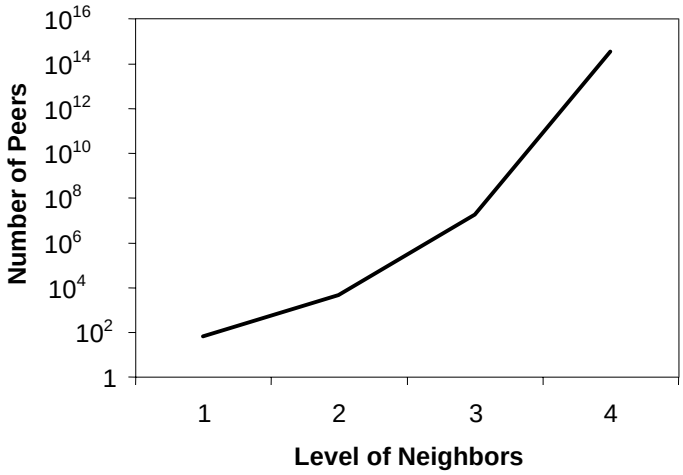


Figure 10. An exponential increase in the number of peers that can be reached

4.5 Factors Determining the Existence of Communities

Given a set V of peer nodes, each having a set of personal interest attributes as a subset of the universal set of all possible interest attributes I, can we say with some certainty that the nodes will form into peer-to-peer communities after the attribute escalation process? Note that set I depends on the particular context of the peer-to-peer system. For instance in the peer-to-peer digital library, I could be the set of all possible genres of books, in another context, I could be the set of all known research areas within Computer Science.

Consider a peer $V_i \in V$, where $V = \{V_1, V_2, \ldots, V_N\}, i \in \{1, 2, \ldots, N\}$ is a set of peer nodes. The set of personal interest attributes of V_i is $P_i \subseteq I$, where

$I = \{\text{all interest attributes}\}$, and the set of claimed interest attributes is $C_i \subseteq P_i$, and $P_i' = P_i - C_i$ is the set of unclaimed attributes.

During the attribute escalation process, every peer sends its claimed attribute set to all of its 1-hop and 2-hop neighbors.

Lets say that $m \leq N$ is the number of 1-hop and 2-hop neighbors of V_i, then eq. (1) gives the set of all claimed attributes received at V_i after the attribute escalation.

$$\hat{C}_i = \bigcup_{j \neq i,\, j=0}^{m} C_j \,,\ 0 \leq j \leq N \qquad \textbf{(1)}$$

In addition, eq. (2.2) will give V_i the set of interest attributes that it shares with other peers.

$$\Psi_i = \left(\hat{C}_i \cap C_i\right) \cup \left(\hat{C}_i \cap P_i'\right) \qquad \textbf{(2)}$$

$$\therefore \quad \Psi_i = \hat{C}_i \cap \left(C_i \cup P_i'\right) \qquad \textbf{(2.1)}$$

$$\therefore \quad \Psi_i = \hat{C}_i \cap P_i \qquad \textbf{(2.2)}$$

Since we define communities as groups of peers that have common interests, eq. (2.2) also indicates the set of communities in which a peer is a member. For the sake of simplicity we assume that every common interest attribute indicates a separate community.

Therefore the factors that determine whether a particular peer is a member of a community are:

1. The number of 1-hop and 2-hop neighbors (m)
2. The number of interest attributes available ($|I|$)

3. The number of interest attributes in the personal set ($|P|$)

4. The number of interest attributes in the claimed set ($|C|$)

Let us analyze the various factors in detail.

The first factor relates to the links in a peer-to-peer network. A peer having more 1-hop and 2-hop neighbors is less likely to obtain an empty $\hat{C}$ after escalations. The worse case scenario occurs when all *m* neighboring peers have empty claimed attribute sets.

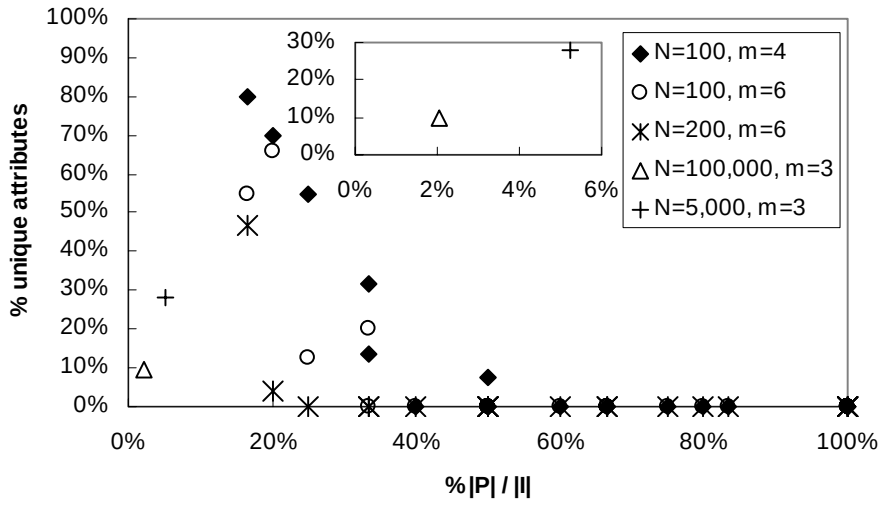


Figure 11. Percentage of interest attributes not common with any peer as a factor of |P| / |I|

The probability that the claimed attribute set of a peer is empty depends on set *P*. The worse case scenario (high probability) can occur when a peer declares no personal interest attributes; therefore the claimed attribute set will also be empty. In addition, when set *I* is very large, and the personal attribute sets of the peers in the network are

very small, Ψ is more likely to be $\emptyset$. It would be interesting to study the relation between the percentages of unique attributes (not contributing to community formation) vs. the ratio of $|P|$ to $|I|$. Fig. 11 shows this relation.

In smaller peer-to-peer networks ($N \leq 200$), we observe that more than 50% of personal interest attributes remain unique, i.e. they do not contribute to community formation, when the ratio of $|P|$ to $|I|$ is below 40% and m=4. Increasing the number of neighbors (m=6) and the number of nodes (N=200) in the network reduces the percentage of unique attributes to 4% even when the ratio of $|P|$ to $|I|$ is lowered to 20%.

However, in larger peer-to-peer networks ($N \leq 100{,}000$), less than 30% of personal interest attributes remain unique even with a $|P|$ to $|I|$ ratio below 6% and m=3. Therefore, with a higher number of nodes in the network, peers will share more of their personal interest attributes with some other peers even though each peer's list of personal interests is only a small fraction of the total number of interest attributes available.

4.6 Summary

This chapter showed that the attribute based clustering of peers can be made to work, by defining an overlay network, consisting of peer links. It demonstrated that the peer formation algorithm stabilizes *after just one iteration* of the escalation technique. Additionally, it introduced an efficient community discovery procedure using weights and threshold. Our simulations of the discovery procedure have confirmed that peers can quickly discover numerous memberships in different peer-to-peer communities using very little computation and communication messages.

CHAPTER 5

INFORMATION DISSEMINATION USING PEER COMMUNITIES

5.1 Motivation

Peer-to-peer communities aid in the better dissemination of useful information amongst peers. For example, suppose node X belongs to a person interested in Amazonian Biological Catapults (ABC). After X declares this interest, it becomes a member of the community of ABC enthusiasts. Henceforth, all information X wants to share can be placed in a public directory and will be readable/searchable by all members of ABC. This concept can be extended to discover resources, physical devices, and network components. It also has some interesting security and access control issues (including trust management). The above example is interesting in the context of peer communities with no overlapping interests and is especially applicable in applications that follow the publish-subscribe-notify model (Gummadi and Hohlt, 2000).

After we run our algorithms detailed in the earlier chapter that uses interest attributes for assisting in the formation of communities, peers will discover the communities that they participate in. However, since interest attributes are constantly changing values, the formation of communities needs to occur on a regular basis to keep the peer-to-peer system up-to-date and to keep the peers subscribed to the most suitable, existing communities. Then, again, a periodic increase in communication messages might not be suitable for low bandwidth networks, as regular communication will be affected by this increased traffic. Therefore, instead, we opt for *Distributed Discovery* followed by push-pull *Peer-to-Peer Gossiping*.

5.2 Challenges for Information Dissemination

Prior approaches for information dissemination within a peer-to-peer network include flooding or hop-limited broadcast. Lamport and Chandy's distributed snapshot algorithm (1985) is not relevant to solve this problem because of two reasons:

1. The Lamport-Chandy algorithm makes an assumption of FIFO channels. This might not be true in our peer-to-peer network due to the use of overlay links.
2. Termination detection of the Lamport-Chandy algorithm requires knowledge of the total number of nodes in the distributed system, and these nodes have to remain alive throughout the execution of the algorithm. This is not practical in a dynamic peer-to-peer network where peers can appear and disappear at random.

Unlike these approaches, our technique begins with a discovery phase to gather data on peers that would be interested in receiving certain information. During the discovery phase certain properties of the peer-to-peer community are discovered. These properties include the approximate number of members and the information about the member peers. Since peers can go offline at anytime, the discovery phase cannot depend on the peers of a community to be online.

For undirected intra-community information dissemination, we propose Peer-to-Peer Gossiping. This is a push-pull approach that is resilient to peer failures and does not critically depend on any single peer or message. It involves communication (gossiping) amongst selected peers of a community (called *seers*) to achieve an objective that is similar to the case of rumor spreading (Karp et al., 2000). The selection of seers is based on the discovered data. As long as the discovered data is available and recent, the push

phase can be repeated numerous times with new information. Whenever required, a peer can retrieve the information from a nearby seer via a pull phase. The two major differences between our method and rumor spreading are:

1. The lack of prior knowledge about the number of peers that exist
2. The lack of random selection of peers

In this chapter, our technique for peer-to-peer gossiping is described. It shows that our Distributed Discovery is a low overhead, simple protocol that identifies seers, and is easy to terminate. Our proposed algorithm for the push phase makes gossip information available to a large percentage of interested peers within a very short number of links. In addition, these two techniques facilitate the management of quickly changing community structures via undirected intra-community communication. Both discovery and gossiping techniques are efficient, de-centralized, robust and highly scalable.

5.3 Distributed Discovery of Seers

The primary goal of this algorithm is to gather information about the peer members of a community in order to select the peers that will become seers. As an intentional offshoot, this algorithm also gathers information about the community, such as the approximate number of online members, and the union of all the sets of claimed attributes that belong to the peer members.

5.3.1 Peer Involvement and Seers

Peers have involvement values associated with every community in which it is a member. Involvement is proportional to the number of peers in the neighborhood that claim the signature attributes of a particular community. Therefore peers with a higher

value of involvement associated with a community *C* have more peers within their neighborhood that are also members of *C*. We use the term *seers* when referring to these peers. Seers of a community are known directly or indirectly to most peer members of that community. This also means that information stored at the seers will be available to a large percentage of peers within the community

Definition 5

Peer Involvement

The average of link weights for elements of the intersection set $I = C_i \cap S$ *is directly proportional to the involvement of node i which has the claimed attribute set* C_i *in a peer-to-peer community with signature S.*

Vector ID	Peer ID	Peer Involvement	...	...	...

Figure 12. Distributed Discovery vector format

5.3.2 Unbound Distributed Discovery

We employ a vector (Fig. 12), which is sent to every peer *j* in its neighborhood by the peer initiator (I_S) of community *S*; such that $S \subseteq$ claimed attribute set of *j*. Any peer can be the initiating peer. To guarantee that every community has at least one initiator, we programmed every peer to attempt to become the initiator of its communities. A globally unique vector ID can alert peers to drop vectors with higher IDs and inform the losing initiator, thereby ensuring that only the vector with the lowest ID

survives. This also means that in the end, all initiators, except one, will lose their status as the initiators of the algorithm.

Table 3

Algorithm for Unbound Distributed Discovery (at Initiator)

Line	Pseudo-Code
1.	`DISTRIBUTED-DISCOVERY-AT-INITIATOR`
2.	`  /* executes at the initiator */`
3.	`  Create vector 'v' for community 'P'`
4.	`  Insert my information`
5.	`  Send 'v' to direct neighbors claiming 'P'`
6.	`  Send 'v' to 2nd-degree neighbors claiming 'P'`
7.	`  /* Wait till frequency peaks and then drops */`
8.	`  Wait to receive end messages or loss of status message`

Eventually, the collective traversals by the copies of the surviving vector will ensure that no peer is visited twice, thereby discovering information from all members of the community *S* in deterministic time. Every peer that receives the vector appends its information to the vector and sends copies of the vector to its neighbors by using the same criteria as the initiating peer. With a community size of *n* peers, a total of *O(n)* vectors travel through the peer-to-peer network. The peers that receive the vector and have no neighbors that have yet to receive the vector construct an end message with the vector and send it to the initiating peer, whose identity can be obtained from the first element of the vector. Table 3 lists the algorithm for Unbound Distributed Discovery executed at the initiator and Table 4 lists the algorithm executed at receiving peers.

