This is to certify that the
dissertation entitled

BUILDING RELIABLE APPLICATIONS IN OVERLAY NETWORKS

presented by

XIAOMEI LIU

has been accepted towards fulfillment
of the requirements for the

Ph.D. degree in Computer Science and Engineering

Major Professor's Signature

2/29/2008

Date

BUILDING RELIABLE APPLICATIONS IN OVERLAY NETWORKS

By

Xiaomei Liu

A DISSERTATION

Submitted to
Michigan State University
in partial fulfillment of the requirements
for the degree of

DOCTOR OF PHILOSOPHY

Computer Science and Engineering Department

2008

ABSTRACT

BUILDING RELIABLE APPLICATIONS IN OVERLAY NETWORKS

By

Xiaomei Liu

Overlay networks become increasingly popular due to the many benefits they can provide: easy deployment, self-organization, scalability, and robustness. However, some features of overlay networks such as openness, lack of admission control, and lack of membership management make it challenging to provide high quality and reliable overlay services. In this thesis, the requirements of reliable services including service availability, data/content authenticity, and user safety, are addressed. The problems that hinder the fulfillment of these requirements in overlay networks are identified. The solutions for solving these problems and building reliable overlay applications are presented.

Two problems that hinder the fulfillment of service availability are identified. The first one is the cut vertices that exist in the overlay topology. This is a common challenge for overlay applications. Cut vertices are topological "critical" nodes whose failure can disconnect the network and harm service availability of the overlay applications. Connection adjacent matrix (CAM), a fully distributed mechanism that can be applied in different overlay applications, is proposed to detect and neutralize the cut vertices. CAM can accurately locate and neutralize cut vertices, as well as offload

the traffic of cut vertices.

The second problem that hinders service availability is the response loss problem that prevents the peer-to-peer (P2P) file sharing systems from providing reliable response return service. Three techniques are proposed to overcome the problem and provide reliable response return services. With limited traffic overhead, all three techniques reduce response loss rate by more than 65% and are fully distributed. These techniques are also designed to be simple enough to deploy in existing P2P systems.

In order to enhance data authenticity, an investigation is deployed on the storage and spread of trust values in the P2P reputation system. A hierarchical reputation management system (hiREP) is proposed to guarantee the reliability of the service content so that the users can trust the content provided by the overlay application. hiREP adopts a hierarchical structure, with an onion based communication mechanism and a public key distribution system that requires no third party certificate authority. hiREP can effectively evaluate the trust values and efficiently manage the storage and the delivery of trust values.

In order to increase user safety, the mutual anonymity in overlay multicast systems is investigated. The anonymity in an overlay multicast system helps make the system a safer place for the participating users. This thesis defines the requirements of the anonymous multicast system, and proposes a mutual anonymous multicast (MAM) protocol that includes the design of a unicast mutual anonymity protocol, and construction and optimization of an anonymous multicast tree. MAM is self organized and completely distributed.

Among the identified problems and proposed solutions, cut vertex is a common issue in the overlay topology and CAM can be applied in different categories of applications, while each of other solutions can be applied in a certain category of applications.

To my parents

ACKNOWLEDGMENTS

Pursuing a Ph.D. is a hard and long journey. Many people have been of great assistance to me. Here I want to express my sincere gratitude to them. First and foremost, I would like to thank my advisor, Dr. Li Xiao, for her professional guidance, great support, and unending patience. It is she who shows me the path to academic research. I appreciate the numerous hours that she spent with me to discuss my research, comment my ideas, and improve my written and presentation skills. I am grateful for the opportunities she provided to communicate with other researchers and the freedom she granted to explore different areas of research. She made it possible for me to complete this work.

I would like to express my gratitude to Dr. Matt Mutka, Dr. Sandeep S. Kulkarni, and Dr. Mei Zhuang for sparing their precious time to serve on my dissertation committee. They gave many valuable comments, as well as important suggestions on extending and improving my research proposal.

I thank all faculty and staff in the computer science and engineering(CSE) department. I thank Dr. Lionel Ni and Dr. Abdol-Hossein Esfahanian for their encouragement when I first came to Michigan State University. I thank Kim Thompson for proofreading my papers.

I am grateful to my colleagues in the ELANS lab. Their friendship made my years in Michigan State University joyful and delightful. Special thanks to Yunhao Liu for his help on my research studies. He was a valuable co-author. I owe him many interesting discussions, the inspiration on my research topics, as well as many

valuable suggestions on my papers. I also thank for their generous help with my research and studies: Chen Wang, Zhengyun Zhuang, Yong Ding, Guokai Zeng, Pei Zhen, and Zhiwei Cen.

Last but not the least, I want to thank my parents and my brothers, for their unconditional love, support, and encouragement. I also want to thank my husband, Risheng, for his love and patience.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

Introduction

The rapid development of communication and computing technology makes it possible for applications to run on end nodes. These end nodes compose overlay networks, which are built on top of the underlying physical networks that include many other end nodes, routers, and servers. Overlay networks become increasingly popular and have drawn much attention from research communities due to their support to a large variety of the Internet applications, such as peer-to-peer (P2P) file sharing system [7, 9, 11, 100, 119], overlay multicast [19, 20, 28, 32, 33, 34], and overlay routing [18, 78, 91, 92]. These applications enjoy the many benefits provided by overlay networks including high flexibility, easy deployment, self-organization, load balancing, scalability, and rapid improving computing capacity of end systems.

1.1 Research Background: Overlay Networks

Loosely speaking, any network that is on top of other networks can be called an overlay network. The Internet started out as an "overlay" running on top of the public switched telecommunication network (PSTN) [35]. Over time, the Internet evolved into the principle platform for the global public communication infrastructure, and is spawning its own collection of "overlay" networks that are running on top of it [35].

Overlay networks today are generally referred to as application overlays, which are located in the application layer of the OSI network layer model. This thesis focuses on the application overlays, which are composed by the end nodes. The terms of *overlay*, *overlay network*, and *application overlay* are used interchangeably in the remainder of the thesis to refer to application overlays composed by end nodes.

Overlay networks are composed of overlay nodes and overlay links. They interface with users and construct a user level network topology on top of an existing network infrastructure. An example of the overlay network structure is shown in Figure 1.1. The overlay network uses a subset of end nodes and links of its underlying networks. These end nodes also function as the overlay nodes in the overlay network. The overlay links can be viewed as paths including one or multiple physical links in the physical networks. Compared with the underlying physical networks, there is no specific"router" in an overlay network. Instead, traffic is forwarded from one overlay node to another directly via the overlay link between these two nodes. At the underlying physical network, the traffic that travels along an overlay link between two overlay nodes follows the actual physical links that form that overlay link.

Overlay networks provide many benefits that render the fast development of the applications. Applications atop of overlay networks are running on the end nodes instead of centralized servers. This removes the needs to obtain the administration permission for deploying the applications. In addition, it is not necessary to consider the requirement from underlying routers in the deployment of overlay applications, since the underlying network infrastructure is transparent to overlay networks. As a result, the scale of overlay networks has increased rapidly in recent years. For example, the scale of peer-to-peer systems has been increased from thousands of nodes in 2000 to millions of nodes now [6].

Overlay networks add new dimensions to the Internet's usability. For instance, the inability of the Internet to support multicast is patent. With the emergence of

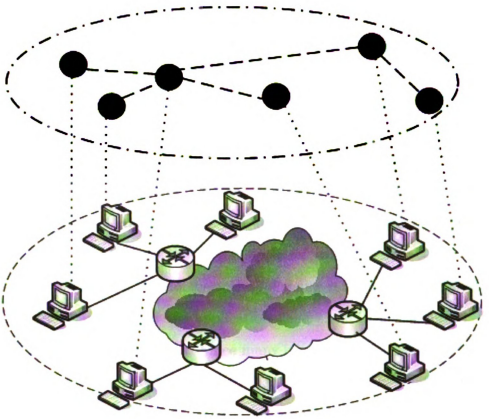


Figure 1.1. The overlay network and its underlying network

the overlay networks, multicast services on overlay networks have been proposed and implemented, which fulfills the demand of a variety of applications such as video conferencing, Internet-based education, NASA TV, software updates, etc. [19, 28, 27, 33, 34, 72, 95, 105, 111, 32].

Overlay networks distribute the cost of running the applications across the nodes in the system and utilize the resource of end nodes. For instance, file sharing peer-to-peer systems distribute the main cost of sharing - bandwidth and storage- across all peers in the system. By distributing the system operational cost across the end nodes in the system, overlay networks are able to provide support for services otherwise provided by the expensive centralized servers. The system can thus scale without using many expensive servers.

1.2 Motivation and Challenges

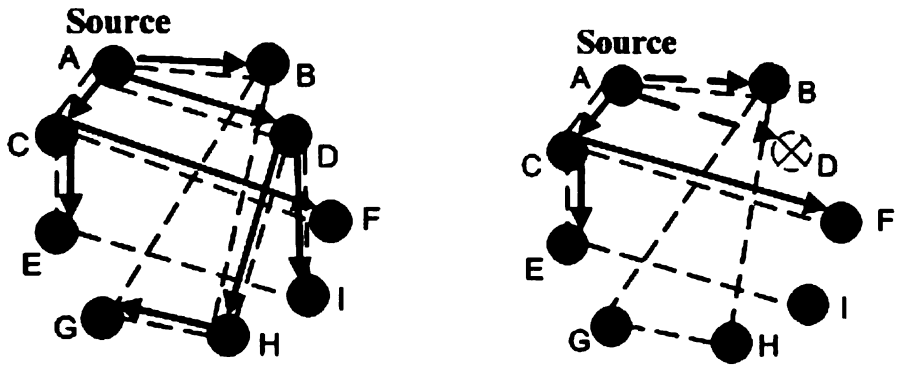
Composed by end nodes exclusively, overlay networks are self-organized and decentralized. Nodes in the system connect with each other based on local knowledge. This removes the need for central servers to collect the network information and organize the nodes in the system. It also makes overlay networks robust to the single point of failure and avoid traffic bottlenecks caused by the servers. On the other hand, the inherent feature of end nodes and the way they connect with each other create challenges for the reliability of the overlay applications.

A reliable application should meet the following requirements: first and most important, the availability of the application itself should be guaranteed. This is applied to all overlay applications. Since many overlay applications provide content service including video, voice, and data to their users, the authenticity of the service content should be guaranteed to meet the expectation of users. In other words, the service provided by the overlay applications should be trustworthy. Moreover, user safety should be provided to make the users feel safe and comfortable to use the applications.

1.2.1 Service Availability

The basic and essential requirement for the reliability is to guarantee the availability of the service. The service provided by the application should be always available for the users, even when the network failure occurs. In other words, the service should be "fault tolerant" for the network failures. The requirement of availability is challenged in the overlay networks by the dynamics and the self-organization of the end nodes.

Since the overlay is loosely connected by the self-organized end nodes, there is no centralized management in the overlay to control the network topology. The end nodes choose neighbors they connect with either randomly or via some locally defined



(a) A distribution tree of overlay live streaming

(b) Outgoing streaming disrupted when nodes leave the system

Figure 1.2. Node transience in the overlay live streaming

algorithms. This makes the presence of topological "critical" nodes in the networks unavoidable. An example of this is the high-degree nodes that connect with a large amount of other nodes in the system. S. Saroiu et al. [90] show a small amount of high-degree nodes can efficiently "shatter" the overlay network, which makes the network highly vulnerable in the face of well-constructed, targeted attacks.

Compared with servers, the end systems that compose an overlay network are unreliable. They come and go very frequently [79, 90, 93]. For instance, previous studies show that a peer's lifetime in the P2P system varies from less than 10 minutes in FastTrack [90, 93] to 60 minutes in Gnutella and Napster. As a result, overlay networks are highly dynamic.

The dynamics of the system caused by the end nodes may lead to data/traffic loss and the interruption of the service for the users. This makes the system "unreliable" and induces extra cost to recover the loss data or resume the interrupted service. For example, the node transience may cause the loss of stream packets and downgrade the quality of the overlay in the multicast overlay live streaming [79]. Figure 1.2 shows the stream disruption caused by the node transience. The multicast tree for the live streaming with the source node A is shown in Figure 1.2(a). When node D leaves the

system, its downstream nodes G, H, and I are disconnected. As a result, they will experience streaming disruption before they are reconnected to the streaming tree. This is shown in Figure 1.2(b).

1.2.2 Authenticity and Service Content Reliability

The requirement of authenticity is about the service content including voice, images, and data provided by the applications. The service provided by the application should be trustworthy for the users. For instance, in the file sharing systems or the content distributing systems, the system should guarantee the authentication of the data it provides to its users. The fulfillment of this requirement of the reliability is challenged by the open feature of overlay networks: devoid of admission and membership control.

There is no admission control in overlay networks. Anyone can freely join or leave the system. This open feature of overlay networks invites the malicious behavior of the attackers. For instance, malicious nodes may return invalid responses to the search request sent out by the users in the P2P file sharing system or provide fake files in the file downloading process. The music industry has utilized this feature to prevent illegal downloading of copyrighted music. A large amount of "polluted" data have been injected into popular Internet file sharing systems such as KaZaA, and are claimed as popular song copies. It is expected that the experience of being "polluted" during the downloading process will discourage users from downloading songs from these systems. This suggests that attackers can behave in a similar way to distribute malfunctioning data including trojan horses or virus. Attackers can also issue DDOS attacks in the system by issuing large amounts of fake search requests.

Devoid of centralized servers, there is generally very limited or no membership management in the overlay networks, which makes it hard to keep an effective node identification system in the overlay networks to identify each node with a unique and fixed node "ID". As a result, it is difficult to build an effective trust mechanism,

which generally requires that each member in the system has a relative fixed and unique identification that can be used to keep track of the reputation of the member in overlay networks. The lack of the fixed node "ID" also makes it possible for a node to do things in the name of other nodes. The attackers are thus able to launch attacks with multiple identifications, also known as the sybil attack [40]. In addition, the lack of "real" node IDs and the open feature of the overlay networks make it difficult to identify and track the trace of attackers when attacks are launched in the system.

1.2.3 User Safety and Other Considerations

Besides guaranteeing the availability and trustworthy content of the service, a reliable overlay application should also protect the information of the users who use the system. Their private information or any other information that they do not want to share should be hidden from others. Devoid of admission control and membership management, anyone can join the system and may track the information of the system as well as other users. In order to protect the user information, user anonymity should be provided for the overlay applications. This is especially important for some critical services such as military and emergency applications, where strategic information and critical updates need to be hidden from external observers. User anonymity can also promote censorship resistance, freedom of speech without the fear of persecution, and privacy protection.

We summarize the requirements of reliability and the features of the overlay networks that create challenges on these requirements in Table 1.1. Table 1.1 shows that the successful implementation of reliable overlay applications need overcome the challenges brought by the overlay features such as the randomness of end nodes in forming overlay topology, high dynamics, lack of admission control and membership management. At the same time, reliable applications should bear the performance requirement of the existing system. The fulfillment of one requirement of reliabil-

Table 1.1. Reliability Requirements and Challenges

Reliability requirement	**Overlay features that create challenges**	**Required applications/users**
Service availability	Randomness in forming overlay topology; dynamics of end systems	All overlay applications
Content authenticity	Devoid of admission control and membership management; devoid of central servers	Applications provide content service
User safety(anonymity)	Devoid of admission control and membership management	Critical applications; Applications or users that need to hide information from third parties

ity should not hinder the fulfillment of another requirement of the reliability in the overlay application.

The usage of overlay networks comes with a price in the network performance due to the inefficient topology and the demand on processing the messages in the application level by the overlay nodes. Many efforts have been made to improve the performance and reduce the operating cost of the overlay networks. Therefore, the reliable applications constructed in this context should not cause large extra cost in the overlay network.

In practice, one requirement of the reliability in the system may conflict with another requirement of the reliability. For instance, constructing reputation systems is a common methodology to help guarantee the reliability of the service content. However, reputation systems need the information of individual nodes to generate and evaluate the "trust" of each node. This conflicts with the requirement for the user safety, which tries to protect the information of the individual node.

1.3 Solutions and Contributions

In order to improve reliability of overlay applications, we identify the specific problems that hinder overlay applications from meeting the requirements of reliability: service availability, data authenticity, and user safety. Problems in building reliable overlay applications can be divided into two categories. The first category of problems are the problems that exist in all overlay applications. The solution for such problems are thus a universal solution that can be applied for all overlay applications. Another category of problems are problems that are caused by the requirement/operation of a certain type of overlay applications. The solutions for this category of problems will thus benefit that type of applications.

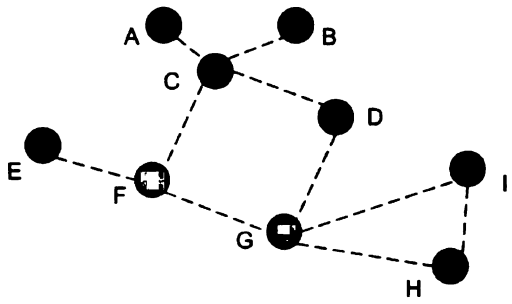
For the first category of problem, we investigate the cut vertices in the overlay topology, which exist in different overlay applications and hinder the service availability of the applications. We propose the connection adjacent matrix (CAM) algorithm to neutralize the cut vertices and optimize the overlay topology [67]. CAM is a universal solution that enhance service availability of different types of the overlay applications. For the second category of problems, we first identify the response loss problem that also hinders service availability. The response loss problem is caused by the dynamics of P2P file sharing systems. A series of reliable response return mechanisms are proposed to resolve the response loss problem and improve service availability [63, 64]. We then investigate the issues in the P2P reputation system to improve data authenticity. We propose a hierarchical reputation (hiREP) system to effectively store and spread the trust values in the P2P reputation system [65]. hiREP helps improve the authenticity and content reliability of the files shared among the users. Further more, we explore the anonymity requirements in the overlay multicast system as an effort on improving user safety. We define the mutual anonymity in the overlay multicast and propose a mutual anonymous multicast (MAM) system to protect the user information and provide the user safety [109, 110].

1.3.1 CAM: Improve Service Availability by Removing Cut Vertices in the Overlay Topology

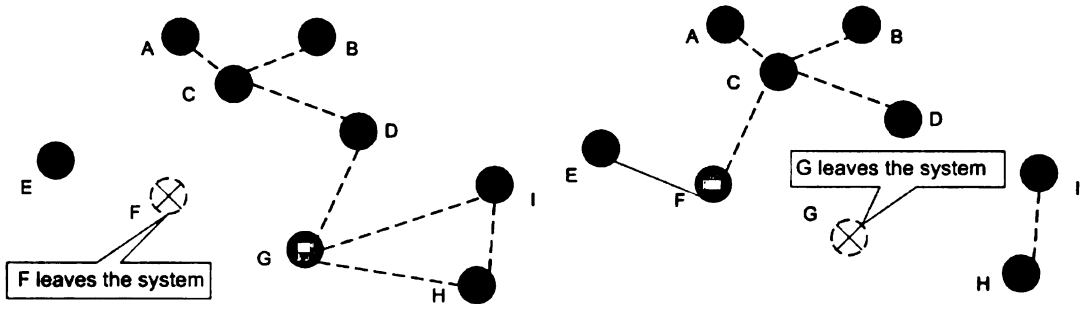
Overlay nodes are highly self-organized. They make connections either with some randomly selected nodes or via locally defined algorithms. In both cases, there is no centralized control to manage the network topology and thus it is hard to avoid of the presence of topological "critical" nodes. Compared with common nodes, their failures are more likely to lead to network failures and harm service availability.

In order to improve service availability, it is important to identify and "remove" the critical nodes in the overlay topology. We investigate one category of "critical" nodes: **cut vertices**. Cut vertices are the nodes that will disconnect the network when they leave. In Figure 1.3(a), node F and G are cut vertices. The network is disconnected when the cut vertices leave the system. This is shown in Figure 1.3(b): Node E is disconnected from other nodes in the system when node F leaves; Node H and J are disconnected with other nodes when node G leaves.

In order to limit the presence of cut vertices, we propose CAM algorithm to detect and neutralize the cut vertices [67]. Traditional methods of detecting cut vertices are centralized and cannot be used in large scaled overlay networks, which are highly dynamic as well. CAM is a fully distributed mechanism that detects the cut vertices before they fail and neutralizes them into normal overlay nodes. CAM not only minimizes the possibility of network decomposition on the cut vertex failure but also offloads the traffic that is handled by the cut vertices. Instead of collecting the overall network topology information, CAM renders each node in the system send out messages to its neighbors and collect local/partial topology information of the nearby nodes. Based on the received information, CAM constructs CAM graphs and figures out whether a node is a cut vertex or not. If the node is a cut vertex, CAM will proactively neutralize the node into a non-cut vertex.



(a) An overlay network with cut vertices



(b) Network is disconnected when cut vertices are gone

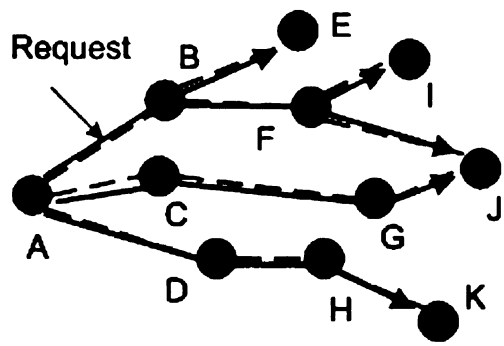
Figure 1.3. Overlay cut vertices that disconnect the network

1.3.2 Reliable Response Return Mechanisms to Improve Service Availability in P2P File Sharing Systems

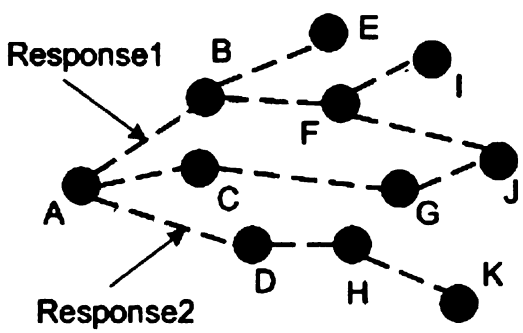
The dynamics of end systems causes the **response loss problem** and hinders service availability in the unstructured P2P file sharing system, which is most commonly used in today's Internet [30, 63, 69, 70, 64]. In an unstructured P2P file sharing system, file locating is generally based on the flooding search mechanism: each node makes duplicate copies of a request it receives and then broadcasts to all its directly connected neighbors except the one that delivered the incoming request. If the node has the requested file, it sends a response back to the requestor along the incoming path of the request. Since the response return process in a P2P system is not flooding based, a response will be lost if any one node or link in the corresponding incoming path fails. Since file locating (and the returned responses) is an important service provided by unstructured P2P file sharing systems, the loss of responses downgrades service availability of the systems.

Figure 1.4 shows response loss problem in the P2P file sharing system. In Figure 1.4(a), requestor A broadcasts its request for files to the system. Upon receiving the request, responders send responses back to requestor following the path that the request is delivered to the responders. This is shown in Figure 1.4(b): Response 1 via path $I \rightarrow F \rightarrow B \rightarrow A$ and Response 2 via path $K \rightarrow H \rightarrow D \rightarrow A$. In Figure 1.4(c), node B leaves the system and as a result, response 1 is lost.

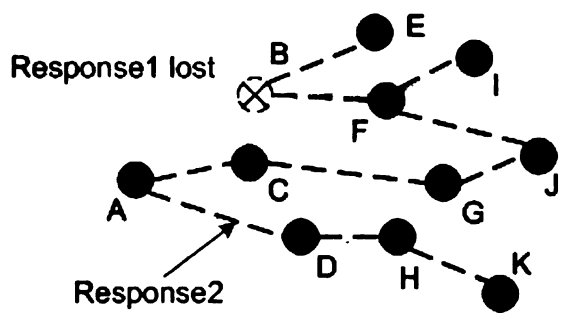
We investigate the response loss problem in the unstructured P2P system and our investigation shows that the response loss rate can be as high as 35% due to the high dynamics of the system. Three mechanisms are proposed to improve the availability in P2P overlay file sharing systems by solving the response loss problem induced by the end system: redundant response delivery (RRD), adaptive response delivery (ARD), and extended adaptive response delivery (e-ARD) [63, 64]. All three techniques handle the response loss problem regardless of node failure or node



(a) Requestor broadcast requests in P2P system



(b) Responses return to requestor in normal case



(c) Response is lost due to node transience

Figure 1.4. Response loss due to the overlay node transience

departure. In RRD, the responder proactively sends back duplicate copies of the same response via different paths to avoid response loss. In ARD, peers in a response path automatically choose alternative paths in case of node failure or departure. The e-ARD mechanism extends the ARD mechanism with the introduction of backup response delivery agents (bRDA) to avoid response loss in P2P systems with limited or no broadcasting searching mechanisms.

1.3.3 hiREP: Enhance Data Authenticity in P2P Systems

The open feature of peer-to-peer system invites the spread of the malfunctioning data. Additional reputation systems are thus constructed to guarantee data authenticity. In a reputation system, each node is associated with a trust value, which is computed based on its performance in the system. Devoid of the centralized control, it is challenging to store and spread trust values securely and efficiently in the P2P reputation systems. Previously proposed P2P reputation systems adopt flooding based polling mechanisms for trust value requesting process, which need to inquire into each node in the system [36, 96]. The polling mechanisms create heavy traffic, do not guarantee voter anonymity, and make it hard for peers to filter out the fake trust values.

In order to construct an effective and efficient P2P reputation system and provide "content reliability" to users, the hiREP system adopts a hierarchical, low cost method to store and distribute the trust values [65]. hiREP focuses on constructing a trusted reputation agent group for each node to handle the trust value information. Instead of managing the trust values by centralized servers or each individual node, each node keeps a group of reputation agents that it trusts to store and manage the trust values. Note here a reputation agent can be trusted by multiple nodes. A node only inquires of its trusted reputation agents the trust information. The onion routing is adopted for the communication between a peer and its trusted agents to protect the information of the nodes. At the same time, each node is assigned a unique nodeID

together with the usage of public key systems to provide authenticity of the votes for the trust value information.

1.3.4 MAM: Provide User Safety in Overlay Multicast Systems

Besides service availability and data authenticity, we extend our study in building reliable overlay applications to user safety. We investigate the anonymity issue in the overlay multicast. We define the mutual anonymity to reflect the unique requirements of implementing anonymity and protecting user information effectively in the overlay multicast system. Compared with unicast, anonymity in multicast needs to provide not only sender and receiver anonymity, but also the group anonymity. In addition, the information of a sender needs to be protected from a group of receivers instead of one receiver. The information of a receiver needs to be protected from other receivers besides sender. In order to guarantee user safety, MAM is proposed to provide mutual anonymity including the anonymity of the sender, the receiver, the group and the multicast tree [109, 110]. MAM protocol adopts the design of a unicast mutual anonymity protocol, as well as the construction and optimization of an anonymous multicast tree. MAM is self organized and completely distributed.

We summarize the solutions presented in this thesis in Table 1.2. All solutions help to build reliable overlay applications via enhancing the different aspects of the reliability. CAM and response reliable return mechanisms of RRD, ARD, and e-ARD help to enhance service availability. hi-REP contributes to the authenticity of the service content and makes the P2P file sharing systems more trustworthy. MAM provides user safety by implementing mutual anonymity in overlay multicast systems. CAM optimizes the overlay topology and is a universal solution that can be applied in different categories of overlay applications. Each of the other solutions can be applied to a certain type of applications.

Table 1.2. Solutions in Building Reliable Overlay Applications

Solutions	**Problem or issues addressed**	**Enhancement in reliability**	**Overlay applications applied**
CAM	Cut vertex	Service availability	All overlay applications
RRD, ARD, and e-ARD	Response return loss	Service availability	P2P file sharing system
hiREP	Trust value management	Content authenticity	P2P file sharing system
MAM	Mutual anonymity	User safety	Overlay multicast

1.3.5 Contributions

The contributions of this thesis are as follows:

1. Address the challenges for building reliable overlay applications. Identify both the common problem and problems that exist in certain types of overlay applications: the cut vertices in overlay topology, the response loss problem in P2P overlay file sharing systems, the effective management of overlay P2P reputation systems, and the mutual anonymity in overlay multicast. The thesis also investigates the influence of these problems on overlay applications.

2. Propose solutions for the challenges and problems on building reliable overlay applications. These solutions helps improve the reliability of overlay applications in the following aspects: CAM works as a universal solution to enhance service availability for different overlay applications by detecting and neutralizing the critical nodes of cut vertices. The RRD, ARD, and eARD mechanisms are proposed to alleviate the response loss problem and improve service availability (the number of returned responses) for the P2P file sharing system. hiREP is introduced as an effective reputation management system to address data authenticity of P2P overlay file sharing systems and help provide trustworthy

service content. MAM is proposed to implement the mutual anonymity in overlay multicast, which protects user safety (the user information and membership information) of the overlay multicast groups.

3. Model the different overlay applications, including the unstructured P2P file sharing system, the P2P reputation system, and the overlay multicast system. Deploy extended simulations based both on real world traces and overlay application models to evaluate the effectiveness and performance of the proposed approaches.

1.4 Thesis Organization

The remainder of this thesis is organized as follows. A literature review of related work is presented in Chapter 2. In Chapter 3, we present the CAM algorithm and the proof of its correctness. Chapter 3 also includes the trace driven simulation that investigates the influence of the cut vertices as well as the effectiveness of CAM based on the real world traces.

Chapter 4 defines and discusses the response loss problem. It then introduces the RRD, ARD, and eARD mechanisms that are proposed to increase the reliability of P2P overlay file sharing system by reducing the response loss. The chapter also provides simulations to evaluate the performance of the three techniques, as well as a thorough discussion on the pros and cons of the three mechanisms.

hiREP is discussed in Chapter 5, starting with the introduction of a general model of the P2P reputation system. The major components, the reputation agent community, the nodeID and the public key system, as well as how they work together to construct an efficient and effective reputation system, are then introduced. The effectiveness of hiREP is then checked under different attack models and by simulations on top of a large-scale P2P network model.

Chapter 6 discusses the mutual anonymity in the overlay multicast system and presents the solution MAM. The requirements for the mutual anonymity in the overlay multicast system are discussed and defined. The protocol design of MAM is introduced in detail. The chapter also includes the analysis of the cost and the network connection latency, as well as the the anonymous degree under different attack models for the overlay multicast system with MAM. A simulation to evaluate the performance of MAM based on the model of overlay multicast system is also provided.

Chapter 7 concludes the thesis and discusses the future direction in building reliable applications in overlay networks.

CHAPTER 2

Related Work

This chapter includes a literature review of the research studies that are related with the proposed solutions on building reliable overlay applications. The first two sections reviews the studies related to our solutions for improving service availability: The next section discusses the research studies on optimizing overlay topology that related to cut vertex and CAM. Section 2.2 presents the related work for the solutions of the reliable response return system, which enhance service availability in the highly dynamic P2P systems. Section 2.3 checks the related work on the reputation mechanisms, which ensure data authenticity and help overlay applications to provide trustworthy service content. The related work on user anonymity that protects user safety of the overlay multicast applications is listed in Section 2.4.

2.1 Topology Optimization of the Overlay Networks

Devoid of centralized servers and administration, overlay networks use ad hoc methods to construct their topology: overlay nodes connect with each other either randomly or based on local algorithms without the knowledge of overall network information.

This makes it inevitable the appearance of topological "critical node". As we mentioned in Section 1.2, the failure of these topological "critical" nodes can cause the network failure and hinder service availability of overlay applications. Therefore, optimizing overlay topology by reducing the number of the topological "critical" nodes can improve service availability for overlay applications.

One category of topological "critical node" is cut vertex. Cut vertex is an important concept that has been introduced in graph theory and studied extensively. Existing algorithms used to detect cut vertices in graph theory need to collect overall topology information of the network and construct a depth first search (DFS) tree including all network nodes. One category of such algorithms detect cut vertices by checking the sub-tree of each node in the DFS tree [24]. These algorithms are generally composed of the following operations. First, construct a DFS tree from any node. Then, for each node v in the DFS tree except the root, check the neighbors that v's descendants connect with. If none of the neighbors of v's descendants is v's ancestor, v is a cut vertex. The root is a cut vertex if and only if it has more than one neighbor.

Another approach is to group the graph nodes into several bi-connected components [24]. A bi-connected component is a component that cannot be disconnected by deleting any vertex in it. For any two vertices in a bi-connected component, there exist at least two disjointed paths between them. All edge vertices that connect any two bi-connected components are thus cut vertices. In the algorithms that adopt the second approach, distinguishing disjointed paths and constructing the bi-connected components require traversal across the network. For instance, Sharir's algorithm of finding the bi-connected components needs to build a DFS tree involving all nodes in the graph [24].

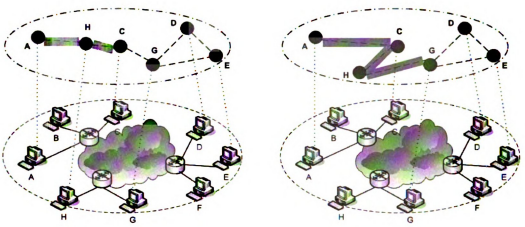
Both categories of the traditional cut vertex detection require the global topology information of the network. Nevertheless, it is extremely difficult, if not impossible,

to obtain such topology information in today's large scale, fully decentralized, and self-governing overlay network systems.

Besides cut vertices, high degree nodes can also be "critical" with respect to network topology. P. Keyani et al. [55] proposed a mechanism to modify the P2P overlay network topology to reduce the number of high degree nodes. Compared with cut vertices, the detection of high degree nodes is trivial. Therefore, the authors adopted a reactive methodology. They proposed to detect the external attack by monitoring the rates at which the first and second degree neighbors of the end nodes leave the system. When the failure rate of seconder degree neighbors is 50% higher than the first degree neighbors, an attack is detected. The end nodes will form a new network topology of exponential topology, where each node has roughly same number of neighbors. The homogeneous nature of the exponential topology makes the network less vulnerable than a network with the scale free topology such as power law topology in the overlay P2P systems. In order to form exponential topology, the end nodes in the system need to connect to random neighbors with no preference. This is implemented by the end nodes via sending random discovery ping(RDP) messages in a deep first search (DFS) way to their neighbors. The method focuses on external attacks. In addition, the long convergence time of this method makes it impractical in a large scale overlay network.

Node failures caused by network dynamics are widely noticed by the research community. Most recently proposed algorithms for overlay networks have begun to address the problems in the system design [63, 69, 117, 13]. However, these algorithms treat all nodes the same way and do not pay special attention to "critical" nodes. The failure of critical nodes can create more serious problems in the network than what these algorithms are able to handle.

Besides increasing overlay application reliability, another direction on optimizing overlay topology is to match the overlay topology with the underlying physical topol-



(a) Inefficient Overlay Topology (b) Efficient Overlay Topology

Figure 2.1. Mismatch of the overlay topology

ogy. As we discussed before, the overlay networks use a subset of end nodes and links of the underlying physical network. The underlying network structure is transparent for the overlay networks. The underlying physical network information is not provided to the overlay networks. The devoid of underlying physical network information makes it difficult for an overlay network to build its network according to the underlying physical network topology, which induces the inefficient overlay network topology. Figure 2.1(a) shows the inefficient overlay topology caused by topology mismatch. It is obviously that the overlay links between *A-H-C-G* are inefficient, where *A* and *C* are in the same subnet, while H and G are in another subnet. All traffic from *A* to *C* needs to pass through *H* first, which causes longer delay in the overlay link and unnecessary traffic.

The inefficient topology of overlay networks downgrades the performance of the overlay applications and increases the cost of such applications. The problem that the overlay network topology does not match its underlying physical network is called topology mismatch problem. Efforts have been made in both our previous research and other research studies to solve the problem and optimizing the overlay topology [69, 71, 118, 59, 30, 84]. Although these methods helps improve the overlay net-

work performance by reducing traffic cost and overlay connection delay, they do not contribute to improving overlay application availability when node failure happens.

2.2 Reliable Response Return in the P2P Systems

P2P file sharing systems can be divided into three categories according to how the files and nodes are organized in the system: centralized, decentralized structured (simplified as structured), and decentralized unstructured (simplified as unstructured) [73].

The earliest model adopted by P2P system is centralized architecture, such as Napster [9]. In this model, central index servers are used to maintain a directory of shared files stored on peers so that a peer can search for the whereabouts of a desired content from an index server. However, this architecture creates a single point of failure, and its centralized nature of the service makes systems vulnerable to denial of service attacks [47]. Today, the centralized model is out of date with the crack down of Napster.

Decentralized P2P systems have the advantages of eliminating reliance on central servers and providing greater freedom for participating users to exchange information and services directly between each other. The structured P2P systems are generally based on the distributed hash table (DHT), and the file placement is tightly controlled with the network topology to make the subsequent searches easily satisfied, such as Chord [100], Pastry [88], Tapestry [119], and CAN [83].

In unstructured P2P systems, files are randomly distributed among nodes and, consequently there is no correlation between the file placement and the network topology. File locating is generally based on the flooding search mechanism: each node makes duplicate copies of a request it receives and then broadcasts to all its directly connected neighbors except the one that delivered the incoming request. If the node

has the requested file, it sends a response back to the requestor along the incoming path of the request. The flooding search mechanism is simple and robust to the node failure: the request will spread over the system even if a few nodes in the system fail. However, since the response return process in a P2P system is not flooding based, a response will be lost if any one node in the corresponding incoming path fails.

The dynamics of overlay networks has been noticed by research communities recently. Many solutions proposed for overlay networks now take the system dynamics of the overlay network into their design consideration [22, 31, 34, 32, 95, 115].

However, for unstructured P2P file sharing systems, most researchers consider that flooding search mechanisms are robust against the node failure induced by the end system transience. The node failure during the query process in such systems is thus hardly accepted enough attention, although it is well known that P2P systems is highly dynamic. The robustness against node failure is obvious if a flooding mechanism is used in the entire search process. However, in order to reduce the search cost, the flooding mechanism is not adopted in the response process. This leads to the loss of responses. As the file locating as well as the file information included in the returned responses is an important service provided by P2P file sharing systems, the loss of responses impedes service availability of the P2P file sharing systems.

Y. Chawathe et. al [30] also noticed the response loss problem in unstructured P2P systems and expected that their proposed optimization technique, which targeted to optimize the network topology by periodically adjusting the neighbor connections of each peer, may make the case worse. They proposed two methods to remedy this problem: the first is to stop forwarding the query earlier than to stop forwarding the response, and the second is to set a time threshold and reissue the query when it can not be satisfied within the time limit.

While the first method may help to make up the response loss caused by their proposed technique of adjusting overlay topology, it cannot help the response loss

problem caused by inherent P2P system oscillation because the first technique needs the cooperation of each peer. The second technique may reduce the unnecessary query reissuing traffic in the case of slow response, but it provides no benefit in reducing traffic and potential extra response delay caused by the lost responses. We also noticed the problem in our previous work [69]. We realize that, with more and more optimization techniques being used to limit the query traffic, this problem will become more serious. This also motivates us to further investigate the problem.

Research has been done to provide fault tolerance in structured P2P systems [15, 83, 100, 106, 119, 120]. However, in structured P2P systems, fault tolerance needs to be provided in the entire search process, while, in the unstructured P2P systems, the fault tolerance only need to be provided to the response messages. Therefore, the solutions should require no modifications to the query request process and should be seamlessly combined with existing query mechanisms.

In structured P2P systems, each node is tightly controlled and the node information can be used to construct fault tolerance strategies. However, in unstructured P2P systems, the only information a peer has is on itself and its neighbors. For example, each node in a Chord system [100] maintains a "successor-list" of its nearest successors to provide fault tolerance during node failure. In order to do this, all nodes need to be assigned a hashed key and orderly organized. This is impossible in unstructured P2P systems where nodes are completely autonomous and connections between nodes are formed randomly.

Message delivery in an ad hoc network may also fail due to node movement. However, routing mechanisms used in an ad hoc network [53, 80, 81] cannot be used to solve the problem in unstructured P2P systems. Unlike an ad hoc network, where the source node and destination node know each other; there is the requirement of anonymity during the query process in an unstructured P2P system. In addition, the delivery algorithm in an ad hoc network generally adopts broadcasting in a wireless

network, which should be avoided in the response delivery process. To the best of our knowledge, no effective solution has been proposed to deal with the response loss problem incurred by the system itself in the unstructured P2P system.

Mechanisms have been proposed to provide fault tolerance for unstructured P2P systems against DoS/DDoS attacks [38, 41, 55, 68]. However, the response loss problem is induced by the system itself instead of outside attacks. Yet one can see from the rest of this thesis, the proposed mechanisms can be also combined with these approaches to reduce response loss under attacks.

2.3 Reputation Systems

Reputation systems are common methodologies being adopted in overlay applications to provide data authenticity. In most of these systems, a trust value is assigned to each of the end system/user. The authenticity of the data is then decided based on the trust of the data provider. Here we check a series of studies on the reputation systems.

Many reputation systems have been proposed for e-commerce society to provide trust between individuals who engage in online transactions without knowing each other [14, 39, 49, 74, 116]. However, these systems are either too complicated to implement on P2P peers or rely on some centralized servers to provide necessary information and/or organize the nodes of the system. For instance, the reputation systems in well known industrial environments such as eBay, Epinions, Amazon, are based on centralized architecture, where reputation feedbacks are solicited and deposited in a single repository, and controlled by a single organization [1, 4, 5, 39].

For structured P2P systems, the tightly controlled structure and the system information can be used to distribute the trust value. Therefore, most efforts focus on how to construct efficient trust value computing models.

Based on the assumption that few nodes are malicious nodes, Aberer and Depotovic [16] propose a binary trust model which utilizes only the complains for the peers that are involved in transactions to compute trust values. Researchers in Georgia Institute of Technology [113] show that computing peer's trust value only based on complains is not accurate enough and propose to compute trust value based on more metrics. They also import context factors to render general trust metric better fit in communities of different transactional or community-specific contexts. A feedback admission control mechanism is proposed to guarantee the effectiveness of reputation feedbacks by checking the legitimacy transactions [99]. In another research work, they extend their trust model and formalize the trust computation model with five trust metrics: Satisfaction feedback, number of transactions, feed-back credibility, transaction context factor and general trust metric [114].

EigenTrust tries to rule out the negative effects of malicious peers who malign the reputation of other peers [54]. It computes trust value based on both local experience of each peer and reputation feedbacks collected from other peers in the system. All aforementioned mechanisms utilize topology information and specific search/routing algorithm of the structured P2P system to distribute the trust value messages in the system.

Instead of computing reputation for peers, R. Morselli et al. [76] propose a mechanism to protect the authenticity of cached index with the signed digests of generated node sets. However, this method doesn't provide the guarantee for the authenticity of the file provider.

For unstructured P2P systems, it is hard and expensive to organize peers. Therefore, it is a challenge to store and spread trust values securely and efficiently in unstructured P2P systems.

Both TrustMe and P2PREP use flooding based mechanisms to transmit the trust values [36, 96]. Instead of storing trust values locally, TrustMe introduces trust-

holding agents (THA), which are assigned to each peer during its bootstrapping process by the bootstrap server, to store the trust values and protect the privacy of trust evaluation peers. After each transaction, a peer broadcasts transaction results about its transaction partner to the entire system, while only the THAs of the partner store the results. During the trust value query process, requestor broadcasts trust value query message to the entire system and only THAs of the potential provider send trust values back to the requestor.

P2PREP uses pure voting based mechanism to construct the reputation system. In P2PREP, the trust value of the peers is locally computed by and stored in their transaction partners. The requestor broadcasts trust value query messages to the entire system. Upon receiving the request, all of the peers that own the trust value of the potential provider return the trust value back to the requestor. In both systems, public key system is used to guarantee the authenticity of the trust value and broadcast communications are used to deliver trust values.

P2PREP and TrustMe guarantee the authenticity of trust votes, but they also create heavy traffic overhead due to the adaptation of flooding based trust value delivering system. P2PREP cannot guarantee the anonymity for the voters since it depends on the IP address of voters to verify the trust value, while TrustMe obtains anonymity in the cost of doubling the trust polling process traffic. In addition, trust holding agent assignments and key distributions for each peer add extra load for the bootstrap servers.

