Advances in Multimedia

Toward the Next-Generation Peer-to-Peer Services

Guest Editors: Yi Cui, Ben Zhao, and Shigang Chen

Toward the Next-Generation Peer-to-Peer Services

Advances in Multimedia

Toward the Next-Generation Peer-to-Peer Services

Guest Editors: Yi Cui, Ben Zhao, and Shigang Chen

Contents

Hindawi Publishing Corporation
Advances in Multimedia
Volume 2007, Article ID 52679, 1 page
doi:10.1155/2007/52679

Editorial
Toward the Next-Generation Peer-to-Peer Services

Yi Cui,[1] Ben Y. Zhao,[2] and S. Chen[3]

[1] *Department of Electrical Engineering and Computer Science, Vanderbilt University, VU Station B 351824, Nashville, TN 37235-1826, USA*
[2] *Department of Computer Science, University of California-Santa Barbara, 1123 Harold Frank Hall, Santa Barbara, CA 93106-9560, USA*
[3] *Department of Computer and Information Science and Engineering, University of Florida, CSE Building, Room 460, Gainesville, FL 32607, USA*

Received 11 December 2007; Accepted 11 December 2007

Recent years have witnessed the blossom of the P2P application model on the Internet. Popular examples include P2P file sharing networks, streaming multimedia and content distribution networks, and P2P metaservices such as peer-to-peer lookup services, reputation services, and network underlays. In this special issue, we aim to present both a timely retrospect of recent achievements in P2P services and an outlook of new challenges ahead. The invited papers address a variety of topics, including the following.

The first paper, which is "A measurement study of the structured overlay network in P2P file-sharing systems," conducts a measurement study on the E-Mule overlay network. It presents a novel crawler program design that provides a perspective of overlay networks from a single user viewpoint, which helps identify new vulnerabilities such as DDOS attacks.

The second paper, "Analysis and implementation of gossip-based P2P streaming with distributed incentive mechanisms for peer cooperation," studies the randomized gossip-based scheme for P2P streaming and proves that an incentive mechanism can be created for a live streaming P2P protocol while preserving the asymptotic properties of the gossip-based scheme. It further proposes a functional architecture and protocol format for this new streaming paradigm.

The third paper, "Enhanced P2P services providing multimedia content," aims to remove the dependence of current P2P query systems on unique identifiers or keywords. It proposes an original image and video sharing system where users can interactively search for interesting resources using content-based image and video retrieval techniques.

The fourth paper, "A hybrid query scheme to speed up queries in unstructured peer-to-peer networks," studies the problem of locating resources in unstructured P2P networks. By identifying problems in existing approaches such as flooding and random walk, the paper proposes a new hybrid query scheme that groups peers into clusters based on similar interests, and improves communication efficiency by mixing intercluster queries and intracluster queries.

The fifth paper, "Globally decoupled reputations for large distributed networks," points out the vulnerability of existing P2P reputation systems to unfair rating attacks, and recommends that reputation systems decouple each peer's reputation as service provider from that as service recommender, making reputations more resistant to tampering. It further proposes a scalable approach to compute decoupled service and feedback reputations, and demonstrates the effectiveness of the new model against previous nondecoupled reputation approaches.

Yi Cui
Ben Y. Zhao
S. Chen

Hindawi Publishing Corporation
Advances in Multimedia
Volume 2007, Article ID 27958, 9 pages
doi:10.1155/2007/27958

Research Article

A Measurement Study of the Structured Overlay Network in P2P File-Sharing Systems

Mo Zhou, Yafei Dai, and Xiaoming Li

CNDS Lab, School of Electrical Engineering and Computer Science, Peking University, Beijing 100871, China

Received 30 January 2007; Revised 20 May 2007; Accepted 4 July 2007

Recommended by Yi Cui

The architecture of P2P file-sharing applications has been developing to meet the needs of large scale demands. The structured overlay network, also known as DHT, has been used in these applications to improve the scalability, and robustness of the system, and to make it free from single-point failure. We believe that the measurement study of the overlay network used in the real file-sharing P2P systems can provide guidance for the designing of such systems, and improve the performance of the system. In this paper, we perform the measurement in two different aspects. First, a modified client is designed to provide view to the overlay network from a single-user vision. Second, the instances of crawler programs deployed in many nodes managed to crawl the user information of the overlay network as much as possible. We also find a vulnerability in the overlay network, combined with the character of the DNS service, a more serious DDoS attack can be launched.

1. INTRODUCTION

In a P2P system, each peer participated in the system contributed some resource, such as computation ability, memory, and storage to complete some huge tasks together, and each peer can get the benefits of the whole systems by providing service for each other. Some typical P2P applications include massive content-delivering [1], file-sharing [2–5], P2P streaming [6], and cooperative computation [7]. This paper mainly concerns about the file-sharing P2P systems.