Table 4

Algorithm for Unbound Distributed Discovery (at Receiving Peer)

Line	Pseudo-Code
1.	DISTRIBUTED-DISCOVERY-AT-RECV-PEER
2.	/* executes at each receiving peer */
3.	**Receive** vector ‘v’
4.	**If** I have already received ‘v’
5.	**Send** NACK to sender of the ‘v’
6.	**End-prog**
7.	**Else**
8.	**Send** ACK to sender of the ‘v’
9.	**End-if**
10.	**Insert** my information
11.	**List** neighbors (up to 2-levels) claiming ‘P’
12.	**Remove** sender of ‘v’ from list
13.	**If** list has peer identities
14.	**Foreach** peer in list
15.	**Send** ‘v’ to peer
16.	**Receive** acknowledgement from peer
17.	**End-for**
18.	**Else**
19.	/* This means I am at the end */
20.	**Create** end message with ‘v’
21.	**Send** to Initiator
22.	**End-prog**

```
23.       End-if
24.       If NACK received from all peers
25.         /* This means I am at the end */
26.         Create end message with 'v'
27.         Send to Initiator
28.       End-if
```

The initiating peer waits a certain amount of time to receive end messages. The union of all end messages provides information about the members of community S to the initiator peer I_S. Since all peers know the identity of the I_S, they can obtain this information in a deterministic time if needed. This leaves one question unanswered: *how can the waiting period for I_S be set?*

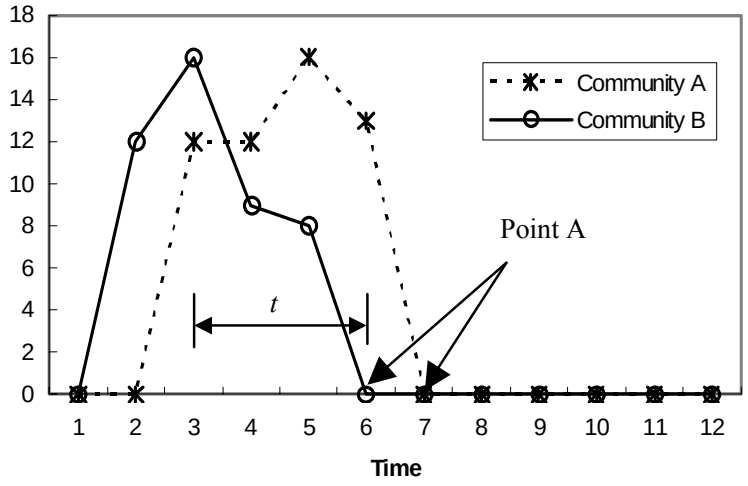


Figure 13. Frequency of end-message arrival at the initiator

In our experiments, we found that the initiating peer received end messages with the frequency graph shown in Fig. 13. The graph shows the number of end messages that the initiators of two separate communities received. The network size was 1000 nodes.

20% of random peers were made to fail. Therefore, by making the initiating peer wait *t* cycles after the frequency drops to zero (Point A), most of the end messages can be collected. Here, *t* is calculated as the time it takes for the frequency to reach its peak from zero. If more messages arrive during this wait time, Point A is reset and the value *t* is recalculated for the new peak frequency.

5.3.3 Hop-bound Distributed Discovery

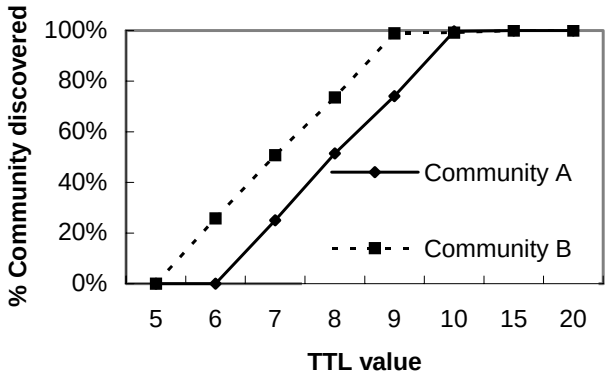


Figure 14. Percentage of the community discovered as the value of hop count (TTL value) is increased

For communities that are extremely large, the Distributed Discovery algorithm will require a long time[3] to conclude. Therefore the initiator will have to remain online for a long time to receive incoming vectors. To overcome this drawback, we propose an alternative hop-bound Distributed Discovery that works by sending a maximum hop count (*h*) value, along with the vector, so that a sub-set of the community can be discovered. The hop-count h can be set based on the initiator's ability to wait. At a later

[3] At most $O(\log_m n)$ for a community with *n* peers each having an average of *m* neighbors.

stage, a merging algorithm can be executed to combine various sub-sets into one community. Fig. 14 shows the increasing percentage of the community discovered, as the value of hop count is increased. Note the linear behavior of the hop-bound distributed discovery. The test was conducted on a network with 4,500 nodes.

The merging algorithm can be executed as a low priority activity. It is not essential to the operation of algorithms, such as community-based search. That being said, the merging algorithm helps structure the peer-to-peer network so that the search algorithm works more efficiently. Following is the construct of the merging algorithm:

Case 1: There exists more than one initiator within *h* hops.

If the initiators have neighbors within or beyond *h* hops, then by virtue of the Distributed Discovery algorithm, the vector with the lower identification number survives. Therefore only one peer will remain as initiator until termination. The ousted initiator will know the identity of the extant initiator. All the results that are sent to the ousted initiator are forwarded to the extant initiator of the community.

Case 2: There exists more than one initiator beyond *h* hops.

If the end vectors received by the initiator indicate that the hop count had been reached, then there is a possibility that there are potential community members beyond *h* who might have been involved in their own hop-based discovery. The initiator therefore sends a message to the peers that sent the end vectors requesting them to obtain the identity of the initiator from their neighbors that were not involved in the current discovery. The initiator with the lowest peer identification value takes over as the new initiator of the merged community. However if no such initiator was found, then this

operation of locating an initiator beyond *h* will be repeated periodically so that eventually a merge operation takes place.

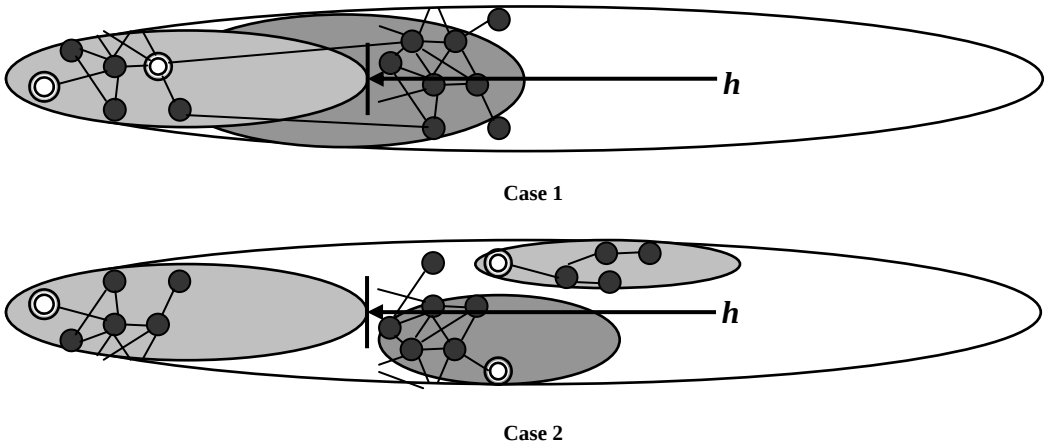
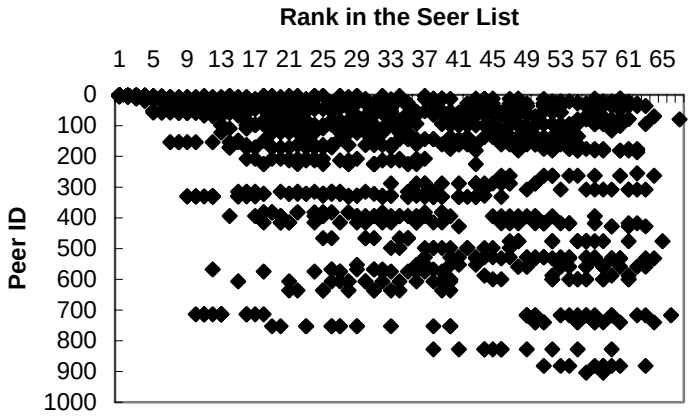


Figure 15. Two cases that might occur during a hop-bound distributed discovery

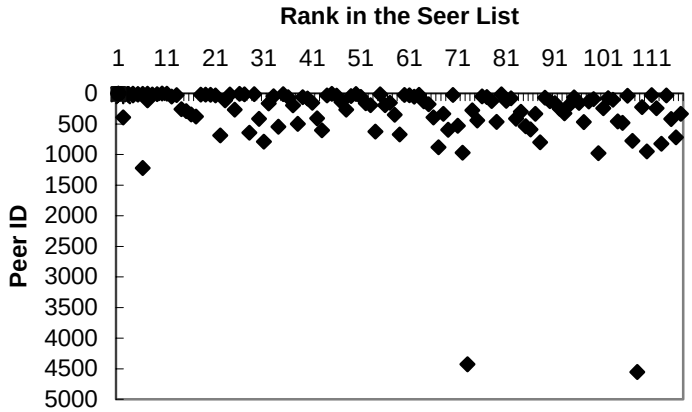
Fig. 15 portrays the above two cases. The large ellipse that encloses everything indicates the actual size of a community ($h=\infty$). The smaller ellipses within indicate sub-sets of the community ($h \in N$, where N is the set of natural numbers) that are discovered. Peers (dark gray circles) and their links (connecting edges) are shown in the figure. Initiating peers are marked with concentric circles.

5.3.4 Distribution of "Seer" Status amongst Peers

In our experiments we found that older peers, i.e. peers that have been part of the network for a longer time, are more likely to become seers of their communities. We attribute this to the fact that older peers have had more time to accumulate more links. Furthermore, when older peers are picked as seers by the initiator of the Distributed Discovery, their Link Weights are amongst the highest when compared to the Link Weights of the other seers.



(a) 1,000 node network



(b) 5,000 node network

Figure 16. Distribution of "Seer" status amongst peers

Fig. 16 plots the peers that have been selected as seers due to their high Link Weights. Fig. 16 (a) is from experiments on a 1,000 node network and Fig. 16 (b) is from experiments on a 5,000 node network. The X-axis is the Rank in the Seer List. Seers with higher Link Weights are closer to the top of the list and have lower rank values. The Y-axis lists the peer id from 0 to *N*-1. Older peers have lower peer id values. In both the

graphs, note the clustering of points to the top left corner. This is because older peers are often selected as seers; and in the list of seers of a community, older peers usually have the highest Link Weights.

5.3.5 Obtaining Bloom Filter Summaries

Bloom filters (Bloom, 1970) are compact data structures that probabilistically represent the elements of a set. They support queries of the form, "*Is X an element of the represented set?*" and have been used in a variety of ways (Gribble et al., 2000; Gribble et al., 2001; Hodes et al., 1999; and Kubiatowicz et al., 2000).

We extend our proposed Distributed Discovery protocol further by gathering Bloom filter summaries from each participating peer. Each peer that receives the vector creates its own Bloom Filter (see Table 5 for algorithm) and merges it using a simple bitwise-OR into the existing filter (see Table 6 for algorithm). After the end messages are received, the initiator merges all the filters, forming, a Bloom Filter that represents a compact summary of the attributes claimed by the peer members of a particular community.

In order to reduce the rate of false positives that result due to the probabilistic nature of these data structures, we chose k=8 hash functions, and set the Bloom Filter size to m=2048 bits. Based on formula (3) (described in Fan et al., 1998, and Ripeanu and Iamnitchi, 2001), we get $p_{err} \approx 1.E^{-05}$ for n=70 possible claimed attributes.

$$p_{err} \approx \left(1 - e^{-\frac{kn}{m}}\right)^k \quad \text{(false positives)} \qquad (3)$$

The reasons for this modification are justified in chapter 6 where we employ the Bloom Filter for a Community-Based Search algorithm.

Table 5

Algorithm for Constructing a Bloom Filter

Line	Pseudo-Code
1.	`CONSTRUCT-BLOOM-FILTER`
2.	`  /* executes at each peer */`
3.	`  bloomFilter = new bool[m] initialized to false`
4.	`  foreach attribute in Claimed Attribute List`
5.	`    foreach hashFunction` h_i
6.	`      bloomFilter[`h_i`(attribute)] = true`
7.	`    end-for`
8.	`  end-for`

Table 6

Algorithm for Merging Bloom Filters

Line	Pseudo-Code
1.	`MERGE-BLOOM-FILTERS`
2.	`  /* executes at each peer on arrival of new vector */`
3.	`  mergedFilter = new bool[m] initialized to false`
4.	`  foreach bFilter in List of Bloom Filters to merge`
5.	`    foreach bElement in bFilter`
6.	`      i = bloomFilter.IndexOf(boolElement)`

```
7.            mergedFilter[i] = mergedFilter[i] bit-OR bElement
8.          end-for
9.        end-for
```

5.4 Push-Pull Gossiping

As indicated earlier, our algorithm utilizes the seers within a community to carry the information or updates, so that it will be available to most peer members.

At the end of the Distributed Discovery algorithm, the initiator knows the involvement values of each peer member of the community. The initiator then creates a set of seers by selecting the peers that have the highest involvement values. By the definition of involvement, the peers in this set have more community members in their neighborhood when compared to other neighborhoods of peers that are not in the set.

We show that by correctly choosing the size of the seer set, the majority of the community members will lie within the neighborhood of at least one peer from the set. Therefore any information that is sent to the initiator can be multicast to the members of the seers set so that it can be available (if required) to the majority of community members, within their neighborhoods.