Besides P2PREP and TrustMe, M.Gupta et al. [52, 56, 57] propose a centralized reputation system on the top of unstructured P2P networks. This centralized system requires an extra reputation computation agent (RCA), which is responsible for the trust value maintenance for the entire system. As no protection is provided, RCA can be the target/victim of attackers. M. Kinateder et al. tries to use Chaum mix (a mechanism that is similar and stimulates the onion routing) in their proposed

UniTEC reputation system. However, the reputation request process in UniTEC is essentially a broadcasting based mechanism and a trusted third party is needed in UniTEC as the certificate authority.

2.4 Mutual Anonymity in Overlay Multicast Systems

In this section we review the related research studies of overlay mutual anonymity that enhance user safety in overlay multicast applications. The related work includes three categories of studies: (1) basic multicast (no anonymity) (2) anonymous unicast and (3) multicast anonymity.

2.4.1 Basic Multicast

Multicast services allow one host to send information to a large number of receivers. Multicast services are required by many applications including video conferencing, video-on-demand, multi-player games, as well as peer-to-peer file sharing. Originally, the multicast research focused on the network layer. However, no real multicast service has been provided at the network layer. The main issues in the network layer multicast are deployment and scalability issues. Recently, the focus has been moved up to the application layer with the work proposed in the overlay multicast [19, 28, 27, 33, 34, 72, 95, 105, 111, 32]. In the overlay multicast, end nodes participating the multicast application share responsibility of forwarding information to other nodes. It is well believed that overlay networks provide much flexibility in design and implementation, as well as enable quick deployment of the multicast functionality.

2.4.2 Anonymous Unicast

Some work has been reported on the overlay anonymous unicast [44, 112]. The essential techniques to achieve the unicast anonymity can be classified into the following categories: routing, addressing, layered encryption, and traffic covering.

In the routing approach, there can be either flooding, where the anonymity is achieved by broadcasting messages across the system, or indirect forwarding, i.e. the use of intermediate nodes (forwarders) to hide correlation between the sender and the receiver [29, 94, 103]. The addressing approach can be implicit, where the address contains no information either on the actual location of the addressee or on the physical reachability of the addressee [46, 45], or explicit, where the address contains information that can be used in a straightforward manner to route a message to the addressee [108]. Layered encryption is often used in anonymity protocols [29, 103]. Traffic covering can prevent the traffic timing analysis [44, 50].

These techniques often work together to achieve anonymity. For example, indirect forwarding needs layered encryption to encrypt the identities of forwarders. Flooding needs implicit address. Layered encryption can also be adopted in flooding to provide extra anonymity. Flooding is too costly (not efficient) but simple, which can be used for achieving local anonymity and achieve distributed receiver anonymity. It can also be combined with indirect forwarding to achieve scalability and efficiency.

There are two ways to choose forwarders in the indirect forwarding approaches. It can be in a centralized fashion, such as Onion [108], or a distributed fashion, such as Crowds [87] and Tor [43]. In the centralized fashion, some centers (maybe the sender or receiver) choose the whole list of forwarders and use layered encryption techniques to encrypt them. The list of forwarders will be piggybacked in the message. The problem is the center needs to know the global network information, which is not scalable in a large scale network. In the distributed fashion, during the message forwarding, the next-hop forwarder is decided by the current forwarder (there are

certainly some variations). The mechanism is scalable and can be applied in sender anonymity. However, it is hard, if not impossible, for the latter to be used in receiver anonymity.

2.4.3 Anonymous Multicast

Little work [48, 108] has been reported on anonymous multicasting. Anonymous multicast communication service is not available yet. Grosch discussed the importance of implement anonymity in IP layer multicast and proposes optimization for anonymous IP multicast. N. Weiler [108] has proposed the use of a proxy, called the SAM server, to hide some receivers. The main idea is first to add a SAM server as a normal node into a multicast tree, then attach receivers to the SAM server so that they are hidden by the server from other members. The concept of SAM server is a kind of extension to proxy or mixer in unicast. There are some drawbacks to this system. If there are multiple receivers attached to a SAM server, there exists another multicast anonymity problem among these receivers. The SAM server can be a target of attack. Also, the SAM servers should be trusted. Some types of multicast anonymity have not been addressed, such as multicast mutual anonymity and multicast group anonymity.

CHAPTER 3

Improving Service Availability by Reducing Cut Vertices

In this chapter we discuss the cut vertex problem that exists in overlay topology. The failures of the cut vertices can induce network failures and harm service availability of the overlay applications. We also present our distributed solution of CAM that can detect and neutralize cut vertices in overlay networks, which eventually improve service availability of the overlay applications.

3.1 Cut Vertices in Overlay Networks

Overlay networks provide base infrastructures for many areas including P2P systems, content distribution, and overlay multicast. In order to provide service availability, the overlay network should be capable to support qualified service under all circumstances, especially when failure happens. A node failure here refers to three situations: a node leaves the system, a node fails due to its own reason, and a node fails due to the outside attacks. The failure of "critical nodes has a greater influence on the system than the failure of normal nodes. A node may become "critical" when the functionality/service it provides is important (e.g., the server in a client-server system), and/or

its physical position is special (e.g., a cut vertex or a high degree node).

Most overlay networks are highly distributed. The resources and services are provided by each node in the system instead of specific servers. This removes the potential "specialty" of a node caused by the functionality/service requirement, but cannot remove the "specialty" of a node induced by the network topology. Nodes in an overlay network are highly self-organized. They make connections either with randomly selected nodes or via locally defined algorithms. In both cases, there is no centralized control to manage the network topology and thus the presence of topological "special" nodes in such a system is unavoidable. S. Saroiu et al. [90] show that the failure of a small amount of high-degree nodes can efficiently "shatter" the overlay network, which makes the network highly vulnerable in the face of well-constructed, targeted attacks. Here we discuss the influence of another type of "critical" nodes, cut vertices, on overlay networks.

Consider a network as an undirected graph. Cut vertices are such nodes whose deletion will create new components in the original graph. For a connected graph (component), removing cut vertices partitions the graph. In this chapter, "graph", "component", and "vertex" are concepts defined in graph theory: a "graph" is used to represent a network; a "component" is a connected graph; and a "vertex" is another name for a node. Vertex and node will be used interchangeably in the remainder of this thesis.

Traditional methods of detecting cut vertices generally require the global information of the network topology. These approaches only work well if the network topology does not change frequently and the scale of the network is from small to medium. This is not the case, however, in most of the overlay networks. The end systems that compose an overlay network come and go very frequently. According to the investigation of previous studies [90, 37, 93], the average lifetime of a node in an overlay network varies from less than 10 minutes in FastTrack to 60 minutes in

Gnutella and Napster. This leads to high dynamics of the overlay network. Furthermore, the scale of an overlay network is expected to be huge, from several thousands to millions of nodes [32, 8]. A third factor that prevents current overlay networks from adopting these approaches is that, due to their feature of being fully distributed, most overlay networks lack the centralized control to maintain the global network topology information.

In order to detect cut vertices in today's large scale and highly dynamic overlay networks, a distributed cut vertex detection mechanism needs to be proposed to run on the end nodes locally without the involvement of centralized servers. We thus propose connection adjacent matrix (CAM), a fully distributed mechanism, to detect cut vertices. Based on the CAM algorithm, each node in the system periodically sends out probe messages to its neighbors and decides whether it is a cut vertex based on the received feedback. CAM is composed of three stages: cut vertex detection, cut vertex computation, and cut vertex neutralization. At the detection stage, a cut vertex candidate sends out component detection messages for each of its connections. If any two detection messages of different connections meet with each other, an arrival message will be sent back to the message issuer. At the computation stage, the candidate constructs a CAM graph in which nodes represent the connections of the candidate. If it receives an arrival message of two connections, the candidate will add an edge to the corresponding nodes in the CAM graph. The candidate can thus decide whether it is a cut vertex by checking the CAM graph: if the graph is disconnected, the candidate is a cut vertex. CAM then normalizes this cut vertex to a non-cut vertex at the neutralization stage.

The rest of the chapter is organized as follows. The next section describes the CAM algorithm. This is followed by the proof of the correctness of the algorithm in Section 3.3. Section 3.4 discusses the simulation methodology. The severity of the cut vertex problem and the performance of CAM are presented in Section 3.5. We

conclude our work in Section 3.6.

3.2 CAM: A Distributed Cut Vertex Detection Algorithm

Consider an overlay network as a graph. The basic idea of CAM is to check whether this graph is still connected after a node is removed. If the graph is partitioned, the node is a cut vertex; otherwise, it is not a cut vertex. CAM is composed of three stages: cut vertex detection, cut vertex computation, and cut vertex neutralization. We present the details of each stage in the rest of this section.

3.2.1 Cut Vertex Detection

A node in the system cannot be a cut vertex if it has zero or one connection. Otherwise, the node considers itself as a cut vertex candidate and initializes a cut vertex detection process. Before the detection, the candidate assigns a unique numerical identifier, starting with one, to each of its connections/edges (we use the terms "connection" and "edge" interchangeably in the rest of this chapter). This identifier is called the connection number of the connection. For example, if a candidate has n connections, it will label them from 1 to n.

At the beginning of the detection, the candidate sends a component probe message to each of its neighbors. The message contains the candidate's IP address, a timestamp, a TTL threshold, and the connection number of the edge that connects this neighbor with the candidate. Each node in the system has a connection list. There is one entry for each candidate in the connection list with the format of <candidate IP address, timestamp, connection number 1, connection number 2, ...>. The node deals with the received message based on the information stored in the connection list. Upon receiving a message, one of the following situations may

arise.

1. The node has already received the message, or the message is old. The node drops the message.

2. There is no entry for the candidate that issues this message. The node creates an entry for it.

3. The timestamp in the received message is newer than the one stored in the corresponding connection list entry. The candidate replaces the old time stamp and connection numbers stored in the connection list with the new ones.

4. The timestamp of a recently received message is the same as the one stored in the corresponding connection list entry but the connection number of the message is not the same. The node adds the new connection number to the corresponding entry and sends an *arrival message* back to the candidate. Each arrival message contains two or more connection numbers and a timestamp. A node does not send any arrival messages until it receives probe messages from at least two different connection numbers.

A node forwards the message it receives to all its neighbors except the message sender if the following conditions are met: this is a "new" message with the latest timestamp; the node did not issue any arrival message for the cut vertex candidate who issues this message; and the message's TTL has not expired. To illustrate the cut vertex detection stage, we present examples of cut vertex detection in three different situations. The initial CAM TTL values in these examples equal three. Nodes reduce the TTL value by one each time right before they forward the probe messages to their neighbors.

The first example is shown in Figure 3.1, where the candidate is clearly a cut vertex. The connections to the candidate are labelled 1, 2, 3, and 4, respectively,

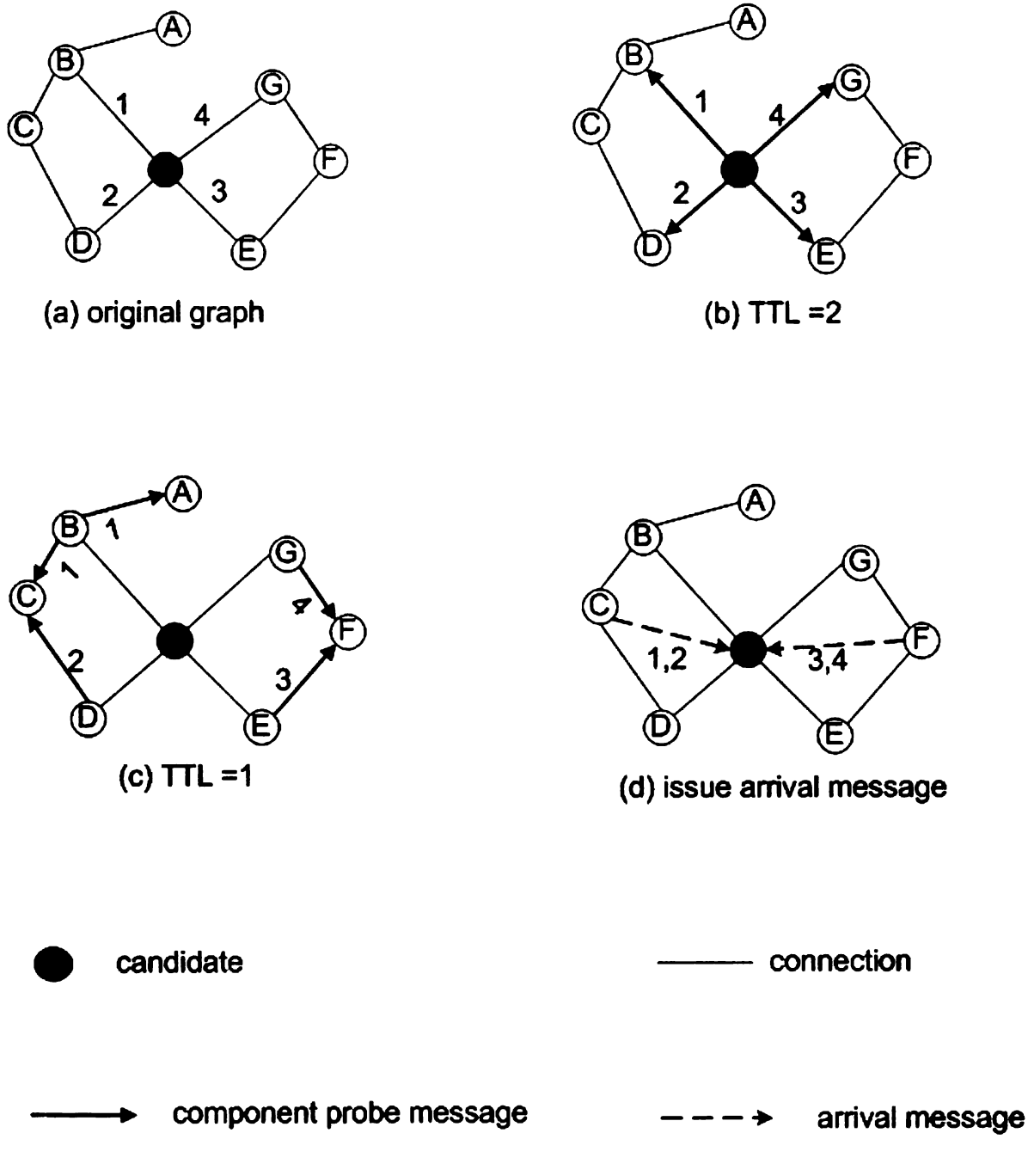


Figure 3.1. Cut vertex case 1: the candidate node is a cut vertex

in Figure 3.1(a). In Figure 3.1(b), the candidate sends a probe message for each connection to nodes B, D, E, and G. Note that the TTL value has already been reduced by one by the candidate before it sends the probe messages to its neighbors. In Figure 3.1(c), nodes B, D, E, and G forward the received probe messages to other neighbors. At this point, nodes C and F received probe messages from two distinct connection numbers. In Figure 3.1(d), node C sends back to the candidate an arrival message with the connection numbers 1 and 2. Node F sends back to the candidate an arrival message with the connection numbers 3 and 4.

In the case shown in Figure 3.2, the candidate is not a cut vertex. In Figure 3.2(b), the candidate sends probe messages for connections 1, 2, and 3 respectively to its neighboring nodes B, A, and F. Nodes B, A, and F rebroadcast the probe messages to their neighbors C, E, and G as shown in Figure 3.2(c). Node C and E rebroadcast messages with connection numbers 1 and 3 respectively to node D. Here C receives two probe messages of different connection numbers for the same candidate at the same time. In this situation, C will broadcast only one probe message with one connection number to its neighbors. At this point, nodes C and G have received probe messages from two distinct connection numbers. In Figure 3.2(d), node C sends back to the candidate an arrival message containing connection numbers 1 and 2. Node G sends an arrival message containing connection numbers 2 and 3. As node C and G received messages of more than one connection number, they no longer forward the messages. In Figure 3.2(e), node D sends an arrival message back to candidate node containing connection numbers 1 and 3.

In the case shown in Figure 3.3, the candidate is not a cut vertex. Nevertheless, there is no arrival message sent back to the candidate since the TTL expired early. In Figure 3.3(b), the candidate sends probe messages of connection numbers 1 and 2 to its neighboring nodes A and I respectively, with a TTL value of 2. In Figure 3.3(c), node A and I rebroadcast a probe to their neighbors B and H. TTL values

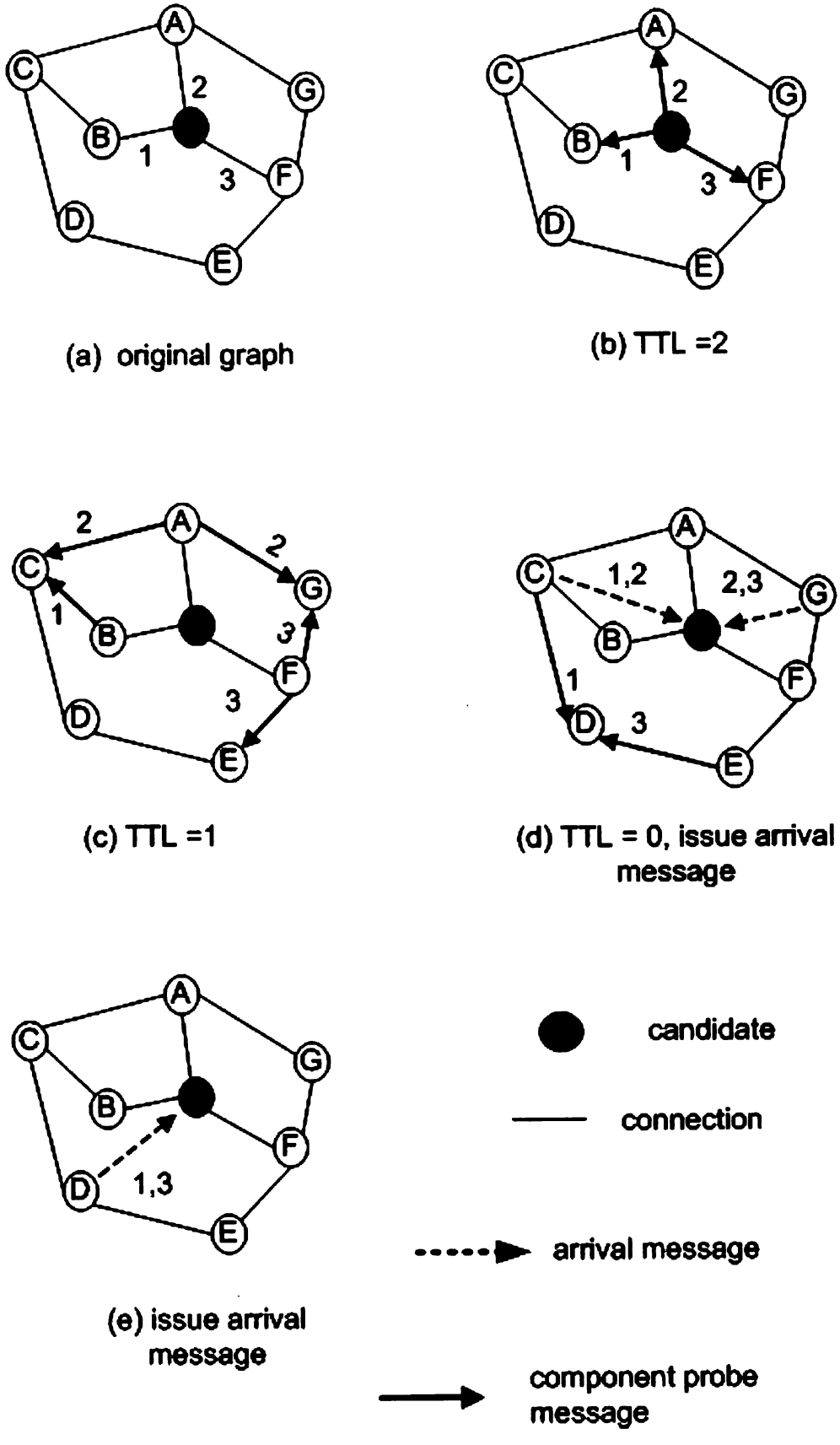


Figure 3.2. Cut vertex case 2: the candidate node is not a cut vertex

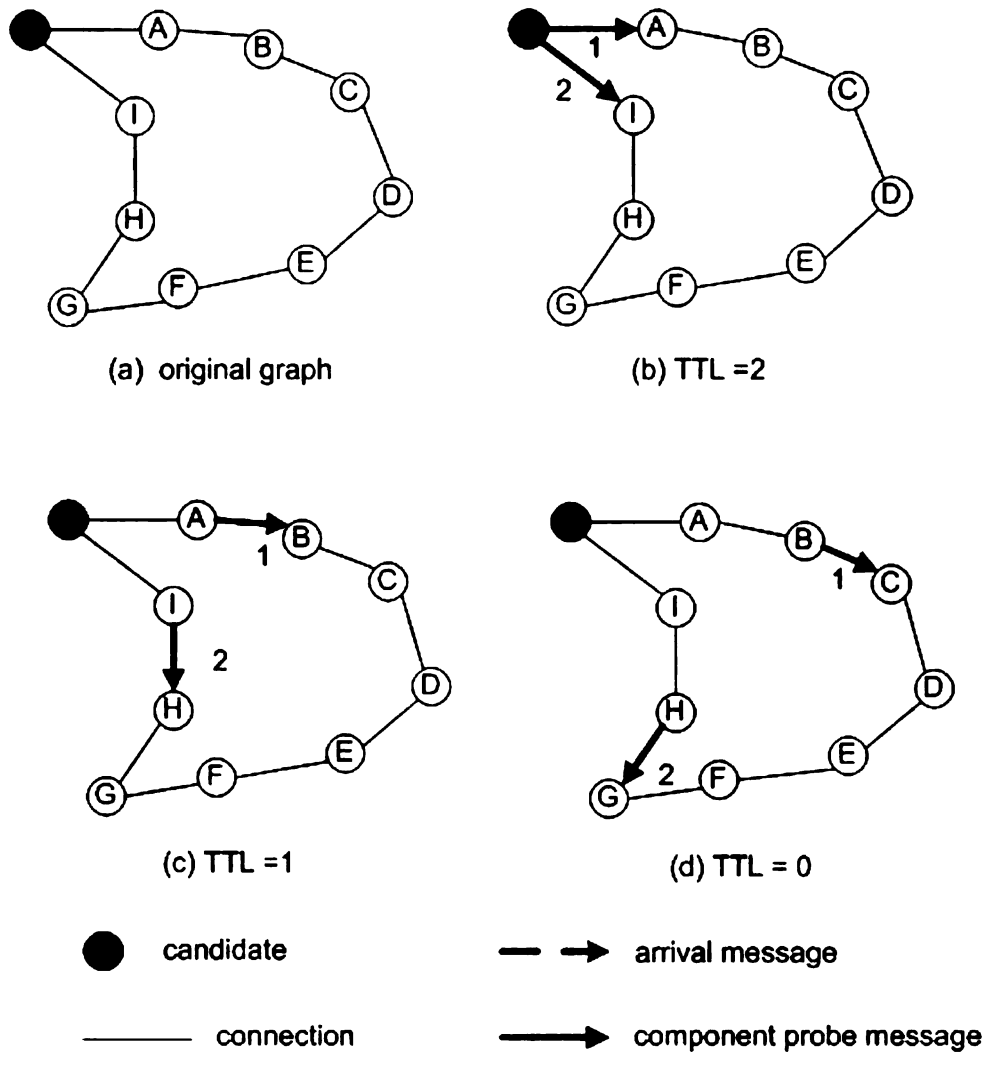


Figure 3.3. Cut vertex case 3: no arrival message is sent back to the candidate node since the TTL is not big enough

are reduced to 1. In Figure 3.3(d), node B and H rebroadcast probe messages to their neighbors C and G. TTL values are reduced to 0, which prevent node C and G from re-broadcasting the probe messages. As no node receives probe messages of two different connection numbers, no arrival message is sent back to the candidate. The simulation results in Section 3.5 also demonstrate that the accuracy of CAM can be increased by increasing the value of the TTL threshold, while increasing the value of the TTL threshold also results in an increase in the probe time and the traffic overhead. Therefore, we have to trade the accuracy for the probe time and the traffic overhead.

3.2.2 Cut Vertex Computation

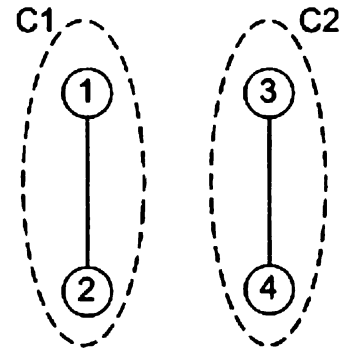
Each candidate maintains an arrival list and a $|c|$-by-$|c|$ binary matrix, where $|c|$ is the number of connections the candidate had. The format of the arrival list is similar to the connection list: <IP address, timestamp, connection number 1, connection number 2, ...>. The IP address is the IP address of the node that sends the arrival message back to the candidate. The binary matrix is called the candidate's connection adjacency matrix or CAM, whose row/column numbers represent the connection numbers of the candidate's connections.

If an arrival message that includes connection number x is received by the candidate from a network node v, the candidate will add x to the corresponding entry of v in the arrival list. For any entry (x, y) in CAM, where x is the row number and y is the column number, if the corresponding connection number of x and y can be found in the same entry of the arrival list, the value of this CAM entry is set to 1. Otherwise, the value of this CAM entry is set to 0. In other words, if any node has sent back to the candidate an arrival message containing connection numbers x and y, a 1 is placed in the (x, y) and (y, x) entry of the candidate's CAM.

After waiting an expected time out period, the candidate interprets its CAM as an

connection	#1	#2	#3	#4
#1	0	1	0	0
#2	1	0	0	0
#3	0	0	0	1
#4	0	0	1	0

(a) CAM

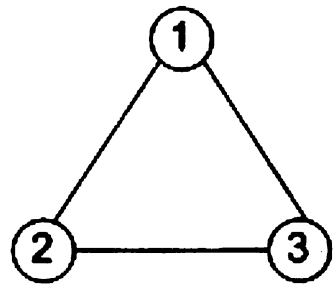


(b) CAM graph

Figure 3.4. CAM and CAM graph in case 1: candidate node is cut vertex since the CAM graph has two components

connection	#1	#2	#3
#1	0	1	1
#2	1	0	1
#3	1	1	0

(a) CAM



(b) CAM graph

Figure 3.5. CAM and CAM graph in case 2: candidate node is not cut vertex since the CAM graph is connected

adjacency matrix representation of an undirected graph, whose vertices correspond to the candidate's connections. This graph is called the candidate's CAM graph. An edge exists between node x and node y in the CAM graph if and only if the value of the CAM entry (x, y) is 1. If the candidate's CAM graph has more than 1 component, the candidate is a cut vertex. The CAM and the CAM graph of the candidate nodes in examples 1, 2, and 3 are shown in Figure 3.4, 3.5, and 3.6.

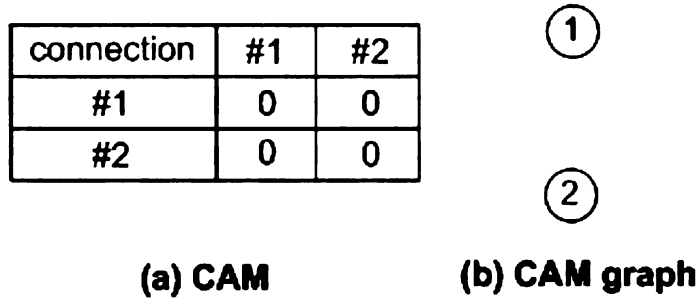


Figure 3.6. CAM and CAM graph in case 3: no arrival message is issued due to TTL expiration

3.2.3 Cut Vertex Neutralization

By adding new connections among different components in its CAM graph, a cut vertex can reduce itself to a non-cut vertex. We call this process cut vertex neutralization. The disconnected components of a node's CAM graph are merged into one connected component in the cut vertex neutralization process. The cut vertex neutralization follows the following principles:

1. The cut vertex always chooses its neighbors to make the extra connection; it chooses other nodes only when its neighbors cannot make the connection.

2. For neighbors associated with the same CAM component, the cut vertex always chooses the neighbor with the biggest available bandwidth to make the extra connection.

3. In the case that both CAM components have more than one node and the corresponding new connection for these two components is made by two neighbors of the cut vertex, the cut vertex will disconnect its own connection with one of the neighbors involved in the new connection. Here we recommend cutting the connection that has longer round trip time between the cut vertex and the neighbor.

Consider a detected cut vertex v that has n CAM graph components C_1, C_2, C_3, C_n. At the beginning of the neutralization process, v will choose its overlay network neighbor that is associated with C_i. If C_i has more than one node, the network neighbors will be selected based on their available bandwidth. v always chooses the one that has the biggest available bandwidth. If none of the overlay neighbors is capable of making a new connection, v will choose the overlay node that sent v the arrival message based on the arrival list. Assume v selects overlay nodes o_1, o_2, o_3, o_n to make the new connections. v then sends a connection message to o_1, o_2, o_3, o_n to indicate how the nodes should connect to each other, e.g., o_1 connects to o_2; o_2 connects to o_3; $\ldots o_{n-1}$ connects to on. Consider a new connection $o_i o_j$ that both o_i and o_j are neighbors of v. If both CAM components that o_i and o_j belongs to have more than one CAM node, v will disconnect vo_i or vo_j based on the cost of these connections.

The network topologies of the aforementioned examples after cut vertex neutralization are shown in Figure 3.7. In case 1, B, D are associated with component C1; E, G are associated with component C2. After making an extra connection between D and E to neutralize the cut vertex, the connection between D and the cut vertex is disconnected based on the third principle aforementioned, as well as the round trip time of the connection of D and the cut vertex is larger than that of E and the cut vertex. For case 3, since both CAM components have only one neighbor associated with them, no connection will be disconnected after the neutralization process.

Based on the third principle of CAM neutralization, a detected cut vertex will disconnect its connection to a neighbor involved in the new connection in a specified situation. Here we prove the recommended disconnection doest not create new cut vertices. The situation specified in third principle is shown in Figure 3.8. Assume K is a cut vertex detected by CAM. Here we assume that the network that K belongs to is a connected network. (Otherwise, the CAM operation will not affect the components

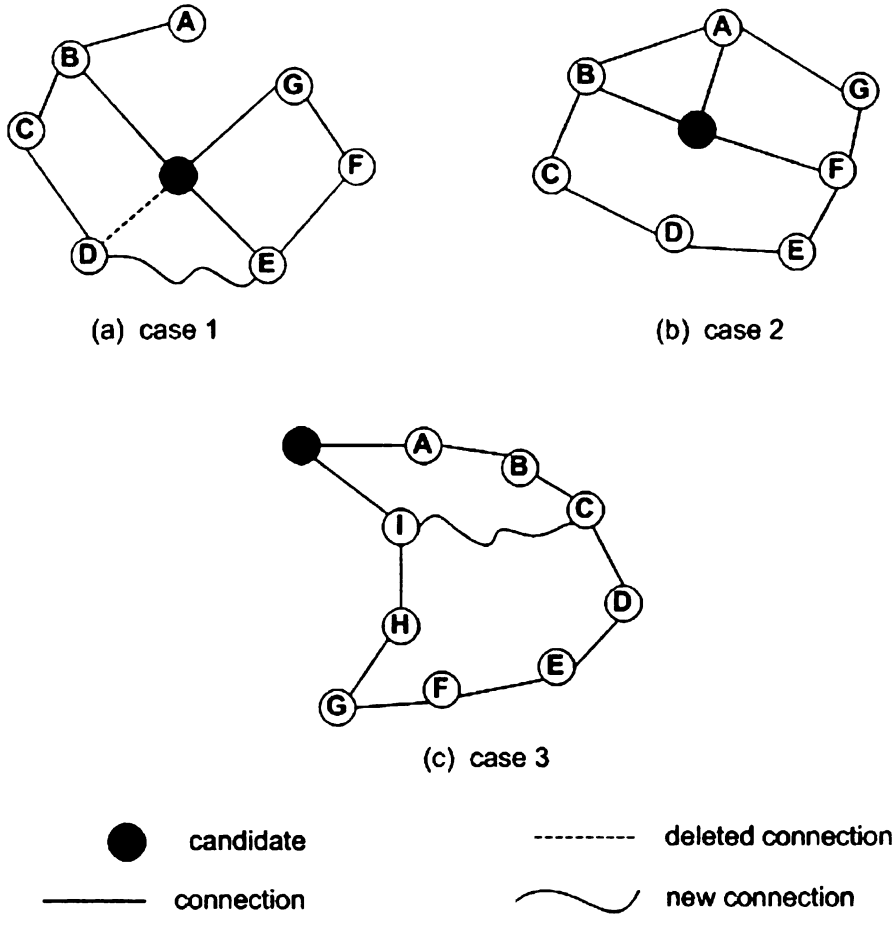


Figure 3.7. Cut vertex neutralization in case 1, case 2, and case 3

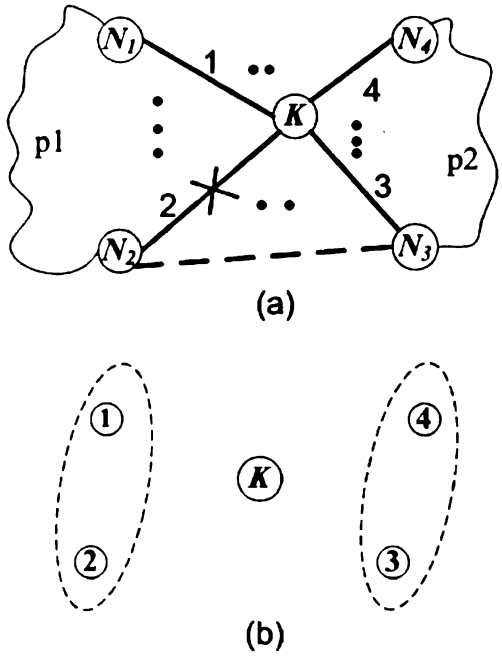


Figure 3.8. Cut vertex based on neutralization principle 3: Cut vertex K and its CAM graph

that K does not belong). There are a path p1 between K's neighbors N1 and N2, and a path p2 between K's neighbors N3 and N4. N3 and N4 are not in the path of p1. N1 and N2 are not in the path of p2. K and its neighbors are show in Figure 3.8 (a). The CAM graph of K is shown in Figure 3.8 (b). Connection numbers 1, 2, 3, and 4 are assigned to connections KN1, KN2, KN3, and KN4 respectively.

Theorem 3.2.1 *Remove either KN_2 or KN_3 while add a new connection N_2N_3 will not create a new cut vertex.*

Proof: Since there is no difference in removing KN2 from removing KN_3, we only give the proof of removing KN_2. The proof for removing KN3 will be similar. The proof includes three cases:

1. Assume a new cut vertex X (X is any node in the network except N_2 and N_3) is created when KN_2 is removed as shown in Figure 3.9. We prove this case by contradiction. As X is a cut vertex, the removal of X will create at least one

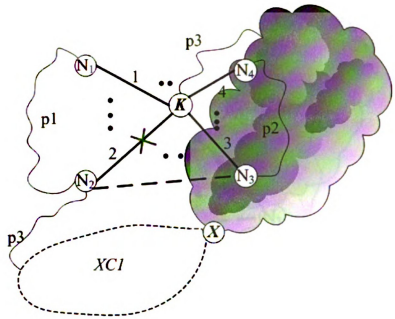


Figure 3.9. Assume the new cut vertex is any node but N_2 or N_3

more component in the network. Let's check a new component XC_1 created by the removal of X. XC_1 can be reconnected to the network by adding KN_2 back to the network (X will not be a cut vertex when KN_2 is added back to the network). This suggests that there is at least one path p_3 via KN_2 as shown in Figure 3.9 that connects XC_1 with other part of the network. In addition, the removal of X will not disconnect p_3. After remove KN_2, K and N_2 are still connected via N_2N_3K due to the new connection of N_2N_3,. Accordingly, p_3 is still connected and XC_1 is connected to the other part of the network via p_3. This contradict with that XC_1 is a new component created by the removal of X.

2. Assume N_3 is turned into a cut vertex by the removal of KN_2. This case is similar to case 1, but N_2N_3 will be disconnected with the removal of N_3. Since N_3 is not in the path of p_1, N_2 and K can be connected by $N_2p_1N_1K$ when N_3 is removed as shown in Figure 3.10. Therefore, p_3 is still connected, as well

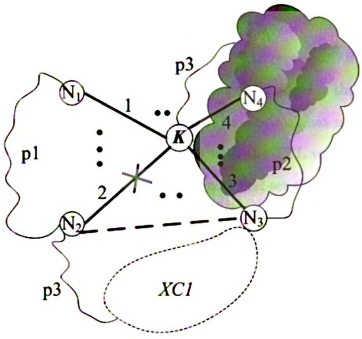


Figure 3.10. Assume N_3 is turned into a cut vertex when KN_2 is removed

as XC_1.

3. Assume removing KN_2 turns N_2 into a cut vertex. This suggests N_2 is not a cut vertex when KN_2 is not removed. Therefore removing N2 when KN_2 is in the network will not create any new component. However, this is impossible, as removal of $N2$ will also disconnect/remove KN_2.

Proved ■

3.3 Proof of Correctness for the CAM Algorithm

In this section, we present the proof of correctness for the CAM algorithm, starting with the definition of the system model, which is followed by the proofs.

3.3.1 System Model and Definitions

Consider a connected undirected graph: G = (V, E) to represent an overlay network, where V is the set of overlay nodes and E is the set of the edges of the overlay network. Assume that the network topology is static and the network has unlimited resources. Thus, each node can issue the probe message with the TTL value set to infinity. In practice, we trade the accuracy for the traffic cost and set the TTL to a small value.

Definition 3.3.1 *Given a graph $G(V, E)$, a cut vertex candidate v_c refers to a vertex that tries to decide whether it is a cut vertex. The number of edges/connections that v_c has is referred to as $|c|$.*

Definition 3.3.2 *Given a graph $G(V, E)$, a vertex v_p, and one of its connections e_c = (v_p, v_q), assuming the connection number of e_c is c. v_q is the neighbor of v_p that is connected by e_c. Let us remove v_p together with all its edges from G and get a new graph $G'(V - v_p, E - \{(v_p, v_i)|v_i \in V, (v_p, v_i) \in E\})$. The reachable set of e_c, $RS(e_c)$, is the set of vertices that can be reached in G' by a breadth first search (BFS) initiated by v_q. This search process is denoted as $BFS(e_c)$. In other words, $RS(e_c)$ contains the set of all vertices that can be reached by v_q via connection e_c. It is also important to note that there exists a vertex $v_q \in RS(e_c)$ such that $(v_p, v_q) \in E$.*

3.3.2 Proofs

Lemma 3.3.1 *If there is an edge that connects two vertices in the candidate's CAM graph that represent connections e_a and e_b of graph G, then $RS(e_a) = RS(e_b)$.*

Proof: The fact that there is an edge between the vertices representing connections e_a and e_b in the candidate's CAM graph implies that the values of entry (a, b) and (b, a) must be 1 in the CAM. This implies that there must exist some node v_p in the network that has received component probe messages for connections e_a and e_b.

Since v_p received component probe messages for connections e_a and e_b, it is clear that node v_p is traversed by both BFS(e_a) and BFS(e_b). According to the operation of BFS algorithm, after traversing node v_p, BFS(e_a) can reach all the nodes that BFS(e_b) can reach and vice versa. Therefore, $RS(e_a) = RS(e_b)$.

Lemma 3.3.2 *If the vertices that represent connections e_a and e_b are in the same component of the candidate's CAM graph, $RS(e_a) = RS(e_b)$.*

Proof: This is trivial given Lemma 3.3.1.

Lemma 3.3.3 *If the vertices that represent connections e_a and e_b are not in the same component of the candidate's CAM graph, $RS(e_a) \neq RS(e_b)$.*

Proof: We prove this by contradiction. Assume that the vertices that represent connections e_a and e_b are not in the same component of the candidate's CAM graph, but $RS(e_a) = RS(e_b)$. Then there must exist at least one node that has not been reached by probe messages containing connection numbers a and b but that can be reached by both BFS(e_a) and BFS(e_b). According to the operation of BFS, this only happens when the TTL has expired before the probe message arrives at the specific node. This contradicts the earlier assumption that the TTL values of probe messages are infinite. Therefore, the lemma is true.

Lemma 3.3.4 *If $RS(e_a) = RS(e_b)$ for any $1 \leq a, b \leq |c|$ of a cut vertex candidate v_c, then v_c is not a cut vertex.*

Proof: $RS(e_a) = RS(e_b)$ suggests that any node $v_i \in RS(e_a)$ must $\in RS(e_b)$. $RS(e_a) = RS(e_b)$ for all $1 \leq a, b \leq |c|$ suggests this happens in any two reachable sets of v_c. In addition, for any nodes v_i and $v_j \in RS(e_a)$, there exists a path from v_i

to v_j that does not include v_c. Therefore, removing v_c from G leaves one connected component. By definition, v_c is not a cut vertex.

Lemma 3.3.5 *If $RS(e_a) \neq RS(e_b)$ for any $1 \leq a, b \leq |c|$, then v_c is a cut vertex.*

Proof: Let G' be the graph that is formed by removing v_c and all its edges in G. $RS(e_a) \neq RS(e_b)$ implies that there does not exist an edge $(v_i, v_j) \in G'$ where $v_i \in RS(e_a)$ and $v_j \in RS(e_b)$. This implies that $RS(e_a)$ and $RS(e_b)$ are separate components of G'. By the definition of RS, the removal of v_c from G results in the number of components of G increasing by at least 1. Therefore, v_c is a cut vertex.

Theorem 3.3.1 *If the candidate's CAM graph has more than one component, it is a cut vertex.*

Proof: The candidate's CAM graph has more than one component. According to Lemma 3.3.3, we know that the RSs associated with the connection numbers whose vertex representations belong to different components in the CAM graph are not equal. Due to Lemma 3.3.5, we can easily conclude v_c is a cut vertex.

Theorem 3.3.2 *If the candidate's CAM graph has one component, it is not a cut vertex.*

Proof: From Lemma 3.3.2, we can deduce that $RS(e_a) = RS(e_b)$ for any $1 \leq a, b \leq |c|$ when the candidate's CAM graph has one component. From Lemma 3.3.4, we can conclude v_c is not a cut vertex.

3.4 Simulation Methodology

In order to evaluate the performance of CAM, we deployed a serial of simulations based on the real world P2P overlay network traces. What needs to be noted here is that cut vertices exist in most overlay networks. We deployed our performance evaluation of CAM on P2P systems due to two reasons. First, P2P systems are very popular today. P2P traffic overwhelms web traffic on the Internet and becomes the major consumer of the Internet bandwidth [89]. Second, the P2P system is a representation of the overlay networks: it is composed of self-governed end systems; it is autonomous and open; there is no central control server in the P2P system; nodes can join and leave the system at any time; and collecting and maintaining overall topology information in a P2P system is hard, if not impossible.

3.4.1 Performance Metrics

Performance metrics used in this study can be divided into two categories: metrics to evaluate the accuracy of CAM and metrics to evaluate the impact of CAM on the network service. For the first category, we introduce *CAM accuracy rate* (CAR), *CAM false positive rate* (CFPR), and *CAM false negative rate* (CFNR). For the second category, we investigated the *search success rate*, *node traffic load*, and *search cost*. Assume that V is the set of all vertices in a network, C is the set of all cut vertices, and K is the set of all vertices that are identified by CAM as cut vertices. The definitions of aforementioned metrics are given as follows:

CAM accuracy rate (CAR): shows how many vertices detected by CAM as cut vertices are cut vertices.

$$CAR = \frac{|K \cap C|}{|K|}$$

CAM false positive rate (CFPR) and *CAM false negative rate* (CFNR): CAM false positive rate and CAM false negative rate show the error types made by CAM. CAM

false positive rate shows how many nodes that are not cut vertices but identified as cut vertices. CAM false negative rate shows how many nodes that are cut vertices are not detected as cut vertices.

$$CFPR = \frac{|(V - C) \cap K|}{V - C}$$

$$CFNR = \frac{|C \cap (V - K)|}{|C|}$$

Search success rate: search is the main operation deployed in the P2P system. The search success rate is the main metric reflecting the quality of service of a P2P system. The search success rate is defined as the searches that can satisfy the users' requests over all searches issued by users. In order to check the efficiency of CAM on the search mechanism, we deploy the searches on the simulated P2P network before and after the CAM operation and compare their success rates.