There are several kinds of architecture in the P2P file-sharing systems. The central index server architecture, the unstructured overlay network, and the structured overlay network, also known as DHT (distributed hash table) [8–11]. There are also some systems constructed by more than one architecture. Recently, DHT has shown much superiority to the other architecture, because of its good scalability and low search cost.

Measurement study to the P2P systems will give helpful insights to the improvement of such systems. In the case of file-sharing P2P systems, we usually concern about these aspects of the system.

(i) Resources or Files. How many files are shared in this system? What files are shared? Reference [12] has shown some study in this aspect in central index server architecture and unstructured overlay network.

(ii) Users. How many users are connected to this system? How about their experience in the system? Reference [13] has shown the user experience in structured and unstructured overlay networks.

(iii) Network. How much network bandwidth cost is needed to maintain the function of the system? What kind of message consumes the most bandwidth in the system? If we find some rules in the messages transferred in the P2P network, we can redesign the P2P protocol to reduce the network bandwidth cost.

Many measurement studies have been carried out to the P2P file sharing system itself, while our motive is to learn how the structured overlay network makes the P2P file-sharing systems more usable. The resource meta-information distribution and locating is easier when the structured overlay network is available in a P2P file-sharing system, hence we focus on how this happens by measuring the DHT. We do this by finding a proper application and gathering data from it. Overnet [14] is the first massively used DHT in the practical file-sharing application, eDonkey [15]; but Overnet is not open sourced as research upon it needs some reverse engineering. eMule [4] is an open-source alternative for the eDonkey program, it support the ed2K protocol used in eDonkey. eMule also begins to support structured overlay network called kad after version 0.42. Both the overnet used in eDonkey and the kad network used in eMule is

based on kademlia [11], but the two structured overlay networks are not compatible. We can modify the source code of eMule, and let it record all the messages transfered from and to the kad network, and perform various analysis on the data.

2. RELATED WORKS

Reference [12] has performed some study on the central index server P2P systems and the unstructured overlay network P2P systems (napster and gnutella). It studies the character of the hosts in both kinds of the systems, which includes the connecting and disconnecting time profiles of the participating peers and the resource they shared.

References [16, 17] proposed some approach to decrease the search cost in the unstructured overlay network. Reference [16] used a query algorithm based on multiple random walks to provide enough searching ability while reducing the searching cost. Reference [17] used the fact that peers that have the same interests in some category of resources may be more possible to solve each other's demands, and reduced the searching cost by grouping the peers with the common interests.

Reference [18] is the first study to the structured overlay network used in a practical file-sharing P2P system. It used some reverse engineering to perform the measurement because the eDonkey is not an open source program. Reference [18] claims that no open-source system can meet the requirements of the measurement. After eMule comes out, the solid source code of this P2P application can be used to make more sophisticated study. Reference [18] has measured some date in the overnet to estimate the availability of the hosts in the system. From [18], we can make an evaluation to the scale of the overnet, and we can compare it with the scale of the kad network in eMule.

References [19, 20] focus on crawling the resource information of the traditional ed2K network (network with central index servers) in the eDonkey/eMule. eMule uses both ed2k network and kademlia to distribute and exchange the information of the sharing resources. We are focusing on the part of the kademlia network. Reference [21] analyzed some parameters of the kad network in the eMule, and tried to adjust parameters of eMule client to get better performance.

Reference [13] makes some measurement study toward both structured and unstructured overlay network. It mainly concerns about the user experience to the two kind of systems, such as the bandwidth cost, searching performance, and load balance. Reference [13] also discovers that the results did not change too much if the data was measured from different geographical position in the world. So although this paper emphasizes particularly on other aspect of the structured overlay network, we can use this conclusion to reduce the unnecessary work with single user vision in different network position.

Reference [22] showed some basic measurement results of the structured overlay network in eMule. This paper showed more analysis toward those results, and some changes have been made to the programs used in [22], for example, the crawlers are redesigned to get better performance with less resource cost. Some new information is also updated after the [22] was written.

3. METHODOLOGY

The measurement of our study has two aspects. A modified eMule client is used for single user measurement, which has the same function in all the other aspects compared to the normal eMule client, but modified clients will log all the messages of both directions between the kad network and the host, and when the routing table changed, the modified clients will also record it. The other aspect of our measurement is a crawler program deployed at multiple nodes in the PlanetLab [23]. The crawler program is modified from amule [24], a linux port for eMule, this is because we need to run many instances of the crawler on the machines of the PlanetLab, which are running linux. We removed most of the code in amule, only remained the code handling kademlia protocol, and the crawler can use it to interact with other clients in the network.

3.1. eMule and its kad network

eMule is an open-source file-sharing P2P application, and its architecture is hybrid, both central index server and DHT are used in eMule. Peers can search files in servers, and when the servers encounter a failure, the peers can still search in DHT, thus the system become more reliable. The kad network used in eMule is based on kademlia [11], which use the XOR result of two peers' IDs to compute the logical distance of two peers. eMule uses kad network to store and retrieve both key word information for metadata search and file ID information for precise file locating.