We conducted experiments on three different peer-to-peer networks to determine the behavior of our push algorithm. The initiator selected X% of the peer members with the highest involvement values as seers. Then it sent some information to these seers (push) with the hope that this information will be available to Y% of the remaining community members within 2 hops. In Fig. 17, the relationship between X and Y is shown for three different network sizes (1000, 5000, and 10000 nodes). The values

provide the average of two tests conducted on all the communities within those networks.

On the average, we found that pushing information out to only 5%-10% of specially selected community members (seers) makes that information accessible (via a pull operation) to approximately 80% (or more) of the remaining members within their own neighborhoods.

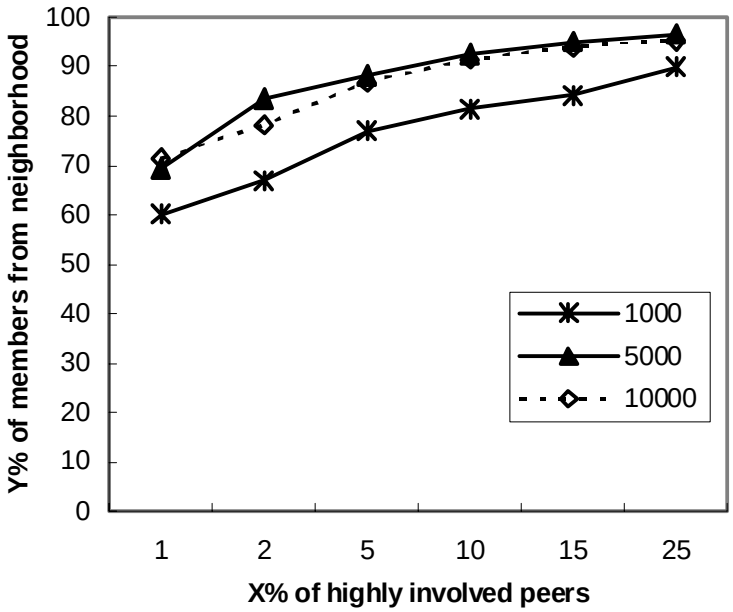


Figure 17. Performance of our push-pull gossiping technique

5.5 Summary

This chapter presented a novel form of undirected intra-community communication using two phases, push and pull, that are preceded by a Distributed Discovery operation. Combined with the distributed discovery of seers, both the push and pull phases help with information dissemination within the community. The

Distributed Discovery algorithm involves gathering information on peer members of a community. A dynamic scheme was provided to determine a termination point for this algorithm.

With the information gleaned from the Distributed Discovery, peers can execute our gossiping protocol using the push-pull phases. We ran experiments to show that pushing the information to only a small number of specially chosen peers allowed a large percentage of peer members of a community to obtain (pull) the gossip information from within their neighborhoods.

CHAPTER 6

INFORMATION SEARCH USING PEER COMMUNITIES

6.1 Motivation

By far, one of the biggest challenges of peer-to-peer systems is the ability to locate information like files or resources. Centralized searching (as used by Internet search engines such as Google) has the downside that the central authority controls the indexing and presentation of the information. Peer-to-peer searching allows anyone to publish information. Searching involves cooperatively passing a query message until a peer that published the desired information is found. As a result, peer-to-peer searching reacts to dynamic changes much more quickly than centralized searching. This chapter briefly describes the mechanics of our search technique and provides some comparisons with known search algorithms.

If the Computer Science (*CS*) and Medical (*M*) communities were disjoint, then the search operations for medical information performed by a node that belonged to the community *CS* would not produce any results. However, if the communities were linked at some point (see Fig. 18), lets say *Q* (*Q* belongs to both communities), then the medical information would be found, but at a great search expense, since, on the average, half of the community *CS* would be searched before a node from the community *M* is found.

To mitigate such problems, we need a community based query propagation method. Thus, to provide efficient searching, it is better to search for one or more target communities, irrespective of the current membership of the searching node.

In Fig. 18, the vertices represent peers and edges represent end-to-end connections between peers. The closely connected collection of peers to the left and the similar but smaller collection to the right are two separate communities that are linked by a common peer.

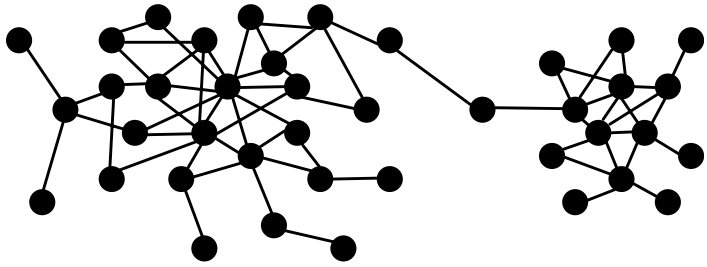


Figure 18. Example of peer communities linked by a common peer

6.2 Challenges for Information Search

Without a central server to index the content of peers, searching for information in peer-to-peer systems becomes a potentially costly operation. The lack of a central regulation also requires more innovative search techniques to tackle the scale and irregularity. The goal is to bring down the cost of searching in terms of number of hops/messages while still covering the largest possible number of peers in the system. This is somewhat complicated by the fact that communities are implicit, self-organizing, dynamic and constantly changing—forming, or breaking down due to changes in the peers. We also needed to compare the effectiveness of our search technique with other known peer-to-peer search algorithms. For this, we created a parameter that measured the quality of the results received in response to a search query.

6.3 Constructing the Search Query

Any peer that needs to search the peer-to-peer network constructs a three-part search query containing:

1. The identity of the peer creating the query
2. The actual query for an item
3. A list of meta-information that describe the item

In an interest-based peer-to-peer network, such as our digital library (Section 1.2), a peer might use interest attributes as meta-information to a query. For instance, if the query is for "books about Vampires," the list of meta-information might include attributes such as, "Twentieth century," "Bram Stoker," and "European authors."

A solution to a query could mean that the peer either owns the requested books or can provide information about the peer that owns the requested books.

6.4 Processing the Query

To facilitate the search operation, the querying peer P_Q sends the query to peer P_S, which is chosen due to its membership within the appropriate communities that are determined by the meta-information attributes of the query. Section 5.3.4 described how Bloom filter summaries of claimed attributes were obtained for a particular community. At this stage, the bloom filters are checked to determine whether a peer claims any attribute from the meta-information list of the query. False positives can occur thereby increasing the overhead of locating P_S. If P_Q matches this description, then it is chosen to process the query. Else, P_Q looks for P_S from amongst its immediate neighbors.

After the query is constructed, three approaches are possible for selecting P_S

1. P_S is the closest seer of a community whose signature S contains at least one attribute from the meta-information list
2. P_S is the seer in the community with S matching the maximum meta-information attributes
3. P_S is the seer of a community whose signature matches the most important attributes from a weighted meta-information list of attributes.

The requirements of the system will determine which of the approaches will be implemented. For the purpose of our experiments, we used the first and the second approach. The first approach has the least communication cost for locating P_S and the second approach has the highest communication cost. Therefore, peers in a system with more exacting search requirements could use either the second or the third approach.

If P_S is not located, P_Q asks its neighbors to provide the identity of P_S. In the peer-to-peer networks generated as described in the Section 3.2.2, we found that the latter case occurs about 26% of the time when 10% of the randomly selected peers created distinct queries. We also found that P_S is almost always located after asking the neighbor peers. The cost of locating P_S is amortized over a number of queries because peers remember the identities of the closest peers that are seers of a particular community.

The query is then sent from P_Q to P_S. Note that this operation is similar to the push phase of gossiping, except that P_Q (or one of its neighbors) selects P_S instead of the initiator of the distributed discovery. P_S stores the query on its *blackboard*. Blackboards are similar to web pages, and each peer has a blackboard. Like web pages, any peer can view the content on the blackboard of any other peer, provided that it knows the identity

of that peer so that the blackboard can be reached. To better organize the blackboard information, a peer can maintain separate blackboards for each community in which it is a member.

6.5 Checking Blackboards

Periodically and asynchronously, for each community *C* that it is a member of, a peer visits the blackboard for *C* maintained by each of its neighbors who are also members of *C*. Again, note the similarity with the gossiping algorithm (pull-phase). If the visiting peer can solve any of the queries on the blackboard, then a message is created and sent to the peer that created the query. The message created could be sent via email to the querying peer.

Regardless of the outcome of the above procedure, the visiting peer copies the queries and places them onto its own blackboard for community *C*. Our experiments showed that even such an asynchronous, background communication amongst peers provides quick and efficient solutions to queries.

Populated communities of asynchronously operating peers will quickly solve the queries (includes propagating it forward to peers that can better solve the query via a push to the appropriate P_S) because there will always be a fraction of peer members that will be involved in pull-operations by checking the seers' blackboards. A time-to-live value can be added to queries so that stale queries do not use up the resources of the network.

6.6 Simulation Results

In a digital library, participating peers could create two kinds of queries for books:

1. Queries containing the title of the book
2. Queries containing partial book contents or genre descriptions (content-based queries).

Community-Based Search (CBS) can operate using either type of query. Our tests were performed using queries that were of the second kind. Therefore a solution to a query contained a list of all the peers that matched as many genre descriptions as possible, implying that the peers were likely to be members of the communities $C=\{C_1, \ldots, C_N\}$ whose member/s owned the requested book.

The performance of CBS was evaluated against the performance of two well-known search techniques:

1. Gnutella search, a hop-limited breadth-first search of the peer-to-peer network beginning from the querying peer
2. Hub search, a hop-limited search like Gnutella, except that only one peer, selected for having the maximum number of neighbors, is forwarded the query each time.

Gnutella search queries were terminated after 2 hops or after finding a single matching peer. Hub search queries were terminated after 5 hops or after finding a single matching peer.

The parameters with which the search techniques were evaluated are:

1. The number of messages required for each search method
2. Quality of the solution, which is a measure of the number of peers found that were likely to be members of the maximum communities in *C*, i.e., the peers found had high link weights for the attributes that matched the genre descriptions
3. Satisfaction of the query, which is the percentage of meta-information attributes matched

6.6.1 Experiment Setup 1

We simulated the CBS operation over several networks. The first set of experiments used a 500 node peer-to-peer network created by the rules described in the section 3.2.2. A set (10% of available peers) of random peers was selected to create queries. Approximately 20% of the queries were for peers within one of the querying peer's community, 10% of the queries attempted to find peers in overlapping communities, and 70% of the queries were for peers in some random, remote (non-overlapping) community. The details of the queries were generated from a list of 25 known attributes ($|I|$=25 and $|P|$=20). On an average, queries had 2.18 attributes in the meta-information list. The first approach was used for selecting P_S.

Fig. 19 shows that CBS consistently requires fewer messages in order to process the queries. Furthermore, CBS scales well when the number of queries increases. On the other hand, the number of messages generated during the Hub search method (Hub) begins to increase rapidly as more queries are created, because each hub forwards queries to all its neighbors. Although the Gnutella (Gnu) technique exhibits linear behavior (since hop-limit was set at 5), it still does not outperform CBS in terms of the number of

messages generated. The average number of messages per query was 1 for CBS (single push operation); 1.26 for Gnutella; and 2.56 for Hub search.

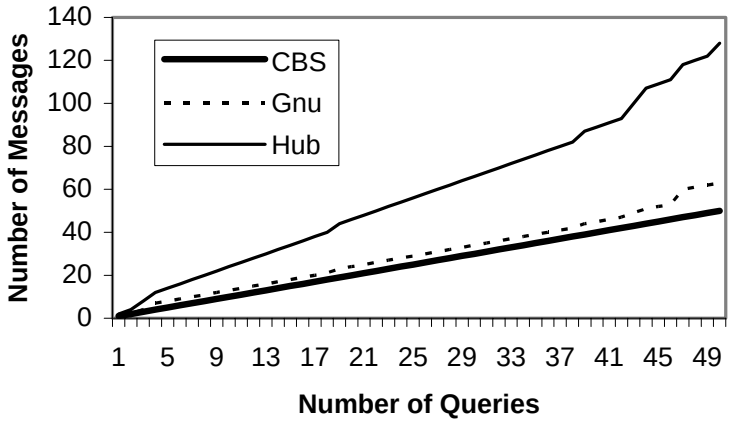


Figure 19. Setup #1 - Evaluation of number of messages as the number of querying peers increases (X-axis)

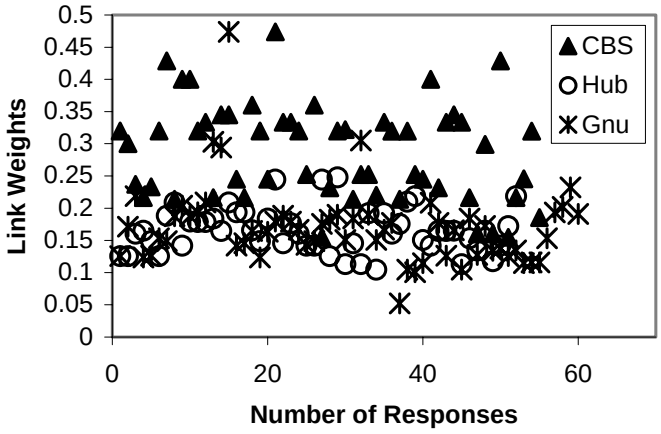


Figure 20. Quality of the query solution

The graph in Fig.20 displays the link weights of the peers found by the various search techniques. Higher link weights indicate that the responding peers have higher

involvements in the communities whose member/s might own the requested book. The average link weight of responding peers was 28.67% for CBS; 16.95% for Gnutella; and 16.59% for Hub search.