Search cost: search cost is one of the parameters that concern the network administrators. Large search cost impedes the scalability of P2P systems. Thus, we expect that CAM should not greatly increase the search cost of the system. We use the messages issued during the search process to evaluate the search cost.

Cut vertex offload rate: We also introduce Cut vertex offload rate to check how much traffic load being offload from cut vertices by CAM. We measure the offload rate based on the messages handled by cut vertices during the search process. Assume messages handled by a cut vertex before CAM are M_{ia}, messages handled by cut vertex after CAM are M_{ib}. The network size is n. The cut vertex offload rate is measured by:

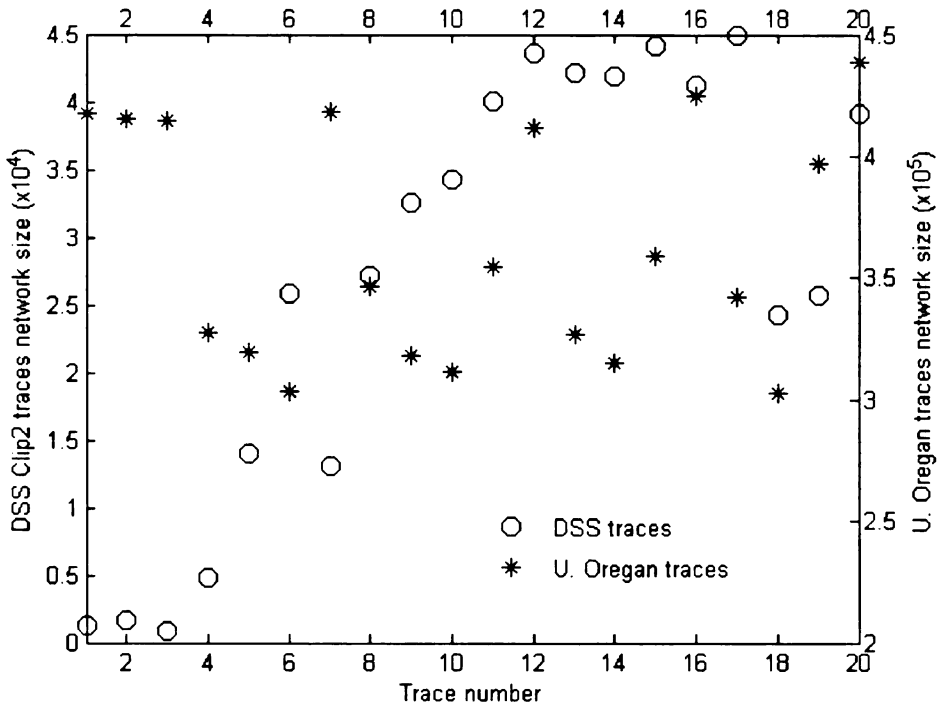
$$\frac{1}{n}\sum_{i}^{n}\frac{M_{ia} - M_{ib}}{M_{ia}}$$

Besides the above performance metrics, we have investigated the change of the number of components and new cut vertices caused by cut vertex failure, as well as the traffic load for average node vs. cut vertices in the search process.

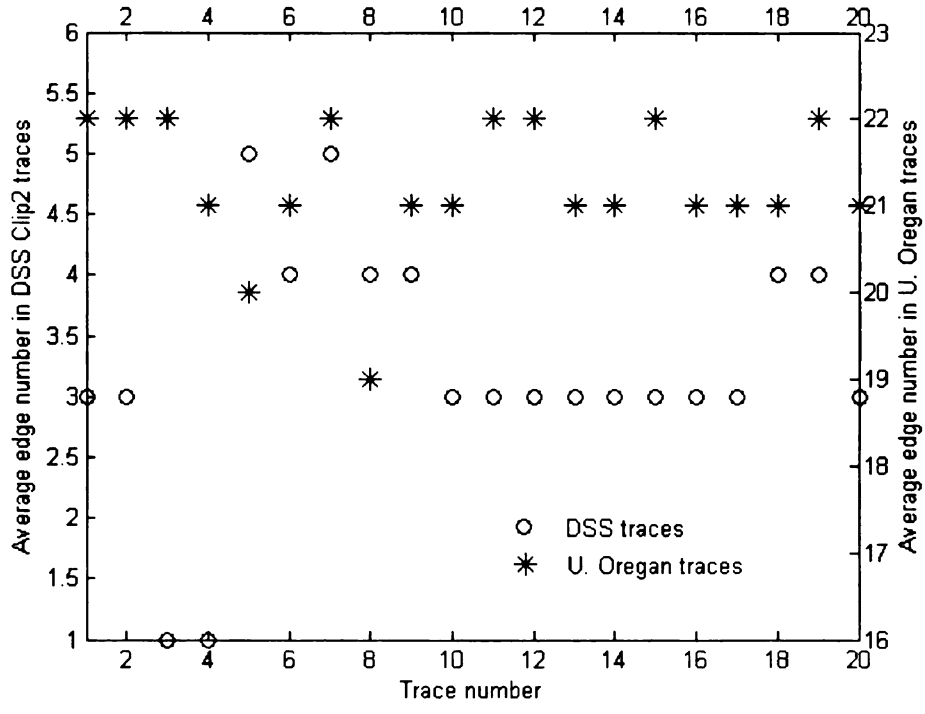
3.4.2 Simulation Setup

We generate the P2P overlay networks based on two series of real world traces: the DSS Clip2 traces that were collected from Dec. 2000 to June 2001, and the traces that were collected by the researchers of the University of Oregon in May 2006 [3, 10]. Clip2 traces were collected over a long period of time. It includes both traces before and after March, 2001, the DSS Clip2 traces were available on http://dss.clip2.com, but are not available now. We can provide the traces to those who are interested upon request. The traces from the University of Oregon can be found at http://mirage.cs.uoregon.edu/P2P/info.cgi.

We selected 20 traces from DSS Clip2 traces and 20 traces from the University of Oregon traces respectively. We show the basic features of these traces in Figure 3.11 and Figure 3.12. The network sizes of the DSS Clip2 traces range from 225 to 47245. For the University of Oregon traces, we removed all non-leaf nodes since leaves only connect to the corresponding ultrapeer and do not forward any traffic [102]. We do not consider leaf nodes and the term "node" only refers to the non-leaf nodes in the University of Oregon traces. The network sizes without leaf nodes in the University of Oregon traces range from 302,732 to 438,370. The average connections per node of the traces are from less than 1 to 5 in DSS traces and from 19 to 22 in the University of Oregon traces. This reflects a big change in the P2P file sharing system in the past few years. The amount of components in DSS traces varies from 246 to 4473, and from 6195 to 15990 in the University of Oregon trace. As for the cut vertices, the number of cut vertices in DSS traces varies from 51 to 7262, while the number of cut vertices in the University of Oregon traces varies from 5555 to 10630. Consider the increase of the network size; both the number of components and the cut vertices in the P2P network "drop" a lot in the past few years. On the other hand, although in a small portion, cut vertices still exist in the current P2P networks. This makes successful and accurate detection of cut vertices even more difficult.

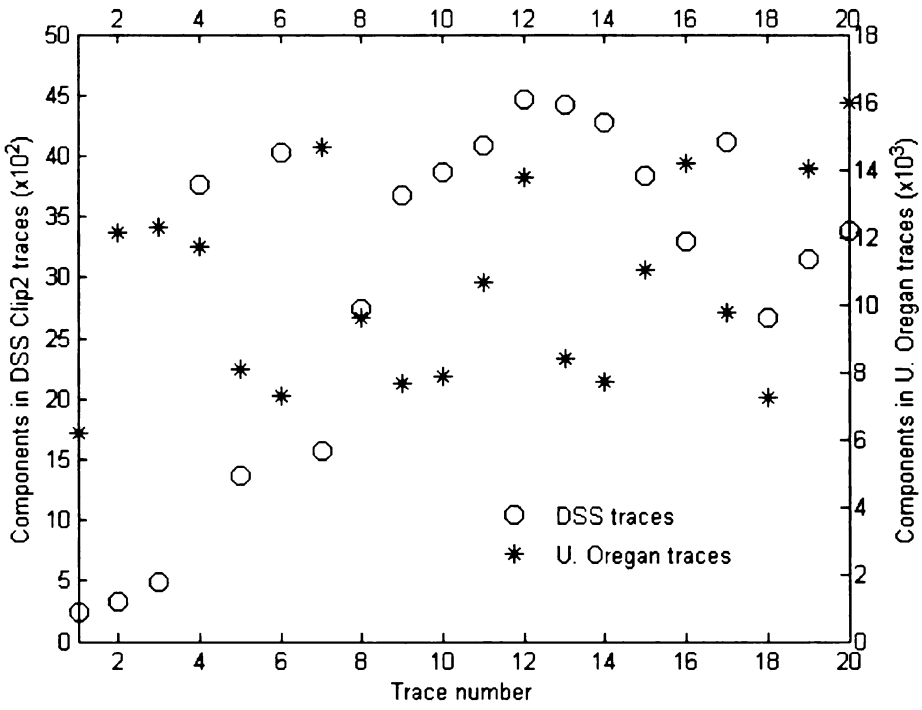


(a)

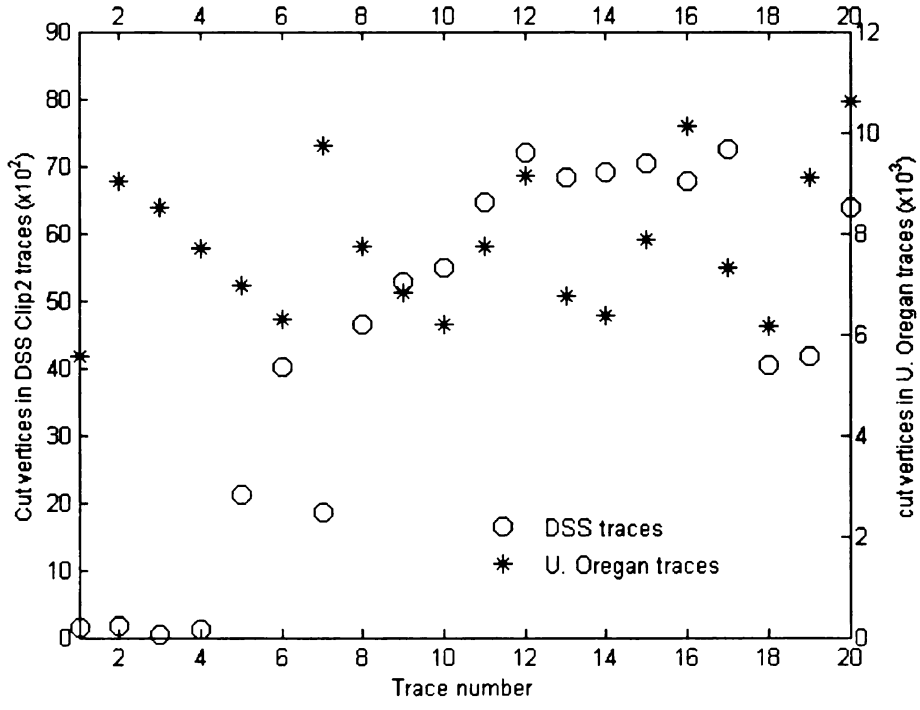


(b)

Figure 3.11. Topology feature of collected traces: network size and edges per node



(a)



(b)

Figure 3.12. Topology feature of collected traces: components and cut vertex

Flooding search is one of the most popular search mechanisms in the P2P system among peers or super peers due to its simplicity and robustness. We deploy our simulation based on the flooding search. We simulate flooding search mechanism by conducting the Breadth First Search (BFS) algorithm starting from a specific node. Previous observations show that the object popularity distribution in a P2P system does not follow a Zipf distribution like WWW objects [51]. Accordingly, we allocate the objects randomly instead of following a Zipf distribution in this study.

3.5 Performance Evaluation

The aim of our cut vertex detection algorithm is to detect the cut vertices accurately before they fail and improve the reliability of the network. To evaluate the accuracy of CAM, we compute the cut vertices in each simulated network using the traditional DFS algorithm, and compare the results with those of CAM. In order to evaluate the impact of CAM on the reliability of the network, we investigate changes in the quality of the service (search success rate) and the network topology features caused by cut vertex failure with and without CAM deployment. We also investigate CAM's impact on the traffic load of cut vertices and check the ratios of high degree nodes in cut vertices.

3.5.1 Accuracy of CAM Detection

We deploy simulations to evaluate the CAM accuracy rate, as well as the CAM false positive rate and CAM false negative rate to further understand the different types of the CAM failure. The results of the simulation for DSS traces are presented in Figure 3.13. Figure 3.14 shows the accuracy rate of CAM for the University of Oregon traces. From these results, we observe that the false negative rate remains as zero in all cases. With the increase of the CAM TTL value, the false positive rate keeps reducing and

drops to almost zero when the TTL equals 4. In the DSS trace, the accuracy rate is over 70% even when the TTL value of CAM is 1. In the University of Oregon traces, the CAM is not accurate when the CAM TTL value is 1 or 2. However, the accuracy rate jumps to over 98% when CAM TTL is 3 and is almost 100% when CAM TTL equals to 4. This suggests that CAM can successfully identify all the cut vertices with a small CAM TTL value, and that CAM errors mainly result from the false alarms CAM reports when it considers non-cut vertices as cut vertices if the TTL is not large enough.

3.5.2 The Cut Vertex Failure and Cut Vertex Traffic Load

In order to investigate the influence of cut vertex failure on the network topology and the network service, we have investigated changes of the network components, the cut vertices, and the search success rate. The results are shown in Figure 3.15 to Figure 3.18. We gradually removed the cut vertices and measured the network performance at the same time. Due to the page limitation of this paper, we cannot show all of our simulation results for the 40 traces. We chose the results from four traces (shown in Table 3.1) that were collected from different period, of different sizes, and have different average connection numbers.

Figure 3.15 shows the increase of components with the failure of the cut vertices. Y-axis shows the ratio of the components in the network when cut vertices fail over that when no cut vertex fails. We observe from Figure 3.15 that the number of network components increases to as many as 7 times (DSS trace) and 3.87 times (U. Oregon trace) of that when no failure happens. Figure 3.16 shows that the new cut vertices that are induced by the failure of cut vertices are about 35% and 10% of the original cut vertices in DSS traces and U. Oregon traces.

We show the drop of the search success rate in Figure 3.17. In the DSS trace, the success rate drops from greater than 70% to less than 10% with the removal of

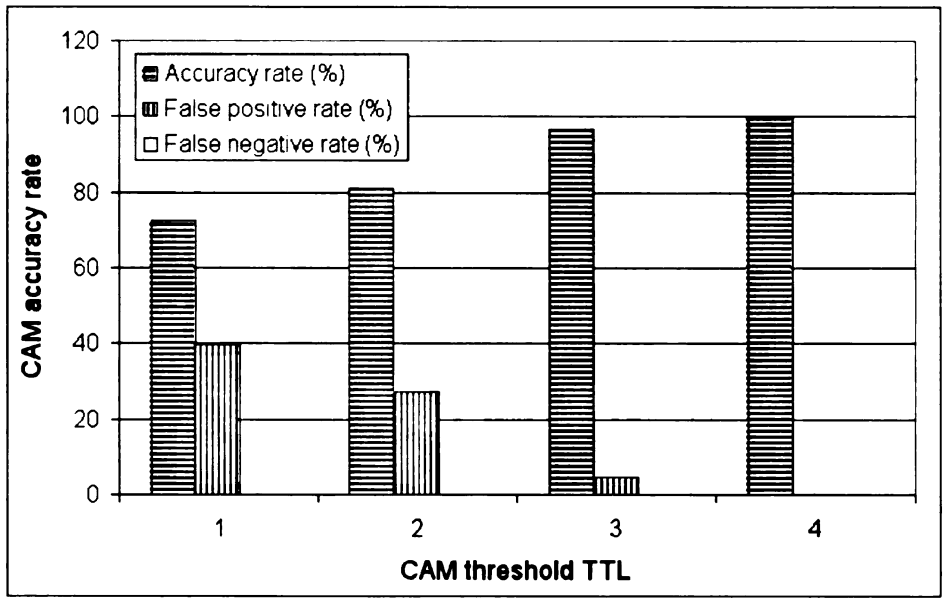


Figure 3.13. Accuracy rate of CAM (DSS trace)

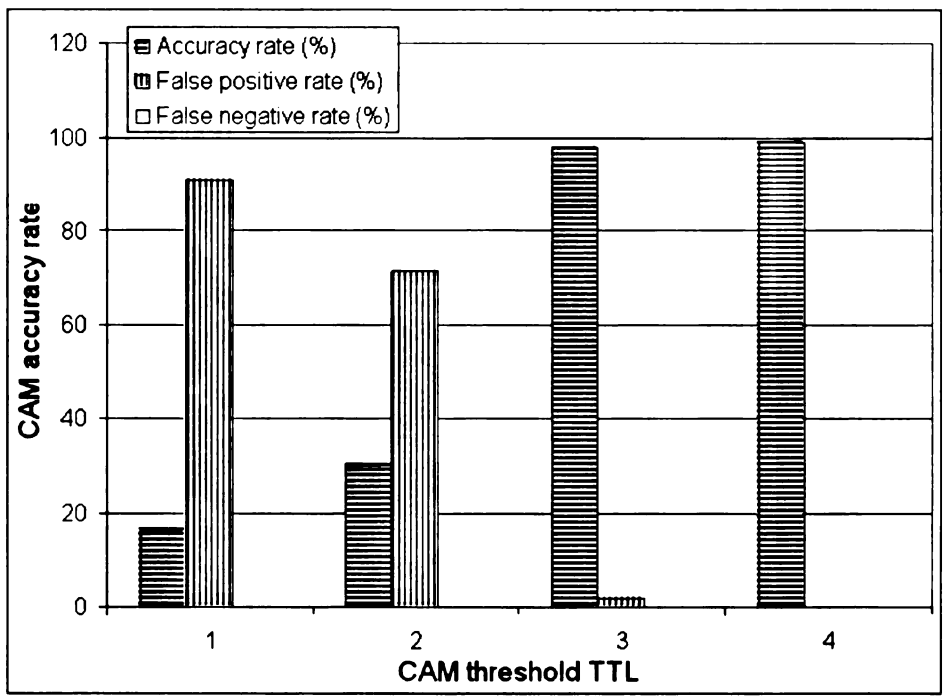


Figure 3.14. Accuracy rate of CAM (U. Oregon trace)

Table 3.1. Characteristics of Traces

Trace number	Collection date	Network size	Neighbors per node	Components	Cut vertices
8(DSS)	5/29/2001	27,192	4	2741	4635
19(DSS)	6/15/2001	25,703	3	3157	4170
3(U.Oregon)	5/23/2006	414,848	22	12285	8500
14(U.Oregon)	5/29/2006	315,121	21	7705	6360

cut vertices. Specifically, the search success rate drops about 50% when 30% of cut vertices are removed. Given the fact that the percentage of cut vertices is no more than 17% in all the DSS traces, 30% of cut vertices equal only 5.1% of the nodes in the system. In the U. Oregon trace, we can observe a drop of 48% in search success rate, from 62% to 32%, when all cut vertices are removed, which is about 2% of the nodes. Both results suggest that the influence of cut vertex failure is comparable to the influence of the failure of highest degree nodes in the system: a failure of the best connected 4% of nodes will partition the system [90].

Figure 3.18 shows the ratio of traffic load of cut vertices vs. that of ordinary nodes. We issued 50,000 searches for each network and recorded the number of the messages that were forward by each node respectively. We can observe from the results that cut vertices can forward as much as 7 times the message compared to common nodes in DSS trace and about 50% more than common nodes in the U. Oregon trace. From the results shown in Figure 3.15 to Figure 3.18, we can conclude that P2P file systems have been improved a lot since 2001. However, although in a very small percentage, there are still some cut vertices in the system and their failure can still increase the number of network components and downgrade the system performance. In addition, cut vertices need to process more messages compared to common nodes in both systems.

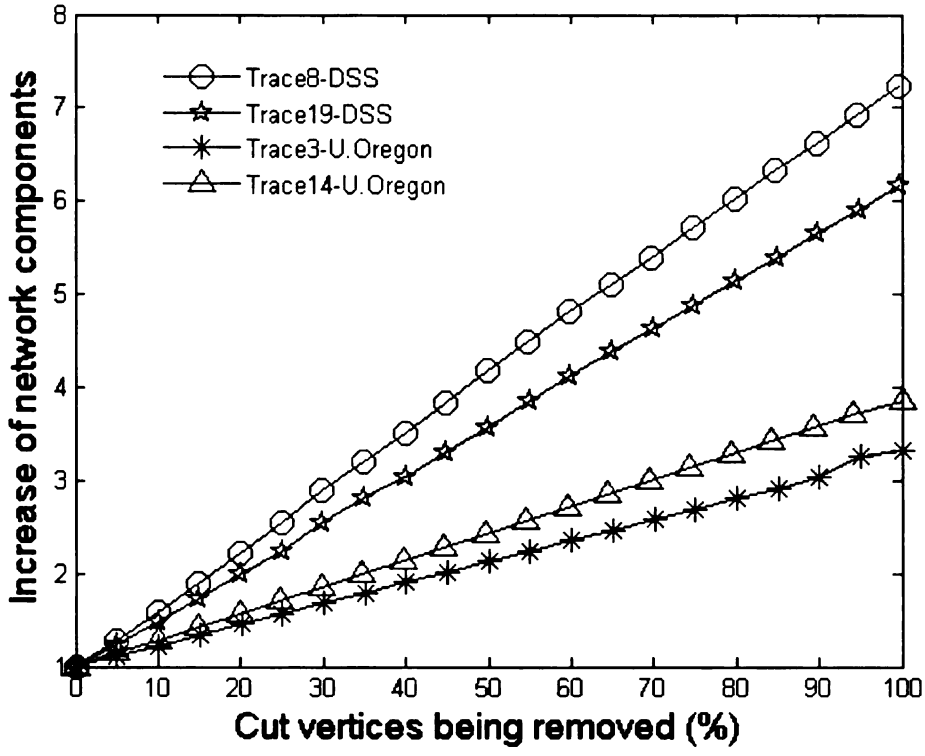


Figure 3.15. Component increase upon cut vertex failure

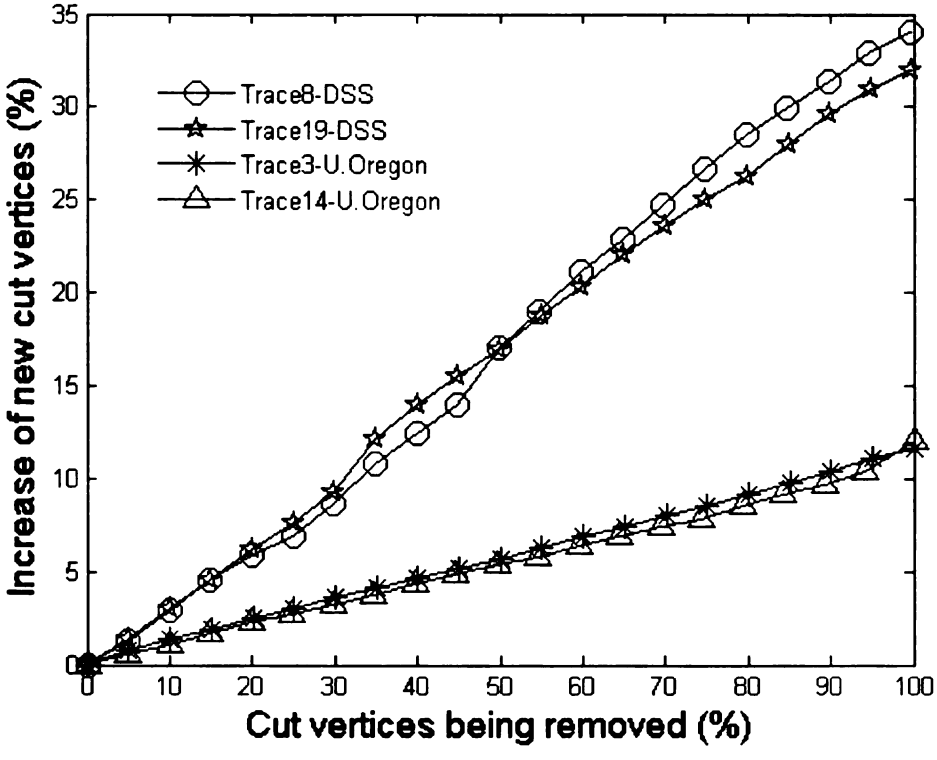


Figure 3.16. New cut vertices induced by cut vertex failure

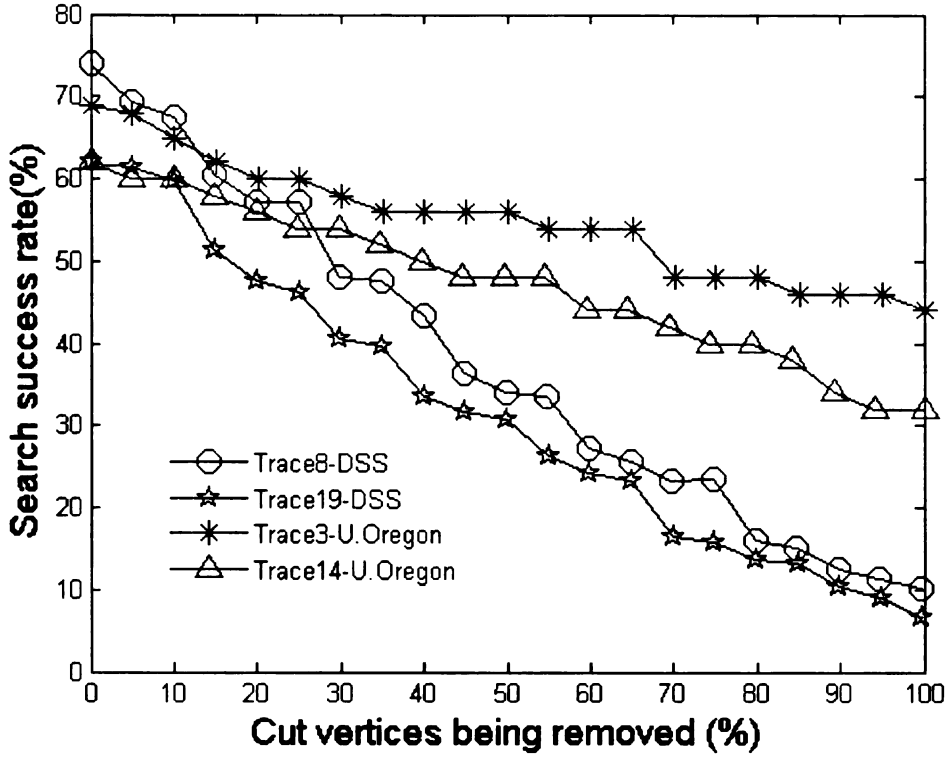


Figure 3.17. Search success rate vs. cut vertex failure

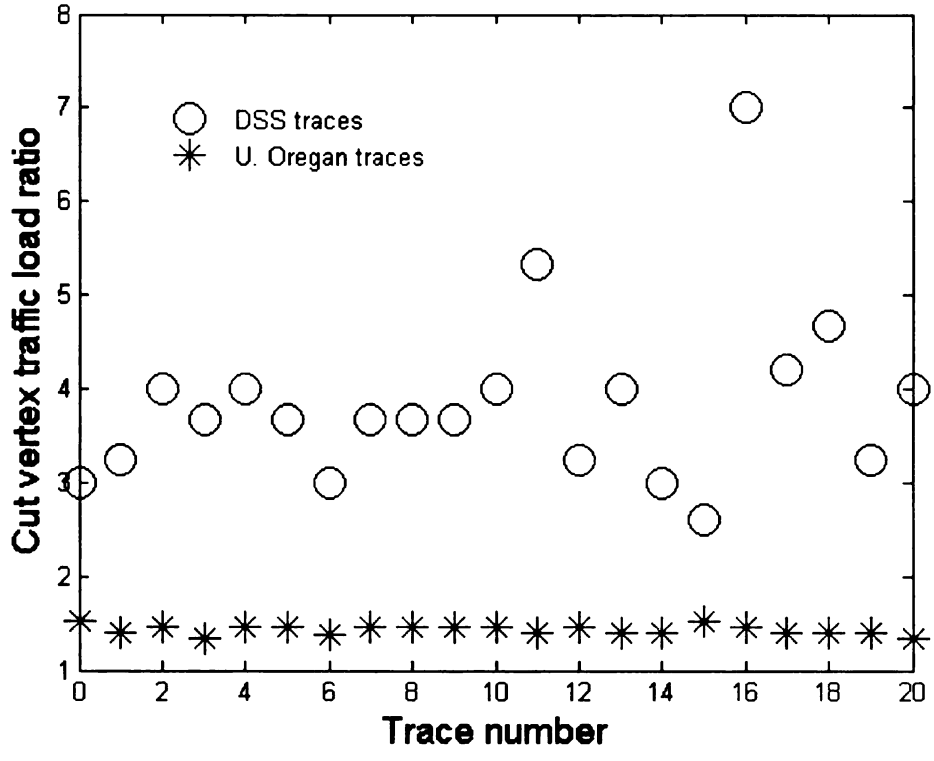


Figure 3.18. Traffic load ratio of cut vertices vs. common nodes

3.5.3 Influence of CAM on the Network Topology

We present here how CAM affects the network topology. Given the consistent results of trace 8 (DSS), 19 (DSS), 3 (U. Oregon), and 14 (U. Oregon), we representatively present the results of trace 8 (DSS) and 3 (U. Oregon), which have relatively large network sizes. Figure 3.19 and Figure 3.20 show the increase of components induced by the cut vertex failure with CAM. They show that, in both the DSS and the U. Oregon trace, the component number increases much more slowly than that in a network without adoption of CAM. At the same time, the number of components keeps increasing when the TTL threshold is increased from 1 to 3, but decreases when the TTL threshold is 4. For a CAM TTL of 3, the components induced by the cut vertex failure reduce from over 7 times the original components to less than 4 times in the DSS trace. In the U. Oregon trace, the components introduced by the cut vertex failure reduce from more than three times the original components to about twice the original components.

Figure 3.21 and Figure 3.22 show how many new cut vertices are introduced by the cut vertex failure after CAM is deployed. We observe that in both the DSS trace and the U. Oregon trace, more new cut vertices are introduced by the cut vertex failure when CAM TTL is increased from 1 to 3. The "generation" rates of new cut vertices are very close when CAM TTL is 3 and 4. In addition, with the CAM deployment, more cut vertices are introduced by the cut vertex failure compared to that in a network that does not adopt CAM.

3.5.4 Influence of CAM on the Network Service and Cut Vertex Traffic Load

We present the impact of CAM on the network service and the cut vertex traffic load in the search process here. Figure 3.23 and Figure 3.24 show the search success rate

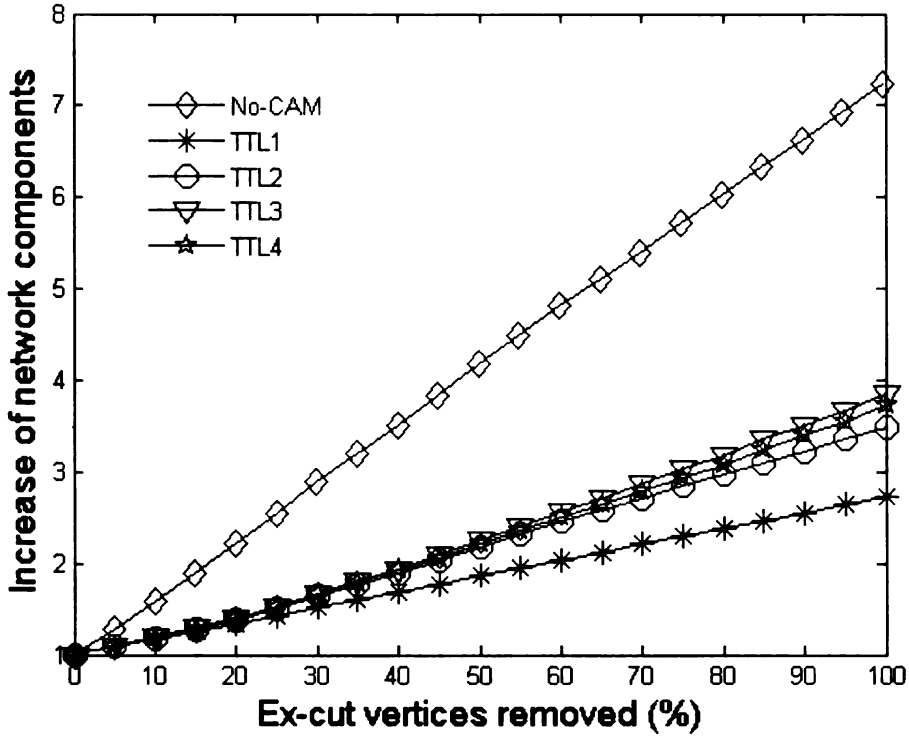


Figure 3.19. Component number increases after cut vertex failures with CAM (DSS trace)

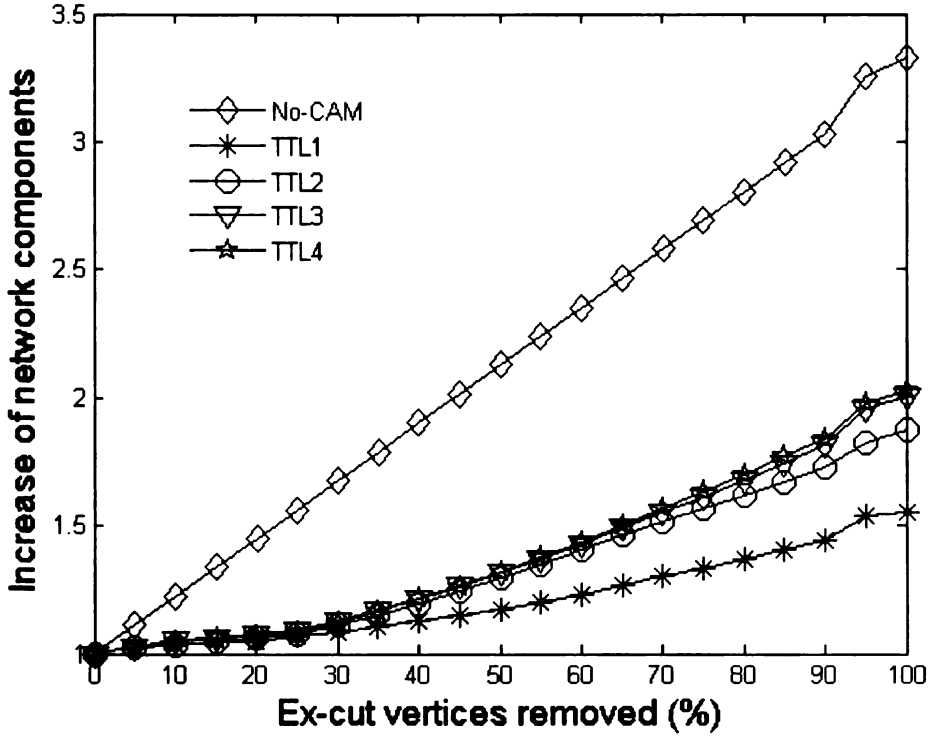


Figure 3.20. Component number increases after cut vertex failures with CAM (U. Oregon trace)

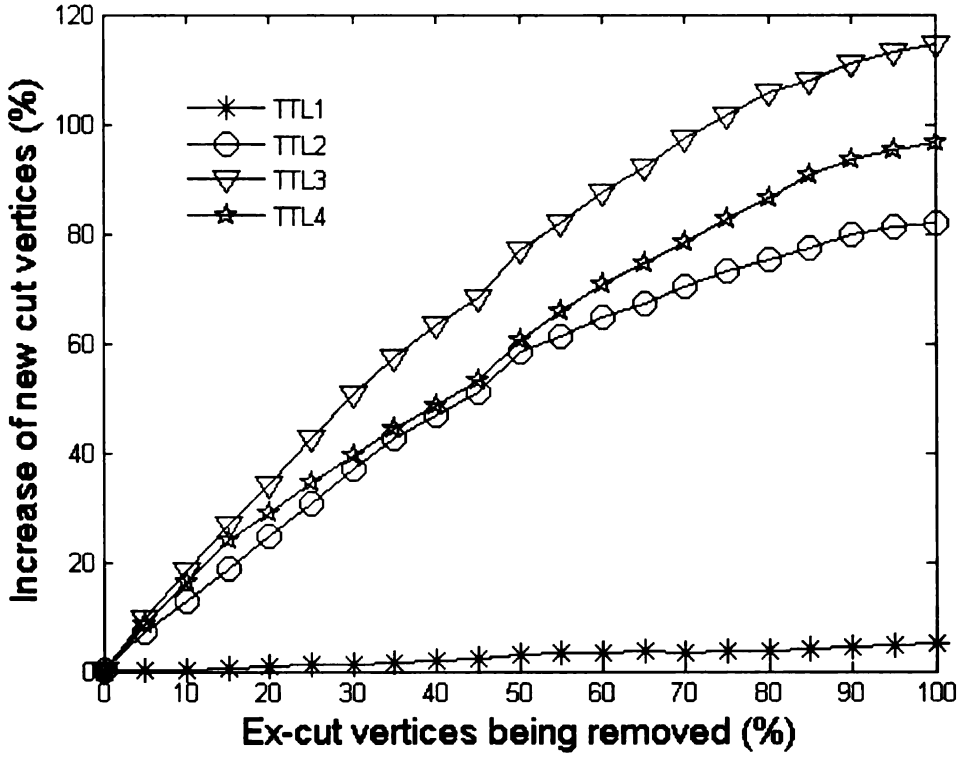


Figure 3.21. New cut vertices after ex-cut vertex failures with CAM (DSS trace)

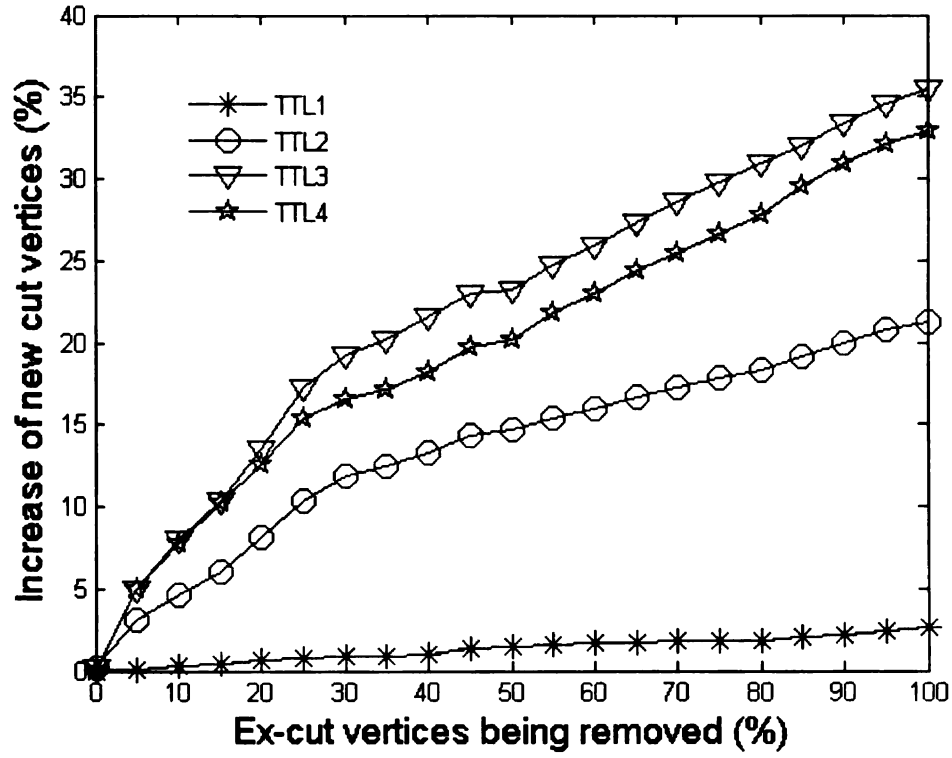


Figure 3.22. New cut vertices after ex-cut vertex failures with CAM (U. Oregon trace)

when cut vertices fail with the adoption of CAM. From the results, we can observe that search success rates are increased up to more than 500% of that of the network not adopting CAM in the DSS trace and around 50% in the U. Oregon trace. The greatest increase in the search success rate is achieved when the TTL value is one. The increase in the search success rate slows down with the increase of CAM TTL value. However, great increases of the search success rate (more than 300% in DSS trace and 35% in U. Oregon trace) can still be achieved even when the CAM TTL value is increased to 3, especially when a large number of cut vertices fail. The search success rate of the TTL value of 4 is very close to that of the TTL value of 4, which suggests the existence of the upper bound of the CAM TTL threshold.

We can also observe that, although CAM can neutralize all cut vertices, the search success rate still drops with the removal of cut vertices in the original graph. This may be due to the new cut vertices introduced by the cut vertex failure, as shown in Figure 3.21 and Figure 3.22. The number of components introduced by the cut vertex failure is shown in Figure 3.19 and Figure 3.20. The trend of the curves in Figure 3.19 and Figure 3.20 suggests more rapid increases in the number of components than that of the new cut vertices. The reason for the faster increase of components number may be that, along with the failure of the original cut vertices, new cut vertices also contribute to the new components.