CBS also outperformed the other search techniques in the satisfaction of the query. The peers that responded using CBS matched on average 81% of the meta-information attributes. For Gnutella and Hub search, the responding peers only matched 70% and 67% respectively. Therefore, search queries routed using CBS found more relevant peers since the responding peers were members of communities that were related to more meta-information attributes than Gnutella and Hub search with the same termination conditions as described earlier.

6.6.2 Experiment Setup 2A

The second set of experiments modeled a peer-to-peer network formed by computer science students from 5 different universities. The purpose of the network was to communicate and exchange resources, such as help files and sample source code, about different programming languages. A 5,000 node peer-to-peer network was created by our rules, as described earlier, to represent approximately 1,000 students per university. The list of known interest attributes included 96 programming languages that are currently being used ($|I|=96$). The list was obtained from Usenet groups (Google Groups) (See Appendix for the list). The Internet survey (Programmer's Heaven) indicates that most engineers know 5 or more programming languages. Therefore in our model, we assigned an average of 5 programming languages to the set of personal attributes for each peer. On an average each peer claimed 2.26 attributes in its claimed

attribute set. The claimed attributes in this model represents the programming languages that the peer uses frequently. The model took into consideration that certain programming languages were very popular while most others were not so popular. Hence we assigned the programming languages using a power-law distribution so that 34.36% of the languages were assigned to about 79.69% of the peers. This resulted in 83 communities being populated with around 0.1% of the peers in the network, and just 2 communities being populated with 10% or more of the peers. Fig. 21 shows the power-law distribution in the sizes of the communities that were formed. The logarithmic X-axis plots the frequency and the logarithmic Y-axis plots the community sizes as a percentage of the number of nodes, i.e. 5,000.

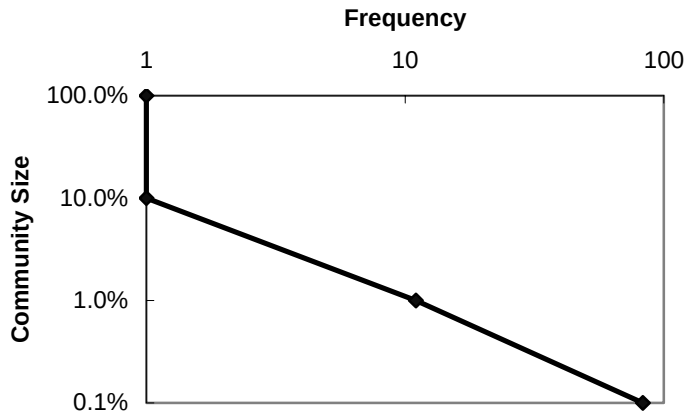


Figure 21. Power-Law Distribution in sizes of communities formed

A set (10% of available peers) of random peers was selected to create queries. On an average, queries had 2.18 attributes in the meta-information list. The creation of queries also considered the disparity in programming language popularity and followed the same power-law distribution for attributes in the meta-information lists. The second

approach was used for selecting P_S, i.e. P_S matches the maximum meta-information attributes.

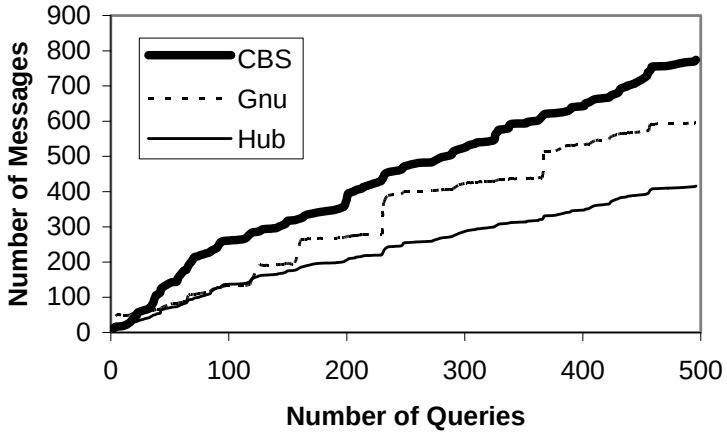


Figure 22. Setup #2A - Evaluation of number of messages as the number of querying peers increases (X-axis)

Fig. 22 shows that CBS requires many more messages in order to process the queries. Nevertheless, CBS scales well when the number of queries increases. Although the Gnutella (Gnu) technique follows the Hub search method (Hub) up to around 120 queries, it soon branches off indicating the use of more messages. The average number of messages per query was 102.19 for CBS; 77.13 for Gnutella; and 54.94 for Hub search.

The graph in Fig.23 displays the link weights of the peers found by the various search techniques. An increasing magnitude of Link Weights forms the logarithmic Y-axis. The integer value of the meta-information attribute forms the X-axis. Higher link weights indicate that the responding peers have higher involvements in the communities whose member/s might own the requested resource. Note that responses are not evenly

distributed. Most of the responses are for lower attribute values (see clustering to the left of graph). This is because most queries were for popular programming languages and hence most responses were for those programming languages. However, queries related to less popular programming languages did not result in high quality responses in Gnutella or Hub search. CBS, on the other hand, consistently provides high quality responses even when queries are for less popular items and therefore perhaps further away from the querying peer. The average link weight of responding peers was 24.30% for CBS; 23.51% for Gnutella; and 28.76% for Hub search.

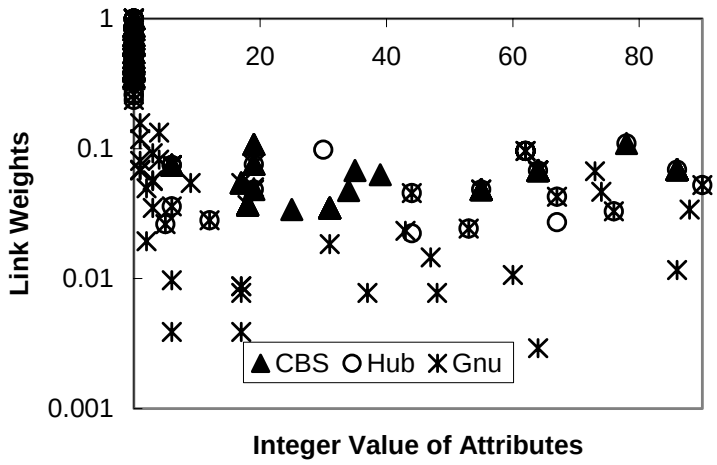


Figure 23. Setup #2A – Quality of the query solution

CBS also outperformed the other search techniques in the satisfaction of the query. The peers that responded using CBS matched on average 57% of the meta-information attributes. For Gnutella and Hub search, the responding peers matched 56% and 20% respectively. Therefore, again search queries routed using CBS found more relevant peers for the same reasons described before.

6.6.3 Experiment Setup 2B

The third set of experiments used the same model described in section 6.6.2 (i.e. same network, similar method of creating queries and assigning attributes). The only difference to the CBS algorithm was that the selection of P_S was done using the first approach, i.e. P_S is the closest seer containing at least one attribute from the meta-information list.

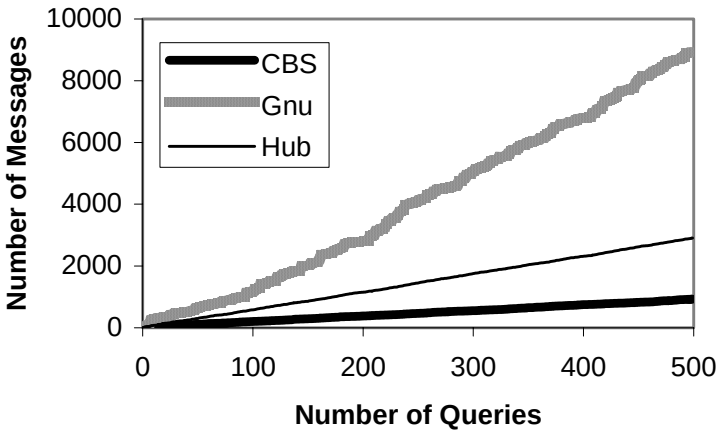


Figure 24. Setup #2B – Evaluation based on number of messages for search operation as the number of querying peers increases (X-axis)

Fig. 24 shows that CBS consistently requires fewer messages in order to process the queries. Furthermore, CBS scales extremely well when the number of queries increases. On the other hand, the number of messages generated during the Hub search method (Hub) and Gnutella search method (Gnu) begins to increase rapidly as more queries are created. The average number of messages per query was 106.42 for CBS; 927.18 for Gnutella; and 330.58 for Hub search.

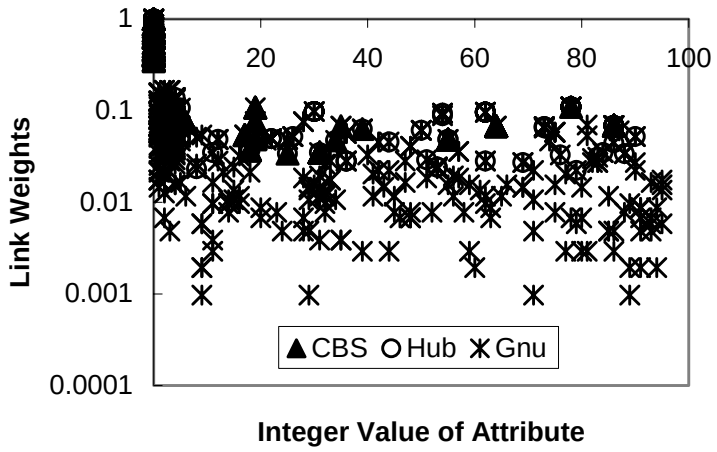


Figure 25. Setup #2B – Quality of the query solution

The graph in Fig.25 displays the link weights of the peers found by the various search techniques. An increasing magnitude of Link Weights forms the logarithmic Y-axis. The integer value of the meta-information attribute forms the X-axis. Again note that responses are not evenly distributed. The graph shows that queries related to less popular programming languages did not result in high quality responses in Gnutella or Hub search. CBS, on the other hand, consistently provides high quality responses even when queries are for less popular items and therefore perhaps further away from the querying peer. The average link weight of responding peers was 19.85% for CBS; 6.83% for Gnutella; and 18.90% for Hub search.

CBS also outperformed the other search techniques in the satisfaction of the query. The peers that responded using CBS matched on average 56% of the meta-information attributes. For Gnutella and Hub search, the responding peers matched 55% and 42% respectively. Therefore, again search queries routed using CBS found more relevant peers for the same reasons described before.

6.7 Summary

This chapter described the mechanics of our Community-Based Search (CBS) query propagation technique. Our proposed solution for searching the peer-to-peer network takes advantage of interest-based communities of peers. This chapter demonstrated how our community-based search query propagation provides more efficient searching by targeting one or more communities, irrespective of the current membership of the searching peer. The community-based search technique consistently provides high quality responses even when queries are for less popular items and therefore perhaps further away from the querying peer. CBS also allows search operations to be based on content rather than just filenames, as in many existing peer-to-peer search techniques.

CHAPTER 7

TRUST MANAGEMENT USING PEER COMMUNITIES

7.1 Motivation

Current peer-to-peer systems are often targeted for global information sharing, replicated file storage, and searching by using an end-to-end overlay network. Although these systems usually involve information exchange between peers, they have either protected peers' anonymity (Clark et al., 2000; and Dingledine, Freedman and Molnar, 2000), or required transacting peers to trust each other implicitly (Gnutella).

Both these approaches are vulnerable to attacks by malicious peers who could abuse the peer-to-peer system to spread viruses, incorrect, or damaging information. Therefore in order to enable practical information sharing in such decentralized and dynamic systems, a viable trust model needs to be incorporated that will allow peers to have varying amounts of dynamically changeable trust amongst each other.

Traditionally, trust has been implemented through exhaustive policy lists that needed to be created at system design time, such as role-based access control (RBAC) lists (Ferraiolo and Kuhn, 1992). In contrast, our trust management system is dynamic and requires minimal global knowledge. Further, the decentralized nature of our algorithms makes it suitable for peer-to-peer systems.

7.2 Challenges for Trust Management

The main challenges that need to be addressed are: how to describe if a peer is trustworthy, what low-cost verification algorithm can be executed by a peer to determine

the trust value of some other peer, how are trust values about peers exchanged within the system, how can dishonest peers be punished.

This chapter proposes an approach for trust management in peer-to-peer systems. It introduces a role-based model for trust amongst peers and shows that it is scalable, dynamic, revocable, secure and transitive. The trust model assigns role-based trust values to peers proportional to their status in the system. The status of a peer depends on its relationships with other peers. Our proposed solution permits asymmetric trust relationships that can be verified by any peer in the system through a simple, low-cost algorithm. Since trust values are proportional to the status of a peer, it is essential to ensure that relationships between any two peers will be legally binding and have non-repudiation; that is peers cannot falsely deny their relationship with another peer. However it is equally essential that peers have the ability to revoke their relationships with malicious peers to punish them for false or damaging information. Finally, this paper introduces a metric that combines a peer's trust value for each of its roles. The combined trust value is a single, relative, probabilistic guarantee that offers peers with a simple, verifiable trust metric about other peers in the peer-to-peer system.

7.3 Dynamic Coalitions

The research described in this chapter was related to a larger project known as *Dynamic Coalitions* (Dasgupta, Karamcheti and Kedem, 2000). Dynamic Coalitions enables a set of partners to work together while sharing information, resources, and capabilities in a controlled and accountable fashion. The partners themselves are

organizations composed of people, departments, computational entities, and agents who perform tasks consistent with the internal rules of their organization.

Coalitions are supported by several innovative techniques such as transitive delegation, cryptographic file systems, capacity sandboxing, reverse sandboxing, and fine-grained access control. These techniques facilitate scalable authentication and revocable authorization of agent computations even when they span resources of different organizations. In addition, they improve overall efficiency by permitting migration of computations to, and a caching of services in, partly trusted environments of another organization.

Let us enforce that every peer belong to at least one pre-determined group corresponding to the department or organization of its human user. For home users, the domain name of an Internet connection is used to identify the pre-determined group of the peer. Thus the basic construct of peer-to-peer systems can be used to implement a practical Dynamic Coalition environment where coalitions are created between peers in different groups.

7.4 Peer-to-Peer Trust Model

We propose an optimistic trust model that provides probabilistic guarantees based on the status / popularity of the peers. Peers have the ability to revoke their relationships with malicious peers and thus cause the trust values of wrong-doers to be reduced. The probabilistic guarantee provides a web-of-trust style estimate based on a peer's past transactions. The accuracy of the guarantee depends on the thoroughness of the peer in

discovering and validating the trust values of other peers. Therefore, non-critical transactions need not consume the resources of the peer-to-peer system.

This section describes our model for trust using peer-to-peer communities. It explains how trust can be assigned and discovered. The following sections discuss how trust can be revoked, and protected against non-repudiation.

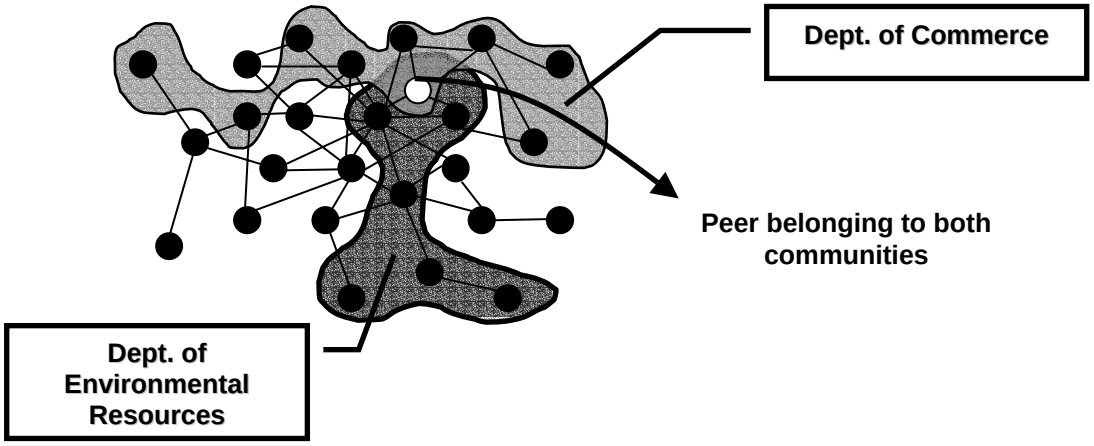


Figure 26. Example of a peer belonging to overlapping communities

7.4.1 Peer Roles and Involvement

Previously it has been pointed out that peer-to-peer communities are implicitly formed, self-organizing structures that depend on the declared (claimed) interests of peers. As a result, peers may belong to more than one, possibly overlapping community. In the case of a constrained application, such as a digital library, community structures will span across departmental or organizational boundaries. For instance, if the digital library were implemented by government departments to share documents and resources, a conceivable community might include peers from both, the Department of Commerce (Maritime Administration) and the Department of Environmental Resources that are

concurrently interested in pollution in US ocean water-ways. This is an example of a cross-departmental community (see Fig.26). Peers might also be part of intra-departmental communities, such as the community of Maritime Administration, or the community of Transportation within the Department of Commerce.

The different communities within which a peer can participate due to its claimed interest attributes constitute the roles of the peer. Every peer will have at least one role corresponding to its pre-determined group. Link Weights, by definition, indicate the number of peers known directly (1-hop neighbors), or indirectly (2-hop neighbors) to a peer within each of its roles (communities). Below a definition for *involvement* is provided, which, like Link Weights, is associated with each role Ψ of a peer V and is proportional to the number of peers within the neighborhood (1-hop and 2-hop neighbors) of V that are also part of Ψ. We call peers with high values of involvement, seers (See 5.3.1 for definition).

For the purposes of simplicity, the examples discussed in this chapter consider peer-to-peer communities each formed due to single shared interest attributes. This means that the signature S of every community will be a single attribute set. Therefore, the intersection set $C_i \cap S$ can only contain one claimed attribute which has an associated Link Weight that is also the Involvement value of the peer in the community S. Nevertheless, our definition for Involvement provides a way to extract values in more complex scenarios where communities of peers share more than just a single interest attribute in common.

7.4.2 Trust, Links and Link Weights

7.4.2.1 First Attempt: Trust and Links. We initially associated trust values with peer links due to the following reasons: (1) Peers create and maintain links to other peers whom they know and therefore trust (optimistically); (2) Since links are bi-directional, information provided by peers that have more links might be more trustworthy. The association of trustworthiness of information (authoritativeness) with links is used by Google in its PageRank metric (Brin and Page, 1998). The PageRank of a web page measures the authoritativeness of its content; (3) Peer links offer a simple, natural trust model that can easily be revoked. If after some transaction, a peer loses trust in its neighbor, it can break (remove) that link, thereby reducing the number of links at its neighbor.

Examples of analogous systems with a similar association between trust and links include: citation graphs in scientific publications, where experts who are well-known and highly regarded by most other authors tend to be highly connected nodes (Kleinberg, 1998; Adamic and Adar, 2000; and Page et al. 1998); and eBay points, where the rank of users is proportional to the number of transactions (purchase / sale) that they have completed with other eBay users.

Despite its wide-spread use, the association of trust with peer links does not provide an elegant solution to trust management. Often, peers that are highly linked-to (hubs) make mistakes, provide incorrect information, or assist in spreading damaging information unintentionally. Pastor-Satorras and Vespignani (2001) argues that viruses

or damaging information from hubs can epidemically spread and persist within a scale-free network, such as peer-to-peer network.

We believe that by making a slight modification, links can provide practical and accurate trust guarantees in decentralized systems. The most important detail that has been lacking in previous trust models is the consideration that peers participate in many different communities (roles). Therefore, in the citation graph, although an author of papers in Biology is highly cited, it is conceivable that the author's explanations of Electrical Engineering concepts are incorrect. Likewise, on eBay, a popular antique seller is not necessarily a trusted expert on electronic equipment.

7.4.2.2 Second Attempt: Trust and Links Weights. It is necessary to consider the roles of a peer when deriving its trust value. We thus propose the use of Link Weights as an indication of role-based trust. With reference to Fig. 2, peer *V* knows more peers within the community of peers interested in "Biography", than it knows within its other communities. As a result, information provided by *V* and classified as "Biography" is more likely to be accurate than information provided by *V* and classified as "Magazine".

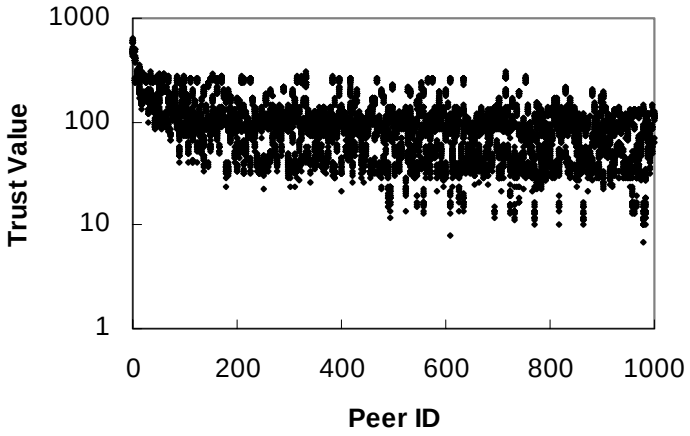
Let us imagine another peer named *W* that has a Link Weight of 10 associated with the interest "Biography." Using our model, *V* would have a trust value of 10 for *W*, but *W* would have a higher trust value of 23 for *V*. The association of Link Weights with trust values allows for asymmetric trust relationships that imitates trust relationships amongst humans in a social network.

7.4.3 Trust Value Distribution

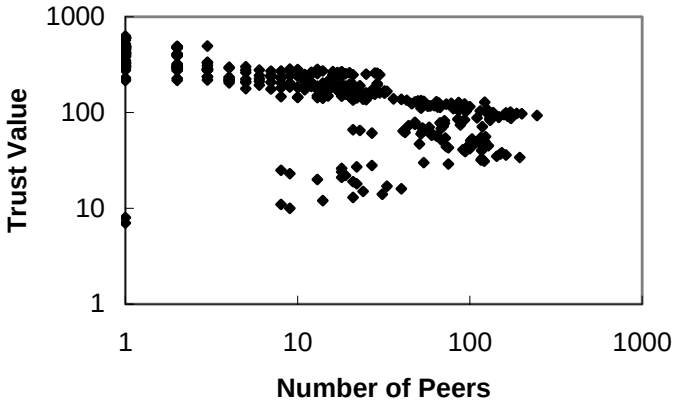
We plot the trust value distribution in a large-scale peer-to-peer network containing 1000 nodes. The graphs in Fig. 27 show a non-scale-free distribution of trust values. This is important because it highlights the dissimilarity in the distribution of trust values when it is obtained from links and our model, where trust values are obtained from Link Weights.

In order to correctly model the peer-to-peer network, we incorporate the scale-free property into the network topology (See 3.2.2 for technique used to form peer-to-peer networks). Therefore, when associated with peer links, trust value distribution decays as a power-law (see Fig. 28), like the degree distribution of peers. This results in almost all peers having low trust values except for a small group of peers that have exceptionally high trust values.

In contrast, Fig. 27 illustrates that the majority of peers start out with a median trust value (around the center of a range of values), while a small group of peers have either higher or lower trust values. In the network considered, the average trust value was 100, maximum was 628, minimum was 7, and mode and median were 93. This distribution is most suitable for an optimistic trust model such as ours because a peer can enter into transactions with other peers whose trust values are median and most likely comparable to its own. Malicious peers will find their trust values dropping unlike in a scale-free distribution where trust values usually cannot be lowered because most peers start out with low trust values. For critical transactions, information can be sought from peers with higher trust values.



a. Trust Values of all peers (see median band)



b. Frequency of peers with different trust values

Figure 27. Trust Value Distribution when trust is associated with Link Weights (1000 peers)

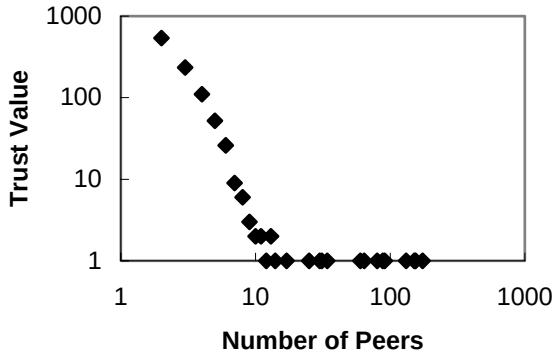


Figure 28. Trust Value Distribution when trust is associated with links (1000 peers)

7.4.4 Verification and Validation

In section 4.3.1 we proposed an *Attribute Escalation* algorithm that uncovered implicit communities and enabled the formation of new communities. We propose a simple modification to the Attribute Escalation algorithm that will allow trust values of a peer to be guaranteed. Instead of simply sending out the list of claimed attributes, each peer V will construct the following message M and send it individually to each of its neighborhood peers $\{V_1, V_2, \ldots, V_n\}$.

$$M = \left\{IP_{dest}, IP_{source}, \left\langle CA_{source} \right\rangle\right\}, E_{source}(M)$$

where,

M is the constructed message,

IP_{dest} is the destination identity (a neighborhood peer),

IP_{source} is the sender identity (i.e. V),

$\langle CA_{source} \rangle$ is the claimed attributes list of the sender,

$E_{source}(M)$ is the M's signature by the sender's private key.

Every peer is responsible for storing messages received from its neighborhood peers in a publicly accessible blackboard (see section 6.4). Blackboards are like websites and the content on a peer's blackboard can be viewed by any peer within the system.

Let us return to the example in Fig. 2. When *V* claims a Link Weight (and therefore trust value) of 23 for "Biography," any peer W in the peer-to-peer system will be able to verify this value by visiting *V*'s blackboard and re-calculating the Link Weight from the posted messages. This calculation is a simple counting operation with a complexity of *O(n)* (See section 4.3.1 for detailed algorithm). Prior to verifying the Link Weight however, *W* might chose to validate the signatures of the messages posted on *V*'s blackboard in an attempt to uncover fabricated messages that were used to artificially increase *V*'s Link Weight value. We call these fabricated messages false messages.

It might seem intuitive that before entering into a critical transaction with peer *V*, an exhaustive process needs to be employed where every one of *V*'s messages has its signature validated. However, we show that contrary to intuition, peers need only validate a small percentage of messages to uncover one or more false messages (if they exist) with a high degree of probability.

To begin a brief theoretical analysis is provided and then it is backed with results obtained from experiments.