If the TTL threshold value equals one, CAM will consider all nodes except those with only one or no neighbor as the cut vertices. During the cut vertex neutralization, a connection will be added between any two neighbors of the cut vertex. Although CAM will cut connections between the cut vertex and some of its neighbors, the connections being cut are always less than those being added. In P2P systems, this will induce the increase of the traffic cost in the search process. In order to prove this, we measured the traffic cost of the search process. The results are presented in Figure 3.25 and Figure 3.26. We observe that, in both the DSS and U. Oregon

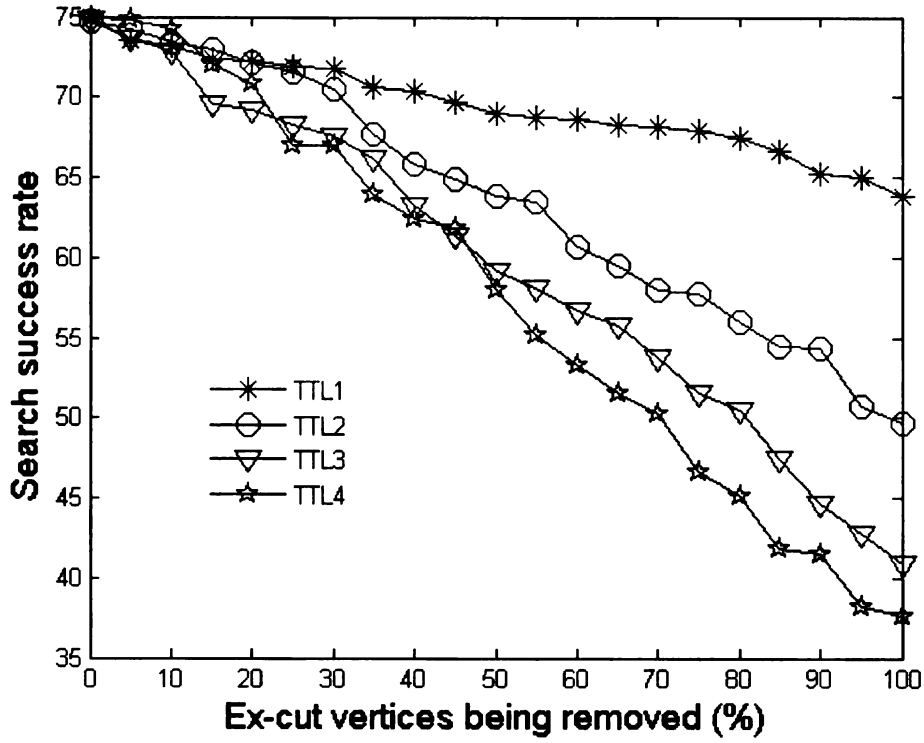


Figure 3.23. Search success rate with CAM (DSS trace)

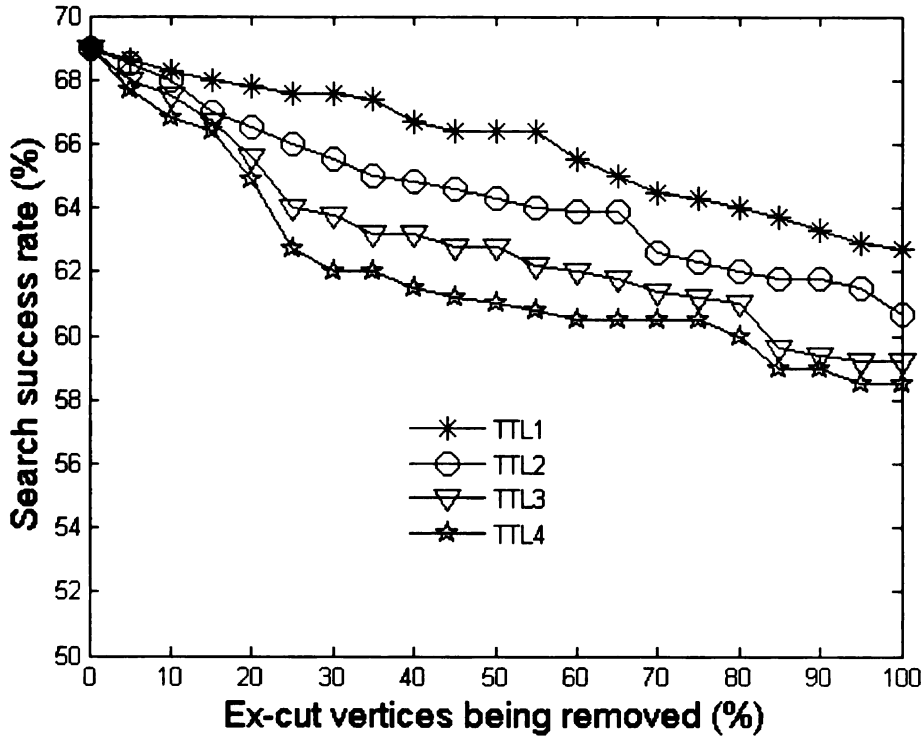


Figure 3.24. Search success rate with CAM (U. Oregon trace)

traces, the search traffic cost is much higher when the TTL is set to 1 than that of other cases: up to more than six times in the DSS trace and up to over 3 times in the U. Oregon trace compared with that when the TTL is set to 3. The search cost continues decreasing when the TTL increases from 3 to 4, but only a marginal increase can be observed. Considering all these factors, we recommend setting the TTL threshold value to 3.

Figure 3.27 and Figure 3.28 show how much traffic is offloaded after the CAM deployment in the DSS trace and the U. Oregon trace respectively. We can observe that the cut vertices are offloaded from about 32% to 53% in DSS traces, and 38% to 61% in U. Oregon traces. We can also observe that more traffic is offloaded from cut vertices when CAM TTL is small. This may be because with a smaller CAM TTL value, there is a bigger opportunity that the component probe messages that belong to the same CAM component could not meet in the same P2P node. Accordingly, more CAM components are detected with a small CAM TTL value and have a bigger opportunity to create new connections between nodes other than the cut vertex, and thus remove old connections of the cut vertex.

3.5.5 Influence of CAM on the Network Service under a Dynamic Environment

It has been shown that most overlay networks are highly dynamic. For P2P systems, different research studies suggest different values on peer lifetime in the system, but most of the studies show the lifetime of peers falls in the range of less than 10 minutes to less than a hour [90, 93, 51, 21, 25, 61, 82, 101]. Accordingly, we evaluate the influence of CAM on the network service in a dynamic environment.

We simulated the joining and leaving behavior of peers via turning on/off logical peers. Peer lifetime is generated according to the distribution observed in paper [101]. The lifetime is decreased by one when each second passes and a peer will leave in the

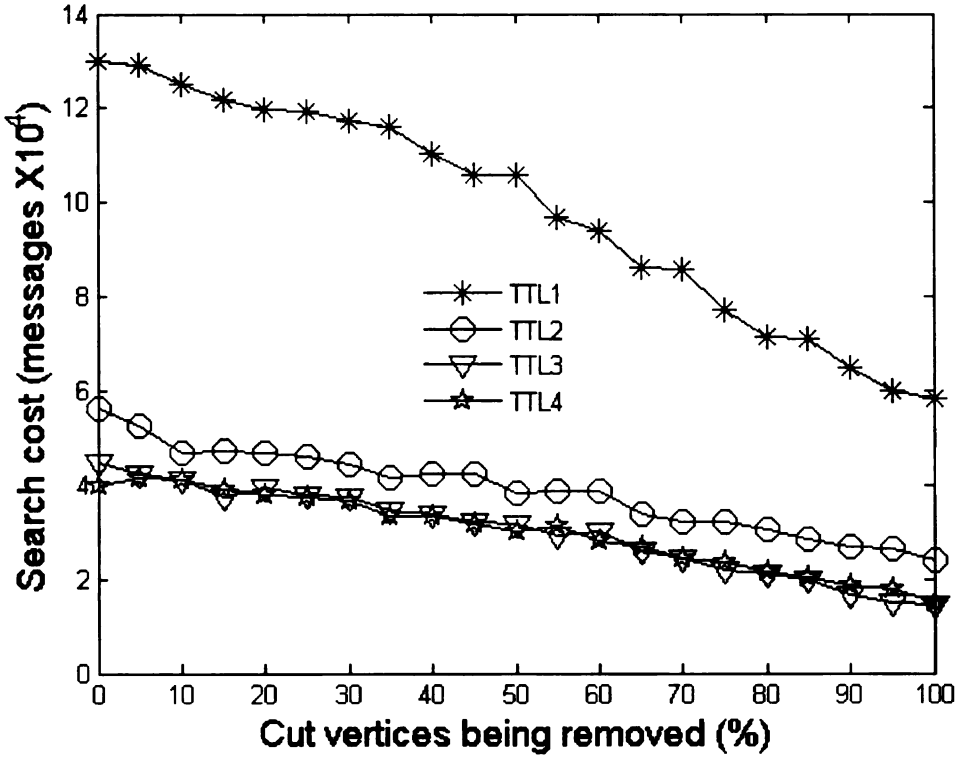


Figure 3.25. Overall search cost (DSS trace)

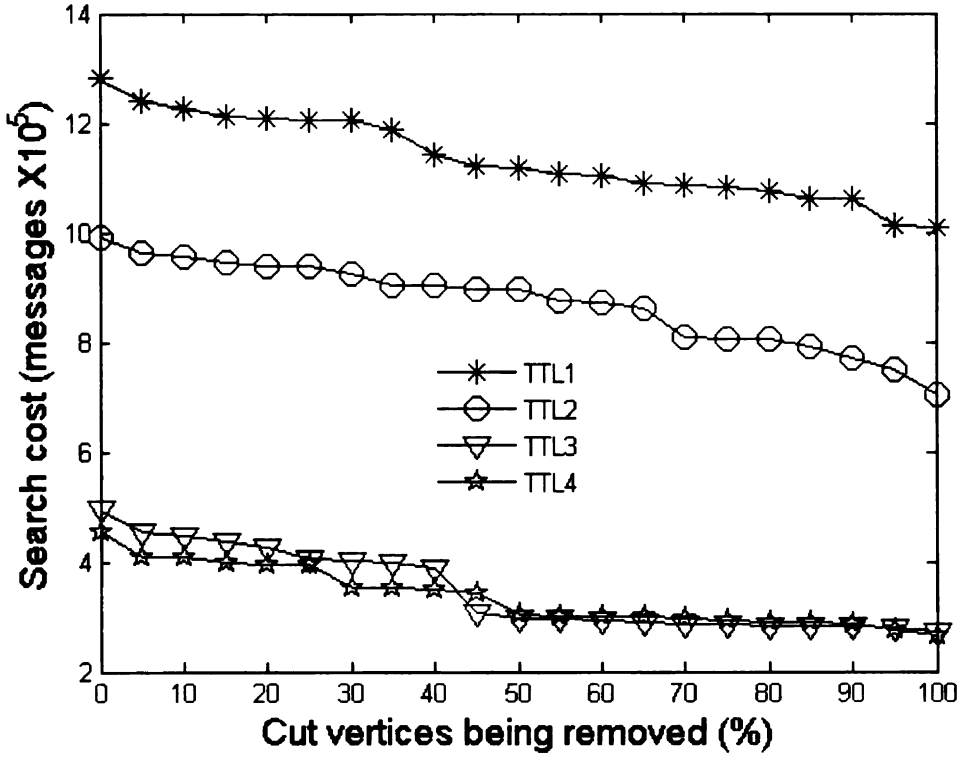


Figure 3.26. Overall search cost (U. Oregon trace)

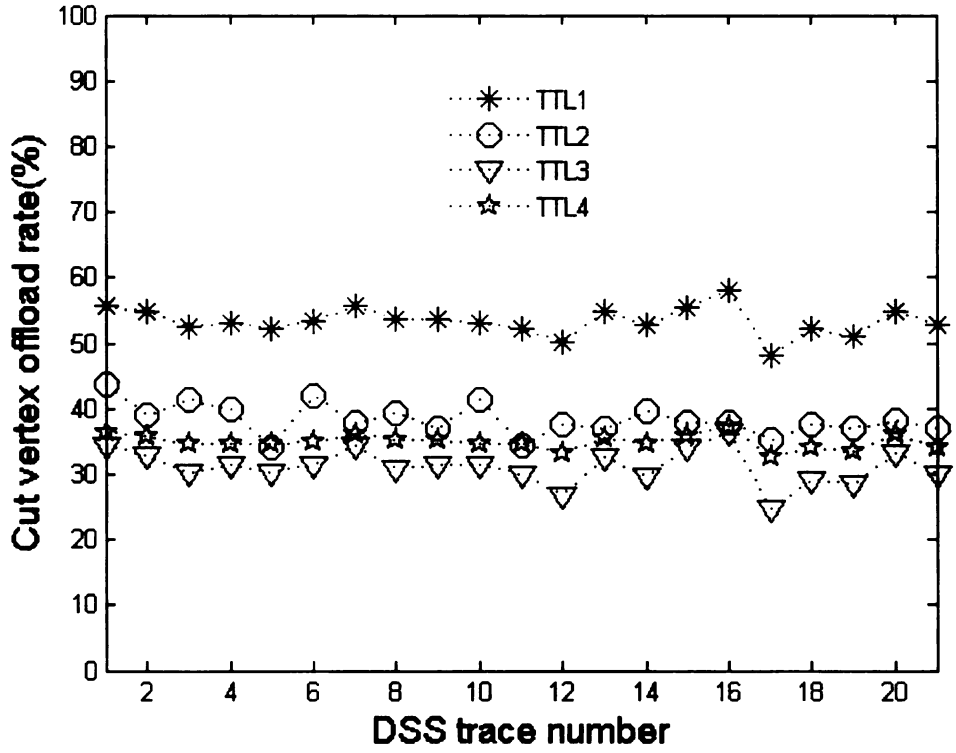


Figure 3.27. Cut vertex traffic offload (DSS trace)

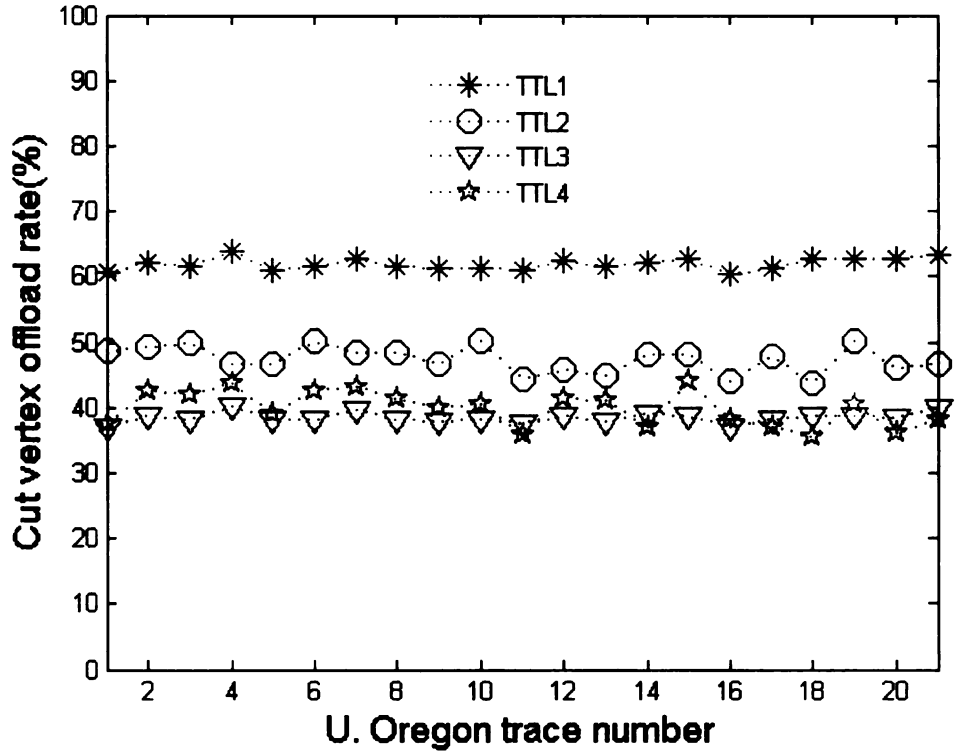


Figure 3.28. Cut vertex traffic offload (U. Oregon trace)

next second when its lifetime reaches zero. During each second, there are a number of peers leaving the system. To maintain the property of network topology during node leaving and joining processes, we randomly pick up (turn on) the same number of peers from the network to join the overlay system. For each peer, a *maximum-neighbor-connection* is predefined following its original degree in the DSS/U. Oregon traces. Each peer is required to keep the number of its neighboring connections no greater than its *maximum-neighbor-connection* during the simulation.

The performance of CAM in a dynamic environment is presented in Figure 3.29 and Figure 3.30. Against our expectation, the search success rates of the network in a dynamic environment are about 10% higher than that in a static environment in the DSS trace and about 6% higher in the U. Oregon trace. The increase in search success rate may be due to the fact that some cut vertices are turned into non-cut vertices with the connections added by the newly joined nodes.

3.5.6 Cut Vertices vs. High Degree Vertices

Previous studies show that high-degree nodes failure in a P2P system can also partition the network [90, 55]. Here we are interested in whether high-degree nodes in P2P overlay networks are cut vertices or vice versa. We investigate the percentage of the nodes that have higher than average degree vs. the percentage of cut vertices that have higher than average degree. Figure 3.31 shows the percentage of high degree nodes in the network. Figure 3.32 shows the percentage of the cut vertices that are high degree nodes. We can observe that, in both the DSS and the U. Oregon traces, cut vertices are more likely high degree nodes. At the same time, there are always some cut vertices that are not high degree nodes. We can also find a large portion of high degree nodes that are not cut vertices in U. Oregon trace: about 50% nodes are high degree nodes vs. only 2% nodes are cut vertices. Although a high degree node is more likely to be a cut vertex, high degree nodes are not necessarily cut vertices

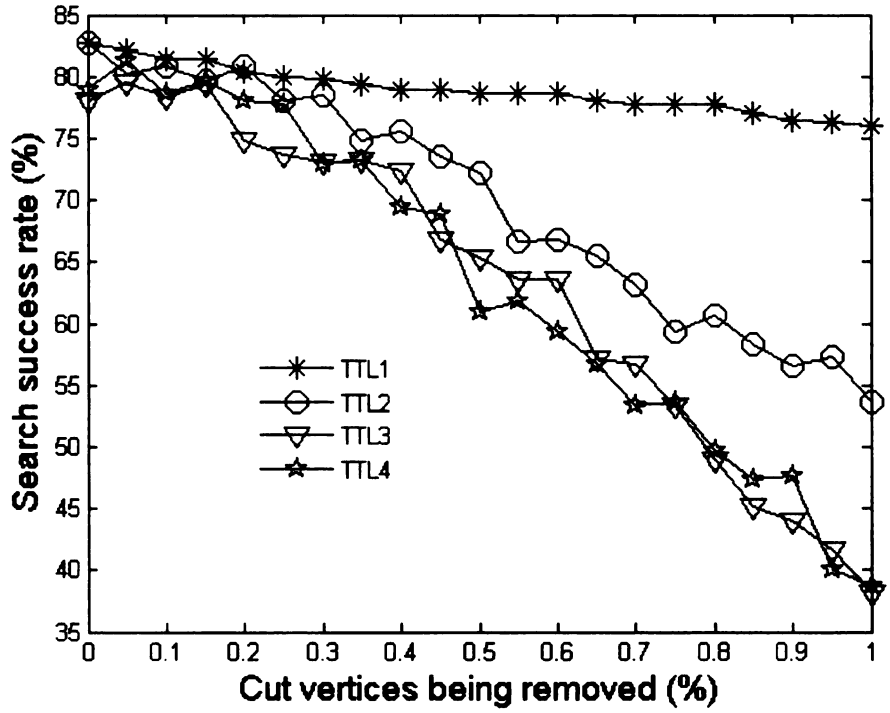


Figure 3.29. Search success rate vs. cut vertex failure after CAM operation in dynamic situation (DSS)

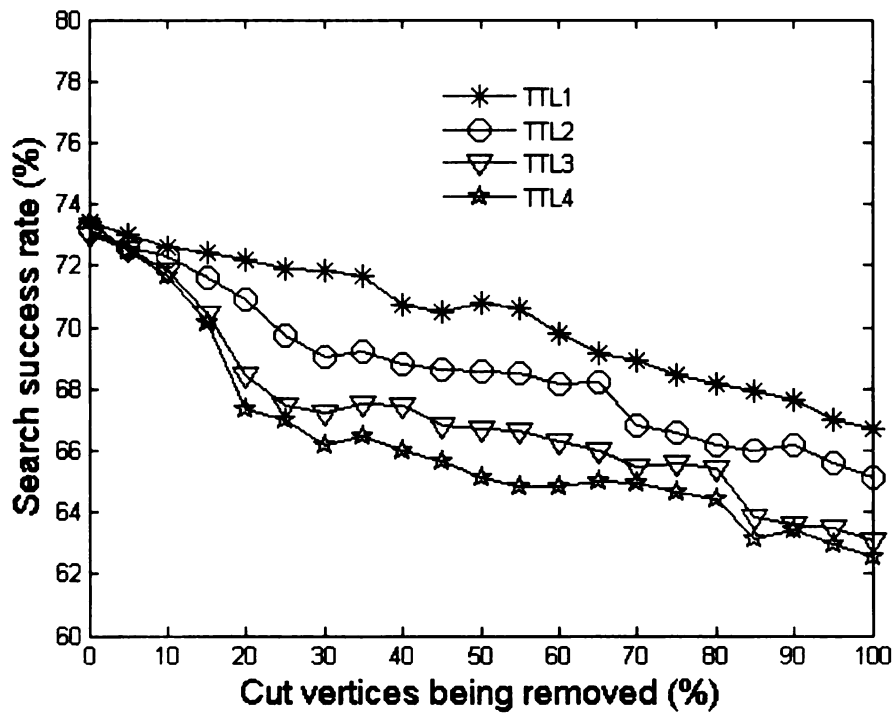


Figure 3.30. Search success rate vs. cut vertex failure after CAM operation in dynamic situation (U. Oregon)

and vice versa.

3.6 Summary

In this chapter, we investigate the cut vertex failure problem in an overlay networks. To our knowledge, we are the first to investigate this problem in overlay networks. We show that although the percentage of cut vertices in a network is not large, (around 16% in DSS traces and 2% in U. Oregon traces), the failure of these nodes can partition the network and downgrade the service availability. In our simulation, the failure of cut vertices can significantly reduce the search success rate (85% in the DSS trace and 48% in the U. Oregon trace).

In order to accurately detect and then neutralize the cut vertices in today's large-scale, highly decentralized, and dynamic overlay network, we proposed a fully distributed mechanism, CAM, to detect cut vertices in overlay networks. CAM can be applied to each node locally. We prove the correctness of the algorithm and also identify its accuracy with the simulation. CAM can successfully identify all cut vertices. The detection traffic overhead can be restricted by setting a small CAM TTL value, which may mistake a small number of non-cut vertices as cut vertices.

The results on the influence of the cut vertex failure and the performance of CAM are summarized in Table 3.2 (CAM TTL = 3). We can see that, in both traces, the cut vertex failure creates many new cut vertices as well as network components. This downgrades the network service (which is evaluated by search success rate). In addition, cut vertices process more traffic than common nodes.

With the CAM TTL threshold value set to 3, CAM can obtain a high accuracy rate of 96% (DSS traces) and 98% (U. Oregon traces) in detecting cut vertices. The neutralization step of CAM reduces network components created by cut vertex failure, which leads to a large increase in the quality of network service when the cut vertices

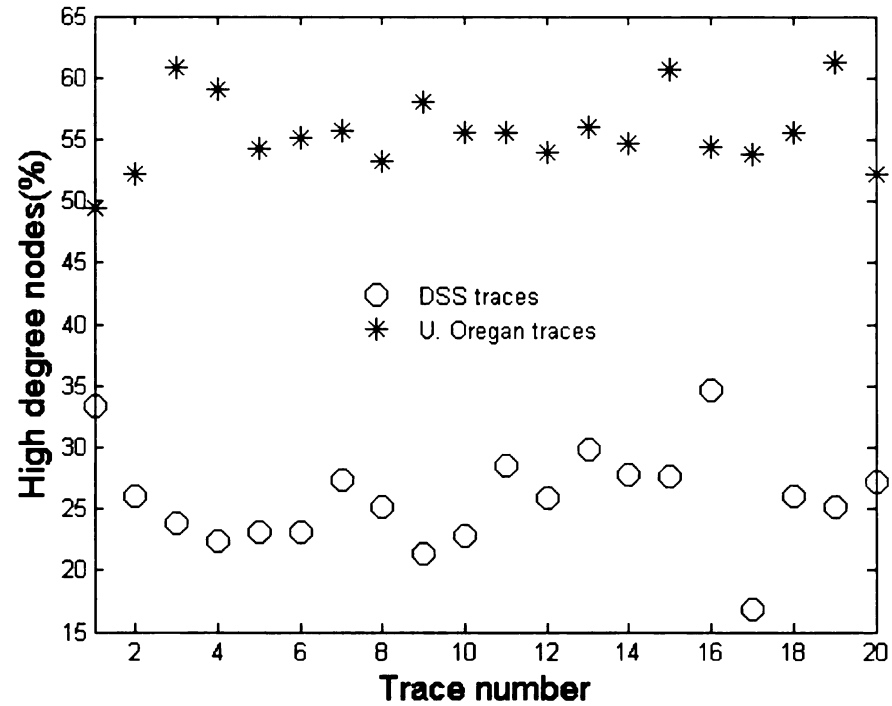


Figure 3.31. Percentage of high degree nodes

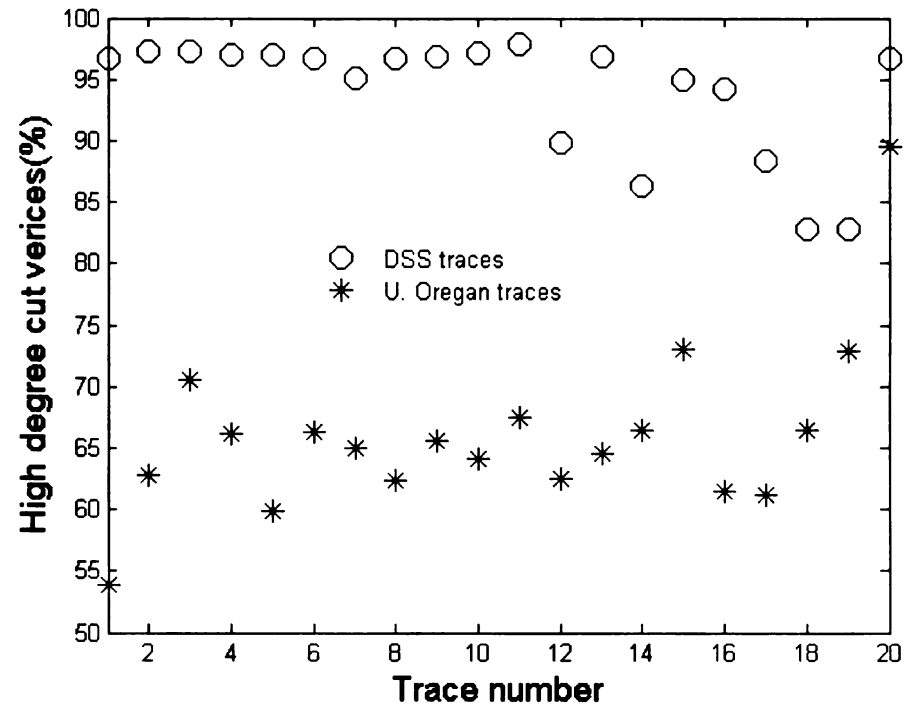


Figure 3.32. Percentage of high degree cut vertices

Table 3.2. Performance Evaluation of CAM

Trace Name		DSS Trace	U. Oregon Traces
Detection accuracy (TTL =3)		96%	98%
Influence of cut vertex failure	#of network components	7 times original components	4 times original components
	# of new cut vertices	34% of original cut vertices	12% of original cut vertices
	Search success rate	85% drop	48% drop
	Overload of cut vertices	7 times non-cut vertices	150% of non-cut vertices
Influence of CAM (TTL = 3)	# of network components	3.84 times original components	More than 2 times original components
	# of new cut vertices	114% of original cut vertices	35% of original cut vertices
	Search success rate	300% worst case increase	35% worst case increase
	Overload rate	32%	38%
High-degree node ratio (cut vertices)		55.6%	25.7%
High-degree node ratio (common nodes)		66%	93.7%

fail, as well as offloads the traffic handled by cut vertices.

CHAPTER 4

Response Loss in the P2P File Sharing Systems

In this chapter we discuss the response loss problem that is caused by the oscillation of P2P file sharing systems. This problem causes the loss of returned responses and thus hinders service availability of P2P file sharing systems. We start this chapter with the introduction of the search process of P2P systems in the next section. We then discuss the response loss problem and present our solutions for this problem, followed by the simulation studies to check the impact of the problem and the effectiveness of the solutions.

4.1 Search Process in P2P File Sharing Systems

Aimed at efficient utilization of Internet edge resources, P2P file sharing systems have received much attention since the emergence of Napster [9]. Today, peer-to-peer traffic has overwhelmed web traffic as a leading consumer of Internet bandwidth [89].

In unstructured P2P systems, file placement is random and has no correlation with the network topology. These systems generally adopt a flooding based search

mechanism among peers or super-peers: A source peer initiates the search process by sending a query to all its neighbors. Each peer that receives the query makes duplicate copies of the query and broadcasts to all its directly connected neighbors except the one that delivered the incoming message in each forwarding step. The duplication process only terminates when a predefined TTL value of the query reduces to zero, or a satisfying result has been found.

The flooding search mechanism is simple to implement, robust to node failures, and effective to partial/keyword search. A requestor can receive numerous responses from different responders. If some nodes fail and can not forward or respond to the query, the query can be forwarded and responded to by other nodes. On the other hand, although a structured P2P system greatly reduces the overhead of file locating, its weakness in partial/keyword search as well as the additional DHT maintenance cost impede its application in the real world. The simplicity and the robustness against node failures make the unstructured P2P system the prevalent model of the P2P file sharing systems.

Another benefit of the flooding search mechanism is that it provides anonymity for the query requestor: no information of the query requestor is included in the query request message. The peers who received the request only know the query message ID and the neighbors who sent this message to it. To reduce the search traffic in the system, the response delivery process in the unstructured P2P system does not adopt a flooding mechanism. If a peer receives a query message and can satisfy this query, it will send a response along the same path the query came from: each peer on the path sends a response to the first neighbor who sent the incoming query to it.

4.2 Response Loss Problem

In this section, we first discuss the response loss problem in detail and then present our solutions to solve the problem. Before the discussion of the response loss problem, we would like to define forwarding neighbor and primary forwarding neighbor.

Definition 1: During the query process, if a neighbor of peer p forwards a copy of this query message to p, then this neighbor is p's *forwarding neighbor*.

Definition 2: Among all the forwarding neighbors of p, the first one that forwards a copy of this query to p is called *primary forwarding neighbor*. In existing P2P systems, p always sends responses back to its primary forwarding neighbor.

The P2P system is a highly oscillating system in that peers come and go frequently, and even when they are in the system, they may adjust their connections with a high frequency. Previous studies show that a peer's lifetime varies from less than 10 minutes in FastTrack [93] to 60 minutes in Gnutella and Napster [26]. Uptime of logical connections is obviously even shorter than individual peers, from 1 minute to less than 24 minutes [37]. In addition, many new techniques trying to improve the performance of P2P systems also require peers to adjust their connections to find better neighbors [98] or achieve optimized overlay topologies [30, 98, 69]. This further increases the dynamic nature of P2P systems.

To reduce the response traffic, responses will be sent back to the requestor along the query incoming path instead of by flooding. In addition, to keep requestor anonymity, no requestor information is stored in the requestor/response message. Peers on the response path only depend on the local knowledge of their primary forwarding neighbor to route the response. Therefore, the response message will be thrown away if the primary forwarding neighbor fails. For example, in Figure 4.1, peer S issues a query and peer T has a response for the query. The response is sent back along the incoming query path $A \rightarrow B \rightarrow C \rightarrow D$. If any one of A, B, C or D leaves after it forwards the query or any connection in any two of these nodes changes,

a response loss occurs. In this case, network resources for both response return and original query propagation are wasted.

File locating process is an important service provided by P2P file sharing system. The file information is included in the returned responses. The loss of returned responses is thus downgrade service availability of P2P file sharing system. One may argue that loss of some response messages is not a major issue since multiple responses may be found for one query. Nevertheless, our simulations show that the loss is not negligible, as we found 35% of the responses are lost in a P2P system. In existing P2P systems, the most popular search method is the keyword search, which suggests the search results will not be exactly what a user expects most of the time. Increasing the number of search results for a query results in increasing the likelihood that a user will find what he/she really needs. Also, it is obvious that all network resources consumed to find the response are wasted when a response is lost.

4.3 Solutions for Response Loss Problem

In order to improve service availability, the author propose three techniques, redundant response delivery(RRD), adaptive response delivery(ARD), and extended adaptive response delivery(e-ARD), to solve the response loss problem and guarantee the successful delivery of responses. All of the three techniques are based on the observation that the amount of forwarding neighbors of a peer in a flooding based query mechanism is more than one in most cases. However, the third mechanism presented of e-ARD also takes care of the case that only one forwarding neighbor is available during the query process.

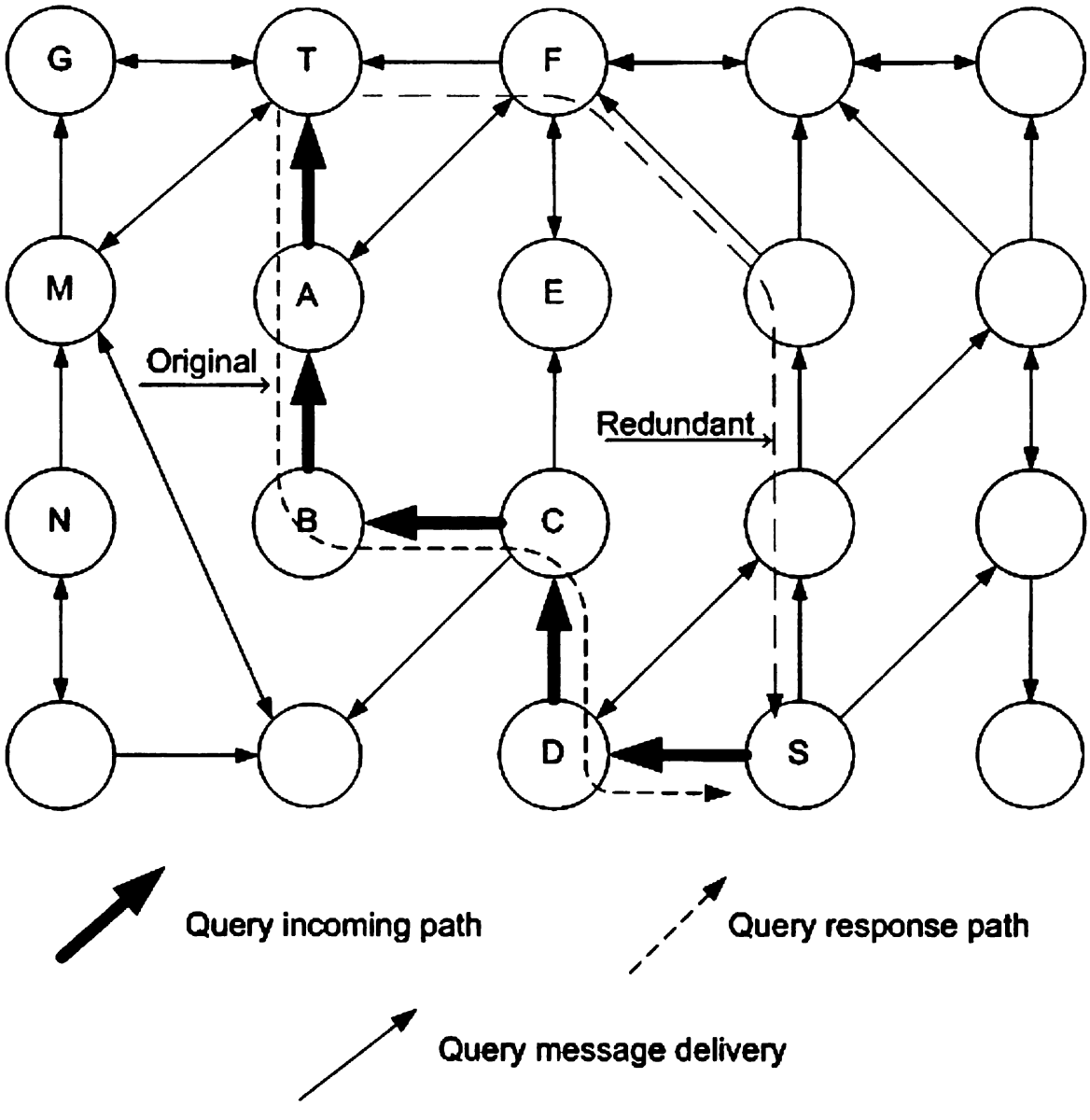


Figure 4.1. Response loss problem in P2P System

4.3.1 Redundant Response Delivery (RRD)

Making a backup copy of vulnerable/critical components of the system is a common technique to improve the system's fault tolerance. The redundant response delivery (RRD) scheme also tries to alleviate the response loss problem via backup paths. The success of using backup paths here is based on whether the same query message can be forwarded to the same node from more than one path. This is intuitively true for a flooding query delivery system in a highly connected network. It is also attested to by the result of our simulation: without specifying how the query is transferred except flooding and network topology settings being kept consistent with those of the real world, RRD does significantly reduce response loss rate compared with the original response delivery mechanism.

What needs to be noted here is that RRD is totally different from a flooding/broadcast delivery system in that in the RRD scheme, only the responder issues responses to multiple selected neighbors, while other nodes in the path only forward the response to its primary forwarding neighbor. In this sense, RRD is more like the k-walker mechanism discussed in [41, 73]: the requestor distributes several walkers in the system; each will detect the node that can satisfy the query via the DFS mechanism. However, nodes in RRD forward responses according to the recorded path instead of a randomly chosen neighbor.

RRD is illustrated here with an example. Figure 4.1 supposes that peer S floods a query, and peer T has a response for the query. According to the Gnutella 0.6 protocol [11], if peer A is T's primary forwarding neighbor, T will send back the response via A and discard the same query message from other forwarding neighbors, such as G, M and F. The path for T to send back a response to S is $A \rightarrow B \rightarrow C \rightarrow D$. If any of the peers on the path fails or leaves, the response will be lost. Instead of returning a response through one path, the RRD scheme adds one or more redundant response paths. For the same example, in addition to sending a response back immediately to

A, T also selects one of the other forwarding neighbors (F, G, and M) as back up neighbors with redundancy probability γ , and sends a response back through each of the selected neighbors. All other peers except the responder will drop response messages with the same message ID previously received.

The probability γ is a performance control factor that will affect the performance gain of RRD. Assume a node in the system has c neighbors. There are h_j nodes in each path starting from neighbor j. The failure probability of each node is p_{ji}. In addition, assume there is no overlap among these paths. The probability that a response returned by this node is lost is:

$$f_{loss} = \Pi_{j=1}^{c}\left(1 - \gamma \times (1 - P_{ji})^{\sum_{i=1}^{h_j}}\right)$$

It is obviously that the increase of γ results in the decreases of f_{loss}. Users can reduce the probability of response loss by setting a larger value of γ . However, f_{loss} does not reduce linearly due to the existence of path overlap and the variance of p_{ji}. In order to further check the influence of in RRD's performance, the author has deployed a series of experiments on it and presents the results in Section 4.5.3. RRD creates very limited computation overhead for local nodes in choosing multiple neighbors to send back the responses. Nevertheless, it creates extra network traffic overhead, which is increased with the increase of redundant paths, if we ignore the path length, path overlap, and traffic variations among paths. However, the response traffic along one path is very limited, in order of $O(1)$ and is restricted by the maximum TTL, which is suggested as 7 [11]. The extra response traffic along redundant paths is also limited and also in order of $O(1)$, restricted by maximum TTL and γ.

4.3.2 Adaptive Response Delivery (ARD)

In the adaptive response delivery scheme, peers deliver the response to a different forwarding neighbor when the primary forwarding neighbor fails. In order to adap-

tively deliver response upon the node failure, each peer keeps a forwarding neighbor list for each query message within a certain period of time. The format of each item in the forwarding neighbor list is <Message ID, Forwarding Neighbor>. The primary forwarding neighbor of this peer will not be recorded in its forwarding neighbor list.

The basic idea of ARD is as follows. If a peer in a response path finds that the next hop cannot be reached, he will check his forwarding neighbor list to see if other neighbors also forwarded the same query message to him. If so, he will select in the list the first that arrived and reroute the response to this forwarding neighbor. Otherwise, he will send a failure message with the query message ID and the information about the unreachable neighbors to his previous hop who will try to reroute the response. The Gnutella 0.6 protocol suggests that the TTL of a response should be set to at least the hops value of the responding query plus 2 [11]. ARD needs to set a larger TTL in order to route the response back to the initiator. The response message is terminated only when it returns to the originator or its TTL value is reduced to zero.

Take Figure 4.2 as an example for ARD. Peer A issues a query that is flooded to many peers. Peer P receives his first copy of the query from F and finds that he has a response for the query. Assume the query path for the query that first reaches P is $A \rightarrow B \rightarrow F \rightarrow P$. We consider the case where B fails or leaves after he forwarded the query to F. When F receives a response from P and tries to send the response to B, F will find that he cannot reach B. With ARD, F will first check his forwarding neighbor list for this query and select from the list the first to arrive. In this example, F chooses G as his new forwarding neighbor. When G receives the response, he again finds that B is non-reachable and picks E to return the response. Peer E will eventually send the response back to A via C or D depending on which one first forwards the query message to E.

Another case is that both B and G leave or fail. Peer F will inform his previous hop, P, that he cannot deliver the response through all his forwarding neighbors, i.e.,

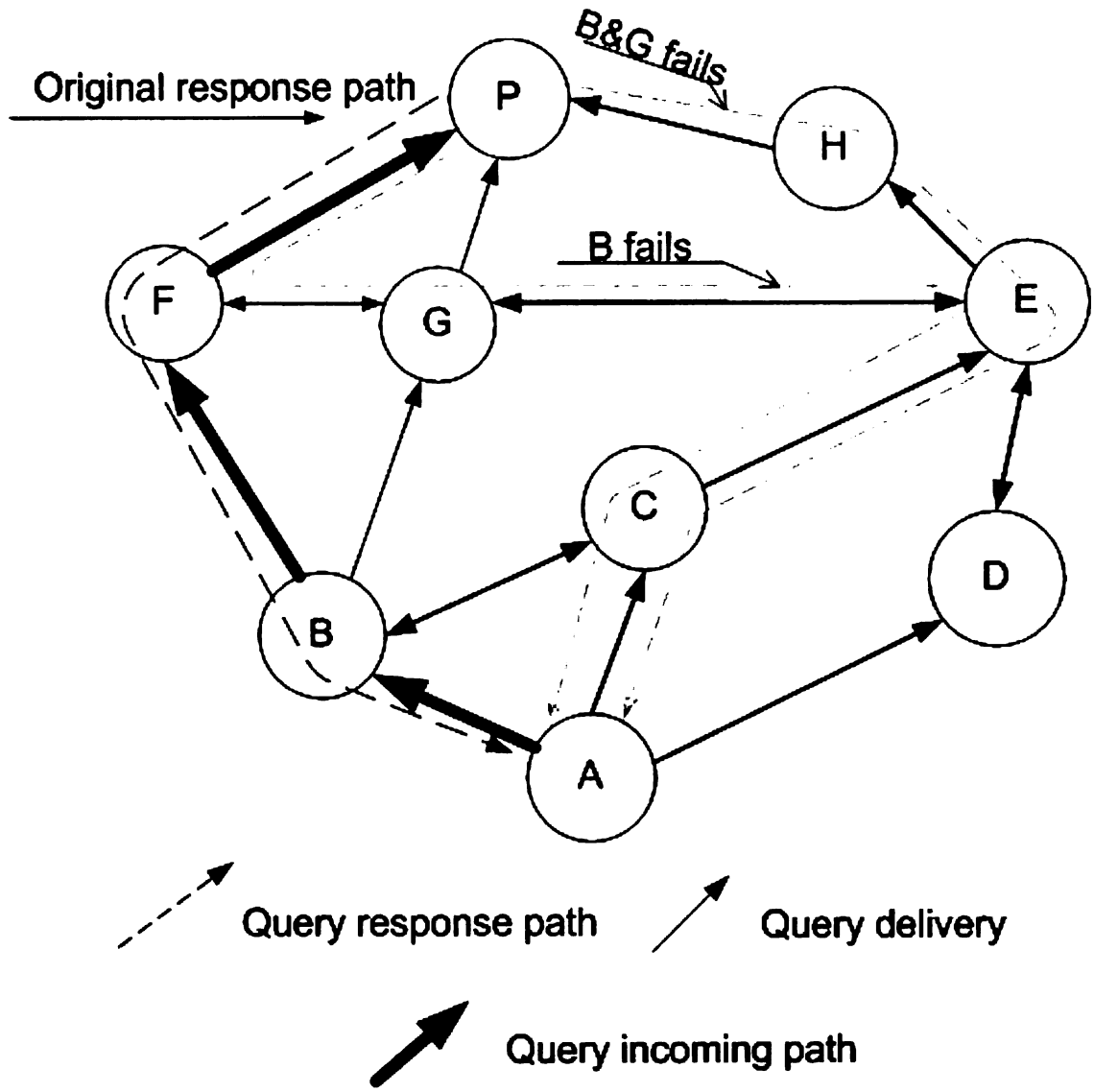


Figure 4.2. Adaptive response routing

B and G. Peer P will pick H to reroute the response through. Although G is also in P's forwarding list for this query, P will not select G since P has been informed by F that G is not reachable.

The update frequency of the forwarding neighbor list should also be considered. Update frequency is $\frac{1}{T_{wait}}$ if we define T_{wait} as the average lifetime of a forwarding neighbor record in the forwarding list. A forwarding neighbor record will be removed from the list if T_{wait} expires. Therefore it is important to select the proper value of T_{wait}. If T_{wait} is set too low, a forwarding neighbor record may be removed before the response is sent back to this peer and ARD will not be very effective. If T_{wait} is set too high, the overhead for each peer to keep the forwarding neighbor lists will be very high. The value of T_{wait} is related to the average response time that is defined as the time difference from when a responder sends a response to when the initiator receives the response.