Let *N* be the number of messages on peer *V*'s blackboard. Assume *k* messages are false. Therefore, *N*-*k* messages are not false. Also assume that peer *W* randomly selects m messages to validate. Now the probability that *W* will not discover any false messages is given by:

$$\rho = \frac{N-k}{N} \times \frac{N-k-1}{N-1} \times \frac{N-k-2}{N-2} \times \cdots \times \frac{N-k-m}{N-m} \qquad \textbf{(4)}$$

So the probability that *W* will discover a false message is:

$$\rho' = 1 - \rho \qquad \textbf{(4.1)}$$

Fig. 29 plots the relation between percentage messages verified and ρ'. The curves vary for k = 10% of N, k = 20% of N, and k = 30% of N. N was chosen to be 100. An increasing percentage of m/N forms the X-axis, while ρ' forms the Y-axis. The graph shows that selecting just 10-20% of the messages to validate will uncover false messages with probability of 70-95%. If a peer fabricates more messages, then the validation of messages will quickly uncover false messages.

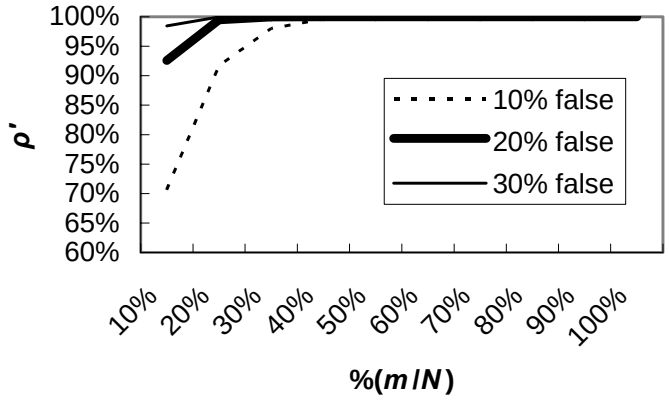


Figure 29. The relationship between percentage of messages chosen to validate (*%m/N*) and probability of uncovering false messages (ρ')

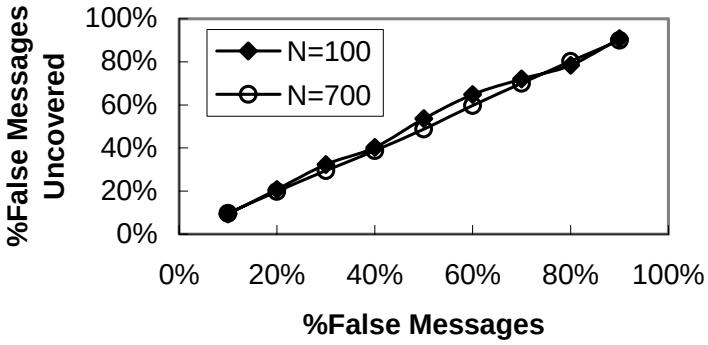


Figure 30. Plot of the percentage of false messages uncovered (Y-axis) as the percentage of false messages is increased (X-axis). %(m/N) = 10%

The theoretical analysis indicates that peers will uncover false messages even when a small, randomly selected set of messages is validated. We also observed this behavior in our experiments. Fig. 30 presents two cases: N = 100 messages and N = 700 messages, when the percentage of messages randomly selected for validation (*%m/N*) was set at only 10%. The graph shows that even when only 10% of the messages are false, 10% of those false messages were uncovered by a peer that randomly selected 10% of the original number of messages to validate.

7.5 Using the Trust Model in Dynamic Coalitions

Dynamic Coalitions are temporarily formed between peers belonging to different communities that each represents a separate organization / department. The trust model we proposed can be used to provide probabilistic trust guarantees to each peer in the coalition. Table 7 lists the algorithm that we use to obtain trust values of a peer in a Dynamic Coalition. Since peers can belong to more than one community, the *FindTrust*

method finds all the trust values (Link Weights) of a peer *V*. The method can be invoked by any peer *W* (not necessarily part of the Coalition). Initially, the *CommonCommunities* method checks *V*'s claimed attributes (posted as messages on its blackboard) for any common attributes between *W* and *V*, indicating possibly shared communities. This is a linear search operation with complexity of *O(n)*. If there are no common attributes, *W* asks all its immediate neighbors if any of them share communities with *V*. In the worst case scenario (neighbors need to execute *CommonCommunities*), the operation has $O(n^2)$ complexity. The best case scenario (neighbors already know common communities from past transactions) is an *O(n)* operation. As a final attempt, if still no common attributes exist, *W* asks its 2-hop neighbors the same question (worst case: $O(n^2)$, best case: *O(n)*) before giving up trying to find trust values for *V*. Remember that after the attribute escalation algorithm, a peer knows the identities of all its 2-hop neighbors and therefore does not have to find their identities at this stage. In order to reduce bandwidth utilization and processing time, a peer might decide to forego finding trust values from its 2-hop neighbors. Our experiments revealed that 2-hop neighbors need to be consulted 32% of the time when 10% of randomly selected peers invoked *FindTrust*.

Table 7

Algorithm to Find Trust Values of a Peer in a Coalition

Line	Pseudo-Code
	`/* function prototype declaration */`
1.	`bool CommonCommunities(int);  /* peers share communities? */`
2.	`int[] ListTrusts(int);  /* list all known trust values */`
3.	

```
4.      bool AskNeighbors(int);  /* 1-hop neigh share communities? */
5.      int[] Lower(int[]);  /* multiply trust values by 0.5 */
6.      bool Ask2HopNeighbors(int);  /* 2-hop neigh share? */
7.      bool VerifiedTrusts(int[]);  /* verify and validate values */
8.
        /* FIND-TRUST (peer) */
9.
        int[] FindTrust(int PeerID)
10.
          int[] list_Trusts; /* trust values */
11.
12.
          if (call CommonCommunities(PeerID))
13.
            list_Trusts = call ListTrusts(PeerID);
14.
          else-if (call AskNeighbors(PeerID))
15.
            list_Trusts = call ListTrusts(PeerID);
16.
            list_Trusts = call Lower(list_Trusts);
17.
          else-if (call Ask2HopNeighbors(PeerID))
18.
            list_Trusts = call listTrusts(PeerID);
19.
            list_Trusts = call Lower(list_Trusts);
20.
            list_Trusts = call Lower(list_Trusts);
21.
          else-if
22.
            WARNING “No trust values!”;
23.
            return NULL;
24.
          end-if
25.
26.
          if (call VerifiedTrusts(list_Trusts))
27.
            return list_Trusts;
28.
          else
29.
            WARNING “Found False Messages”;
30.
```

```
31.          return list_Trusts;
           end-if
```

For each attribute found in common with *V*, the corresponding Link Weight is stored in *list_Trusts*. Link Weights provided by 1-hop neighbors will be multiplied by 0.5 and values provided by 2-hop neighbors get multiplied by 0.25. All trust values are validated using the process described in the earlier section.

The list of trust values provides a peer in a coalition with a probabilistic trust guarantee about another peer. Tampered values do not go undetected (due to verification and validation), making these values secure. Additionally, trust values can be transitively obtained from other peers and scaled down depending on the peer providing the values.

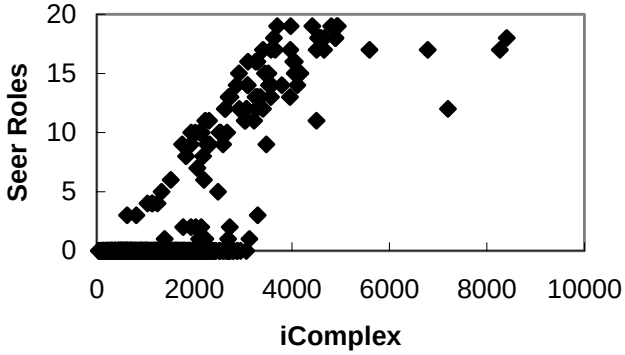
As an alternative to our current scaling down process, trust values transitively obtained from another peer could be multiplied by the trust value of that peer. However, it has not yet been explained how a peer can have a single trust value. The next section will present our idea for a collective trust value of a peer that can be used for a more realistic scaling process amongst other benefits.

7.5.1 Aggregating Trust Values into an *iComplex*

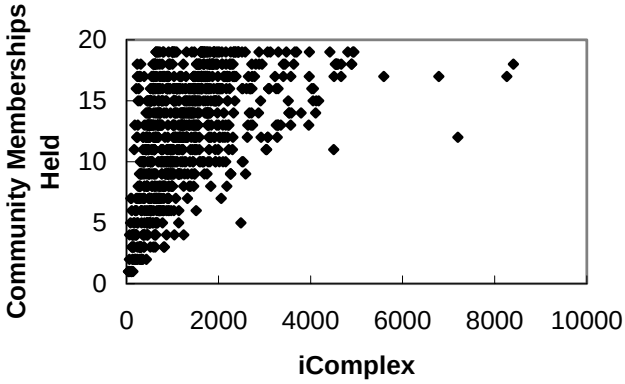
The list of trust values associated with each peer can be used to provide probabilistic guarantees to other peers. However in a practical implementation of peer-to-peer communities, a single shared interest attribute will not always be the signature of a community. At the end of section 3.1 it was assumed that every community signature

contained only a single attribute. Let us see what happens when this simplification assumption were temporarily removed.

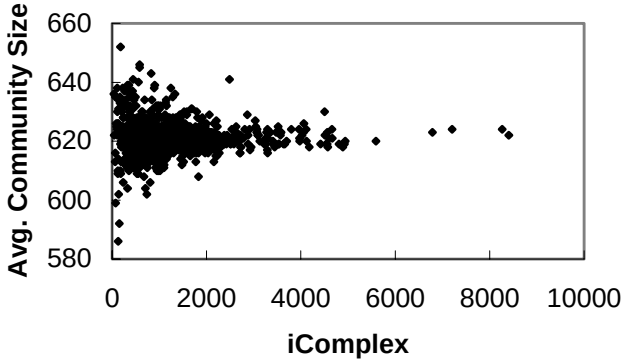
Firstly this means that $|S|$ could be greater than one. Imagine a digital library with several communities of peers. Suppose there exists a community (of science-fiction enthusiasts) with signature S_1 = {"Fiction", "Technical"}, and another community (of fiction enthusiasts) with signature S_2 = {"Fiction"}. Finally assume that Peer *V* (from Fig. 2) is a member of both these overlapping communities. Based on the Link Weight values from Fig. 2 and the definition of Involvement (Section 5.3.1), *V* is more involved in the community of fiction enthusiasts than in the former community. If there exists another peer *W* with Link Weight values, "Fiction"=18 and "Technical"=18 (therefore *W*'s Involvement in community S1 is 18), then information provided by *W* and classified as "science-fiction" is more likely to be accurate than information provided by *V* having the same classification.



a. *iComplex* Vs Number of communities in which peer is a seer



b. *iComplex* Vs Community memberships held



c. *iComplex* Vs Average Community Size

Figure 31. Behavior of *iComplex* when calculated as a sum of all trust values

Involvement values, which are associated with each community within which a peer is a member, play an important role in determining accurate role-based trust values of a peer. Now a proposal is made for aggregating all involvement values corresponding to a peer into an *iComplex*. An iComplex value is calculated by each peer individually and stored on their own blackboards. Since iComplex values are, in essence, aggregations of trust values, the verification and validation process described earlier will still apply. As a system design, all peers need to agree upon the aggregation function to calculate their iComplex values. Examples include but are not restricted to: sum of all trust values, or average of all relative trust values. A relative trust value $rt_1 = t_1$ divided by the approximate size of community $S_1=\{C_1\}$, where t_1 is a trust value for attribute C_1, and the approximate size of the community is obtained through Distributed Discovery (see chapter 5).

Fig. 31 shows the behavior when sum is the aggregate function used to compute iComplex values. The first graph illustrates that higher iComplex values implies that the peers are seers (highly-involved) in more communities and therefore these seers have higher trust values than other peers in each of their roles. The next graph shows the relationship between iComplex and number of memberships held by a peer. This graph is significant because it demonstrates the effectiveness of an iComplex value aggregated using a simple sum function. The graph shows that peers cannot increase their iComplex values by simply joining many communities. In fact peers that are members of many communities are most likely to have low iComplex values (notice the clustering of points close to the Y-axis). The final graph shows that peers cannot obtain high iComplex

values by joining very large communities. The peers with the highest iComplex values were members of average sized communities.

Therefore, iComplex values calculated by adding all trust values of a peer can provide a reliable, collective trust value. Moreover, peers will not be able to synthetically increase their iComplex value by simply joining more communities or joining larger communities. A higher iComplex usually indicates that the peer is a seer of many communities and therefore trusted by the peers of those communities.

7.5.2 Using iComplex for Information Assurance

An incremental change to the format of the responses can be made by requiring the responding peer to send its iComplex value as well. This provides the querying peer with information on the probabilistic trust values of the responding peers. The iComplex values received will allow a querying peer to rank responses based on the probabilistic trust values of the responding peers.

7.5.3 Attacks and Threat Assessment

Without the iComplex value, malicious peers could misinform a querying peer about the peer that owns a particular book / resource. A misinformed querying peer will then obtain incorrect / damaging data from the peer identified by malicious peers. In a business-world implementation of digital libraries, malicious peers might dishonestly divert traffic away from certain other peers.