We will show that ARD, a simple resilient forwarding technique, is able to increase response success rate with low traffic cost, especially when many peers fail in a system. Compared with RRD, the additional traffic overhead in ARD is small because ARD only reroutes a response if necessary. The simulation shows that the traffic caused by the adaptive delivery algorithm is close to that of the original response delivery mechanism.

One may also be concerned that ARD may perform poorly in the case that only a few or none of the peers have multiple forwarding neighbors. However, in the case of RRD, this is rare in the unstructured P2P system where the flooding mechanism is being used. Even when this case happens, a peer can return the response to the node that delivered it, which can provide a second chance to the response message by rerouting it to some other peers.

4.3.3 Extended Adaptive Response Delivery (e-ARD)

The success of RRD and ARD relies on how many duplicate copies of a query message can be sent to one node via its different neighbors. Neither RRD nor ARD will function effectively in the case where only one forwarding neighbor is available for a peer. Although this situation rarely occurs in most of the existing unstructured P2P systems, it may happen more frequently with the adoption of new techniques targeted to cut query traffic overhead by removing unnecessary node connections and limiting query message duplication.

To address this limitation, the ARD technique is extended to be effective even in the case that there is only one forwarding neighbor available for a peer in the system. The new technique is called extended Adaptive Response Delivery mechanism (e-ARD). In the e-ARD mechanism, an IP address used for adaptive response delivery is appended to each query message. When the next hop neighbor in the response transfer path fails, the peer can, in addition to forwarding the response to an alternative neighbor, forward the response to the node of this IP address. One important issue that cannot be ignored here is that the requestor anonymity must be maintained during the process, that is, with this additional information of a backup IP address, a third party cannot tell the identity of the query requestor. Therefore, the address of requestor can not be simply appended to the query message. In e-ARD, the requestor anonymity is achieved with the introduction of the backup response delivery agent (bRDA).

backup Response Delivery Agent (bRDA)

In the e-ARD scheme, a type II query request/response message is defined by adding a new field of bRDA address based on the query request and response message specified in Gnutella 0.6 [11]. The type II query request/response message is shown in Tables 4.1 and 4.2. During the query process, type II request/response messages will be

Table 4.1. Type II Request Message

	Minimum Speed	bRDA Addr	Search Criteria	Extension blocks
Byte Offset	0 1	2 5	6- end with 0x00	Optional

issued. When a requestor makes a query request, it will put its own address in the field of the bRDA address of the request message and broadcast it to all its neighbors. The peer who receives the query will decide with a wrapping probability of whether it will replace the bRDA's IP address in the query message with its own IP address. The node that decided to append its own IP address to the query message is called the backup response delivery agent (bRDA). As a requestor always appends its address to the query message, a requestor is a bRDA.

When a bRDA appends its own IP address in the query message, it also stores the old IP address in the query message in its forwarding neighbor list. The forwarding neighbor list of each node in e-ARD is also extended to add the address of the previous bRDA. Each entry in the forwarding neighbor list is in the format of <message ID, pre-bRDA address, forwarding neighbor 1, forwarding neighbor 2, etc.>. For the node which is not the bRDA, the pre-bRDA address of the related record is NULL. A responder will copy the bRDA address of the received query to its response message. During the response delivery process, a node will check its neighbor list and see if it is the bRDA of the related message. If it is, it will remove the IP address from the response message and append the previous bRDA address stored in its forwarding neighbor list to the response message. Here update frequency of the forwarding neighbor list is also an important factor for the effectiveness of e-ARD. The impact of this update frequency on the average query response time is comparable to that in ARD.

Table 4.2. Type II Responder Message

	Number of Hits	Port	Responder Addr	Speed	bRDA Addr	Result Set
Byte Offset	0	1 2	3 6	7 10	11 14	15-

Wrapping Probability α

It is important to determine a proper value of wrapping probability. If it is too big, most of the nodes in the query-incoming path will become bRDA. This will consume too many local resources and reduce the response return efficiency. One extreme case would be to set as 1. This reduces e-ARD to ARD since the backup path address appended to the query message will be just the address of the primary forwarding neighbor address of each peer. On the other hand, if α is too small, then one can always consider that the bRDA address in the query message is the address of the requestor and thus cannot guarantee the anonymity of the requestor. Previous studies show that the older a peer is, the longer it is expected to remain in the system [25]. With the above considerations in mind, the definition of α should satisfy the following requirements:

- There are one lower bound α_0 and one upper bound α_1 for α such that $\alpha_0 \leq \alpha \leq \alpha_1$.
- α increases as peer lifetime increases.
- The accelerated speed of α decreases as peer lifetime increases.
- Two peers with similar lifetime should also have similar values of α.

Therefore, we define α as:

$$\alpha = \alpha_0 + (\alpha_1 - \alpha_0) \times (1 - \frac{c}{\phi(T_{on}) + c}) = \alpha_1 - \frac{c(\alpha_1 - \alpha_0)}{\phi(T_{on}) + c}, \tag{4.1}$$

where α_0 is the lower bound, α_1 is the upper bound, and c is a fixed value that can be set by the user to further decide the distribution of α across $[\alpha_0, \alpha_1]$ given the scope of T_{on}.

In order to choose α_0, we introduce f as the probability of the bRDA address being changed at least once by any of the nodes in the query path. If we assume that it takes $k(k > 1)$ hops for a query to be satisfied, then $f = 1 - (1 - \alpha)^{k-1}$. The goal here is to find out the minimum value of α that makes f close to 1. It is obvious, however, that f monotonically increases respect to α. Therefore, we consider that f is close enough to 1 if it is greater than 90%. Considering that the TTL of a query message in most of the unstructured P2P systems is set as 7, the lower bound of α can be 0.32. To make the analysis simpler, we assume that the node lifetime is not less than 1 minute, and take 0.35 as the value of α. To further test out the lower bound, we consider a case where all nodes in the query path take 0.35 as the value of α and that query path length is only as short as three hops. We can see f will be 0.578 which means there is still a fairly large chance for the bRDA address to be changed. It is obvious that the upper bound of should be a value of (0.5, 1). We thus recommend α_1 of 0.75 which is in the middle of (0.5, 1). Replacing α_0 and α_1 with 0.35 and 0.75, we can simplify the formula to $\alpha = 0.75 - \frac{0.4c}{\phi(T_{on})+c}$.

Without the second requirement, $\phi(T_{on})$ would be a linear function of T_{on}. With the last requirement, we need to add a non-linear factor of T_{on} to tune the acceleration of α. $\phi(T_{on})$ thus turns into a function with the format of $T_{on} \times \Delta T_{on}$, with ΔT_{on} being the non-linear function of T_{on}. As research indicates that the average peer lifetime ranges from less than 10 minutes to 60 minutes [93], we check the power function, log function, and exponential function with T_{on} from 1 to 60. The trend of how α increases under different non-linear functions is shown in Figure 4.3(b), assuming $c = 70$. It is obvious log function is a good choice in that it does not increase as fast as exponential and power function, while not as slow as linear function. From

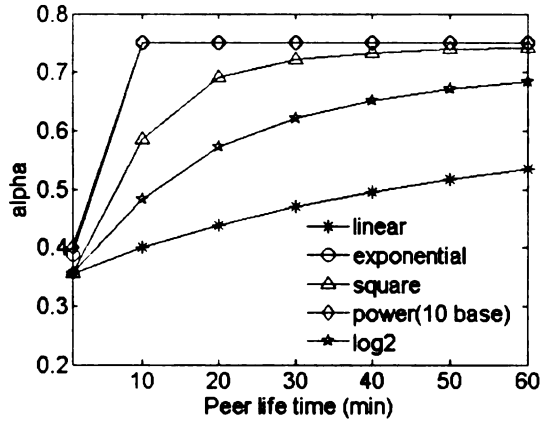


Figure 4.3. Alpha under different nonlinear functions (assume $C = 7$)

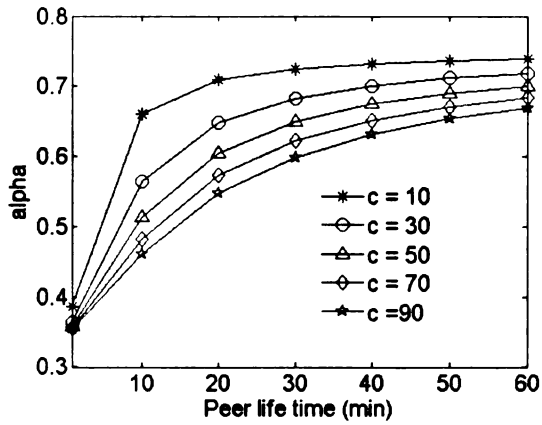


Figure 4.4. Scope of alpha varies upon different C values

Figure 4.4, we can observe how different c values affect the scope of α. As the value of peer lifetime in our simulation is from 1 minute to 60 minutes, we take $c = 70$, which renders α fall within the scope of [0.36, 0.68].

Adaptive Response Delivery

When a responder forms a response message, it will insert the bRDA address it obtained from the query message in the response message. When a node receiving

the response message finds that its primary forwarding neighbor for this response message fails, one of three situations occurs: (1) There are other neighbors in its forwarding neighbor list and they are still alive. The node will choose one of these available forwarding neighbors according to the ARD and forward the response back through it; (2) There are no other forwarding neighbors for this response or all the neighbors on list for this query are off line. The node will use the bRDA address appended to the response message to build a UDP connection to this agent and forward the response message to it. (3) The backup response delivery agent itself is failed, behind a firewall or any other circumstances that the UDP connection cannot be built. In e-ARD, the node informs its previous hop node and ask it to choose other node to forward the response message as in ARD. At the same time, a backup response delivery agent will always need to replace the IP address in the response with its stored IP address when it receives a response. (While a TCP connection may still be built up via a mechanism similar to the push operation in Gnutella network in this case, the author does not recommend such an operation since the cost to create such kind of connection is too expensive for a one time communication.)

Figure 4.5 illustrates the effect of the backup delivery agent in an adaptive response routing of the e-ARD scheme. P and K, which fail during the response return process, are the primary forwarding neighbors of peer O and N respectively. In Figure 4.5.*a* and Figure 4.5.*b*, O is on the query incoming path. Therefore, whether O is a bRDA or not, it always sends a response to a bRDA (here is A) which is also on the query incoming path when no other forwarding neighbor is available for O.

In Figure 4.5.*c* and Figure 4.5.*d*, O sends a response to its forwarding neighbor N, which is not on the original query incoming path. In Figure 4.5.*c*, N is not a bRDA itself and thus it still sends a response back to agent A, whose address is appended in the response message. The response message has been rerouted to the original query incoming path since the bRDA address appended to the response message

is the address of the agent that is in the original query incoming path. In Figure 4.5.*d*, N itself is a bRDA. It changes the bRDA address field according to the bRDA address stored in its forwarding neighbor list and sends a response message to M accordingly. Here the response message did not reroute to the original query incoming path although the appended bRDA address in the message is also the address of an agent who is in the original query incoming path. In all cases, the length of the response path is reduced due to the introduction of bRDA.

The e-ARD mechanism can be used in P2P systems adopting a DFS based search mechanism or other non-flooding based search technique such as k-walker. In addition, forwarding responses to a backup response delivery agent in e-ARD can reduce the number of hops in a response return process and thus further cut both response time and traffic cost in the response process. This will partly pay for the overhead of creating a UDP connection.

4.4 Simulation Methodology

A large-scale network simulation has been done to evaluate the performance of our techniques. Both Gnutella [11] and KaZaA [7] are popular unstructured P2P systems. As the documentation about KaZaA is not available to the public, the simulation is based on a Gnutella-like context.

4.4.1 Performance Metrics

RRD, ARD and e-ARD is evaluated using different performance metrics. The first important performance metric, response return rate, evaluates the effectiveness of the proposed techniques. Response return rate is defined as follows:

$$Response\ return\ rate = \frac{all\ responses - responses\ lost\ in\ backward\ path}{all\ responses}$$

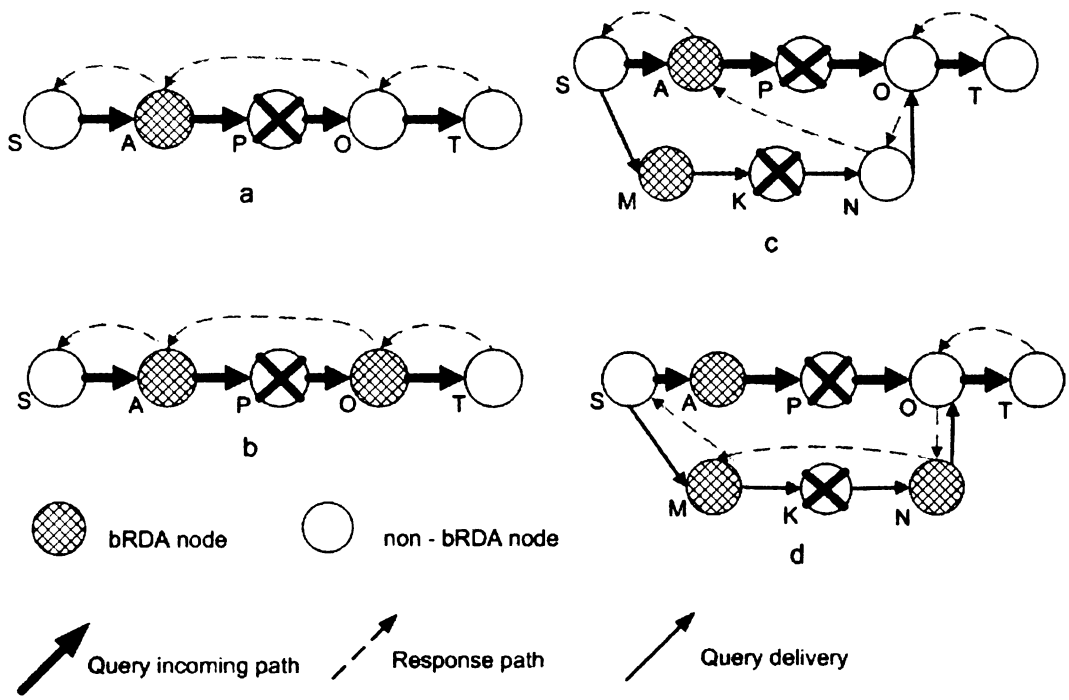


Figure 4.5. Extended adaptive response delivery and the effect of backup response delivery agent

The second metric, response traffic cost, evaluates the efficiency of the proposed mechanisms. The response traffic cost is the traffic of sending a response message from a responder to a query initiator. The response traffic cost is computed as a function of consumed network bandwidth and other related expenses. Specifically, we assume all messages have the same length in this work. When messages traverse an overlay connection during the given time period, the traffic cost (T_c) is given as: $T_c = M \times L$, where M is the number of messages that traverse the overlay connection, and L represents the number of physical links underlying the overlay connection. The target of all optimization mechanisms is to achieve high response return rate with low cost.

The third performance metric that is evaluated is the quality of service: here we refer the quality of service(QoS) to whether the users are satisfied with the service of the system. We use response time, which is commonly used in P2P researches, to evaluate the QoS performance of the system. The response time is defined as the time from when a query is issued by a requestor to when the response is received by the requestor. The delay of a logic link are computed as follows: first, we take the average delay of the physical links as the basic unit in the simulation, and compute the unit-less raw "delay" of a logic link. Assume the mean delay for all the physical links is $E(T)$. If the sum of the delays of physical links under a logical link is $Sum(T_i)$, the raw delay of a logical link is $Sum(T_i)/E(T)$. Next, to map the P2P network latency into our simulation, we generated a group of overlay link delays according to the observation in paper [26]. We then sort the overlay link according to their raw "delay" and assign the generated delay to each link.

4.4.2 Parameter Settings

The system configuration are summarized in Table 4.3. During each group of experiments in the simulation, the value of some parameters may be changed to test the

Table 4.3. Parameter Setting of the Simulation

Name	Default	Description	Name	Default	Description
Logical network size	8000	Number of peers in the logical network	Redundancy probability	0.2	Probability of responder to select each neighbor as redundant response backward neighbor in RRD
Neighbors per	6	Average number of neighbors each peer has	Wrapping const	70	Constant to adjust accelerant of wrapping probability α in e-ARD
Query rate	0.3	Average number of queries each peer issues per minute	Forwarding neighbor list uptime	Average response time	Time forwarding neighbor record expires
Peer lifetime	10	Average peer online time(minutes)			

performance of three techniques under different circumstances. Note that for the network size, we have done simulations based on network sizes of 2000, 3000, 5000 and 8000. The results from networks with different sizes are consistent which confirms that our techniques are scalable. Here we show the simulation results from a logical network size of 8,000.

4.4.3 Simulation Framework

P2P networks are overlay networks. In order to capture the overlay feature, two types of topology, physical topology and logical topology, are generated in the simulation. The physical topology is generated to represent the characteristics of the Internet under the P2P system. The logical topology represents the overlay P2P topology built atop of the physical topology. All P2P nodes are in a subset of nodes of the physical topology. The communication cost between two logical neighbors is calculated based on the physical shortest path between this pair of nodes.

Network topology and density have a large impact on how many neighbors will send duplicate query messages to a peer. The network topology in the simulation should be consistent with the real network topologies to accurately evaluate the performance of the proposed mechanisms. Previous studies show that both large scale Internet topology and P2P overlay topologies follow the small world and power law properties, and topologies generated using the AS Model have such properties [90, 97, 100, 104]. BRITE is one of the topology generation tools that provide the option to generate topologies based on the AS Model [2]. A physical topology of 22,000 node are generated based on BRITE. Four logical topologies with 2000, 3000, 5000 and 8,000 nodes are then generated based on the physical topology. In order to check the networks with different connectivity, the average number of neighbors of each node (of logical topology) are varied from 4 to 10.

The unstructured P2P system with flooding search mechanism is the prevalent model of existing P2P systems; accordingly, the proposed techniques are evaluated mainly based on this category of systems. Observations presented in paper [51] pointed out that the object popularity distribution in a P2P system does not follow a Zipf distribution like that of www objects mentioned in [17, 23].

Here we simulate the flooding search process via executing a Breadth First Search (BFS) based algorithm from a specific node. A search operation is simulated by

randomly choosing a peer as a requestor and requesting keywords according to the distribution described in paper [51]. As for the query dispatch frequency, we set every node in the system to issue, on average, 0.3 queries per minute, which is calculated from the data collected by K. Sripanidkulchai [98]. For instance, 12,805 unique IP addresses issued 1,146,782 queries in 5 hours.

Based on generated P2P network topologies, the joining and leaving behavior of peers are simulated via turning on/off logical peers. The peer lifetime is generated according to the distribution observed in paper [90]. The lifetime is decreased by one with each second, and once a peer's lifetime reaches zero, it will leave the following second. During each second, there are a number of peers leaving the system. In order to keep the power law property during node leaving and joining processes, the same number of peers from the network are randomly picked up (turned on) to join the overlay system. For each peer, a maximum-neighbor-connection is predefined following power law. Each peer is required to keep the number of his neighbor connections no greater than his maximum-neighbor-connection during the simulation.

4.5 Performance Evaluation

Simulation results about performance evaluation are presented in this section, starting with the performance evaluation in a Gnutella like context environment and followed by the performance evaluation of e-ARD in a P2P system with DFS search mechanism.

4.5.1 Response Return Rate

Figure 4.6 compares response return rates of a Gnutella-like system, RRD scheme, ARD scheme, and e-ARD scheme with peer uptime ranging from 10 minutes to 60 minutes. The query frequency of each node is 0.3 queries per minute. The simulations

are deployed in a 60 minute period with data collected every 20 seconds. This process is repeated multiple times. The average response return rate based on these collected data in the evaluation are used. Results in Figure 4.6 show that Gnutella-like systems suffer from 35% response loss when the average peer uptime is 10 minutes. RRD, ARD, e-ARD can increase the response return rate by up to about 35%, 47%, 51% respectively, compared with Gnutella-like systems. The performance of the e-ARD scheme is better but close to ARD, while both of them achieve a near perfect response return rate. The reason may lie in that in a Gnutella-like system, most of the nodes have multiple query forwarding neighbors and can forward a query to other forwarding neighbors instead of bRDA in the e-ARD scheme.

Given that the peer lifetime is 10 minutes, Figure 4.7 compares the response return rates of different schemes versus the average query frequency, and shows that the response return rate is not sensitive to query frequency. The relationship between response return rates and the average number of response neighbors per node in the Gnutella-like system and ARD are presented in Figure 4.8. We can observe that the return rate increases as the average number of neighbors increases in a Gnutella-like system: the increase of the average number of neighbors suggests a network with higher density, which eventually results in shorter response return path. If we assume there are k nodes between requestor and responder in a query response path, with the same failure probability σ , the probability that no node failure happens is $f = 1-1(1-\sigma)^k$. Therefore, a shorter response return path will result in lower probability of peer failure during the response return process and thus a higher response return rate. Nevertheless, the response return rate of ARD is not sensitive to the average number of neighbors due to its near optimal performance. The effectiveness of e-ARD is similar to that of ARD since the number of neighbors of the peers has almost no impact on the routing via bRDA. The same case also applies for the impact of network topologies on response traffic cost.

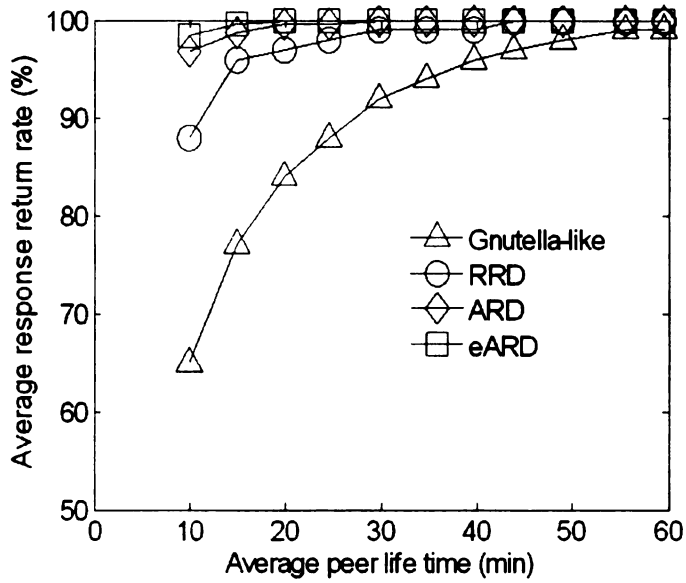


Figure 4.6. Response return rate versus peer lifetime

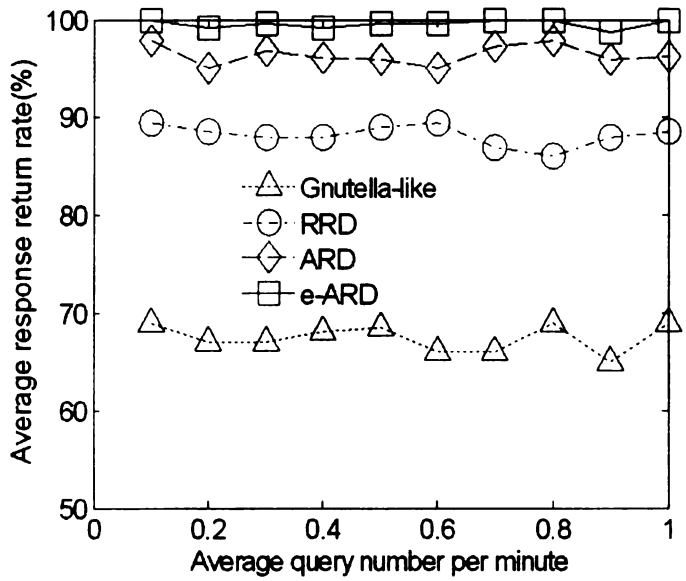


Figure 4.7. Response return rate versus query frequency

4.5.2 Response Traffic Cost and Response Time

The average response traffic costs of four different response delivery schemes are shown in Figure4.9. For the RRD scheme, only one additional path is selected for returning a duplicated response in the simulation. Compared with a Gnutella-like system, the RRD scheme incurs a 102% increase in traffic cost. This is because the RRD scheme always sends a duplicate response message through another path, which is generally longer than the original response path, whether node failure happens or not on the original response path. ARD only increases the response traffic cost by about 9% and e-ARD creates even less extra traffic at 6%, compared to the original system during the response process. The e-ARD scheme creates limited overhead due to two issues. First, when there are no forwarding neighbors alive, a peer in e-ARD does not send a response back to the peers of the last hop immediately but can forward it to a bRDA first. Second, sending a response to bRDA generally reduces the number of hops of routing the response message back.

On the other hand, extra response traffic cost of e-ARD is close to ARD. This happens due to the next two issues: first, the extra overhead created by the construction between the peer and bRDA will cut some of the benefit in limiting the extra traffic cost. Second, in the P2P system with a flooding query mechanism, the case that no other forwarding neighbors are alive happens rarely. Accordingly, in most of the cases where the primary forwarding neighbor fails, both ARD and e-ARD only execute the first option: send a response to some forwarding neighbor alive. Figure 4.10 shows that, given a fixed average peer uptime of 10 minutes, the change of query frequency does not affect the extra traffic costs induced by the three mechanisms. Figure 4.11 compares response traffic cost of a Gnutella-like system and ARD with different average number of neighbor connections. We can see that the response traffic cost decreases as the average number of neighbors increases. The reason is that when a peer has more neighbors, it has a higher probability of finding a candidate in his

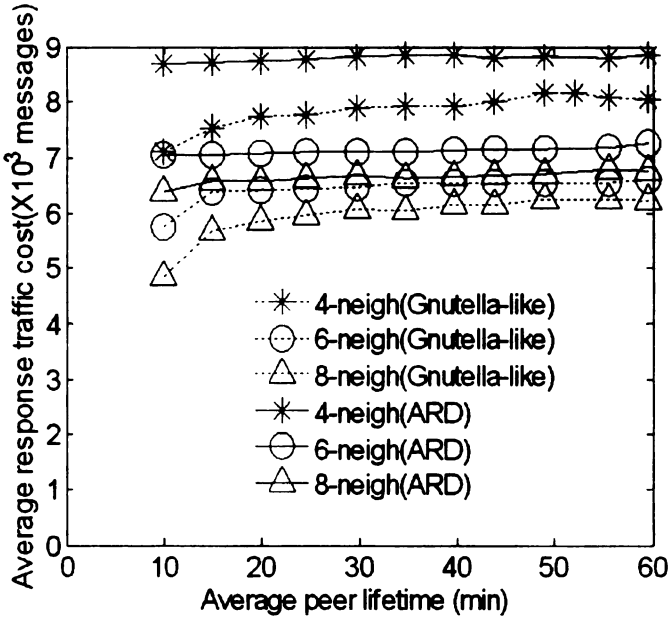


Figure 4.8. Response return rate under different network topologies (ARD)

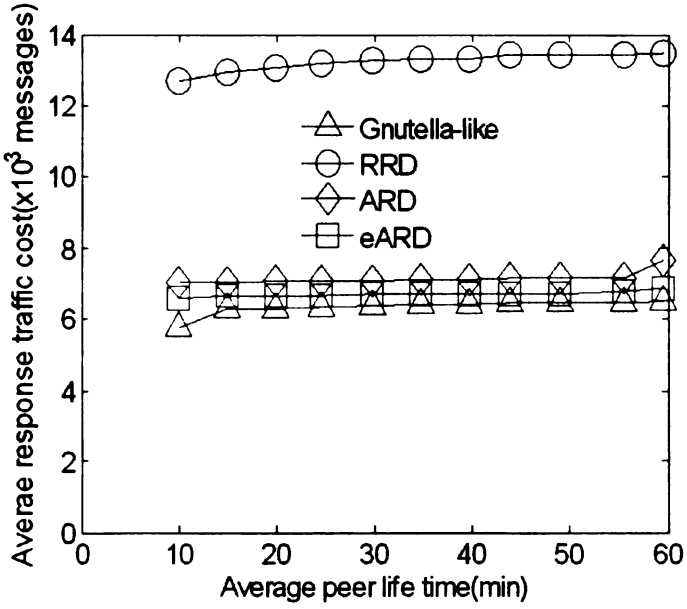


Figure 4.9. Response traffic cost versus peer lifetime

forwarding neighbor list to reroute the response instead of going back to his previous hop in the case that the peer cannot reach the next hop to send back a response.

To investigate the impact of the extra traffic cost created in the entire query request/response process, the ratio of response traffic to request traffic of all the systems is shown in Figure 4.12. We can see that request traffic is from one to two orders of magnitude greater than response traffic. Therefore even the traffic overhead of the RRD scheme can be ignored considering the traffic cost in the whole query process.

In a system that does not adopt flooding query process, the ratio of response traffic to request traffic may increase. For example, the simulation shows the ratio of response traffic and request traffic in random walk mechanism is around 1:10. Nevertheless, in this category of systems, e-ARD will be used in most of the time, which only incurs very limited extra response return traffic due to the adaptation of bRDA.

Figure 4.13 shows the average response time of the four response schemes. As being expected, the average response time of both the RRD and ARD schemes are close to that of Gnutella-like systems. The response time in RRD is only 2% higher than that of Gnutella-like systems, while ARD is about 4% higher. The average response time of e-ARD is even shorter, only 1.2% more than that in Gnutella-like system.

4.5.3 Analysis of RRD, ARD, and e-ARD

In this section, the influence of different key parameters in the three schemes are evaluated respectively.

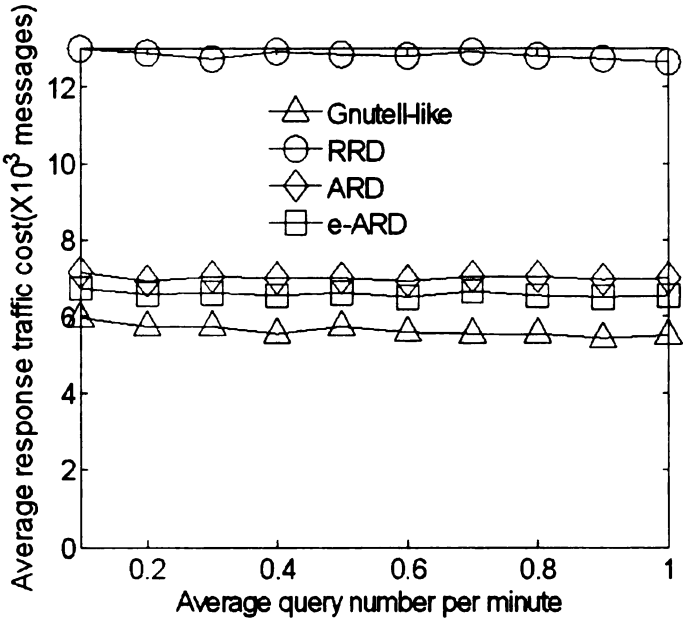


Figure 4.10. Response traffic cost versus query frequency

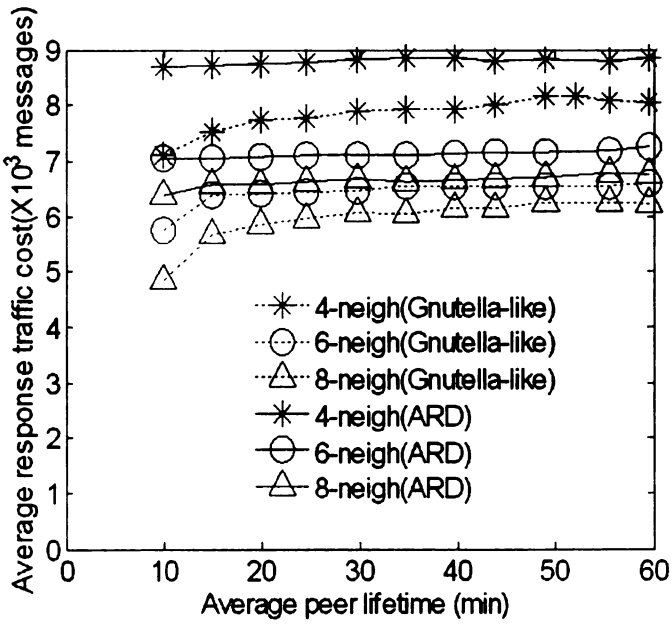


Figure 4.11. Response traffic cost under different network topologies

Performance improvements upon different γ values in RRD

The value of γ limits the number of redundant paths in RRD and affects the performance gain that can be achieved. The response return rates upon different γ values has been investigated. The results are presented in Figure 4.14. The results show that the response return rate does increase with the increase of the γ , but the improvement tends to be very little: with an average peer lifetime of 10 minutes, the response return rate is improved by 33% when γ is increased from 0.2 to 0.4 and further improved when is increased to 0.6. Nevertheless, marginal improvement is achieved when γ is increased to more than 0.6. This may be due to path overlap in the network. According to the investigations in previous studies, the peer lifetime in different P2P systems ranges from less than 10 minutes to 60 minutes. The influence of γ will decrease when the average peer lifetime increases.

Lifetime of forwarding neighbor lists in ARD

Figure 4.15 investigates the impact of the lifetime of forwarding neighbor lists on ARD's response return rate. Only the impact on ARD is investigated, since the influence of bRDA in e-ARD will be trivial. The curves in Figure 4.15 represent the response return rates of the ARD scheme with different lifetime of forwarding neighbor lists, i.e. $\frac{1}{4}$, $\frac{1}{2}$ 1, and 2 times of the average query response time. We can see that when the lifetime of the forwarding neighbor list reaches twice the average response time, the return rate is more than 95% of that for infinite lifetime. The reason is that if the lifetime is too short, a peer failing to deliver a response is less likely to find another candidate to reroute the response because the forwarding neighbor list may not be available any more, which will result in a high response failure rate.

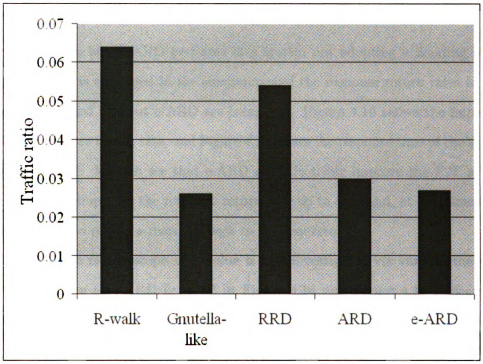


Figure 4.12. Ratio of response traffic over request traffic

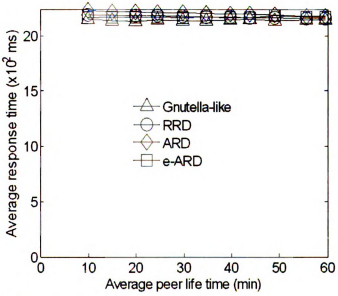


Figure 4.13. Response time versus average peer lifetime

Effectiveness of e-ARD in P2P systems with a DFS based searching mechanism

In order to show how e-ARD performs in a system not adopting a flooding search algorithm, DFS is employed in the simulator, and the response return rates in such a system with and without e-ARD are compared. Figure 4.16 shows the impact of e-ARD in response return rate, and Figure 4.17 shows the response time of the system without flooding. We can see that e-ARD can effectively improve the P2P system performance: it improves the response return rate up to 60% and, at the same time, needs less time to return a response back to the requestor.

The influence of the scope of α value is also investigated via varying the value of c. The curves of e-ARD ($c = n$) in Figure 4.16 and Figure 4.17 indicate the performance of e-ARD when c is set to be n. We can observe that the response return rate increases as the c value increases. This is because when the c increases, the scope of α decreases, which makes fewer nodes as bRDAs. The decrease of the number of bRDAs eventually results in a shorter response return path upon the node failure. For the same reason, the response time drops as the c value increases. One more thing we can observe here is that the performances of e-ARD with different scope of α are close and, therefore, we should choose a c value that makes the values of α large enough to satisfy the anonymity requirement for the requestor.

4.5.4 Comparison of RRD, ARD and e-ARD

RRD, ARD and e-ARD are summarized in Table 4.4. Among all three mechanisms, RRD requires the least modification of the existing response return mechanism, which makes it a quick fix for the current response return mechanism. On the other hand, RRD incurs more traffic overhead than ARD and e-ARD. Although nodes in ARD need to maintain a forwarding neighbor list locally, no new packet formats and extra bRDAs need to be introduced to construct ARD. The effectiveness and efficiency of

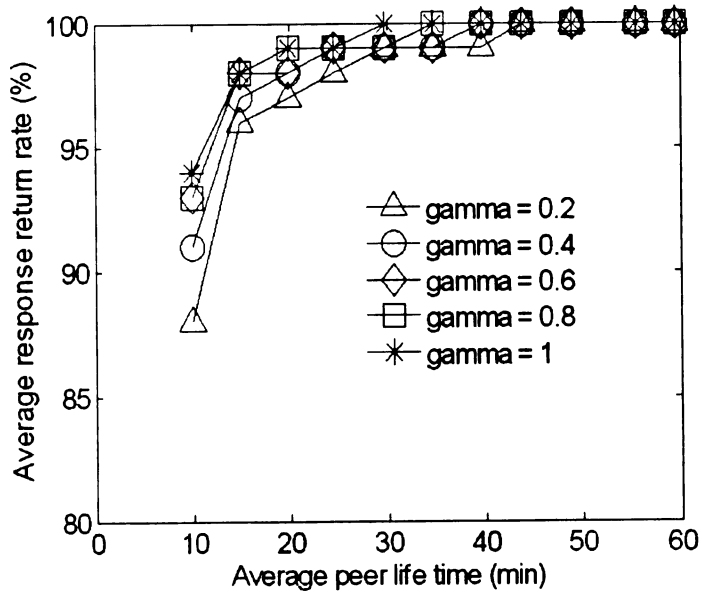


Figure 4.14. Response return rate upon different γ values (RRD)

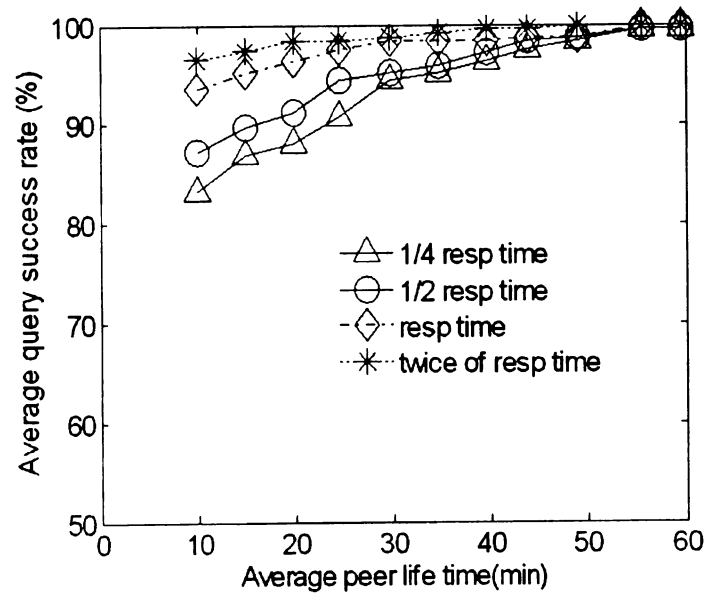


Figure 4.15. Response return rate under different lifetimes of forwarding neighbor lists

ARD outperforms that of RRD and are close to that of e-ARD: up to 47% (ARD) vs. 51% (e-ARD) upon response return rate improvement and 9% (ARD) vs. 6% (e-ARD) upon extra traffic cost. Although the response time of ARD is the longest among the three mechanisms, the response time of ARD is very close to that of RRD and e-ARD. Considering both performance and implementation complexity, ARD is the best choice to remedy the response loss problem in existing response return mechanism.

Both RRD and ARD, however, work effectively based on the assumption that there are more than one forwarding neighbor for each peer in the system. In a system that such an assumption does not hold, e-ARD is the best candidate to avoid the response loss incurred by the dynamic nature of the system itself.

4.6 Summary

In an unstructured P2P system, query responses are sent back to the requestor along the incoming query path. However, the P2P system is a highly dynamic system in that average peer lifetime is from 10 to 60 minutes, and the logical connection between peers lasts from 1 to 24 minutes in average. This leads to a response loss problem, with up to 35% of the responses being lost.

In order to remedy the response loss problem and improve service availability, three techniques are proposed here: RRD, ARD, and e-ARD. All these techniques reduce response loss rate with limited extra cost regarding to the entire query process. RRD requires the least modification of the current response return mechanism to implement. ARD functions more effective and efficiency than RRD, while its implementation complexity is less than that of e-ARD. e-ARD extends ARD and can be used in unstructured P2P systems with a limited or non-flooding search mechanism.

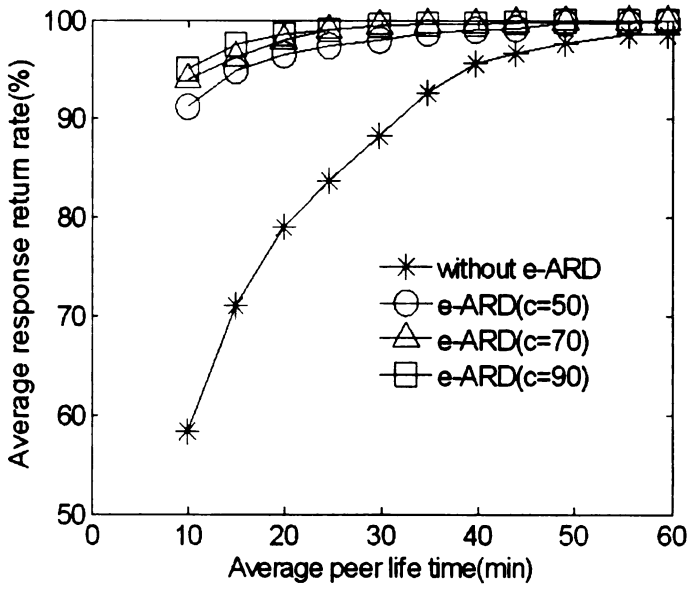


Figure 4.16. Response return rate with and without e-ARD (DFS)

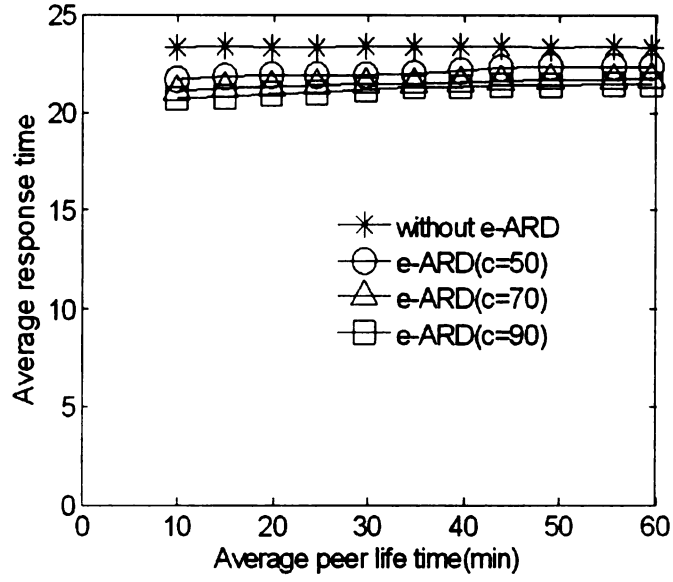


Figure 4.17. Response return rate under different lifetimes of forwarding neighbor lists

Table 4.4. Comparison of RRD, ARD, and e-ARD

	RRD	**ARD**	**e-ARD**
Effectiveness (Response Return Rate)	effective (up to 35%)	more effective (up to 47%)	most effective (up to 51%)
Efficiency (Extra Traffic Cost)	double the response cost in Gnutella-like system and maybe even higher (102%)	less extra traffic overhead (less than 90%)	least extra traffic overhead (6%)
Quality of Service (Response time)	shorter (2% more)	longest (4% more)	shortest (1% more)
Implementation Restrictions	node in the path has more than one forwarding neighbor; performance depends on how the redundant paths overlay	node in the path has more than one forwarding neighbor; performance depends on how many forwarding neighbors a node has in the query path	none
Implementation Complexity	no extra complexity	each node maintains a forwarding list	each node maintains a forwarding list; new message formats are introduced; extra bRDAs are introduced
Application	quick fix for current system	system adopting flooding search system; dense network	system not adopting flooding search system; sparse network

CHAPTER 5

Hierarchical Reputation Management for P2P Systems

Reputation systems are adopted in many overlay systems to provide data authenticity. In this chapter, we explore the reputation management system for P2P systems to enhance data authenticity and provide the trustworthy service content more effectively and efficiently. Our solution of hiREP adopts a hierarchical architecture to limit the traffic created in the trust value spreading process. hiREP guarantees data authenticity and voter anonymity, and also makes it easier for peers to filter out fake trust values.

5.1 Overview of Reputation Management for P2P Systems

Like many other overlay systems, peer-to-peer file sharing systems are fully distributed systems with no central control. Anyone can freely join and leave the system. In addition, it is required that the service requestor and provider remain anonymous. These features make it easy to inject malfunctioning data into the system and hard to

trace the data source, which impede data authenticity of the service content provided to users.

The music industry has already utilized these features to prevent people from downloading free music. Investigations show that large amount of "polluted" data have been injected into KaZaA, one of the most popular P2P system: more than 73% of popular song copies are polluted data [60]. The injection of "polluted" data may protect the music industry from financial loss, but it also demonstrates that attackers can behave in the similar way to prevent one from publishing legal data or distribute virus data in P2P systems.

One may consider that the anti-virus software and firewalls installed in the end users' PCs are enough to protect peers from the infection of such data. However, these approaches require users to download malfunctioning data before the software can analyze the data, which not only wastes network resources but also the users' time, not to mention the innocent users who may not update their anti-virus software in time. It should be the task of the system designer to prevent spreading such data across the system. P2P system can not be extended to support more critical applications without guaranteed trustworthy service.

To solve the problem and provide data authenticity, research has been done to construct the reputation system in a P2P network. The typical query process in a P2P network with the reputation system is shown in Figure 5.1. After receiving responses, a requestor sends out messages to elicit the trust values of the potential providers. Based on the received trust values, the requestor chooses the qualified provider to download files.

The construction of reputation system for P2P networks includes two basic components: how to compute the trust value of each node and how to distribute/access the trust values [96]. The trust value computation model has been studied extensively [14, 16, 54, 74, 75, 76, 99, 107, 114, 113] and is beyond the scope of this thesis.

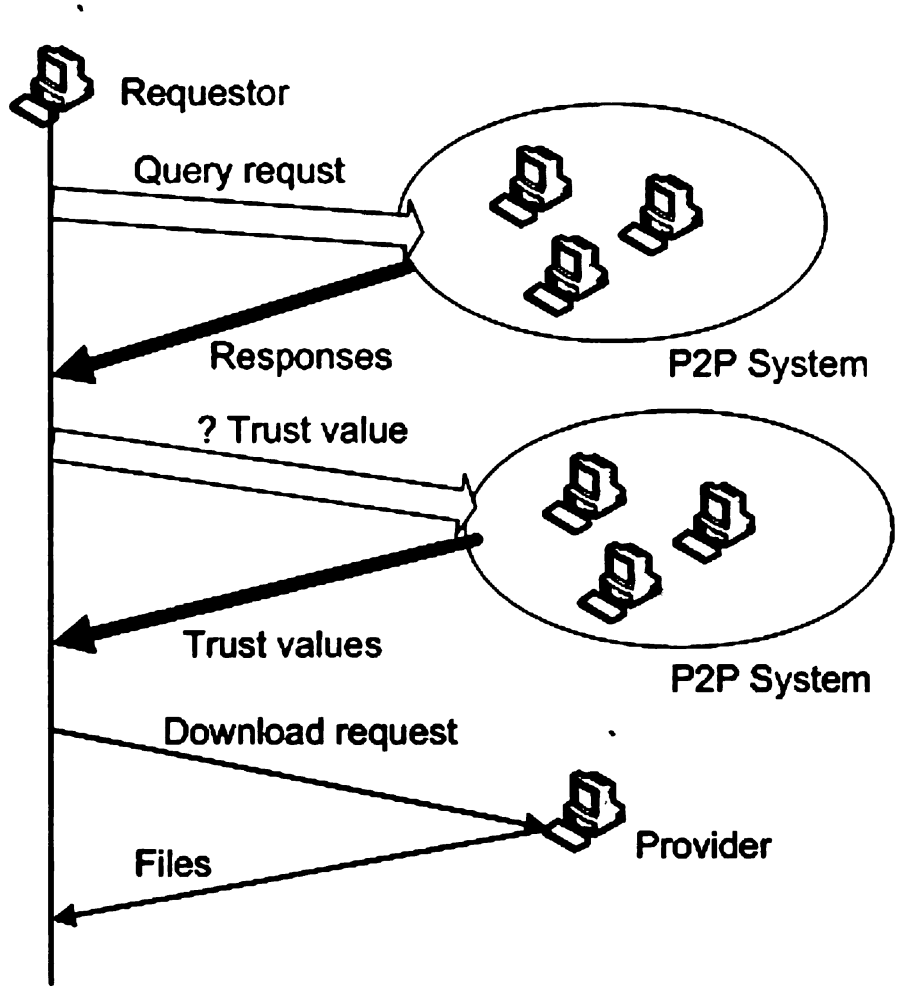


Figure 5.1. Reputation system in P2P systems

This thesis focuses on the storage and spreading of the trust values in the unstructured P2P system. Unstructured P2P system is the prevalent model of P2P systems in actual practice. Its flooding based query system is easy to implement and robust to node failures. However, the overwhelming traffic caused by flooding is the main hurdle of the system scalability and many mechanisms had been proposed to replace the pure flooding system.

With other issues of P2P systems in mind, the following requirements should be met to design reputation systems in an unstructured P2P network: 1) the trust value spreading process of the reputation system cannot base on pure flooding mechanisms that cause overwhelming web traffic. 2) voter anonymity should be guaranteed to

protect voters' privacy, i.e., the real identity of voters should be hidden from other parties. 3) the authenticity of the trust value should be guaranteed, i.e. the system should be robust to the attacks that try to invalidate the reputation evaluation.

In order to fulfill all three requirements, a hierarchical reputation system, hiREP, is proposed for the unstructured P2P systems.

Peers in hiREP can be divided into three types: general peers, reputation agents, and trusted reputation agents (or, trusted agents for short). Any peer with a bandwidth greater than 64k can choose to function as a reputation agent, but only a qualified one can be selected by other peers as a trusted agent. All trusted agents construct a reputation agent community. A peer reports transaction results only to its trusted agents, and checks only with its trusted agents to fetch the trusted values of other peers. hiREP limits the traffic created by trust value inquiries per peer to $O(C)$, where C is the number of the trusted agents per peer.

In order to guarantee the anonymity of voters, hiREP adopts onion routing for the communication of a peer and its trusted agents. Each peer is assigned a unique nodeID together with the usage of public key system to provide authenticity of the votes.

The main characteristics of hiREP include:

- hiREP adopts an effective hierarchical structure to construct unstructured P2P reputation systems. The challenge of constructing efficient hierarchical reputation system in an open and anonymous system derives from how to locate the suitable reputation agents for an individual peer and build reliable and light weighted communication between them. To the best of the author's knowledge, hiREP is the first reputation system with light overhead on top of the unstructured P2P systems.

- Besides avoiding of flooding based polling mechanisms in trust request process, hiREP adopts a token controlled message distributing mechanism for the trusted

reputation list exchange process.

- By using public key hash as the nodeID of a peer, hiREP distributes public keys efficiently in the P2P system without third party certificate authority.

- With the adaptation of onion routing based communication in the reputation request process, hiREP guarantees voter anonymity with fairly light overhead.

We present the design of hiREP in the following section. Section 5.3 analyzes the traffic load that hiREP may bring to the system as well as its robustness under different attacks. A series of simulations are deployed to evaluate the performance of hiREP and presented in Section 5.4. Section 5.5 gives a summary of this chapter.

5.2 Design of hiREP

The design of hiREP is presented here in detail. hiREP focuses on the storage and distribution of trust values. It also guarantees the authenticity of the transaction results reported to the reputation agents. To allow better understanding of hiREP, this section is starting with the motivation of choosing hierarchy structure, followed by an overview of hiREP and the description of each components of the system in details.

5.2.1 Fully Distributed, Centralized, or Hierarchical

A fully distributed mechanism is simple to implement and looks like a natural fit for constructing functionalities atop of an unstructured P2P system, which itself is fully distributed. Without introducing any centralized server, trust values are generally stored locally in the peers that make the trust value evaluations. As the trust values of a particular provider are disseminated in all the nodes that have

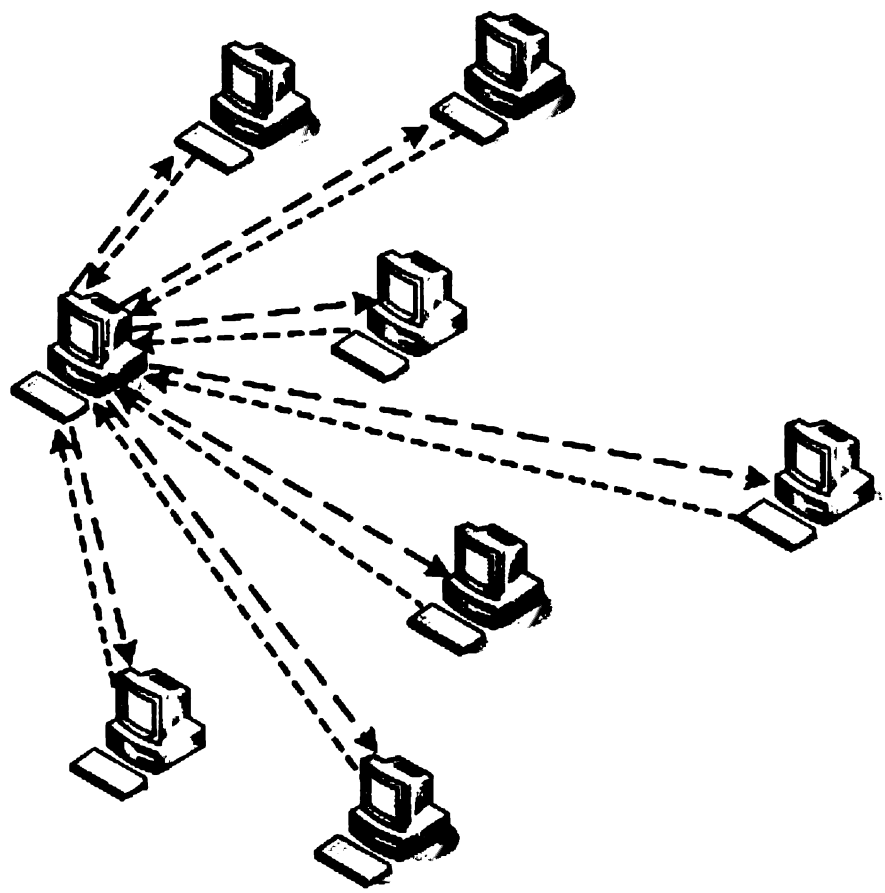


Figure 5.2. Reputation system in fully distributed structure (P2PREP)

transaction experience with it and there is no way for a node to know who owns the trust value, the trust requestor has to poll every node of the system to collect the evaluations. This leads to a flooding based polling process.

Figure 5.2 shows such a fully distributed structure adopted in P2PREP [36]. In P2PREP, the trust value of the peers is locally computed by and stored in their transaction partners. The requestor broadcasts trust value query messages to the entire system. Upon receiving the request, all of the peers that own the trust value of the potential provider return the trust value back to the requestor. The trust value query messages flooded across the system by the trust value requestor bring heavy traffic load to the system.

Like the reputation/credit system in the real world, the reputation systems for the e-commerce society are based on a centralized structure that is shown in Figure 5.3: Credits of individual entity are reported to some reliable centralized reputation management web servers, which send out credit report to the other parties as necessary. However, going back to a centralized system requires extra reliable servers to store the trust values, which is not practical and conflict with the distributed structure of unstructured P2P system. In addition, centralized structures are inevitably accompanied with the problems like traffic bottleneck and single point of failure of servers, and are vulnerable to the DoS/DDoS attacks in an open system.

Adopting a hierarchical structure, hiREP tries to achieve the advantages of both the centralized system and distributed system, and avoid the disadvantages of them. In the hierarchical reputation system, the trust values of nodes are accumulated and stored in a group of reputation agents. Unlike centralized systems, reputation agents in hierarchical system are composed of general peers instead of dedicated servers. By default, a peer can choose any node as its trusted reputation agent. On the other hand, peers only need to contact with a group of relative fixed reputation agents to obtain the trust value they want and evaluate agents instead of massive number

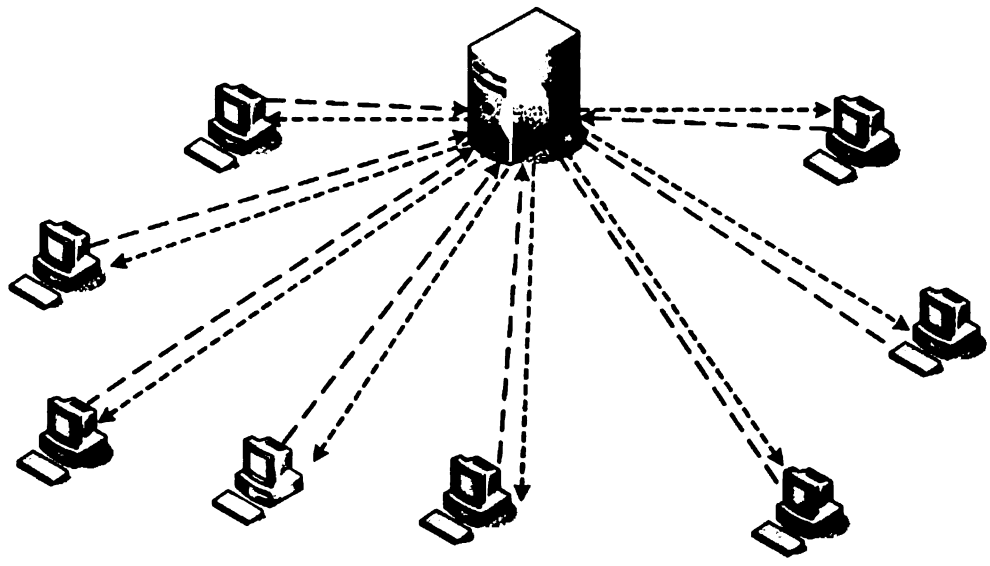


Figure 5.3. Reputation system in centralized structure

of individual nodes to filter out the malicious nodes when an attack happens. The challenge here is how to keep the qualified reputation agents for each peer.

5.2.2 Overview of hiREP

In hiREP system, each peer selects a group of agents as its trusted agents. Any peer with a bandwidth greater than 64k can claim itself a reputation agent, though not every reputation agent can be trusted by other peers. Peers update their trusted agent lists periodically. A reputation agent computes the trust value of each node using its own trust value computation model. A peer sends trust value request messages and reports the transaction results only to its trusted agents. Peers and trusted agents form a hierarchical structure as shown in Figure 5.4. Note although any peer can claim itself as a reputation agent, only trusted reputation agents may store trust values of other peers and send out trust values to other peer that trust them upon request.

Voter anonymity is preserved by adopting an onion routing based communication

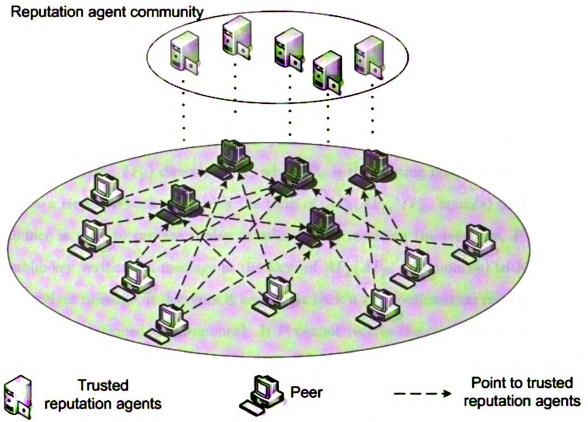


Figure 5.4. Hierarchical structure of hiREP

mechanism between a peer and its trusted agents. A public key system is used to provide data authenticity in communication processes.

5.2.3 nodeID, Public Key System, and Onion

Any peer in hiREP has two types of public key pairs: anonymity key pair (AP, AR), which helps provide anonymity, and signature key pair (SP, SR), which helps provide message authenticity. AR and SR are the private keys; AP and SP are public keys. The nodeID is generated by the peer itself and is the hash of SP generated by a hash function such as SHA-1. By associating SP with a node's nodeID, hiREP is effectively protected from Man-In-The-Middle attack: as nodeID is uniquely determined by SP, it is impossible for attackers to replace the public key of a particular nodeID. nodeID helps a peer to build its reputation in the system and only has relations with SP.

Attackers cannot associate nodeID with a node's identity in real world such as its IP address.

(AP, AR) is associated with a peer's IP address and sent to other peers upon request. As shown in Figure 5.5, when a peer P picks peer K as its onion routing relay (P knows K's IP address as it picks up K), it will send a routing relay request (R_o, AP_p, IP_p) directly to K, where R_o is the routing relay request. K then sends a response back to P with the form of $AP_p(AP_k, IP_k,$ nounce) to P, where nounce is used to prevent replay attacks. Upon receiving the response, P sends a public key verification message in the form of $AP_k(AP_p, IP_p,$ nounce) to K. When K receives message, it decrypts it and sends back a key confirmation message of the form AP_p(confirmed, IP_k, nounce). If P cannot receive the confirmation, it knows AP_k is invalid.

After receiving the anonymity key from its onion routing relays, P can form its own onions which define a path to it. The onion format is:

$$(((((((fakeOnion)AP_p)IP_p)AP_1)IP_1)AP_k)IP_k, sq)SR_p$$

AP_i is the anonymity public key of peer i. sq is the non-decrease sequence number used to indicate the age of the onion. SR_p is used to guarantee the authenticity of the onion. A node has P's onion and SP_p can decrypt the onion using SP_p and sends messages to K. K then peels one more layer of onion using AR_k and sends the message to the next layer relay and so on until it reaches P. As the formats of the onions sent to each relay are exactly the same, even the relay next to P does not know P is the receiver.

5.2.4 Reputation Agent Community Formation

In hiREP system, high performance reputation agents can be selected as trusted agents. All trusted agents form the reputation agent community.

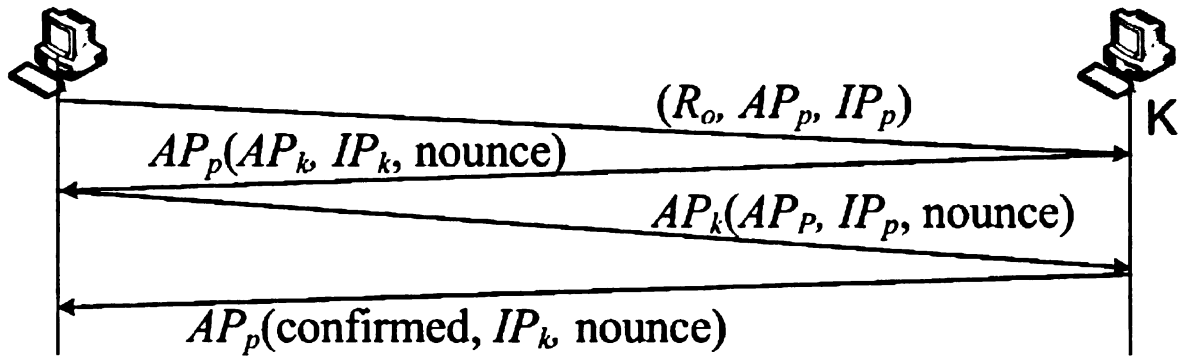


Figure 5.5. Fetch the anonymity key of a routing relay

Trusted agent list request

Each peer keeps a trusted agent list locally. The format for each list entry is like this: {weight, agent nodeID, $Onion_{agent}$, SP_e}. Weight is decided by the expertise of this reputation agent. SP_e is the private key of the agent.

When a peer first joins the system or it wants to collect some good reputation agents with other peers' recommendations, it sends out a trusted agent list request with the format of {R_{al}, token, TTL}. R_{al} is the agent list request. The amount of agent list request messages is limited by both TTL value and token number. The author recommends a default TTL value as 7 to be consistent with the query TTL value in Gnutella [11]. The amount of tokens equals the number of trusted agent lists that a peer wants to collect. A token was used up only when a node returns its trusted agent list to the requestor. The node can return its own nodeID if it has no trusted agent list.

Figure 5.6 illustrates the agent list request process. Requestor R plans to collect six reputation lists from other peers. Therefore, R distributes agent request messages to its neighbors with 6 tokens in Figure 5.6(a). In Figure 5.6(b), A and B return reputation lists to R and use up one token respectively. As C doesn't have any reputation list, it forwards the message with untouched tokens to F. As F receives two tokens from B and C respectively, it uses one by itself and sends the message

with the left tokens to I. I sends a list back to R, uses up the last token, and ends the message forwarding.

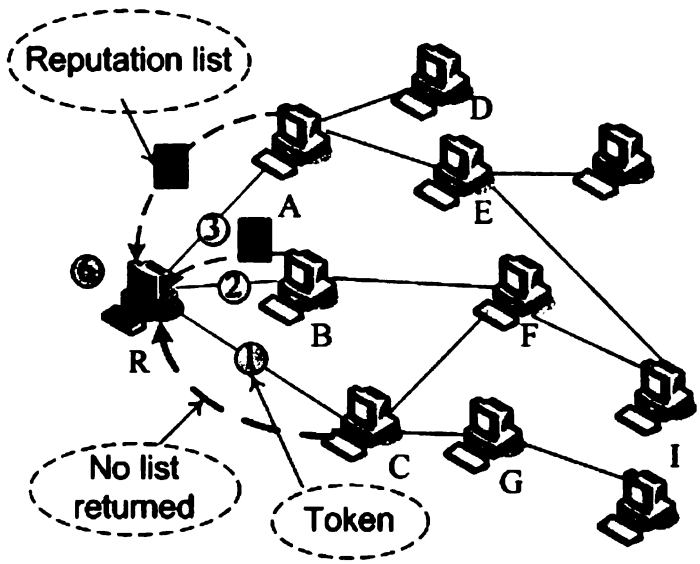
Agent rank and selection

Upon receiving the lists, the requestor ranks each reputation agent in different reputation lists according to their weight: assume the requestor wants to collect n reputation agents. For a reputation agent of the greatest weight, it is ranked as value n; the one of the second greatest weight is ranked as value $n-1$ and so on. If there are more agents in a received agent list than what a requestor needs, say m, all the agents ranked less than $n-m$ will be assigned a rank value 0. For the same agent who gets different rank values from different agent lists, the highest rank value will be its final rank. The requestor then selects its trusted agents according to their ranks. If several agents have the same rank, the requestor picks up its trusted agents from them randomly.

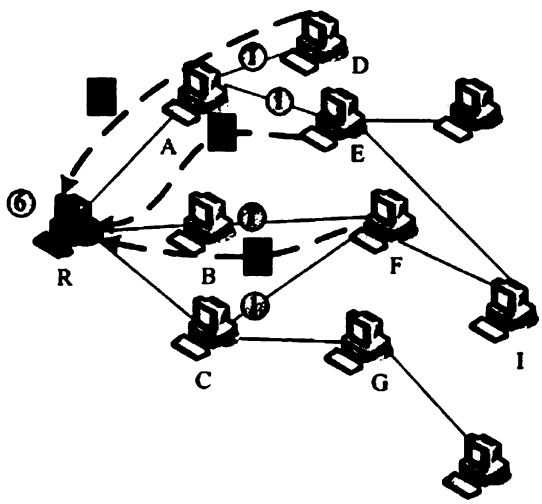
Trusted agent list maintenance

After selecting its trusted agents, a peer will assign an initial expertise value of 1 to each agent and keep updating the expertise values after each transaction. Assume accuracy of agent E in current transaction is A_c, and its cumulative accuracy of previous transactions A_p. The accuracy of agent E is $\alpha A_c + (1-\alpha)A_p$; $\alpha \in (0,1)$. Current accuracy A_c is either 0 or 1. It is 1 only when the evaluation given by this agent node is consistent with the transaction result.

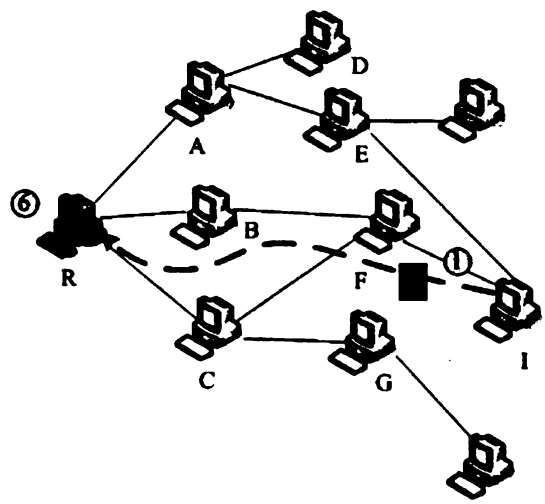
If an agent is offline and its accuracy value is positive, it will be moved to the backup agent cache. Otherwise, it will be removed from the trusted agent list. Backup agent cache is updated following the most recently first principle. When the amount of agents in its agent list is smaller than some threshold, say 50, the peer first probes all back up agents. If the result is not satisfying, the peer will send out an agent



(a) Reputation list request processed by first hot neighbors($TTL = 6$)



(b) Reputation list request processed by second hop neighbors ($TTL = 5$)



(c) Reputation list request will not be forwarded since token is run out($TTL = 4$)

Figure 5.6. Request process of trusted agent list

request message to find other qualified agents.

5.2.5 Trust Value Distribution

Trust value distribution should guarantee both anonymity of voters and the authenticity of trust values. The voters here refer to two parties: trusted agents that send trust values to the trust value requestors, and peers that send their transaction results to their trusted agents.

In hiREP, a trusted reputation agent keeps a public key list to store the public signature keys. The format of the list is: $\{nodeID_1, SP_1; nodeID_2, SP_2; nodeID_n, SP_n\}$, where SP_i is the public signature key of the node that chooses this agent as its trusted agent.

Trust Value Request

When a peer P wants to get the trust value of a particular node from its trusted agent E, P sends out the trust value request message using an onion of E stored in its trusted agent list. The format of trust value request is $\{SP_e(R), SP_p, Onion_p\}$. SP_e is the public key of E. SP_p is the public key of P. $Onion_p$ is the Onion issued by P. R is the request message with the format request, nonce.

Trust Value Response

After receiving the trust value request message, E computes the nodeID of P using the pre-known hash function. E will add the nodeID and public key of P to its public key list if P's nodeID is not in the list. E then sends back to P a trust value response message using $Onion_p$. The format of the message is $SP_p(T)$, SP_e, $Onion_e$. SP_p and SP_e are the same as in request message. $Onion_e$ is a fresh Onion issued by E. T is the response message with the format trust value, nonce, where nonce is the same one included in request message R.

Transaction Result Report

After a transaction, P will report the transaction results with the format of (SR_p(result, nounce), $nodeID_p$) to E using $Onion_e$. E then locates SP_p in its public key list using $nodeID_p$ and tries to decrypt the signed transaction result. If the result cannot be decrypted, the message will be dropped.

Voter anonymity and data authenticity are provided in all three processes. With the adaptation of onions, the real identity of the sender and receiver is hidden from each other and third parties. This provides protection for voters' privacy. The authenticity of both trust values and transaction reports is also ensured by the private key signature of senders.

Until now, we assume the public keys can not be cracked. This assumption can be loosed by allowing peers to update their public key pair periodically. New public keys signed by current private key can be sent out using the most recently received onions. It is also easy for a peer who receives the update message to map and replace an old nodeID to a new nodeID.

5.2.6 Transaction in a P2P System with hiREP

The transaction process in a P2P system with hiREP is shown in Figure 5.7. Like the transactions in other P2P reputation systems, it includes the query process, transaction (file downloading), and transaction reporting. However, the query process and transaction reporting process in hiREP are different from that of other reputation systems. In the hiREP query process shown in Figure 5.7(a), the trust value request will only be sent to requestor's trusted agents instead of broadcasting to the entire system. In other words, only the requestor and its trusted agents are involved in the trust query process. After receiving the trust values, the requestor computes the final estimated trust value of the potential file providers and selects the one with the highest estimated trust value to download the file as shown in Figure 5.7(c). In Figure

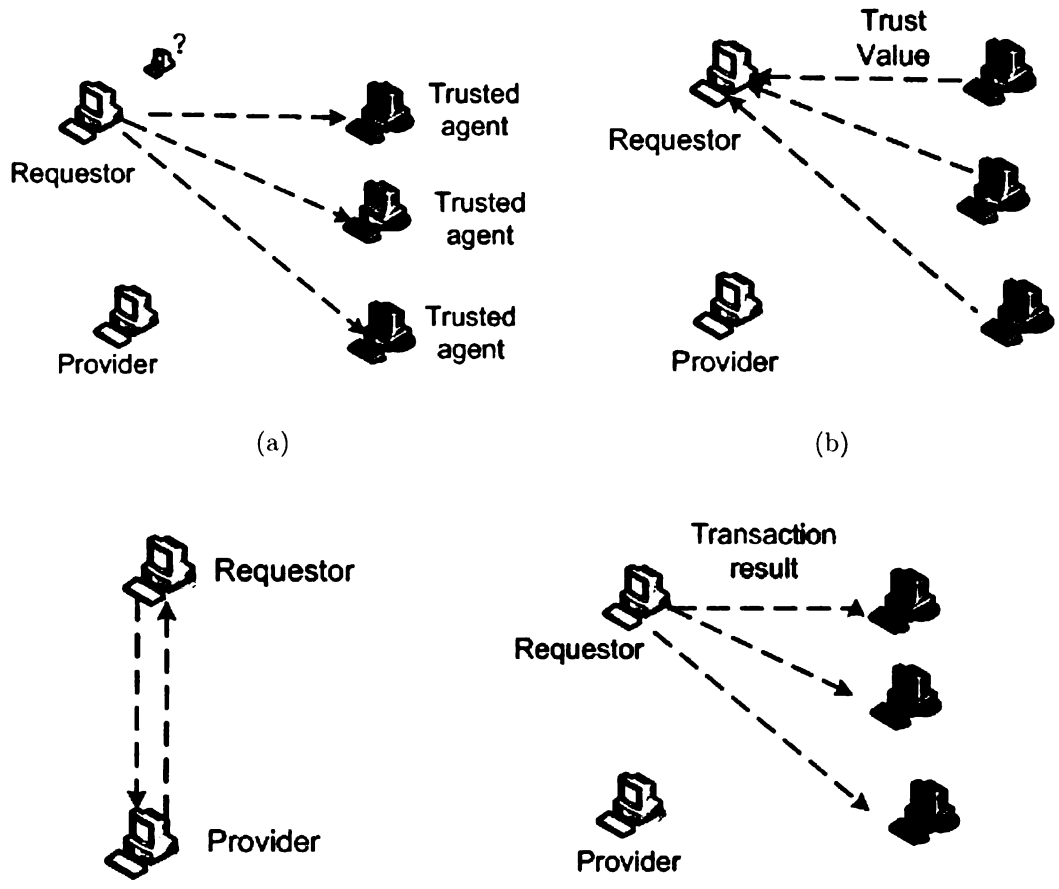


Figure 5.7. Transaction process in hiREP: (a)Requestor checks its trusted reputation agents about the trust value of the provider; (b)Reputation agents send back the trust value of provider to requestor; (c)File downloading; (d)Requestor reports the transaction results to reputation agents

5.7(d), the requestor updates the expertise values of its trusted agents and sends its transaction results to all of its trusted agents as discussed in Section 5.2.5.

5.3 Analysis of hiREP

This section analyzes hiREP in three aspects: the next subsection discusses the traffic overhead that hiREP brings to the overlay networks. Then we discuss the impact of the dynamic nature of the overlay network on hiREP. We also analyze the robustness

of hiREP against different types of attacks.

5.3.1 Traffic Overhead

As the reputation list initialization is executed only once for each peer and all the other messages created in the hiREP system are either sent out only as necessary or piggybacked in other messages, the main traffic overhead of hiREP comes from trust value distribution.

Assume each peer has c trusted agents in average. Each agent's onion has o_i relays and each transaction result reporter's onion has o_j relays. The messages created for trust value distribution in one transaction will be $\sum_{j=1}^{n}\sum_{i=1}^{n})\Big(o_i + o_j\Big)$, assuming each peer execute the process once. Considering that o_i and o_j are generally less than 10, the messages for the trust value distribution of one transaction are in the order of $O(c)$.

5.3.2 Impact of Dynamic Nature of P2P Networks for hiREP

P2P system is well known for its dynamic nature, i.e. nodes in P2P system come and go frequently. One concern is that this may invalid the onion of a reputation agent. However, although most of the nodes in the system are highly dynamic, there exist long-life nodes in the system as well. Although previous observations indicate median uptime for a node in Gnutella is 60 minutes [90], authors of paper [86] observe 25% of peers stay in the system for more than 24 hours. Other investigations indicate that 80% of hosts keeps inside the P2P system for the daily measurement [21]. F. E. Bustamante et al. [25] shows that the older a peer is, the longer it is expected to remain in the system. Thus, peers can reduce onion failure by choosing the long-lived relays.

5.3.3 Robustness Against Attacks

Manipulate trusted agents

Malicious nodes try to hinder peers in selecting proper trusted agents by giving multiple bad recommendations to reputation agents with high performance or multiple good recommendations to reputation agents with poor performance. The former case is discouraged by the system: as an agent is always ranked according to the greatest weight it received, the bad recommendation given by attackers will be ignored. As for the latter case, multiple high recommendations for an agent have the same effect as one single high recommendation.

The system can not completely prevent poor agents from getting high ranks, but attackers cannot render requestors to assign a poor agent a weight greater than all other agents. The point here is to guarantee good agents have chances to be selected and the requestor's reputation list is not overwhelmed with poor performance agents. In an extreme case, the trusted agent selection process will reduce to a random selection between the "real" good reputation agents recommended by sane peers and "fake" good reputation agents recommended by attackers. Poor performance reputation agents are then filtered out in the reputation list maintenance process.

Manipulate Peer Identities

In identity spoofing attacks, attackers send out trust values or transaction results using the identities of other nodes. This is not possible in hiREP. All trust values and transaction results are signed by the private keys which are associated with senders' unique nodeID. It is impossible for attackers to get the private key of the other peers.

In sybil attacks, attackers use multiple identities in a distributed system [40]. This is not avoidable unless the system has some centralized control server to strictly

control the identity a node can have [40]. However, if we consider each identity as a particular node, hiREP can reduce the damage of the sybil attack by filtering out poor performance reputation agents based on its own experience.

Manipulate the Reputation Evaluation

Attackers try to invalidate the trust value evaluation by making good evaluations for "poor" peers and bad evaluations for "good" peers. hiREP guarantees the authenticity of the transaction reports sent to the reputation agents. With the authentic transaction reports, reputation agents can decide the trust value of the peer using the next level computation model. As a trusted reputation agent receives more information for trust computation than a peer based on local experience, it is expected to compute trust values more accurately.

DoS/DDoS Attack

DoS/DDoS attacks may be initialized to disable the service of high performance reputation agents. To issue such an attack, the attacker has to first distinguish the high performance agents. The cost of distinguish such agents is not trivial. As traffic is spread among randomly chosen onion relays and reputation agents, it is hard to identify the high performance reputation agents by analyzing the traffic flow or the content of trust value request/response packets. The attackers have to go through all the process like a normal peer to figure out the high performance reputation agents. In addition, as the size of the reputation community is large, the peers that lose some of its trusted agents can easily replace them by other high performance reputation agents.

5.4 Performance Evaluation

In this section hiREP are evaluated with a series of simulations. The performance metrics is described in the next subsection, following by the simulation settings and simulation results.

5.4.1 Performance Metrics

Two major performance metrics are used in the simulation: traffic cost and trust evaluation accuracy. Traffic cost is used to indicate the network resources consumed in message delivering process, which is more critical than the local machine resources due to the rapid development of PCs. The messages induced in the trust query process are used to represent the traffic costs. The individual bandwidth and the length of links are ignored.

Trust evaluation accuracy is an important metric that is used to evaluate the effectiveness of a reputation system. The system is expected to be able to achieve at least the same level of trust evaluation accuracy as that in a pure voting system. Here we use the mean square error (MSE) between the estimated trust value and the true trust value of peers to indicate the evaluation accuracy of the system.

$$\text{MSE of trust value} = \frac{1}{n}\sum_{i=1}^{n}\sqrt{(v-\widetilde{v_i})^2}$$

where v is the real trust value, and the $\widetilde{v_i}$ is the trust value provided by either the trusted reputation agents (hiREP) or other peers(pure voting system).

5.4.2 Simulation Setup

A P2P network with power law topology is generated using BRITE [2]. Each node is randomly assigned as trusted (trusted value 1) and untrusted (trusted value 0). The reputation agents (nodes with bandwidth larger than 64K) are divided into good

Table 5.1. Simulation Parameters

Name	Default	Description	Name	Default	Description
Network Size	2000	Number of peers in the Network	Trusted agents of a peer	60	Amounts of trusted agents on a peer's trusted agent list
Neighbors per node	3	Average number of neighbors each peer	Poor performance agents	10%	Agents which can not make proper reputation of peers
Good rating	0.6-1	Scope of good reputation rating	TTL	4	TTL limit used in pure voting flooding process
Bad rating	0-0.4	Scope of bad reputation rating	Token number	10	Initial number of tokens for obtaining reputation agent lists
Relies on average in an onion	7	Agencies a peer includes in its onion			

agents and bad agents according to their capability to make trust value evaluation. A good agent gives trust values ranging from 0.6 to 1 to trustworthy peers, and trust values ranging from 0 to 0.4 to untrustable peers. A poor agent makes the inconsistent evaluation: 0.6 to 1 for untrustable peers and 0 to 0.4 for trustworthy peers. The default values of simulation parameters are listed in Table 5.1. They may be varied in the experiments.

Similar as in paper [96], the hiREP is compared with a pure voting system (called polling system in paper [96]). The trust making process is started with randomly selecting a peer as a potential service provider. Each node in the pure voting system computes a trust value and the overall estimated trust value is based on all of them. The flooding process is simulated by deploying a Breadth First Search based search operation.

For hiREP, only the trusted agents of the peers involved in a transaction compute trust values. After a transaction, these peers update the expertise values of and report the transaction results to the trusted agents. The above processes are simulated by re-computing the trust values (by trusted agents) and trusted agent expertise values (by these peers).

5.4.3 Simulation Results

The traffic costs of hiREP and pure voting process are explored in Figure 5.8, where curves of voting-n represent messages incurred by pure voting mechanism in a network with average node degree of n. As messages incurred by hiREP are only decided by the number of trusted agents per node, the messages incurred by hiREP are the same in networks of different node degree. We can see that, even in a network with average node degree of 2, the number of messages produced in hiREP system is less than $\frac{1}{2}$ of that produced in pure voting system. Due to the network size limit in the simulation, the TTL value of trust value request message in a pure voting system are set to be 4.

In the real system, TTL value is generally set to be 7, which suggests more messages will be sent out. We can also observe that more messages in a pure voting system are sent out in a density network than in a sparse network.

Trust accuracy of the hiREP systems and that of the pure voting system are compared in Figure 5.9, with an assumption of 10% malicious nodes in the system. The curves of hiREP-n represent MSE of hiREP systems that adopt different trusted agent threshold: hiREP-4 represents a system where a peer removes a trusted agent from its agent list if the expertise value of this agent is less than 0.4 and so on. We can see that the accuracy of hiREP is at least as good as that in pure voting. After a training process (about 100 transactions), hiREP reports trust value with much higher accuracy. MSE of trust accuracy of hiREP continues to reduce after 300 transactions, where the trust value accuracy is 90% and the system starts being stable. We can observe that a higher threshold results in a shorter convergence time.

We have investigated the effect of malicious nodes which give wrong evaluation intentionally. Figure 5.10 shows that trust value evaluation accuracy in pure voting system decreases much faster than that of the hiREP system. This may be because in pure voting system the trust value provided by each node is treated equally while in hiREP system, only the trust values provided by the agents of high expertise are accepted by other nodes. Therefore, malicious nodes lose their rights to express opinions due to their bad evaluation history. From Figure 5.10, we can observe that evaluation of pure voting may be more accurate when there are very few malicious nodes in the entire system. With the increase of the number of the malicious nodes, the performance of hiREP in trust accuracy will overwhelm that of the pure voting system. In an extreme case that 90% of reputation agents are poor performed, MSE of trust evaluation accuracy in hiREP is still under 25%.