Since iComplex values are ultimately calculated from Link Weights which are dependent on the number of peers in one's neighborhood that share a certain attribute, one way of fraudulently increasing the iComplex value would be to create dummy

neighbors with real peer identities and interests. This is a difficult problem to solve. There have been a few attempts to solve this by: using reputation-based systems or making it difficult to create a new peer identity (Murphy VII and Manjhi, 2002) (by computationally expensive key generation, or associating it with a government issued identity number, such as social security number, voter identification number, and so on).

Finally, because our trust model provides probabilistic guarantees, a peer with a high iComplex value can still provide (with low probability of doing so) incorrect / damaging information as a result of a query. We therefore propose a revocation mechanism (described in the next section) as a means to punish wrong-doers.

7.6 Revocation and Non-Repudiation of Trust

This section discusses how our trust model allows peers to revoke relationships with malicious peers, and the non-repudiation of peer relations. Malicious peers are not only peers that provide incorrect/damaging information, but also are peers that use unfair methods to lower the trust values of their neighbors.

7.6.1 Revocation

We propose a distributed revocation mechanism, where each peer maintains its own revocation list. Therefore a disgruntled peer *W* that has been affected by previous transactions with a malicious peer *V* can simply maintain this information in a revocation list posted on its blackboard.

The validation procedure described earlier involves validating the signatures of the messages posted on a peer *V*'s blackboard in an attempt to uncover false messages. In order to allow for revocation of these messages, we propose an additional action that

peers entering into a transaction with *V* can execute after the validation procedure. We call this action Revocation Check. The action entails: (1) randomly selecting a few validated messages from *V*'s blackboard; (2) determining the peers that authored those messages; and (3) visiting the blackboards of the message authors to check for possible revocations. If the Revocation Check finds that the message authors have placed *V* in their revocation lists, then those messages are called revoked messages.

Section 7.4.4 explains the relationship between the number of messages validated and the number of false messages uncovered. We therefore know that if 10% of the messages of a peer *V* were selected for Revocation Check and 10% of *V*'s neighbors had placed *V* in their revocation lists, then the Revocation Check will find that 10% of the selected messages are revoked messages.

As a result, if *V* has maintained a good record over a large number of transactions, except for a few incorrect/damaging transactions, then its trust value will remain high. Also, a malicious neighbor of *V* would not be able to independently bring down the *V*'s trust value.

Finally, the Revocation Check procedure can ascertain if a malicious neighbor *W* of *V* has unfairly revoked its relationship with *V*. This means that *W* continues to account *V*'s signed messages to calculate its trust values and iComplex value even after placing *V* in its revocation list.

7.6.2 Non-Repudiation

It has been shown how peers can revoke their relationships with malicious peers to punish them for false or damaging information. However, since trust values are

derived from peer links, it is essential to ensure that peers cannot falsely deny their relationship with another peer.

A malicious neighbor *W* of *V* cannot lie about the fact that it is a neighbor of *V*. This is because the signed message (section 3.3) addressed to *V* and created by *W* will be publicly accessible from *V*'s blackboard. Therefore trust values calculated as a result of peer links have non-repudiation.

7.7 Summary

This chapter presented an approach for trust management in peer-to-peer systems. Our optimistic role-based model for trust amongst peers was introduced and it was shown to be scalable, dynamic, revocable, and secure. Our proposed solution permits asymmetric trust relationships that can be verified by any peer in the system through a simple, low-cost algorithm. This chapter introduced a metric known as iComplex that combines a peer's trust value for each of its roles into a single, relative, probabilistic guarantee of trust. Finally, it discussed how our trust model allows peers to revoke relationships with malicious peers, and the non-repudiation of peer relations.

CHAPTER 8

CONCLUSIONS AND RECOMMENDATIONS

8.1 Future Work

The current crop of peer-to-peer systems focuses on sharing of resources (primarily files) amongst peers. Since for most purposes a client-server configuration provides a simple architecture and guaranteed performance, there has been very little interest within industry circles for adopting the peer-to-peer paradigm. Additionally, a completely decentralized system makes revenue generation and accounting non-trivial. This has further hindered industry participation and early-adoption.

Some interesting challenges that can be solved within the peer-to-peer paradigm are: revenue generation and accounting, so that companies or individuals can make money; alternative trust models and reputation systems, to allow for secure and trustworthy collaboration amongst peers; and investigations into the feasibility of integrating the peer-to-peer paradigm with mobile ad hoc networks, multimedia streaming, or grid computing, in order to uncover new and valuable applications.

Within the context of this work, additional work can attempt to further bring down the number of messages during community-based search. This would involve the deployment of a non-trivial caching algorithm amongst peers within a community so that the communication overhead of locating the appropriate seer by asking one's neighbors is reduced.

8.2 Conclusions

Peer-to-peer communities provide a method for arranging large numbers of peers in a self configuring peer relationship based on declared attributes (or interests) of the participating peers. This method is expected to have an impact on sharing of resources, pruning of search spaces, and trust relationships amongst peers in the network.

This dissertation shows that the attribute based clustering of peers can be made to work, by defining an overlay network, consisting of links. Links are user-directed connections based on experience and can be fine tuned for search and sharing performance. The peer formation algorithm is shown to stabilize in two rounds using the escalation technique. Additionally, an efficient community discovery procedure is introduced that uses weights and a threshold. Our simulations of the discovery procedure have confirmed that peers can quickly discover numerous memberships in different peer-to-peer communities using very little computation and communication messages.

Next, a push-pull communication technique was presented that helps disseminate information and propagate search queries within a peer-to-peer network. An initial set of algorithms provided a method for peers to discover their community memberships and certain properties of their communities. These properties include the approximate number of members and the information about the member peers. A repeatable push-pull gossiping protocol is then used to disseminate information only within the community of interested peers. We used simulations to provide evidence of the protocol's efficiency, robustness, and scalability.

Our solution for searching the peer-to-peer space also takes advantage of interest-based communities of peers. Several experiments demonstrated how our community-based search query propagation provides more efficient searching by targeting one or more communities, irrespective of the current membership of the searching peer. The community-based search technique also allows search operations to be based on content rather than just filenames, as in many existing peer-to-peer search techniques.

This dissertation also presented an approach for trust management in peer-to-peer systems. It introduced an optimistic role-based model for trust amongst peers and showed that it is scalable, dynamic, revocable, and secure. Our solution permits asymmetric trust relationships that can be verified by any peer in the system through a simple, low-cost algorithm. A metric known as *iComplex* was established to combine a peer's trust value for each of its roles into a single, relative, probabilistic guarantee of trust. Finally, the dissertation discussed how our trust model allows peers to revoke relationships with malicious peers, and the non-repudiation of peer relations.

REFERENCES

Abdul-Rahman, A. and Hailes, S.: "Supporting trust in virtual communities," *Proceedings of the 33rd Hawaii International Conference on System Sciences*, 2000.

Aberer, K. and Despotovic, Z.: "Managing trust in a peer-2-peer information system," *Proceedings of the 10th International Conference on Information and Knowledge Management*, Paques, H., Liu, L. and Grossman, D. (Eds.), ACM Press, 2001, pp. 310-317.

Aberer, K.: "P-Grid: A self-organizing access structure for P2P information systems," *6th International Conference on Cooperative Information Systems*, Trento, Italy, *Lecture Notes in Computer Science* 2172, Springer Veralg, Heidelberg, 2001.

Adamic, L. and Adar, E.: "Friends and neighbors on the web," (unpublished), 2000.

Adamic, L.A., Lukose, R.M., Puniyani, A.R., and Huberman, B.A.: "Search in power-law networks," *Physical Review* E, vol. 64, no. 4, 046135, 2001.

Agrawal, D., Abbadi, A.E., and Steinke, R.: "Epidemic algorithms in replicated databases," *Proceedings of the Sixteenth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*, Tucson, AZ, 1997, pp. 161-172.

Akamai Technologies. http://www.akamai.com/

Albert, R., Jeong, H., and Barabási, A.-L.: "Diameter of world-wide web," *Nature*, vol. 410, Sept. 1999, pp. 130-131.

Amaral, L.A.N., Scala, A., Barthélémy, M., and Stanley, H.E.: "Classes of small-world networks," *Proceedings of the National Academy of Sciences*, vol. 97, no.21, 2000, pp. 11149-11152.

Axelrod, A. and Cohen, M.D.: *Harnessing complexity: Organizational implications of a scientific frontier*, Basic Books, New York, NY, 2000.

Babaoglu, O., Meling, H., and Montresor, A.: "Anthill: A framework for the development of agent-based peer-to-peer systems," *Proceedings of the 22th International Conference on Distributed Computing Systems*, Vienna, Austria, 2002.

Barabási, A.-L and Albert, R.: "Emergence of scaling in random networks," *Science*, vol. 286, 1999, pp. 509-512.

Bloom, B.: "Space/time trade-offs in hash coding with allowable errors," *Communications of the ACM*, vol. 13, no. 7, 1970, 422-426.

Bolosky, W., Douceur, J., Ely, D., and Theimer, M.: "Feasibility of a serverless distributed file system deployed on an existing set of desktop PCs," *Proceedings of SIGMETRICS*, Santa Clara, CA, 2000.

Brin, S. and Page, L.: "The anatomy of a large-scale hypertextual Web search engine," *Computer Networks and ISDN Systems*, 30(1-7), 1998, pp. 107-117.

Bu, T. and Towsley, D.: "On distinguishing between internet power law topology generators," *Proceedings of INFOCOM*, 2002.

Caronni, G., Waldvogel, M., Sun, D., and Plattner, B.: "Efficient security for large and dynamic multicast groups," *IEEE 7th International Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises,* June 1998, pp. 376-383.

Clark, I., Sandberg, O., Wiley, B., and Hong, T.: "Freenet: A distributed anonymous information storage and retrieval system," *Proceedings of the Workshop on Design Issues in Anonymity and Unobservability*, Berkeley, CA, 2000.

Clarke, I., Hong, T.W., Miller, S.G., Sandberg, O., and Wiley, B.: "Protecting free expression online with freenet," *IEEE Internet Computing*, IEEE Press, vol. 6, no. 1, 2002, pp.40–49.

Cuenca-Acuna, F.M., Peery, C., Martin, R.P., and Nguyen, T.D.: "PlanetP: Using gossiping to build content addressable peer-to-peer information sharing communities," *Proceedings of the 12th International Symposium on High Performance Distributed Computing*, June 2003.

Dasgupta, P., Karamcheti, V., and Kedem, Z.: "Efficient and secure information sharing in distributed, collaborative environments," Proceedings *of 3rd International Workshop on Communication-based Systems*, April 2000.

Demers, A.J., Greene, D.H., Hauser, C., Irish, W., and Larson, J.: "Epidemic algorithms for replicated database maintenance," *Proceedings of the Sixth Annual ACM Symposium on Principles of Distributed Computing*, Vancouver, B.C., Canada, 1987, pp. 1-12.

Dingledine, R., Freedman, M., and Molnar, D.: "The freehaven project: Distributed anonymous storage service," *Proceedings of the Workshop on Design Issues in Anonymity and Unobservability*, 2000.

Druschel, P. and Rowstron, A.: "Past: Persistent and anonymous storage in a peer-to-peer networking environment," *Proceedings of the 8th IEEE Workshop on Hot Topics in Operating Systems*, Elmau/Oberbayern, Germany, May 2001, pp. 65–70.

Ellis, C.A., Gibbs, S.J., and Rein, G.L.: "Groupware: some issues and experiences," *Communications of ACM*, vol. 34, no.1, 1991, pp. 39-58.

Erdös, P. and Rényi, A.: "On the strength of connectedness of a random graph," *Acta Mathematica Acad. Sci.*, Hungary, 12, 1961, pp. 261-267.

Fan, L., Cao, P., Almeida, J., and Broder, A.: "Summary Cache: A scalable wide-area web cache sharing protocol," *Proceedings of ACM SIGCOMM*, Vancouver, Canada, 1998.

Ferraiolo, D.F. and Kuhn, D.R.: "Role based access control," *15th National Computer Security Conference*, 1992.

Festinger, L.: "Laboratory experiments: The role of group belongingness," in Miller, J.G., ed., *Experiments in Social Process*, McGraw-Hill, New York, 1950.

Flake, G.W.: *The computational beauty of nature: Computer explorations of fractals, chaos, complex systems, and adaptation*, MIT Press, January 2000.

Flake, G.W., Lawrence, S., and Giles, C.L.: "Efficient identification of web communities," *Proceedings of the 6th International Conference Knowledge Discovery and Data Mining*, ACM Press, New York, 2000, pp. 150-160.

Flake, G.W., Lawrence, S., Giles, C.L., and Coetzee, F.M.: "Self-organization and identification of web communities," *IEEE Computer*, vol. 35, no.3, March 2002, pp. 66-71.

Gajewska, H., Kistler, J., Manasse, M.S., and Redell, D.D.: "Argo: A system for distributed collaboration," *Proceedings of the Second ACM International Conference on Multimedia*, 1994.

Garey, M.R. and Johnson, D.S.: *Computers and in-tractability: A guide to the theory of NP-completeness*, W.H. Freeman, New York, 1990.

Garfield, E.: *Citation indexing: Its theory and application in science*, Wiley, New York, 1979.

Gibson, D., Kleinberg, J., and Raghavan, P.: "Inferring web communities from link topology," *Proceedings of the 9th ACM Conference on Hypertext and Hypermedia*, 1998.

Gil, Y. and Ratnakar, V.: "Trusting information sources one citizen at a time," *Proceedings of the 1st International Semantic Web Conference*, Sardinia, Italy, 2002.