Besides the traffic cost and trust evaluation accuracy, we have investigated the response time of the trust value request process, which is defined as the time from a

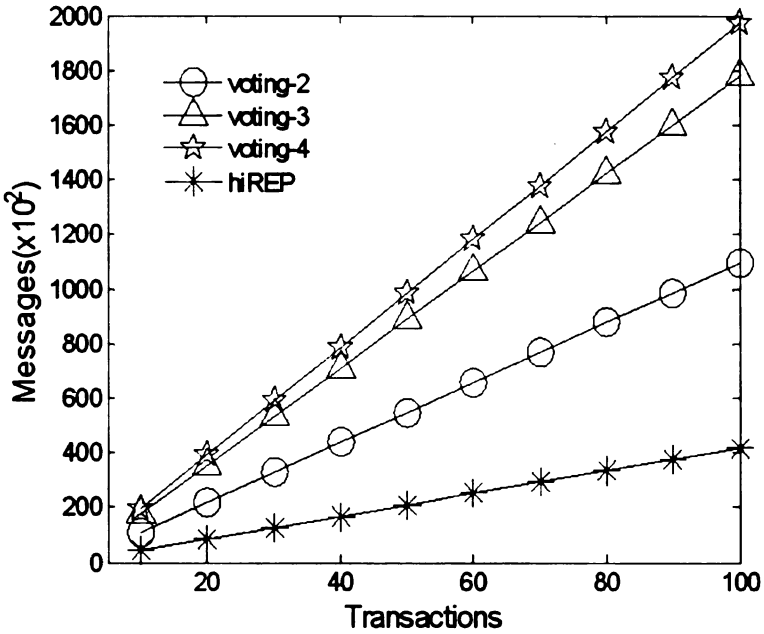


Figure 5.8. Trust query traffic cost of hiREP vs P2PREP

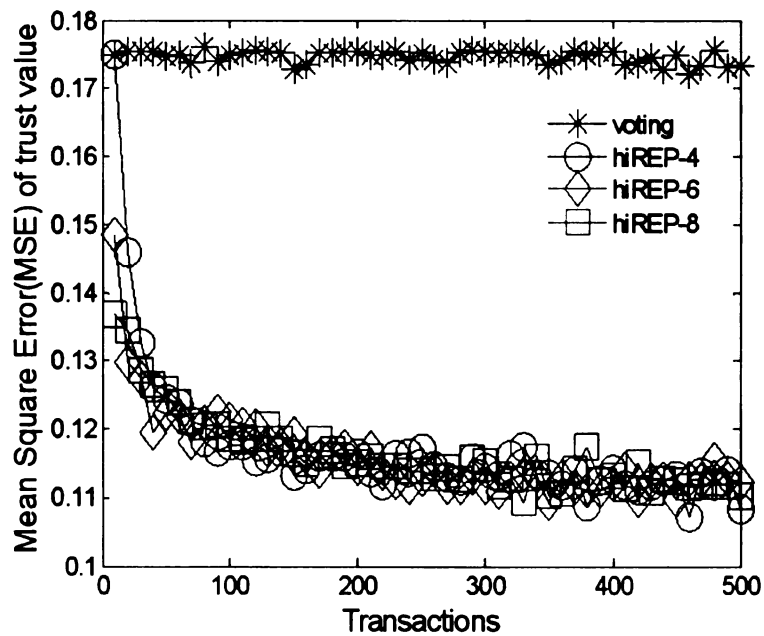


Figure 5.9. Trust accuracy vs. transactions

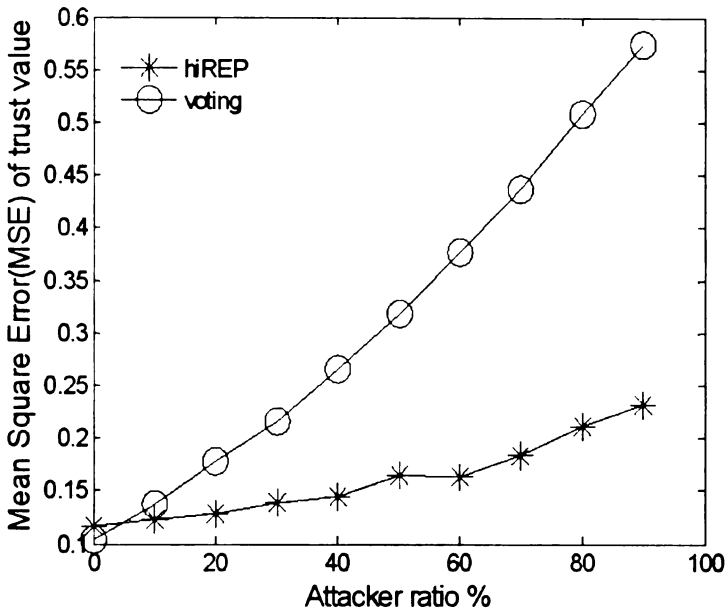


Figure 5.10. Trust accuracy vs. malicious nodes

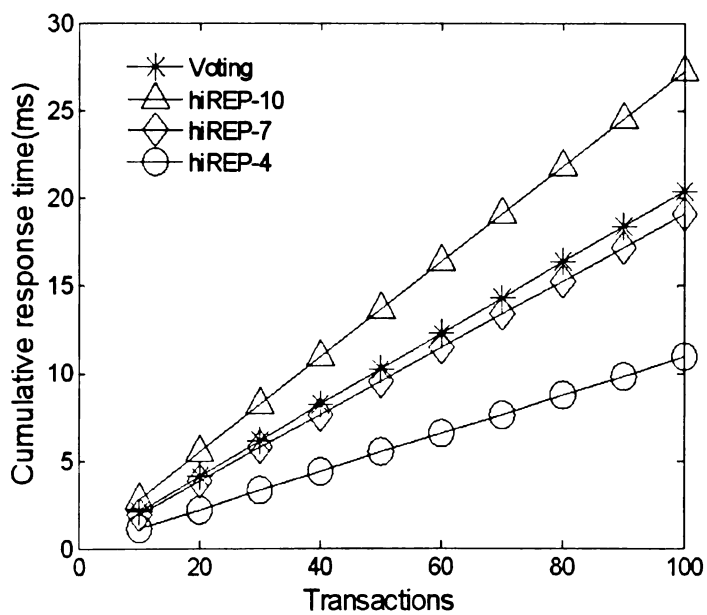


Figure 5.11. Cumulative response time of hiREP system

peer sends out request till it obtains the trust value. Figure 5.11 shows the cumulative response time of the pure voting system and the hiREP system. hiREP-n refers to the hiREP system in which there are n relays in an onion. We can observe that the decreases of the relay number result in the decreases of the response time of hiREP system. The average response time of hiREP is lower than that of the pure voting system.

5.5 Summary

In this chapter, a hierarchical reputation management system, hiREP, is proposed to guarantee the authenticity of the service content in the P2P overlay application and make it more reliable to users. The unique features of hiREP include: 1) a trust value distribution mechanism where the trust value requestor communicates only with a limited number of trusted agents instead of polling every node in the system; 2) an onion based communication mechanism between peers and their trusted agents to guarantee the voter anonymity; and 3) a public key system that does not rely on third party certificate authority for key distribution to guarantee data authenticity in communication.

Our analysis shows that hiREP is robust against the attacks such as trusted reputation agents manipulation, identity spoofing, reputation evaluation manipulation, and DoS/DDoS attack. hiREP also reduces the cost that may be induced by the reputation system and protect the voter anonymity. The simulation results demonstrate that hiREP creates much less traffic cost than a flooding based polling system, as well as provide more trustworthy reputation feedbacks to the nodes in the system.

CHAPTER 6

Mutual Anonymous Overlay Multicast

As an extended effort for building reliable overlay applications, we explore the mutual anonymity in overlay multicast systems. Mutual anonymity in overlay multicast systems helps to prevent the exposure of user and group information from others and thus enhance user safety in the systems. In this chapter, we identify the requirements of mutual anonymous overlay multicast and provide a mutual anonymous multicast protocol (MAM) to protect user information and provide user safety.

6.1 Overview of Mutual Anonymous Overlay Multicast

Multicast services are demanded by a variety of applications, e.g., video conferencing, Internet based education, NASA TV, software updates, etc. However, IP multicast is not widely deployed in the Internet due to its limitation in scalability and the support for higher layer functionality such as reliability and congestion control. Therefore, multicast services on overlay networks have been proposed by researchers [19, 28,

27, 33, 34, 72, 78, 95, 105, 111, 115, 32]. The salient features of overlays include ease of deployment and flexibility. We envision that in the near future, a wide variety of applications will be able to enjoy multicast services on overlay networks. Apart from commercial applications, we believe that government and military organizations will also use such services due to the several advantages multicast has to offer.

It follows to expect that, as multicast services continue to be deployed, existing and future multicast applications will also demand the security features that unicast communications have. Security in multicast communication has been addressed in [26, 58, 77]. Most of the work here focuses on authentication of the senders and receivers and the efficient distribution of the keys to all legal group members and exclusion of members that leave the group. Our focus here is providing anonymity in multicast communications.

Anonymity is an important component of user security and is demanded by many applications. Some of them are: critical multicast services like military or emergency applications, where strategic information and critical updates are transmitted to multiple destinations and require anonymity from external observers. Multi-party video conferencing applications carrying classified information will need anonymity from external observers and other members in the group. Large business organizations may have to multicast database updates to many sites for synchronization, and such applications will demand anonymity from rival organizations.

Solutions proposed for anonymity in unicast communications can not be directly applied to multicast applications. The fundamental difference between multicast and unicast is the concept of a group in multicast. There is a relationship among multiple nodes including the source(s) and destination(s) that are correlated, unlike in unicast where only one sender and one receiver are communicating.

Due to the correlation among nodes, there are special challenges in achieving anonymity in multicast: (1) The anonymity semantics in multicast are different from

those in unicast. For sender anonymity, the sender needs to hide not only from one receiver, but from a subset of, or all the receivers. In receiver anonymity, the receiver may need to hide not only from the sender, but also from other receiver(s). There is a special issue in anonymity in multicast called group anonymity, where the presence of the group is not disclosed to outsiders. The involvement of multiple members makes it very difficult to hide the existence of the group. (2) Multicast services naturally need the existence of a tree. Exposing this tree itself will compromise the degree of anonymity. In contrast, in unicast, the path from a sender to a receiver is much easier to hide. (3) Membership management is a challenging issue in multicast. Member joining and leaving makes anonymity difficult. (4) There are other inherent challenges for anonymous multicast services such as group key management.

There has been very little work on anonymous multicast. Simple solutions have been proposed to achieve multicast anonymity by inserting some proxies (called SAM) to the tree [48, 108]. However, this tree may itself not be efficient. The authors there attempt to provide sender anonymity and receiver anonymity. The authors do not address issues of providing mutual and group anonymity.

Our work on the mutual anonymous multicast (MAM) protocol for the overlay multicast is presented in the rest of this chapter. MAM includes the design of a unicast mutual anonymity protocol, and construction and optimization of an anonymous multicast tree. MAM is self organizing and completely distributed. The self-organized and completely distributed design of MAM can efficiently realize mutual anonymity in overlay multicast systems and protect user information.

The rest of the chapter is organized as follows. The definition of multicast mutual anonymity is given in the next section. Section 6.3 discusses the consideration on MAM protocol design. Section 6.4 presents the detail of MAM protocol design. Section 6.5 analyzes anonymity degree. Section 6.6 evaluates the performance of the MAM protocol. A summary of the work on MAM protocol is given in Section 6.7.

6.2 Definition of Multicast Mutual Anonymity

Before we give the definition of anonymous multicast, we specify roles of nodes in a multicast system. Nodes that belong to the multicast group are called group members. We label a node as a sender if it sends a multicast message to other nodes that are group members. The other group members that receive this message are called receivers. Note that we assume every node could be a sender and a receiver in the service. Nodes that are neither senders nor receivers are called outsiders to the group.

Definition 6.2.1 ***Multicast mutual anonymity.*** *Here a set of members desire to be hidden from others. Members in such a set need to achieve mutual anonymity from each other. Such a set can be a pair, such as the sender and a receiver; or one receiver and another receiver. The set also can be multiple members and may even include all members (i.e. complete anonymity).*

Multicast mutual anonymity can cover multicast sender anonymity (hide the sender's identity) and receiver anonymity (hide one or more receivers' identities). Of course multicast sender anonymity and receiver anonymity can be achieved by simpler protocols than that for multicast mutual anonymity. Multicast mutual anonymity is the focus in this thesis. Another type of multicast anonymity is group anonymity, which hides the existence of a multicast group session from all outsiders. Traffic covering approaches can be used to achieve multicast group anonymity, which is beyond the scope of this thesis.

Three types of nodes in a mutual anonymous multicast system are defined here.

1. *Anonymous member nodes*, AM nodes in short, are the member nodes whose identities need to be hidden from all member/non-member nodes.

2. *Non-anonymous member nodes*, NM nodes in short, are the member nodes that need not to be hidden from others.

3. *Middle outsiders*, MO nodes in short, are the nodes that do not need to receive any packets from the source for their own purpose, but providing packet forwarding service for the multicast system. If needed MO nodes are invited by the system for improving the overall efficiency. They do not hide their identity.

One naive approach to achieve anonymous multicast services is to treat the multicast as a set of unicast communications from the sender to the individual receivers and then directly apply one of the unicast anonymity schemes. While the approach is simple, it is inefficient. In order to achieve high efficiency and reduce the redundancy of multicast message transmissions among multiple receivers, multicast always relies on some structure to deliver a message. The structure is usually a tree, and the tree can be source-based or core-based. Unicast is a special case of multicast, where the structure is a path. The potential solution to anonymous multicast must center on the concept of the tree. This is the main difference and also the source of challenges in achieving anonymity in multicast compared with anonymity in unicast.

6.3 Design Consideration of Anonymous Multicast Systems

We need to consider both multicast tree efficiency and anonymity degree in the design of a multicast mutual anonymity protocol. An example is shown in Figure 6.1, in which we can see that an optimal multicast tree without (Figure 6.1.a) and with (Figure 6.1.c) anonymity concern are very different, where the cost of an AM-NM connection is 6 times that of an NM-NM connection and the cost of an AM-AM connection is 15 times that of an NM-NM connection. We have the following objectives in designing MAM protocol.

1. High mutual anonymity degree: the identity of each anonymous node (AM),

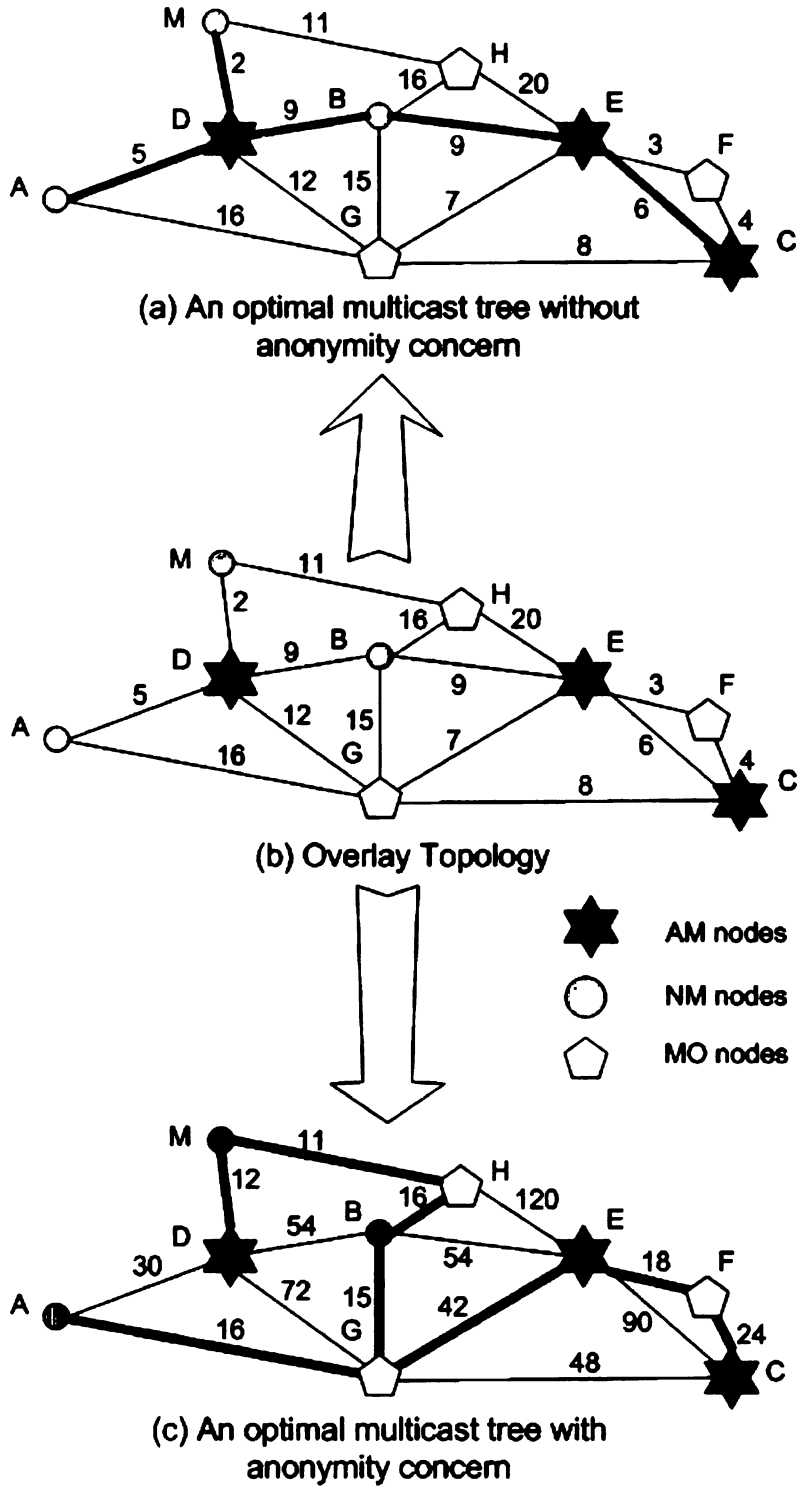


Figure 6.1. An example of multicast tree with and without anonymity concern

whether a sender or a receiver, in a multicast group should be hidden from all group members and outsiders.

2. Delivery efficiency: a smart tree with consideration of anonymity is built with low average delay and low resource usage.

3. Distributed fashion: the construction of the anonymous multicast system must be completely self-organizing and in a distributed manner. No trusted central server is involved. Further, MAM must be robust in a dynamic overlay environment.

4. Self-optimization: MAM will allow all the nodes to incrementally optimize the system, by reconstructing the tree and inviting more middle outsiders (MO nodes) to improve the overall performance.

Here is the basic idea of MAM. A set of NM nodes form an efficient multicast tree in terms of bandwidth and/or delay. Nodes of the tree are degree-bounded. Early AM nodes connect unsaturated NM nodes, or MO nodes (if MO nodes have been invited), on the tree using a unicast initiator anonymity protocol. When there is no unsaturated NM node in the tree, a joining AM node will connect to another unsaturated AM node in the tree using a unicast mutual anonymity protocol. If there are too many AM nodes in the system, MO nodes will be invited to join the multicast tree so that the new AM node can connect with the MO node using a unicast anonymity protocol. When to invite MO nodes depends on the cost ratio of unicast initiator anonymity protocol and unicast mutual anonymity protocol, and the ratio of AM nodes in a system. The pseudo code for a joining node P is shown as below.

Algorithm 1 Pseudo code of unicast initiator anonymity protocol

P contacts a bootstrapping server and gets a list of members
P contacts one active member and gets a full list of members
if P is an NM node **then**
  make a direct connection to an unsaturated NM node
end if
if (P is an AM node) **AND** (P can find unsaturated NM members) **then**
  make AM-NM connections with few unsaturated NMs
else if (P is an AM node)**AND**(P can find unsaturated AM members) **then**
  make AM-AM connections with several AMs
end if
if $\frac{number\ of\ AM\ nodes}{number\ of\ NM\ nodes} > \text{IT}$ **then**
  Invite MO nodes
end if
if timeout **then**
  Tree optimization
end if

6.4 Mutual Anonymity Protocol Design

Here we present the protocol design of mutual anonymity in overlay multicast system. This include a unicast initiator anonymity protocol, a unicast mutual anonymity protocol, and anonymous multicast tree construction. We also analyze the cost and connection latency of anonymous connections at the end of this section.

6.4.1 A Unicast Initiator Anonymity Protocol Design

The idea of Onion and a reverse Onion can be used to achieve initiator anonymity for bi-directional communication. Since an AM node has more than one choice for NM nodes to make AM-NM connections with, we optimize the Onion protocol as follows to keep both strong anonymity and robustness.

In the improved protocol for AM-NM connection, a Remailer (a reverse Onion) is generated by the AM node for the NM node to anonymously send messages to the AM node. In the AM→NM communication direction, the AM node uses an approach

similar to Crowds and Tor, in which each middle node in the path can make a decision to forward the message to another middle node or the NM node. This approach is more robust than Onion in the case of middle node failure.

In order to simplify the protocol description, we use S to denote the AM node, and use R to denote the NM node, as below. Note that S knows R's identity, but R does not know anything about S. Since this connection is initiated by S, we also label S as the initiator. In the rest of the Chapter, we use $\{M\}K$ to indicate that M is encrypted with the key K. K_{p+} denotes p's public key and K_{p-} denotes p's private key.

Step 1: The node S first generates m, the number of middle nodes in the Remailer. S then randomly selects a list of m nodes, p_0, p_1, p_2,..., p_{m-1} to form a Remailer. The lifetime of this one-time Remailer in seconds is also generated. The Remailer is built with S as the last member of the path and with p_i in the middle. It is of the form:

$$\{p_{m-1}, \{p_{m-2}, \cdots \{p_0, \{S\}K_{p_{0+}} \cdots\} K_{p_{(m-2)+}}\} K_{p_{(m-1)+}}\} K_{R+}\}$$

Step 2: S randomly selects a node, q_0, sends it the message: $S \rightarrow q_0$: $\{R$, $\{$Remailer, life-time$\}K_{R+}\}$.

Step 3: A peer q_i can elect itself to act as a deliver with a predefined forwarding probability h. If q_i is self-elected, the message $\{$Remailer, lifetime$\}K_{R+}$ will be delivered to the non-anonymous member node R directly. Otherwise, q_i will randomly select another node, q_{i+1} and forward the message $\{R$, $\{$Remailer, lifetime$\}K_{R+}\}$ to it.

Step 4: On receiving the message $\{$Remailer, lifetime$\}K_{R+}$, R uses its private key to decrypt the encrypted message.

Step 5: R generates a symmetric key K, and encrypts the multicast packet f with K. R then encrypts K with its private key. It keeps sending multicast packets

with the format as below through the Remailer to S:

$$R \rightarrow S : \{f\}K, \{K\}K_{R-}$$

Step 6: S uses R's public key to decrypt the symmetric key K, and uses K to decrypt the content encrypted by K.

At any time, a node R may have one or more Remailers. It will check the age and the expected lifetime for each Remailer periodically and delete obsolete Remailers. Each live Remailer corresponds to one AM node. Each AM node may connect with different NM nodes with the same or different Remailers for two reasons: increasing the difficulty for a NM node to guess the identity of the AM node, and providing multiple paths to the AM node in case of failure of any middle nodes in a Remailer. Too many Remailers for the same AM node will increase overhead, which can be adjusted by setting shorter lifetimes for the Remailers.

6.4.2 A Unicast Mutual Anonymity Protocol Design

When a joining AM node cannot find an unsaturated NM node in the tree, one option for him is to connect to another unsaturated AM node in the tree using a unicast mutual anonymity protocol. Most unicast mutual anonymity protocols were proposed for file sharing systems and may not be applicable here directly because of their low efficiency [43, 85, 94, 103, 112]. Therefore, we need to design a new unicast mutual anonymity protocol. Designing an efficient mutual anonymity protocol is difficult, but is possible here by utilizing the mechanism of MO node invitation.

We can use IP addresses to identify NM/MO nodes because they do not need to be anonymous. Instead, each AM node randomly selects an 18 byte value using a given algorithm to ensure its uniqueness when it joins the system. Note that AM nodes may change their ID_{AM} at any time for anonymity consideration. At the time of its joining, each AM node is bounded with one or multiple MO nodes, which means

an AM node sends Remailers to its bounded MO nodes, and its ID_{AM} and bounded MO nodes' IP addresses, e.g. ID_{AM}-IP_{MO_1},...,ID_{AM}-IP_{MO_i}, ..., will be kept in other NM nodes.

When an AM node (AM_1) decides to make a connection to another AM node (AM_2), it will select one of its bounded MO nodes (MO_i) to establish a connection with and one of AM_2's bounded nodes (MO_j). The connection between AM_1 and MO_i, and the connection between AM_2 and MO_j are established by the unicast initiator anonymous protocol introduced in the previous section. A connection of ID_{AM_1}-IP_{MO_i}-IP_{MO_j}-ID_{AM_2} is therefore established to achieve mutual anonymity between AM_1 and AM_2.

6.4.3 Anonymous Multicast Tree Construction

Many previous studies have intensively studied how to build an efficient overlay and optimize a random overlay, so we will not focus on this issue here. We use an idea similar to the Narada protocol [34] to build our multicast overlay among NM/MO nodes. The basic idea of Narada is to construct an efficient connected mesh first. Narada then constructs shortest path spanning trees of the mesh, each tree rooted at the corresponding source using well known routing algorithms.

As in Narada, every NM node and invited MO node has a full list of all the members. A joining node is able to get a list of group members (not necessary complete or accurate) by an out-of-band bootstrap mechanism, and randomly selects several unsaturated members to connect with. If the new joining node is an AM node that needs to hide its identity, it will randomly select one or multiple unsaturated NM/MO nodes forming anonymous connections with them, which is described in Section 6.4.1.

The multicast tree needs to be maintained. All the NM nodes probe their distances with all the other NM nodes and share the information among the overlay, so that

every single node has an identical distance table including each pair of the NM nodes. As the design is for small sized systems, maintaining such a list is not difficult. With such a table, a good multicast tree including all NM nodes can be easily computed and maintained [34]. The distance between a NM node and an AM node is not available because the AM node is anonymous to NM nodes, but it is also not necessary since the AM node is connected with a NM node via a number of middle nodes making the direct distance between the AM node and the NM node meaningless in optimizing the tree. However, the ID_{AM} of the AM nodes can be kept in the NM nodes, and a NM node knows the number of AM nodes that connect with it via anonymous passage but does not know their identities. In optimizing the tree, this NM node will subtract the number of its connected AM nodes from its bounded degree.

When a joining AM node cannot find an unsaturated NM node in the tree, one option for him is to connect to another unsaturated AM node in the tree using a unicast mutual anonymity protocol described in Section 6.4.2. However, we do not wish to see too many AM-AM mutually anonymous connections for performance reasons. Therefore, in some situations, MAM considers inviting some MO nodes to help by joining the system. We define an Invitation Threshold, IT. When the ratio of AM nodes to NM/MO is greater than the value of IT, the system will try to invite some MO nodes to join. When MO nodes are invited into the systems, joining AM nodes will have chances to join the tree by making AM-NM connections instead of more expensive AM-AM connections.

6.4.4 Cost and Latency of Anonymous Connections

There may be additional cost and latency for multicast systems when we try to provide anonymity to a set of member nodes. Here we discuss the potential influence of the proposed unicast mutual anonymity protocols of MAM on the cost and latency of the overlay multicast.

First, the selection of the number of middle nodes, m, has great impact on the anonymity degree and the cost of the connections. Obviously there is a tradeoff between the anonymity degree and the cost. Specifically, a larger m will result in a higher anonymity degree while incurring larger cost and latency.

Second, the predefined forwarding probability h also partially influences the cost and latency of data delivery in the system. In MAM, for simplicity, we uniformly select the value of h for the peering nodes. In real systems, nodes may select h independently, and the variety of h will improve the anonymity degree provided to the clients.

Third, the average cost of an AM-AM connection is at least two times greater than an AM-NM/MO connection. If we take (1) the dynamic nature of the member nodes, and (2) each AM node may use a set of NM/MO and switch some of them, into consideration, the average cost of an AM-AM connection is more than twice of that of an AM-NM/MO connection.

6.5 Anonymity Degree Analysis

The attacks targeted at overlay multicast system as well as anonymity degree achieved by MAM protocol are discussed in this section.

6.5.1 Attack Model

We assume that the attacker will break into some overlay nodes that are chosen randomly in one round, and try to figure out who the AM nodes are using the information that he obtains from the broken nodes. We assume the attacker can find the single parent and k children of all the nodes that have been broken. We also assume that the broken node keeps forwarding the packets in the same way as that before it is broken. We call the parents of all those broken nodes the potential root of a subtree with AM

nodes, which is called an implicit tree. The attacker will give each potential root a coefficient that is related with the probability regarded by the attacker as the root of the implicit tree by utilizing the information he obtains from all the broken nodes. A node that is more likely to be the root of the implicit tree has a higher coefficient, which means it is more important than other broken nodes and more prone to further attack.

The objective of the attacker is to use the above coefficients for future attacks, e.g., the attacker can launch a congestion attack to the potential root(s) to deny the service of as many receivers as possible, or the attacker can launch another break in attack to the potential root(s) to find the identity of the root of the implicit tree. No matter what the next attack is, the attacker will try to attack the node(s) that are more likely to be closer to the root of the implicit tree since he can potentially deny service to more receivers if he launches a congestion attack or has a higher probability to get the identities of all the receivers in the implicit tree if he launches a break in attack.

One thing to be reminded of here is that two broken nodes that are two layers apart can generate a broken tree with length of three by sharing information with each other. An example is that node A is in the i^{th} layer, while node B is in the $(i+2)^{th}$ layer. After sharing the parent and children information with each other, node A finds that the parent of node B is actually one of its children, so a three layer broken tree is generated by nodes A and B. In this case, an unbroken node can also be on a broken tree as long as both its parent and at least one of its children are broken. We call node A the head of the broken tree if and only if node A is broken while its parent and grandparent are not broken. Similarly, we call node B the tail of the broken tree if and only if node B is broken while none of its children and grandchildren are broken. If node A is in the i^{th} layer and node B is in the j^{th} layer, we call this broken tree a broken tree with the length $j - i + 1$, which is basically

the number of nodes in this broken path. Generally, all the broken nodes can form a broken forest comprised of several broken trees that are subtrees of the implicit tree. We denote the length of a broken tree as the length of the longest broken path in the broken tree.

6.5.2 Anonymity Degree Analysis

The metric we use to analyze anonymity degree is P_{reveal}, which is the probability that the identity of an AM node is revealed. If the AM node itself is broken, this probability is 1, otherwise, we calculate this probability according to a weight. Each node has a weight that stands for how sure the attacker thinks that this node's parent or one of its children is an AM node. Each node could be the root of a broken tree or the tail of a broken path, which we will define later. We believe the longer the broken tree or the broken path is, the more weight the attacker will give to this node. Our analysis is focused on how to get the weight of each node based on the distribution of the length of the broken tree or broken path.

In this section, we assume the multicast tree structure is a k-nary incomplete tree with L+1 layers and the root node is at Layer 0. Here an incomplete tree means that some receivers are not in the L^{th} layer. The receivers can be located from the first layer to the L^{th} layer. We assume in the incomplete tree scenario, each node has either 0 or k children. We introduce the incomplete tree in the hope of achieving better bandwidth efficiency since there is no redundant link in an incomplete tree. Here, we introduce a set of parameters $q_{i,j}$, which is a value given to each node $p_{i,j}$ in the tree. We let $q_{i,j}$ be 1 if node $p_{i,j}$ is a real node in the tree. We let $q_{i,j}$ be 0 if it does not exist in the tree. We also assume that the attacker has successfully broken into N nodes in this tree. Since the attacker chooses the nodes randomly for break in

attack, the probability of each node in the tree being broken is equal, which is:

$$P_{broken} = \frac{N}{\sum_{i=0}^{L} \sum_{j=1}^{k^i} q_{i,j}} \tag{6.1}$$

We first analyze anonymity degree of AM as a sender and then analyze the anonymity degree of AM as a receiver. If the root of the tree is one of the broken nodes, the attacker has already obtained all the information he needs. Otherwise, there is a probability that the real root will be regarded as the root and may be subject to the next attack. The overall probability that the identity of the root is revealed is shown below,

$$P^S_{reveal} = P_{broken} + (1 - P_{broken}) \times P^S_{attack} \tag{6.2}$$

$$P^S_{attack} = \sum_{j=1}^{k} \left(\frac{w^S_{1,j}}{\sum_{i=1}^{L} \sum_{j=1}^{k^i} w^S_{i,j}}\right) \tag{6.3}$$

$$w^S_{i,j} = \begin{cases} 0 & (q_{i,j} = 0) \\ P_{broken} \times \left(\sum_{l=1}^{L} p^S_{i,j}(l \times f(l))\right) & (q_{i,j} = 1, i = 1) \\ P_{broken} \times (1 - P_{broken}) \times \left(\sum_{l=1}^{L} p_{i,j}(l) \times f(l)\right) & (q_{i,j} = 1, i = 2) \\ P_{broken} \times (1 - P^2_{broken} \times \left(\sum_{l=1}^{L} p^S_{i,j}(l) \times f(l)\right)) & (q_{i,j} = 1, i > 2) \end{cases} \tag{6.4}$$

where $w^S_{i,j}$ is the weight given to node $n_{i,j}$ (i.e. j_th node in i_th layer), which is the head of a broken tree; $p^S_{i,j}l$ is the probability that the length of the broken tree with node $n_{i,j}$ as the head is l ; $f(l)$ is a function that increases when l increases.

The exact form of $f(l)$ depends on the attacker's policy. We choose $f(l) = l$ in this thesis.

In an incomplete tree, different nodes at the same layer have different probabilities of being the head of a broken tree of a specific length. A node that has more "deeper" descendant has higher probability of being a head of a long broken tree and vice versa.

$$p_{i,j}^{S}(l) = \begin{cases} 0 & (q_{i,j} = 0) \\ 0 & (l \leq 0 \text{or} l > L) \\ 1 & (q_{i+1,k\times j} = 0, l = 1) \\ (1 - P_{broken})^{k+\sum_{n=1}^{k^2} q_{i+2,k^2\times j-k^2+n}} & q_{i+1,k\times j} = 1, l = 1 \\ 0 & q_{i+1,k\times j} = 1, l > 1 \\ \sum_{j=1}^{\sum_{n=1}^{k^2} q_{i+1,k^2\times j-k^2+n}} (p_{bg}(j) \times & \\ p_{i+2,j}^{S}(l-2|bn=j)) \times (p_{bc}(0) + & \\ \sum_{j=1}^{k}(p_{bc}(j) \times p_{i+1,j}^{S}(l-1|bn=j))) & \text{otherwise} \end{cases} \tag{6.5}$$

Here, $p_{bc}(i)$ is the probability that i children have been broken. Similarly, $p_{bg}(i)$ is the probability that i grandchildren have been broken. $p_i(l|bn = j)$ is the probability that the longest broken tree among j trees is of length l, given that j

children of a parent, which is in the $(i-1)^{th}$ layer, have been broken in the i^{th} layer. $p_{bc}(i)$, $p_{bg}(i)$ and $p_i(l|bn=j)$ can be calculated as:

$$p_{bc}(i) = \binom{k}{i} \times p_{broken}^{i} \times (1 - p_{broken})^{k-i} \tag{6.6}$$

$$p_{bg}(i) = (\sum_{n=1}^{k^2} q_{i+2,k^2\times j-k^2+n}) \times p_{broken}^{i} \times (1 - p_{broken})^{\sum_{n=1}^{k^2} q_{i+1,k^2\times j-k^2+n} - i} \tag{6.7}$$

$$p_{i,k}^{S}(l|bn=j) = \begin{cases} 0 & (l=0) \\ \sum_{m=1}^{j} \left(\binom{j}{m} \times (\overline{p_{i,j}^{S}(l)})^{m} \times \right. & \\ \left. (\sum_{n=1} l - 1 \overline{p_{i,j}} S(n))^{j-m} \right) & (l>0) \end{cases} \tag{6.8}$$

Here, $\overline{p_{i,j}^{S}(l)}$ is calculated as:

$$\overline{p_{i,j}^{s}(l)} = \frac{(\sum_{j=k\times i-k+1}^{k\times i}(l))}{k} \tag{6.9}$$

So far, we have finished analyzing how to get p_{reveal} .

For anonymity degree of AM as a receiver, we denote the AM node we consider as $p_{u,t}$. Its parent is $p_{u-1,[\frac{t}{k}]}$and grandparent is $p_{u-2,[\frac{[\frac{t}{k}]}{k}]}$. Here, we denote [i] as the largest integer that is no more than i. We give the formulae to calculate the

probability that the identity of the AM as a receiver is revealed as below.

$$p^r_{reveal} = \left(1 - (1 - p_{broken})^2\right) + (1 - p_{broken})^2 \times p^r_{attack} \tag{6.10}$$

$$w^r_{i,j} = \begin{cases} 0 & q_{i,j} = 0 \\ 0 & q_{i,j} = 1, q_{i+1,k\times j} = 0 \\ p_{broken} \times (1 - p_{broken}) \times \left(\sum_{l=1}^{L-1} p^r_{i,j}(l) \times f(l)\right) & q_{i,j} = q_{i+1,k\times j} = 1, \\ & q_{i+2,k^2\times j} = 0, \{i,j\} \neq \{u-1, [\frac{t}{k}]\} \\ \left(\sum_{l-1}^{L-1} p^r_{i,j}(l) \times f(l)\right) \times p_{broken} & q_{i,j} = q_{i+1,k\times j} = 1, \\ & q_{i+2,k^2\times j} = 0, \{i,j\} = \{u-1, [\frac{t}{k}]\} \\ p_{broken} \times (1 - p_{broken})^2 \times \left(\sum_{l=1}^{L-1} p^r_{i,j}(l) \times f(l)\right) & q_{i,j} = q_{i+1,k\times j} = q_{i+2,k^2\times j} = 1, \\ & \{i,j\} \neq \{u-2, [\frac{[\frac{t}{k}]}{k}\} \\ p_{broken} \times (1 - p_{broken}) \times \left(\sum_{l=1}^{L-1} p^r_{i,j}(l) \times f(l)\right) & q_{i,j} = q_{i+1,k\times j} = q_{i+2,k^2\times j} = 1, \\ & \{i,j\} = \{u-2, [\frac{[\frac{t}{k}]}{k}\} \end{cases} \tag{6.11}$$

$$p^r_{attack} = \begin{cases} 0 & u = 1 \\ \dfrac{\left(\dfrac{w^r_{u-1,[t/k]}}{\sum_{i=1}^{L}\sum_{j=1k^i} w^r_{i,j}}\right)}{k} & \end{cases} \tag{6.12}$$

$$p_{i,j}^{r}(l) = \begin{cases} 0 & l > u-1 \\ 0 & l < 1 \text{or} i < 1 \\ 1 & l = 1, i = 1 \\ 1 - p_{broken} & l = 1, i = 2 \\ (1 - p_{broken})^2 & l = 1, i > 2 \\ p_{broken} \times p^{r}_{i-1,[\frac{j}{k}]}(l-1) + & \\ (1 - p_{broken}) \times p_{broken} \times p^{r}_{i-2,[\frac{j}{k}]/k}(l-2) & \text{otherwise} \end{cases} \tag{6.13}$$

Here, the definition of $p_{i,j}^{r}(l)$ is similar to $p_{i,j}^{S}(l)$, which is defined before.

6.5.3 Numerical Results and Discussions

In the following discussion, we will consider the numerical results based on the above formulae for anonymity degree in the incomplete tree. The incomplete tree we use is a binary tree. The root node has four grandchildren; one is the root of a complete subtree with 2 leaves in the third layer, one is the root of a complete subtree with 4 leaves in the fourth layer, one is the root of a complete subtree with 8 leaves in the fifth layer and the other is the root of a complete subtree with 16 leaves in the sixth layer. The data are obtained in MATLAB.

Figure 6.2 and 6.3 show the sensitivity of anonymity degree to broken ratio. Different curves represent different combinations of k and L. Anonymity degree is represented by P_{reveal}. Smaller P_{reveal} means better anonymity degree. It is obvious that anonymity degree improves as broken ratio decreases.

When the percentage of broken nodes and L is fixed, anonymity degree improves when k increases. This is because when the tree grows wider, the broken nodes tend to be in different branches. The length of the broken tree tends to decrease. When the percentage of broken nodes and k is fixed, anonymity degree improves when L increase. This is because when the tree grows deeper, the length of the broken tree tends to decrease.

The AM sender anonymity of the incomplete tree in our example in Figure 6.2 is between those of the complete binary tree with all receivers in the third or sixth layer. This is obvious because the AM sender's anonymity improves when the tree grows. We observe that the difference between the incomplete tree curve and the complete tree curve with six layers is very slight. This is because the children of the sender who has fewer descendants will have comparatively small weight, which helps to improve the AM sender's anonymity. Actually we can achieve significant bandwidth efficiency with little sacrifice on AM sender's anonymity.

The AM receiver anonymity of the incomplete tree in our example in Figure 6.3 is worse than that of the complete tree with sixth layers. This is because fewer nodes will be considered as the parent of the receiver, the comparative weight of the AM receiver's parent tends to increase. We observe that among all the AM receivers in the incomplete tree, the higher AM receivers have better anonymity than the lower ones. This is because their parents tend to be the tail of a shorter broken tree, which helps to decrease their weight. This fact holds under the assumption that the attacker does not know the layer of the AM receiver. We observe that the differences among different AM receivers in the incomplete tree case and between the incomplete tree

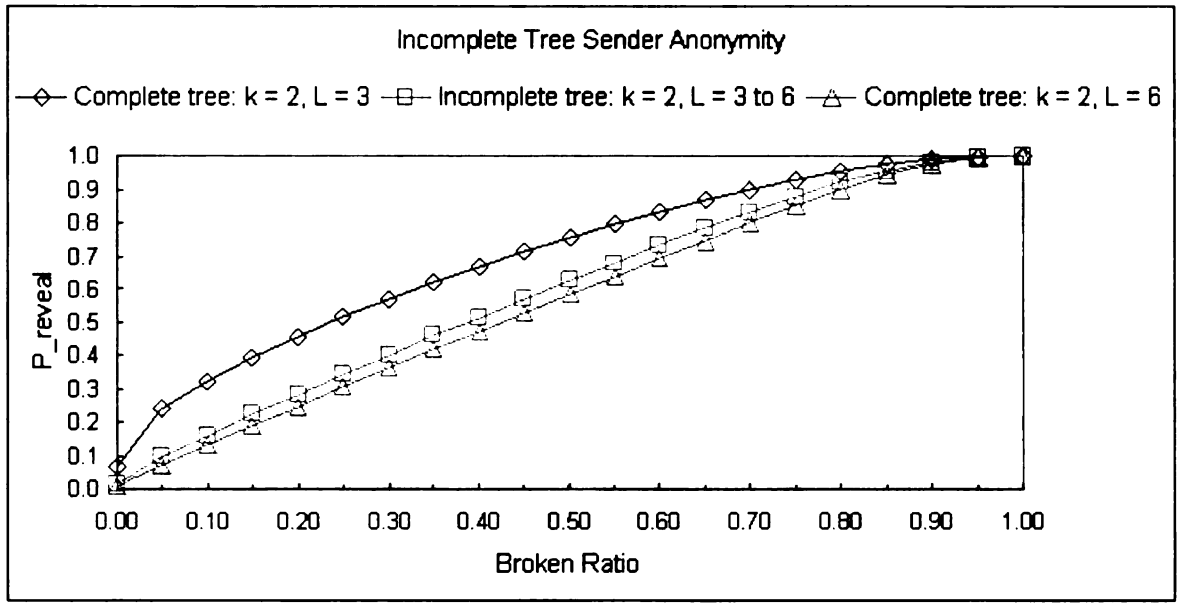


Figure 6.2. Anonymity degree of AM as a sender

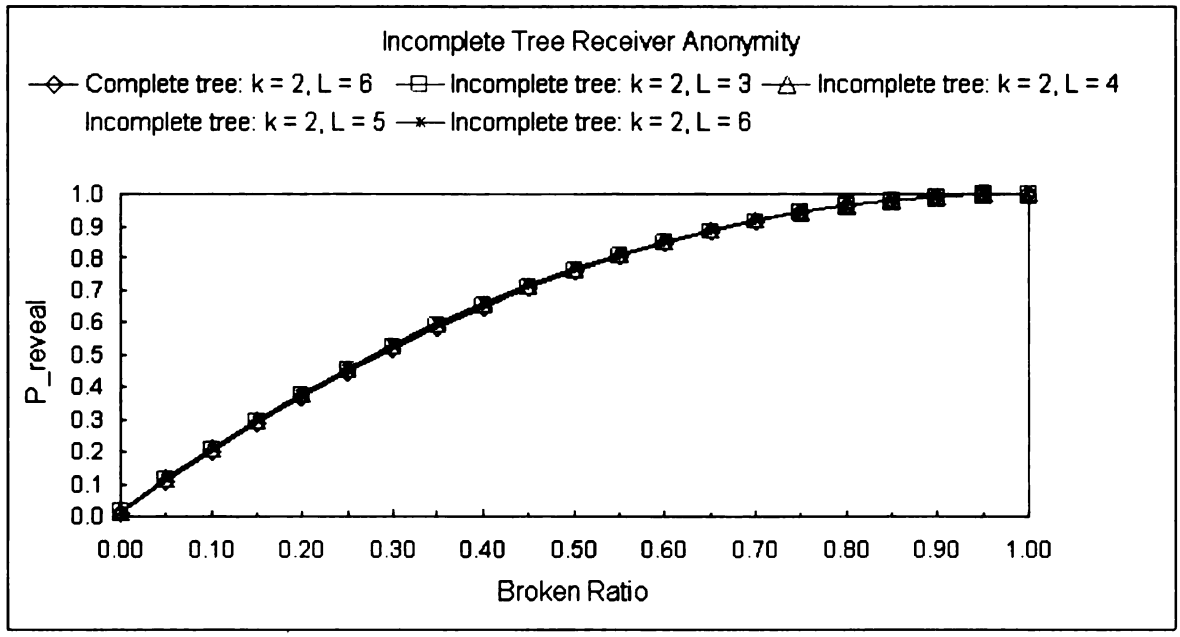


Figure 6.3. Anonymity degree of AM as a receiver

case and the complete tree case are very slight because the AM receiver's anonymity is dominated by the probability that the sender or the receiver is broken, which is determined by the percentage of broken nodes. This means that significant bandwidth efficiency can be achieved with little sacrifice of AM receiver's anonymity.

6.6 Performance Evaluation

We experimentally measure the additional overhead incurred in our design for achieving AM-NM and AM-AM anonymous connections, and use comprehensive simulations to evaluate the effectiveness of MAM.

6.6.1 Simulation Methodology

Two types of topologies, physical and logical topologies are generated in our simulation. The physical topology should represent the real topology with Internet characteristics. The logical topology represents the overlay system built on top of the physical topology. The communication cost between two logical neighbors is calculated based on the physical shortest path between this pair of nodes. To simulate the performance of the MAM protocol in a more realistic environment, the two topologies must accurately reflect the topological properties of real networks in each layer. BRITE [2] is a topology generation tool that provides the option to generate topologies based on the AS Model. Using BRITE, we generate physical topologies with 3,000 to 7,000 nodes. The average number of neighbors of each node ranges from 4 to 10. The 100 to 300 overlay nodes are randomly selected from the nodes in the physical topologies.

To reflect the real overlay systems, in the experiments we report here, member nodes are coming and leaving according to the distribution observed in [90]. The mean of the distribution is chosen to be 1800 seconds. The value of the variance is

chosen to be half of the value of the mean. In each experiment, a number of nodes join the system at the first 120 seconds of the simulation in random sequence. The lifetime of each node will be decreased by one after passing each second. A member will leave in the next second when its lifetime reaches zero. During each second, there are a number of members leaving the system, and we then randomly pick up (turn on) a similar number of members from the physical network to join the system.

In all the experiments, every 50 seconds, random nodes are selected as senders to multicast data at a constant rate, and the simulations run for 60 minutes. In the MAM protocol, the lifetime of *Remailers* is randomly selected from 50 to 200 seconds.

6.6.2 Cost Measurement of Encryption Techniques

We measure the cost of basic cryptography techniques to obtain a base data for our further simulation. In the first experiment, we measured the data on five crypto servers, in which 2,000 overlay servers and clients of the ChinaPCCN are involved. For each area center, local machines initiate (1) 900,000 symmetric encryption requests, each of which encrypts a 1-100KB length file, (2) 900,000 MAC code generation requests, each of which generates a 20 Byte MAC code for a 1-100K length file, and (3) 10,000 RSA encryption and decryption requests, which respectively call for a signature or verification for the 10,000 MAC results generated above.

In our second experiment, we ran the crypto software kits [12] on a desktop PC with an 800MHz Pentium III CPU and 256Mbytes of memory, 20GBytes hard disk and 10/100M Ethernet card. We ran each test 10 times, and use the average data.

On crypto servers, the average 1024-bit RSA decryption count per second is from 17.12 to 103.09 compared to 14.6 for software tools. The encryption is from 322.4 to 1941.7 compared to 275 for software. Normally the RSA encryption is about 10 19 times faster than the decryption process. The 768-bit and 512-bit test results also validate it. In most cases, the 1024-bit key RSA can be regarded as a sufficient

security cryptography algorithm, so in our simulation on the overlay multicast system, we chose the 1024-bit RSA as the crypto process in the onion path and use 45.87 and 864 per second, from the crypto server's results, as the reference values of the 1024-bit RSA performance.

The DES performance is in the range from 2.24Mbps to 8.78Mbps, and the 3DES is from 1.89Mbps to 6.43Mbps. The hash function of MD5 performance is from 0.97Mbps to 11.64Mbps, and SHA-1 is from 3.23Mbps to 11.28Mbps. The software implementation performance is almost at the average level of those five servers. Therefore, we chose 5.41Mbps as the DES speed and 7Mbps as the SHA-1 speed.

6.6.3 Performance Metrics

We compare the performance of three different approaches: Optimal, MAM, and RAND. In Optimal, the anonymous multicast tree is optimized using an offline algorithm. In a nave approach, indicated as "RAND", each joining node randomly selects a member to connect to the multicast tree.

We use two performance metrics: *relative resource usage* (RRU) and *average worst-case delay* (AWD).

The stress of a physical link is defined in [34] as the number of identical copies of a packet carried by a physical link. We define resource usage as $\sum_{j=1} nd_j \times s_j$, where d_j is the delay of link j and s_j is the stress of link j. Resource usage is one of the parameters of seriously concerned to network administrators. Heavy network traffic limits the scalability of overlay networks [87]. RRU is defined as the ratio of the resource usage of MAM or other approaches to the optimal anonymous multicast tree. AWD is the average delay from the source to the farthest node that gets the multicast packets, when nodes are selected at random as the source nodes in multiple runs.

6.6.4 Simulation Results

When there is no member needing to hide its identity, the system will be the same as normal end system multicast. Intuitively, the total cost of the system will increase when the number of nodes need to be hidden increases. We first show MAM's performance by increasing the number of nodes that need to achieve anonymity (AM nodes) in the system.

With 3000 physical nodes and 200 overlay multicast members, Figures 6.4 and 6.5 plot the RRU and AWD of different approaches versus the number of AM nodes in the system. When the ratio of AM nodes in the system is small, MAM's RRU is very close to the optimal solution. MAM's AWD is very close to the optimal solution when less than half of the nodes are AM nodes. We vary the system size from 100 to 400, and the physical network size from 2,000 to 8,000. The results are consistent, indicating that MAM maintains effectiveness, and the RRU and AWD of MAM are not sensitive to the system size or the physical networks size.

If all of the members in a system are AM nodes, even the optimal solution will be as bad as the nave RAND approach, and a system of smaller size can incur greater traffic overhead than a system with larger size. Hence, in MAM, we propose to avoid having all the members as AM nodes by inviting MO nodes into the system. Frankly, it is always helpful if more MO nodes can join the system. However, the overhead of inviting MO nodes is hard to predict: they merely provide service to the system but do not consume the multicast content.

In the "MAM" protocol, MO nodes are not invited. We can see that when the percentage of AM nodes is large, both RRU and AWD degrade significantly. We investigate the effectiveness of inviting MO nodes to join in Figures 6.4 and 6.5 using the "MAM-MO" protocol, in which MO nodes are invited when the ratio of AM nodes in the system reaches 90%. The improvement is substantial for a system with more than 270 AM nodes (90% of the system). The next question is when is the best

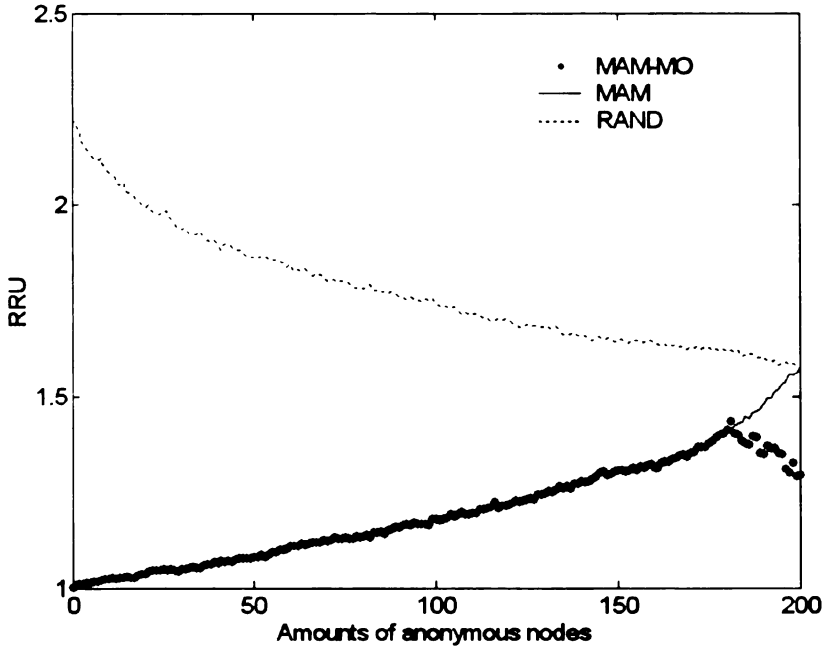


Figure 6.4. RRU vs the number of AM nodes

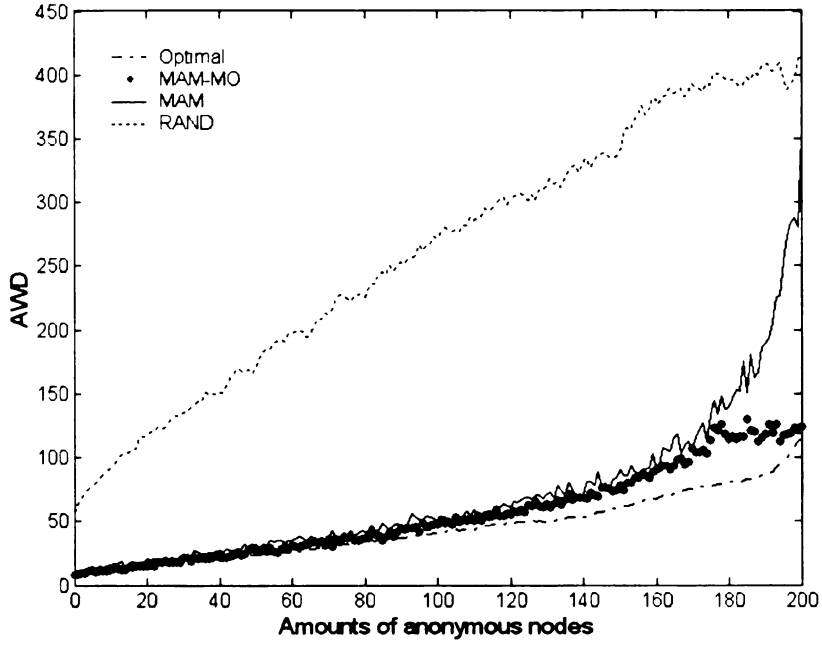


Figure 6.5. AWD vs the number of AM nodes

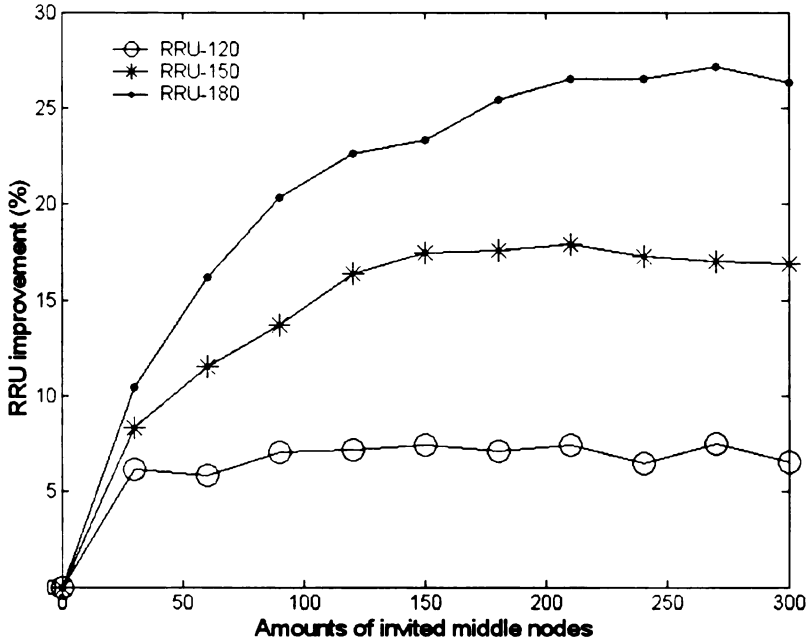


Figure 6.6. RRU improvement vs. # of invited MO nodes

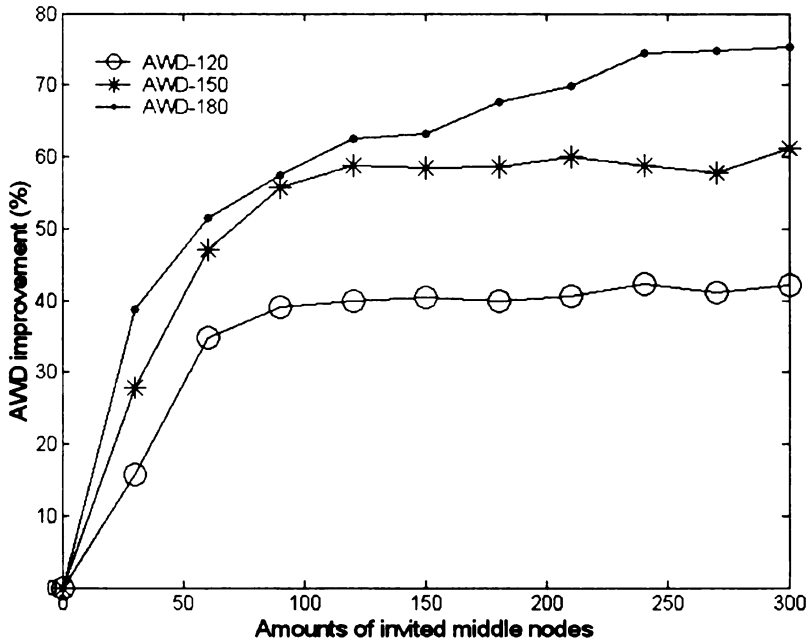


Figure 6.7. AWD improvement vs. # of invited MO nodes

time for the system to invite MO nodes, i.e. what is the best Invitation Threshold IT. Figures 6.6 and 6.7 show the RRU improvement and AWD improvement versus the number of invited MO nodes for a different given number of AM nodes in a system with 200 overlay multicast members. RRU/AWD improvement is defined as the percentage of the RRU/AWD improvement with the MAM-MO protocol over the MAM protocol without MO node invitation. "RRU-n"/ "AWD-n" means the RRU/AWD improvement for a given number of n AM nodes. In general, inviting more MO nodes means better performance with the assumption that we have an infinite number of available MO nodes to be invited.

However, when a certain number of MO nodes have been invited, inviting more MO nodes is not as effective as before. For example, there is a clear jump in Figure 6.6 for RRU-120, which shows that when 30 MO nodes have been invited, inviting more MO nodes gives little additional RRU improvement, where the corresponding IT is 120/(200+30)=52%. Similarly, the ITs for RRU-150 and RRU-180 are 47% and 47%. If we calculate the ITs from Figure 6.7, we have 46%, 52% and 47% for AWD-120, AWD-150, and AWD-180 respectively.

Therefore, our interpretation of the experimental results is that when less than around 50% of the nodes wish to be anonymous, MAM may be directly used with no need to invite MO nodes; otherwise, MO nodes should be invited to keep the ratio of AM nodes in the system at about 50%. Beyond this, inviting more MO nodes is not necessary.

6.7 Summary

In this chapter, we propose the MAM protocol to provide anonymous multicast service and protect user information. The main work presented in this chapter is summarized below:

1. We define different types of anonymity in an anonymous multicast system, and show the rationale of our focus on multicast mutual anonymity. Our analysis shows that the anonymity degree of AM nodes is correlated with the broken ratio, tree degree, and tree depth. Compared to the complete multicast tree, the incomplete multicast tree can achieve a similar anonymity degree with much higher bandwidth efficiency.

2. We propose MAM protocol, and address the critical issues in this protocol, which include an efficient and robust unicast initiator anonymity protocol, an efficient unicast mutual anonymity protocol, and an effective anonymous multicast construction approach. The self-organized and completely distributed design of MAM can efficiently realize mutual anonymity in overlay multicast systems.

3. We define the attack model to an anonymous multicast system, and theoretically analyze the anonymity degree of the MAM protocol.

4. We evaluate the effectiveness of MAM in a dynamic environment by comprehensive simulation study. The study evaluation shows that MAM is an effective approach to construct an efficient anonymous multicast tree. When the percentage of AM nodes in a system is below a certain level, without inviting MO nodes, MAM works as well as the optimal solution. We also show that inviting a certain ratio of MO nodes can be very effective for a system with a large number of AM nodes.

CHAPTER 7

Conclusion and Future Work

7.1 Conclusion

Composed by self-organized end systems, overlay networks are decentralized and utilize the edge resources of the Internet, distributing the operation cost across the nodes in the system. Overlay networks are open and provide new dimensions to the Internet usability. Running on top of the end systems, it does not need to overcome the administration barrier to deploy applications in overlay networks. This makes it easy to deploy applications in overlay networks. As a result, the scale of overlay networks has increased rapidly.

At the same time, overlay networks bring challenges in building reliable applications on top of them. The openness and devoid of membership management of overlay networks invite attackers to distribute malfunctioning data in the system. It is difficult to trace the attackers due to the lack of the centralized control. In addition, the end systems that form the overlay networks are highly dynamic, which can interrupt the services provided by overlay applications. As the end nodes in overlay networks are loosely connected either randomly or according to their local based algorithm, it is hard to avoid the appearance of "critical" nodes such as high-degree nodes and cut

vertices in the overlay networks.

The reliability of reliable applications is multi-folded. The basic and essential requirement of reliability is the availability of the service provided by the application. The service provided by the overlay applications should always be available for users, even when failure happens. For the overlay applications that provide content service to users, the service content should be authenticate. In other words, the service should be trustworthy to users. Moreover, the user safety should also be guaranteed as an extension of reliability.

In this thesis, the author makes efforts on building reliable applications by identifying the specific problems and proposing corresponding solutions on all three different aspects of reliability.

The first problem addressed in the thesis is the cut vertex problem, which hinders the service availability of the overlay applications. Cut vertices are one category of topological "critical" nodes that are induced by the randomness and self-organization that overlay nodes connect with each other. Cut vertices are unavoidable in overlay topology and application independent. The solution for the cut vertex problem is applicable for all overlay applications. Traditional methods of detecting cut vertices require the information of overall topology, which is impossible in today's large scale and highly dynamic overlay networks. In this thesis, CAM, a distributed solution that can effectively detect and neutralize the cut vertices, is proposed to combat the cut vertex problem in overlay networks. In the CAM mechanism, nodes in overlay networks send out probe messages to their neighbors to collect local connection information. The traffic cost induced by the probe messages is bounded by the hops of the probe messages. Based on the received feedbacks, a node can form a connection adjacent matrix(CAM) to figure out whether it is a cut vertex. A cut vertex will then be neutralized by adding extra connections to other nodes. At the same time, connections of the cut vertex are also selectively removed to offload the traffic that

is processed by the cut vertex. CAM greatly improves the reliability of the overlay network upon the failure of cut vertices. The correctness of the CAM algorithm is proved in the thesis. The effectiveness and efficiency of CAM algorithm are evaluated by the simulations based on the real world traces.

Besides the cut vertex problem, another problem that hinders the availability of the service, the response loss problem, is also addressed. The response loss problem is induced by the dynamics of the end nodes and the way the responses are sent back in the unstructured P2P file sharing system. Three different techniques are proposed to overcome the problem: the redundant response delivery (RRD) scheme as a proactive approach, the adaptive response delivery (ARD) scheme as a reactive approach, and the extended adaptive response delivery scheme(eARD) to function in an unstructured P2P system with limited or no flooding based search mechanism. RRD reduces the response loss rate by sending back redundant copies of response from responder via different paths. RRD requires the least modification of the existing response return mechanism and is the easiest technique among the three to deploy. In ARD, each node in the response return path needs to maintain a list of neighbors that forward the request to this node. The response can be sent back to a different neighbor if one neighbor fails. As responders in ARD do not send out redundant copies of responses, ARD induces less traffic than RRD. Both RRD and ARD are based on the assumption that one request will be forwarded by more than one neighbor to a node. eARD introduces the backup response delivery agent (bRDA) to store the response path information and loose the above assumption. All three techniques can effectively increase the response return rate, from up to 35% in RRD to 51% in eARD.

A hierarchical reputation system, hiREP, is proposed to guarantee the authenticity of the service content provided by P2P file sharing systems. hiREP adopts a hierarchical structure to effectively store and spread trust values in the P2P system. A node in the hiREP system only reports transaction results to and fetches trust

values of other nodes from its trusted reputation agents. This restricts the traffic cost induced by trust value storage and spread. Public key systems are used in hiREP to guarantee the data authenticity during the trust value storage and spread process. A nodeID associated with the public key hash of the node is assigned to each node, which enables the key distribution without third party certificate authority. Onion routing based communication is adopted during the reputation request process to protect the user information of voters.

In order to further improve the reliability of overlay applications for users, efforts are made to explore the user safety issues. Specifically, we investigate the mutual anonymity in overlay multicast systems. Different types of anonymity in an anonymous overlay multicast system are defined in the thesis. A mutual anonymity protocol, MAM, is proposed for implementing anonymity in overlay multicast systems. MAM includes an efficient and robust unicast initiator anonymity protocol, an efficient unicast mutual anonymity protocol, and an effective anonymous multicast construction approach. The anonymity degree provided by MAM against attacks is analyzed theoretically. The analysis shows that the anonymity degree of AM nodes is correlated with the broken ratio, tree degree, and tree depth. Compared to the complete multicast tree, an incomplete multicast tree can achieve a similar anonymity degree with much higher bandwidth efficiency. A series of comprehensive simulations are deployed to evaluated the performance of MAM. The simulations show that MAM can effectively construct an efficient anonymous overlay multicast tree.

7.2 Future Work

Looking forward to the future, work can be done on continuing refine and improve the proposed solutions, as well as identifying and solving problems that hinder the implementation of reliability in other categories of overlay applications. Specifically,

the future work will lie in three directions and is listed as follows:

1. Integrate our work of reducing cut vertices with other studies on optimizing the overlay topology to form a comprehensive solution on overlay topology optimization, which will further improve the reliability, and even the performance of overlay applications. For instance, our work can be combined with the work of P. Keyani 's work [55] on reducing high degree nodes to reduce both categories of topological "critical" nodes. They can be further integrated with the topology aware methodologies to help improve the performance. There will be problems and challenges in building such an integrated overlay topology optimization solution. Our investigation proves high-degree nodes are not necessarily cut vertices in overlays [63]. Removing high degree nodes by turning the overlay topology into an exponential topology may induce new cut vertices. In addition, the efforts to match the overlay topology with the underlying physical topology may lead to high degree nodes as well as cut vertices. Especially it is shown that the topology of the Internet follows power-law [42]. It may not be possible to achieve best results in all three aspects, i.e., minimizing the number of cut vertices and high degree nodes, as well as matching the overlay topology with the underlying physical topology. In this case, one/several new metrics may need to be proposed to measure the overall benefit based on cut vertices, high degree nodes, and topology matching, as well as an integrated solution that can optimizing the overlay topology according to the new metrics.

2. Overlay networks are built on top of the underlying physical networks. The characteristics of the underlying physical network may affect the reliability of the overlay on top of it. Our study focuses on the overlay network itself without taking into the account the influence of the underlying physical network. In the future, our research can be extended by exploring the influence of the underlying physical networks on the overlay networks.

For instance, the multihoming technology is now deployed extensively in stub networks to achieve reliability and redundancy on the connection to the Internet. Intuitively, this will have positive influence on the reliability of the overlay networks. However, studies need to be done to measure quantitatively how much positive effect this change of the underlying networks will bring to the upper layer overlays. In order to handle the connection failure scenario, the multihoming technology adopts some methodologies such as prepending the stub network's AS number in the back-up BGP route, advertising the IP address block associated with a broken connection to the upstream BGP router connected with a live connection, or providing a tunnel between the upstream BGP router connected with the broken connection and the live connection. These methods generally take some convergence time to move the packets originally going through the broken connection to the live connection. This may reduce the benefits that the multihoming technology brings to stub networks as well as overlay networks. In order to achieve better performance and reduce the financial cost of ISPs, traffic can be moved among different connections in the load balancing process. This may conflict with the adjustment/optimization of the overlay topology, which also tries to induce traffic among different paths in the overlay network. In this area, we have completed a survey on the current research in the multihoming technology and propose a load balancing mechanism for multi-homed stub networks [62, 66]. We plan to further investigate the effect of the underlying multihoming technology on the overlay topology.

3. Besides optimizing the overlay topology, our research addresses problems that are unique in the P2P file sharing systems and overlay multicast systems. This can also be extended by identifying the problems and issues that are unique in other categories of overlay applications. For instance, the dynamic feature of the end systems also affect the availability of the live streaming application,

which cause the loss of streaming packet and downgrade the quality of the live streaming. Authenticity will also be an issue in the parallel downloading system to guarantee the reliability of the downloaded data content. User safety will be critical in the applications such as micro-pay to protect user information and protect users from malicious behaviors such as identity theft. In this direction, problems need to be identified first, as well as the root causes related to the requirements/operations of the overlay application that leads to these problems. Solutions can then be proposed to solve the problems and increase the reliability of the applications. In addition, these solutions need to be easy to be adopted in the original application. They should also be designed in a way that does not downgrade the performance and reliability that are already achieved in the application.

BIBLIOGRAPHY

BIBLIOGRAPHY

[1] Amazon. *http://www.amazon.com.*

[2] BRITE. *http://www.cs.bu.edu/brite/.*

[3] DSS Trace. *http://dss.clip2.com.*

[4] eBay. *http://www.eBay.com.*

[5] Epinions. *http://www.epinions.com.*

[6] Gnutella Network Size. *http://www.limewire.com/index.jsp/size.*

[7] KaZaA. *http://www.kazaa.com.*

[8] Limeware. *www.limeware.org.*

[9] Napster. *http://www.napster.com.*

[10] P2P traces of the University of Oregon. *http://mirage.cs.uoregon.edu/P2P/info.cgi.*

[11] The Gnutella Protocol Specification 0.6. *http://rfc-gnutella.sourceforge.net.*

[12] RSAREF20. *http://tiranog.ls.fi.upm.es*, 1994.

[13] C. Abad, W. Yurcik, and R.H. Campbell. A Survey and Comparison of End-System Overlay Multicast Solutions Suitable for Network-Centric Warfare. In *SPIE Defense and Security Symposium/BattleSpace Digitalization and Network-Centric Systems IV*, 2004.

[14] A. Abdul-Rahman and S. Hailes. Supporting Trust in Virtual Communities. In *HICSS*, 2000.

[15] K. Aberer, P. Cudre-Mauroux, A. Datta, and Z.Despotovic. P-Grid: A Self-organizing Structured P2P System. In *International Conference on Cooperative Informantion Systems*, 1995.

[16] K. Aberer and Z. Despotovic. Managing Trust in a Peer-to-Peer Information System. In *ACM CIKM*, 2001.

[17] V. Almeida, A. Bestavros, M. Crovella, and A. de Olivera. Characterizing Reference Locality in the WWW. In *the IEEE Conference on Parallel and Distributed Information Systems (PDIS)*, 1996.

[18] David Andersen, Hari Balakrishnan, Frans Kaashoek, and Robert Morris. Resilient Overlay Networks. In *ACM SOSP*, 2001.

[19] S. Banerjee, B. Bhattacharjee, and C. Kommareddy. Scalable application layer multicast. In *ACM SIGCOMM*, 2002.

[20] Suman Banerjee, Christopher Kommareddy, Koushik Kar, Bobby Bhattacharjee, and Samir Khuller. Construction of an Efficient Overlay Multicast Infrastructure for Real-time Applications. In *IEEE INFOCOM*, 2003.

[21] R. Bhagwan, S. Savage, and G.M.Voelker. Understanding Availability. In *2nd International Workshop on Peer-to-Peer Systems (IPTPS)*, 2003.

[22] S. Birrer and F E. Bustamanti. Resilience in Overlay Multicast Protocols. In *IEEE MASCOTS*, 2006.

[23] L. Breslau, P. Cao, L. Fan, G. Phillips, and S. Shenker. Web Caching and Zipf-like Distributions: Evidence and Implications. In *IEEE INFOCOM*, 1999.

[24] F. Buckley and M. Lewinter. *A Friendly Introduction to Graph Theory.* Prentice Hall.

[25] F E. Bustamante and Y. Qiao. Friendships that Last: Peer Lifespan and Its Role in P2P Protocols. In *The 8th International Workshop on Web Content Caching and Distribution (WCW)*, 2003.

[26] R. Canetti, J. Garay, G. Itkis, D. Micciancio, M.Naor, and B. Pinkas. Multicast security: a taxonomy and some efficient constructions. In *IEEE INFOCOM*, 1999.

[27] M. Castro, M. B. Jones, A.-M. Kermarrec, A. Rowstron, M. Theimer, H. Wang, and A. Wolman. An Evaluation of Scalable Application-level Multicast Built Using Peer-to-peer Overlays. In *IEEE INFOCOM*, 2003.

[28] M. Castro, A. Rowstron, A.-M. Kermarrec, and P. Druschel. SCRIBE: A large-scale and decentralised application-level multicast infrastructure. *IEEE Journal on Selected Areas in Communication*, 20(8), 2002.

[29] D. Chaum. Untraceable Electronic Mail Return Addresses, and Digital Pseudonyms. *Communications of the ACM*, 24(2):84–90, 1981.

[30] Y. Chawathe, S. Ratnasamy, L. Breslau, N. Lanham, and S. Shenker. Making Gnutella-like P2P Systems Scalable. In *ACM SIGCOMM*, 2003.

[31] Y. Chen, R H. Katz, and J D. Kubiatowicz. Dynamic Replica Placement for Scalable Content Delivery. In *IPTPS*, 2002.

[32] Y. Chu, A. Ganjam, T. Ng, S. Rao, K. Sripanidkulchai, J. Zhan, and H. Zhang. Early Experience with an Internet Broadcast System Based on Overlay Multicast. In *USENIX Annual Technical Conference*, 2004.

[33] Y. Chu, S. Rao, S. Seshan, and H. Zhang. Enabling Conferencing Applications on the Internet Using an Overlay Multicast Architecture. In *ACM SIGCOMM*, 2001.

[34] Y.-H. Chu, S.G. Rao, S. Seshan, and H.Zhang. A Case for End System Multicast. *IEEE Journal on Selected Areas in Communications*, 20(8), 2002.

[35] D Clark, B. Lehr, S. Bauer, P. Faratin, R. Sami, and J Wroclawski. The Growth of Internet Overlay Networks: Implications for Architecture, Industry Structure and Policy. In *TPRC*, 2005.

[36] F. Cornelli, E. Damiani, S D C. Vimercati, S. Paraboschi, and P. Samarati. Choosing Reputable Servents in a P2P Network. In *ACM World Wide Web Conference (WWW)*, 2002.

[37] C.Rohrs. Query Routing for the Gnutella Network, 2001.

[38] N. Daswani and H. Garcia-Molina. Query-Flood Dos Attacks in Gnutella. In *ACM Computer and Communications Security*, 2002.

[39] C. Dellarocas and P. Resnick. Online Reputation Mechanisms - A Roadmap for Future Research. In *First Interdisciplinary Symposium on Online Reputation Mechanism*, 2003.

[40] John R. Douceur. The Sybil Attack. In *IPTPS*, 2002.

[41] D. Dumitriu, E. Knightly, A. Kuzmanovic, I. Stoica, and W. Zwaenepoel. Denial-of-Service Resilience in Peer-to-Peer File Sharing Systems. In *ACM SIGMETRICS*, 2005.

[42] M. Faloutsos, P. Faloutsos, and C. Faloutsos. On Power-law Relationship of the Internet Topology. In *ACM SIGCOMM*, 1999.

[43] M. Freedman and R. Morris. Tarzan: A Peer-to-Peer Anonymizing Network Layer. In *ACM CCS*, 2002.

[44] X. Fu, B. Graham, X. Dong, R. Bettati, and W. Zhao. Analytical and Empirical Analysis of Countermeasures to Traffic Analysis Attacks. In *International Conference on Parallel Processing (ICPP)*, 2003.

[45] E. Gabber, P. Gibbons, D. Kristol, Y. Matias, and A. Mayer. Consistent, Yet Anonymous,Web Access with LPWA. *Communications of the ACM*, 42(2):42 - 47, 1999.

[46] E. Gabber, P. Gibbons, Y. Matias, and A. Mayer. How to Make Personalized Web Browsing Simple, Secure, and Anonymous. In *Conference on Financial Cryptography*, 1997.

[47] O. D. Gnawali. A Keyword-Set search system for peer-to-peer networks. *Master thesis*, 2002.

[48] Christian Grosch. Framework for Anonymity in IP-Multicast Environment. In *IEEE GLOBECOM*, 2000.

[49] B. Gross and A. Acquisti. Balances of Power on eBay: Peers or Unequals? In *Workshop on Economics of Peer-to-Peer Systems*, 2003.

[50] Y. Guan, X. Fu, X Dong, P. Shenoy, R. Bettati, and W. Zhao. NetCamo: Camouflaging Network Traffic for QoS-Guaranteed Mission Critical Applications. *IEEE Transactions on Systems, Man, and Cybernetics*, 2001.

[51] K. P. Gummadi, R. J. Dunn, S. Saroiu, S. D. Gribble, H. M. Levy, and J. Zahorjan. Measurement, Modeling, and Analysis of a Peer-to-Peer File-Shareing Workload. In *19th ACM Symposium on Operating Systems Principles (SOSP)*, 2003.

[52] M. Gupta, P. Judge, and M. Ammar. A Reputation System for Peer-to-Peer Networks. In *ACM NOSSDAV*, 2003.

[53] D. B. Johnson and D. A. Maltz. Dynamic Source Routing in Ad Hoc Wireless Networking. In *IEEE Mobile Computing (MobiCom)*, 1996.

[54] S.D. Kamvar, M.Schlosser, and H.Garcia-Molina. The EigenTrust Algorithm for Reputation Management in P2P Networks. In *WWW*, 2003.

[55] P. Keyani, B. Larson, and M. Senthil. Peer Pressure: Distributed Recovery from Attacks in Peer-to-Peer Systems. In *IFIP Workshop on Peer-to-Peer Computing*, 2002.

[56] M. Kinateder and S. Pearson. A Privacy-Enhanced Peer-to-Peer Reputation System. In *the 4th International Conference on Electronic Commerce and Web Technologies (EC)*, 2002.

[57] M. Kinateder, R. Terdic, and K. Rothermel. Strong Pseudonymous Communication for Peer-to-Peer Reputation Systems. In *ACM SAC*, 2005.

[58] P. Kruus and J. Macker. Techniques and issues in multicast security. In *IEEE MILCOM*, 1998.

[59] G. Kwon, K.D. Ryu, and Y. Xie. BYPASS: Topology-Aware Lookup Overlay for DHT-based P2P File Locating Services. In *IEEE ICPADS*, 2004.

[60] J. Liang, R. Kumar, and K W. Ross. The KaZaA Overlay: A Measurement Study. In *IEEE INFOCOM*, 2005.

[61] J. Liang, R. Kumar, Y. Xi, and K W. Ross. Pollution in P2P File Sharing Systems. *Elsevier Computer Networks*, 2005.

[62] X. Liu and X. Li. A Survey of Multihoming Technology in Stub Networks: Current Research and Open Issues. *IEEE Network*, pages 32–40, 2007.

[63] X. Liu, Y. Liu, and L. Xiao. Reliable Response Delivery in Peer-to-Peer Systems. In *IEEE MASCOTS*, 2004.

[64] X. Liu, Y. Liu, and L. Xiao. Improving Query Response Delivery Quality in Peer-to-Peer Systems. *IEEE Transactions on Parallel and Distributed Computing (TPDS)*, 17(11):1335–1341, 2006.

[65] X. Liu and L. Xiao. hiREP: Hierarchical Reputation Management for Peer-to-Peer Systems. In *ICPP*, 2006.

[66] X. Liu and L. Xiao. Inbound Traffic Load Balancing in BGP Multi-homed Stub Networks . In *IEEE ICDCS*, 2008.

[67] X. Liu, L. Xiao, A. Kreling, and Y. Liu. Optimizing Overlay Topology by Reducing Cut Vertices. In *ACM NOSSDAV*, 2006.

[68] Y. Liu, X. Liu, C. Wang, and L. Xiao. Defending P2Ps from Overlay Flooding-based DDoS. In *ICPP*, 2007.

[69] Y. Liu, X. Liu, L. Xiao, L. Ni, and X. Zhang. Location-Aware Topology Matching in Unstructured P2P Systems. In *IEEE INFOCOM*, 2004.

[70] Y. Liu, L. Xiao, Ni. L. N., and X. Zhang. Location Awareness in Unstructured Peer-to-Peer Systems. *IEEE Transactions on Parallel and Distributed Computing*, 16(2):163–174, 2005.

[71] Y. Liu, Z. Zhuang, L. Xiao, and L. M. Ni. AOTO: Adaptive Overlay Topology Optimization in Unstructured P2P Systems. In *IEEE GLOBECOM*, 2003.

[72] Y. Liu, Z. Zhuang, L. Xiao, and L. M Ni. A Distributed Approach to Solving Overlay Mismatching Problem. In *IEEE ICDCS*, 2004.

[73] Q. Lv, P. Cao, E. Cohen, K. Li, and S. Shenker. Search and Replication in Unstructured Peer-to-Peer Networks. In *ACM International Conference on Supercomputing*, 2002.

[74] S. Marsh. *Formalising Trust as a Computational Concept.* Ph.d. thesis, University of Stirling, 1994.

[75] S. Marti and H.Garcia-Molina. Limited Reputation Sharing in P2P Systems. In *ACM Conference on Electronic Commerce (EC)*, 2004.

[76] R. Morselli, B. Bhattacharjee, J. Katz, and P. Keleher. Trust-Preserving Set Operations. In *IEEE INFOCOM*, 2004.

[77] M. Moyer, J. Rao, and P. Rohatgi. A Survey of Security Issues in Multicast Communications. *IEEE Network*, 1999.

[78] A. Nakao, L. Peterson, and A. Bavier. A Routing Underlay for Overlay Networks. In *ACM SIGCOMM*, 2003.

[79] V. N. Padmanabhan, H. J. Wang, and P. A. Chou. Resilient Peer-to-Peer Streaming. In *IEEE ICNP*, 2003.

[80] V. Park and M. Corson. Temporally-ordered Routing Algorithm(TORA) Version 1: Functional Specification. *IETF Specification*, 2001.

[81] C.E. Perkins and E. M. Royer. Ad-hoc On-demand Distance Vector Routing. In *IEEE MILCOM*, 1997.

[82] J. Pouwelse, P. Garbacki, D. Epema, and H. Sips. The Bittorrent P2P File-sharing System: Measurements and Analysis. In *IPTPS*, 2005.

[83] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker. A scalable content-addressable network. In *ACM SIGCOMM*, 2001.

[84] S. Ratnasamy, M. Handley, R. Karp, and S. Shenker. Topologically-Aware Overlay Construction and Server Selection. In *IEEE INFOCOM*, 2002.

[85] M. K. Reiter and A. D. Rubin. Crowds: Anonymity for web transactions. *ACM Transactions on Information and System Security, pp. 66-92, November*, 1998.

[86] M. Ripeanu. Peer-to-Peer Architecture Case Study: Gnutella Network. In *International Conference on Peer-to-Peer Computing (P2P)*, 2001. 92

[87] Ritter. Why Gnutella can't scale. No, really. *http://www.tch.org/gnutella.html*, 2001.

[88] A. Rowstron and P. Druschel. Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems. In *International Conference on Distributed Systems Platforms*, 2001.

[89] S. Saroiu, K. Gummadi, R. Dunn, S. Gribble, and H. Levy. An Analysis of Internet Content Delivery Systems. In *OSDI*, 2002.

[90] S. Saroiu, P. Gummadi, and S. Gribble. A Measurement Study of Peer-to-Peer File Sharing Systems. In *Mutimedia Computing and Netowrking (MMCN)*, 2002.

[91] S. Savage, T. Anderson, A. Aggarwal, D. Becker, N. Cardwell, A. Collins, E. Hoffman, J. Snell, A. Vahdat, G. Voelker, and J. Zahorjan. Detour: A Case for Informed Internet Routing and Transport. *IEEE Micro*, 19(1):50–59, 1999.

[92] S. Savage, A. Collins, E. Hoffman, J. Snell, and T. Anderson. The End-to-End Effects of Internet Path Selection. In *ACM SIGCOMM*, 1999.

[93] S. Sen and J. Wang. Analyzing Peer-to-peer Traffic Across Large Networks. In *ACM SIGCOMM Internet Measurement Workshop*, 2002.

[94] R. Sherwood, B. Bhattacharjee, and A. Srinivasan. P5: A Protocol for Anonymous Communication. In *IEEE Symposium on Security and Privacy*, 2002.

[95] S. Shi and J. S. Turner. Routing in Overlay Multicast Networks. In *IEEE INFOCOM*, 2002.

[96] A. Singh and L. Liu. TrustMe: Anonymous Management of Trust Relationships in Decentralized P2P Systems. In *P2P*, 2003.

[97] K. Sripanidkulchai. The Popularity of Gnutella Queries and Its Implications on Scalability, 2001.

[98] K. Sripanidkulchai, B. Maggs, and H. Zhang. Efficient Content Location Using Interest-Based Locality in Peer-to-Peer Systems. In *IEEE INFOCOM*, 2003.

[99] M. Srivatsa, L. Xiong, and L. Liu. TrustGuard: Countering Vulnerabilities in Reputation Management For Decentralized Overlay Networks. In *WWW*, 2005.

[100] I. Stoica, S. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan. Chord: A scalable peer-to-peer lookup service for Internet applications. In *ACM SIGCOMM*, 2001.

[101] D. Stutzbach and R. Rejaie. Understanding Churn in Peer-to-Peer Networks. In *ACM SIGCOMM Internet Measurement Conference*, 2006.

[102] D. Stutzbach, R. Rejaie, and S. Sen. Characterizing Unstructured Overlay Topologies in Modern P2P File-Sharing Systems. In *ACM SIGCOMM Internet Measurement Conference*, 2005.

[103] P. F. Syverson, D. M. Goldschlag, and M. G. Reed. Anonymous Connections and Onion Routing. *IEEE Symposium on Security and Privacy*, pages 44–53, 1997.

[104] H. Tangmunarunkit, R. Govindan, S. Jamin, S. Shenker, and W. Willinger. Network Topology Generators: Degree-Based vs. Structural. In *ACM SIGCOMM*, 2002.

[105] M. Waldvogel and R. Rinaldi. Efficient Topology-Aware Overlay Network. In *ACM HotNets*, 2002.

[106] S. Wang, D. Xuan, and W. Zhao. On Resilience of Structured Peer-to-Peer Systems. In *IEEE GLOBECOM*, 2003.

[107] Y. Wang and J. Vassileva. Trust and Reputation Model in Peer-to-Peer Networks . In *Third International Conference on Peer-to-Peer Computing (P2P)*, 2003.

[108] N. Weiler. Secure Anonymous Group Infrastructure for Common and Future Internet Application. In *Annual Computer Security Applications Conference*, 2001.

[109] L. Xiao, X Liu, W. Gu, D. Xuan, and Y. Liu. A Design of Overlay Anonymous Multicast Protocol. In *IPDPS*, 2006.

[110] L Xiao, Y. Liu, W. Gu, D. Xuan, and X. Liu. Mutual Anonymous Overlay Multicast. *Elsevier Journal of Parallel and Distributed Computing (JPDC)*, 86(9):1205–1216, 2006.

[111] L. Xiao, A. Patil, Y. Liu, L. M. Ni, and A.-H. Esfahanian. Prioritized Overlay Multicast in Ad-hoc Environments. *IEEE Computer Magazine*, pages 67–74, 2004.

[112] L. Xiao, Z. Xu, and X. Zhang. Low-Cost and Reliable Mutual Anonymity Protocols in Peer-to-Peer Networks. *IEEE Transactions on Parallel and Distributed Systems*, 14(9), 2003.

[113] L. Xiong and L. Liu. A Reputation-based Trust Model for Peer-to-Peer eCommerce Communities. In *IEEE International Conference on Electronic Commerce*, 2003.

[114] L. Xiong and L. Liu. PeerTrust: Supporting Reputation-Based Trust for Peer-to-Peer Communities. *IEEE Transactions on Knowledge and Data Engineering*, 16(7), 2004.

[115] Z. Xu, C. Tang, and Z. Zhang. Building Topology-aware Overlays Using Global Soft-state. In *IEEE ICDCS*, 2003.

[116] B. Yu and M. P. Singh. A Social Mechanism of Reputation Management in Electronic Communities. In *the 4th International Workshop on Cooperative Information Agents*, 2000.

[117] S. Yuen and B. Li. Strategy Proof Mechanisms for Dynamic Multicast Tree Formation in Overlay Networks. In *IEEE INFOCOM*, 2005.

[118] X. Zhang, J. Liu, Q. Zhang, and W. Zhu. gMeasure: A Scalable Group-based Network Performance Measurement Service for Peer-to-Peer Applications. In *IEEE GLOBECOM*, 2002.

[119] B.Y. Zhao, L. Huang, J. Stribling, S.C. Rhea, A.D. Joseph, and J.D. Kubiatowicz. Tapestry: An infrastructure for fault-resilient wide-area location and routing. *IEEE Journal on Selected Areas in Communications*, 2001.

[120] S Q. Zhuang, D. Geels, I. Stoica, and R. H. Katz. On Failure Detection Algorithms in Overlay Networks. In *IEEE INFOCOM*, 2005.