Gnutella. http://www.gnutelliums.com/

Golbeck, J., Parsia, B., and Hendler, J.: "Trust networks on the semantic web," *Proceedings of Cooperative Intelligent Agents 2003*, Helsinki, Finland, 2003.

Google Groups. http://groups.google.com/

Google. http://www.google.com/

Granovetter, M.: "Strength of weak ties," *American Journal of Sociology*, vol. 78, 1973, pp. 1360-1380.

Gribble, S.D., Brewer, E.A., Hellerstein, J.M., and Culler, D.: "Scalable, distributed data structures for Internet service construction," *Proceedings of the 4th Symposium on Operating Systems Design and Implementation*, San Diego, CA, 2000

Gribble, S.D., Welsh, M., Behren, R.v., Brewer, E.A., Culler, D., Borisov, N., Czerwinski, S., Gummadi, R., Hill, J., Joseph, A.D., Katz, R.H., Mao, Z., Ross, S., and Zhao, B.: "The Ninja architecture for robust internet-scale systems and services," *Special Issue of Computer Networks on Pervasive Computing*, vol. 35, no.4, 2001, pp. 473-497

Gummadi, R. and Hohlt, B.: "Efficient implementation of a publish-subscribe-notify model using highly-concurrent B-Trees" (unpublished), 2000.

Harchol-Balter, M., Leighton, T., and Lewin, D.: "Resource discovery in distributed networks," *18th Annual ACMSIGACT/SIGOPS Symposium on Principles of Distributed Computing*, 1999.

Hayden, M. and Birman, K.: "Probabilistic broadcast," *Cornell CS Technical Report TR96-1606*, 1998.

Hayes, P., Hauptman, A., Carbonnell, J., and Tomita, M.: "Parsing spoken language, a semantic caseframe approach," *Proceedings of the 11th International Conference on Computational Linguistics*, Bonn, Germany, 1986, pp. 587-592.

Hodes, T.D., Czerwinski, S.E., Zhao, B.Y., Joseph, A.D., and Katz, R.H.: "An architecture for secure wide-area service discovery," *ACM Baltzer Wireless Networks*: *Selected papers from MobiCom 1999*, 1999.

Hong, X. and Gerla, M.: "Dynamic Group Discovery and Routing in Ad Hoc Networks," *Proceedings of the 1st Annual Mediterranean Ad Hoc Networking Workshop*, Sardegna, Italy, September 2002.

ICQ. http://www.icq.com/

Jeong, H., Néda, Z., and Barabási, A.-L.: "Measuring preferential attachment for evolving networks," *European Physics Letters* 61, 567, 2003.

Jive Software, http://www.jivesoftware.com/

Karp, R., Shenker, S., Schindelhauer, C., and Vocking, B.: "Randomized rumor spreading," *41st Symposium on Foundation on Computer Science*, Redondo Beach, CA, November 2000.

KaZaA. http://www.kazaa.com/

Keoh, S.L. and Lupu, E.: "Trust and the establishment of ad-hoc communities," *2nd Internal iTrust Workshop on Trust Management in Dynamic Open Systems,* London, UK, September 2003.

Kleinberg, J.M.: "Authoritative sources in a hyper-linked environment," *Proceedings of the 9th Annual AVM-SIAM Symposium on Discrete Algorithms*, 1998, pp. 668-677.

Kubiatowicz, J., Bindel, D., Chen, Y., Czerwinski, S., Eaton, P., Geels, D., Gummadi, R., Rhea, S.,Weatherspoon, H.,Weimer, W., Wells, C., and Zhao, B.: "OceanStore: An architecture for global-scale persistent storage," *Proceedings of the 9th International Conference on Architectural Support for Programming Languages and Operating Systems*, Cambridge, MA, 2000.

Lamport, L. and Chandy, M.: "Distributed snapshots: determining global states of a distributed system," *ACM Transactions on Computer Systems.* vol. 3, no. 1, February 1985, pp. 63-75.

Marsh, S.: "Formalising trust as a computational concept," Ph.D. Thesis, University of Stirling, 1994.

Matrouf, A., Gauvain, J.L, Ne´el, F., and Mariani, J.: "An oral task-oriented dialog for air-traffic controller training," SPIE 1293, *Applications of Artificial Intelligence,VIII*, April 1990.

Maymounkov, P. and Mazi`eres, D.: "Kademlia: A peer-to-peer information system based on the XOR metric," *Proceedings of the 1st International Workshop on Peer-to-Peer Systems*, Boston, MA, 2002.

Meissner, A. and Musunoori, S.B.: "Group Integrity Management Support for Mobile Ad-Hoc Communities," *Workshop proceedings Middleware for Pervasive and Ad-Hoc Computing*, Rio de Janeiro, Brazil, June, 2003, pp. 53-59.

Moy, J.: Ospf version 2, RFC 2328. Available from http://rfc.sunsite.dk/rfc/rfc2328.html, 1998. (Visited: November 25, 2003)

MSN Chat. http://chat.msn.com/

Murphy VII, T. and Manjhi, A.K.: "Anonymous identity and trust for peer-to-peer networks," (unpublished), 2002.

Murray, H.A.: *Explorations in personality*, Oxford University Press, New York, 1938.

Napster. http://www.napster.com/

Newman, M. E. J.: "Clustering and preferential attachment in growing networks," *Physical Review* E 64, 025102, 2001.

Oppen, D.C. and Dalal, Y.K.: "The clearinghouse: A decentralized agent for locating named objects in a distributed environment," *ACM Transactions on Office Information Systems*, vol. 1, no. 3, July 1983, pp. 230-253.

Page, L.: "PageRank: bringing order to the web," *Stanford Digital Libraries working paper* 1997-0072, 1997.

Page, L., Brin, S., Motwani, R., and Winograd, T.: "The PageRank citation ranking: Bringing order to the Web," *Stanford Digital Libraries Working Paper*, 1998.

Pastor-Satorras, R. and Vespignani, A.: "Epidemic spreading in scale-free networks," *Physical Review Letters*, vol. 86, no. 14, April 2001, pp. 3200-3203.

Pelc, A.: "Fault-tolerant broadcasting and gossiping in communication," *Networks*, vol. 28, no. 3, October 1996, pp. 143-156.

Programmer's Heaven. http://www.programmersheaven.com/c/userpoll/Poll_archive.htm?PollID=24 (Visited on: November 25, 2003)

Pujol, J.M., Sangüesa, R., and Delgado, J.: "Extracting reputation in multi agent systems by means of social network topology," *Proceedings of the 1st International Joint Conference on Autonomous Agents and Multiagent Systems*, pp. 467–474, ACM Press, July 2002.

Renesse, R.V, Minsky, Y., and Hayden, M.: "A gossip-style failure detection service," *Proceedings of Middleware*, 1998.

Ripeanu, M. and Iamnitchi, A.: Bloom filters short tutorial (unpublished), 2001.

Rowstron, A. and Druschel, P.: "Pastry: Scalable, distributed object location and routing for largescale peer-to-peer systems," *Proceedings of the 18th IFIP/ACM International Conference on Distributed Systems Platforms*, 2001, pp. 329-350.

Scott, J.: *Social network analysis: a handbook*, SAGE Publications, 1991.

Steyvers, M. and Tenenbaum, J.B.: "The large-scale structure of semantic networks: statistical analyses and a model of semantic growth" (submitted to *25th Annual Meeting of the Cognitive Science Society*).

Stoica, I., Morris, R., Karger, D., Kaashoek, F., and Balakrishnan, H.: "Chord: A scalable peer-to-peer lookup service for internet applications," *Proceedings of the ACM SIGCOMM*, 2001, pp.149-160.

Strogatz, S.H.: "Exploring complex networks," *Nature*, vol. 410, London, 2001, pp. 268-276.

Waldman, M., Rubin, A.D., and Cranor, L. F. Publius: "A robust, tamper-evident, censorship-resistant, web publishing system," *Proceedings of the 9th USENIX Security Symposium*, 2000, pp. 59–72.

Watts, D. and Strogatz, S.: "Collective dynamics of "small-world" networks," *Canadian Journal of Mathematics*, vol. 8, no. 3, 1956, pp. 399-404.

White, H.D. and McCain, K.W.: "Bibliometrics," *Annual Review Information Science and Technology*, Elsevier, 1989, pp. 119-186.

Yahoo Chat. http://chat.yahoo.com/

Yahoo Groups. http://groups.yahoo.com/

Yang, B. and Garcia-Molina, H.: "Efficient search in peer-to-peer networks," *International Conference on Distributed Computing Systems*, Vienna, Austria, 2002.

Yolum, P. and Singh, M.P.: "Emergent properties of referral systems," *Proceedings of the 2nd International Joint Conference on Autonomous Agents and MultiAgent Systems*, ACM Press, July 2003.

Yu, B. and Singh, M.P.: "A social mechanism of reputation management in electronic communities," *Proceedings of the 4th International Workshop on Cooperative Information Agents*, Klusch, M., Kerschberg, L.(Eds.), Lecture Notes in Computer Science, vol. 1860, Springer, 2000.

APPENDIX A

LIST OF PROGRAMMING LANGUAGE GROUPS ON USENET

1 comp.lang.ada
2 comp.lang.apl
3 comp.lang.asm.x86
4 comp.lang.asm370
5 comp.lang.awk
6 comp.lang.beta
7 comp.lang.c.moderated
8 comp.lang.clarion
9 comp.lang.clipper.visual-objects
10 comp.lang.clos
11 comp.lang.clu
12 comp.lang.cobol
13 comp.lang.dylan
14 comp.lang.eiffel
15 comp.lang.forth.mac
16 comp.lang.fortran
17 comp.lang.functional
18 comp.lang.basic.misc
19 comp.lang.pascal.mac
20 comp.lang.c++.leda
21 comp.lang.pascal.borland
22 comp.lang.pascal.delphi.advocacy
23 comp.lang.pascal.delphi.announce
24 comp.lang.pascal.delphi.components.misc
25 comp.lang.pascal.delphi.components.usage
26 comp.lang.perl.announce
27 comp.lang.perl.misc
28 comp.lang.perl.moderated
29 comp.lang.pop
30 comp.lang.postscript
31 comp.lang.prograph
32 comp.lang.prolog
33 comp.lang.rexx
34 comp.lang.ruby
35 comp.lang.sather
36 comp.lang.python.announce
37 comp.lang.scheme.c
38 comp.lang.mumps
39 comp.lang.oberon
40 comp.lang.smalltalk.advocacy
41 comp.lang.tcl.announce
42 comp.lang.visual.basic
43 comp.lang.basic.realbasic
44 comp.lang.basic.visual.3rdparty
45 comp.lang.basic.visual.announce
46 comp.lang.c++.moderated
47 comp.lang.java.help
48 comp.lang.java.javascript
49 comp.lang.java.machine
50 comp.lang.java.misc
51 comp.lang.java.programmer
52 comp.lang.java.security
53 comp.lang.java.setup
54 comp.lang.java.softwaretools
55 comp.lang.java.tech
56 comp.lang.lisp.x
57 comp.lang.ml
58 comp.lang.modula2
59 comp.lang.modula3
60 comp.lang.java.3d
61 comp.lang.java.advocacy
62 comp.lang.atUNIV
63 comp.lang.java.api
64 comp.lang.java.beans
65 comp.lang.java.corba
66 comp.lang.java.databases
67 comp.lang.java.developer
68 comp.lang.java.gui
69 comp.lang.lisp.franz
70 comp.lang.lisp.mcl
71 comp.lang.icon
72 comp.lang.idl
73 comp.lang.idl-pvwave
74 comp.lang.javascript
75 comp.lang.labview
76 comp.lang.limbo

77 comp.lang.logo
78 comp.lang.misc
79 comp.lang.pascal.ansi-iso
80 comp.lang.objective-c
81 comp.lang.php
82 comp.lang.pl1
83 comp.lang.pascal.misc
84 comp.lang.pascal.delphi.databases
85 comp.lang.pascal.delphi.misc
86 comp.lang.pascal.delphi.components.writing
87 comp.lang.perl.modules
88 comp.lang.perl.tk
89 comp.lang.sigplan
90 comp.lang.verilog
91 comp.lang.vhdl
92 comp.lang.vrml
93 comp.lang.scheme.scsh
94 comp.lang.smalltalk.dolphin
95 comp.lang.basic.visual.database
96 comp.lang.basic.visual.misc

BIOGRAPHICAL SKETCH

Mujtaba Khambatti received an MCS from Arizona State University in 2000, and a BE (computer) from Pune Institute of Computer Technology, University of Pune (India) in 1999. While at Arizona State University, he has been involved in research within the areas of: Peer-to-Peer Communities, Computing Communities, Dynamic Coalitions, and Mobile Ad Hoc Networks.

Invited to join the Outstanding Student Honor Society in 2003 and the National Society of Collegiate Scholars in 2002, he has also won student scholarships to attend two international conferences. During his undergraduate education, he won 3 awards for his senior-year project and was also honored with a silver medal for academic achievement during his senior year.

At Arizona State University, Mujtaba has been the chair of the student chapter of the IEEE Computer Society, the founder of the Windows Interest Group, the founding president of the Engineering and Applied Sciences Graduate Student Association, and the founding member of the Graduate and Professional Student Association. During his student group activities, he co-chaired the IEEE CS Workshop on Research in Computer Science and chaired the 1st Symposium of Research in Engineering and Applied Science.

His research interests include peer-to-peer systems, distributed operating systems, mobile ad hoc networks, and security.