3.2. The recorded data of the single client measurement

We focus on the structured overlay network. To perform the analysis, we need the following data: all the message transferred from and to the single client, the change in the routing table, and the login and logout record of the client. The data files we generated for analysis include six types of records.

(i) Startup record. This record indicates the eMule client started its connection to the kad network. It includes record time, host ip address, host tcp port, host udp port, and host ID used in kad network.

(ii) Ending record. This record indicate the client closed its connection to the kad network. This record only includes the record time.

(iii) Incoming packet record. This record contains the information of a kad network packet received by the host. It includes record time, source ip, source udp port, packet length, and the content of the packet.

(vi) Outgoing packet record. It contains the information of a kad network packet sent by the host. Its structure is similar to the incoming packet record, only the destination ip and destination udp port replace the source ip and source udp port.

Table 1: Number of records.

Record type	Quantities of the record
Startup records	26
Ending records	26
Adding contact records	46279
Removing contact records	16244
Incoming packet records	443203
Outgoing packet records	608515

(v) Adding contact record. This record contains the contact information of a new peer added to the routing table, it includes record time, new peer's ip, tcp port, udp port and ID. We noticed that not all the contact information received from the kad network will be added to the routing table, thus we only record the contact information actually added.

(iv) Removing contact record. When a peer's information is removed from the routing table due to time out or some other reasons, its information will be recorded. This record has the same structure as the adding contact record.

The data for our measurements is collected continuously from a typical eMule user for more than two weeks. He shared some files in his computer, and also used eMule for file searching and downloading. As in [13], the result will not change too much when we measure the data from different physical locations. Thus, we believe that we do not need to collect data from more users. Table 1 shows some statistical information of the data in general. It includes the number of the six types of record we mentioned above.

3.3. The measurement from the crawler program

The crawler program is focused on the user information in the kademlia network of the eMule application. The purpose of the program is to exploit the user information of the eMule kademlia network as much as possible. In a structured overlay network, this can be done by dividing the ID spaces into many parts, and every instance of the program exploits only one part of the spaces. The crawler program acts like normal kademlia node in the view of other eMule clients, but the difference between them is that the crawler has no interests in searching specific content in the kademlia network, it sends node searching requests packets to the nodes in its routing table continuously, hopes that can help it retrieving more user information. The rate of the node searching requests packets sent by the crawler program is controlled in a set level, and the crawler program will not send two node searching requests packets to the same eMule client in a short time. The crawler program will also handle the searching or publishing requests for specific keyword or file from other eMule client correctly, all of these will minimize the impact to the kademlia network.

In the kademlia network, the length of the ID of a node is 160 bits. This provides an explicit division method for dividing the ID space; we can group all the kademlia nodes whose ID has the same highest n bits; we can adjust the granularity of the division by selecting a good value for n. In our measurement, we set the n as 8, this means all the nodes in the same part of the ID spaces have at least 8 same highest bits, and the whole ID space is divided into 256 parts. The reason we divide the space into 256 part is related to the searching tolerance in eMule. We will mention the searching tolerance in the later of this paper, here what we need to know is that any information of a specified file ID or keyword will only appear in nodes whose IDs are in the same part of the ID space. Every instance of the crawler program sends node searching requests packets only to the nodes who are in the same part of the ID space with it. The information retrieved from other eMule clients by the node searching requests will be used recursively for retrieving more user information. Because every instance of the crawler program knows contact information of other instances, they will send the information of the nodes who are not in the same part of the ID space with itself to the corresponding instance of the crawler program. The crawler program who received the user information from other instance of the crawler will treat the information the same as received from normal eMule clients, and use it in a recursive way.

Every node in the PlanetLab has only some restricted resource for every program instance running on it. So the previous design of our crawler system did not completely achieve our goal. This is because with the increasing of the obtained user information, the memory cost of every node running the instance also increases. In order to make the crawlers obtain more user information, and control the resource cost in a certain level, we adjusted the design of the crawler system. Every instance of the crawlers running in the nodes of the PlanetLab will not do anything without receiving orders, and the orders they will receive is also designed rather simple. They will receive only two types of orders. The first is orders to set a home address, the crawlers will transfer all the packets they received to the home address. The home address is the computers we can control, and the usage of the resource is not restricted. The second type of orders is to perform a set of scanning operations on a small set of contacts. After receiving this type of order, the crawlers will begin to crawl user information from the received contacts. If any result is retrieved, the crawlers will send them back to the home address. After scanning all the contacts, the crawlers will stop any action until new orders are received. All the analysis of the results from the crawlers are performed in the home computers. We can see this redesign of our crawler system can make all the crawler run on the PlanetLab nodes with only a limited resource cost. The home computers combine all the results they received, and send another set of contacts to the crawlers and they will continue to scan them. The Figure 1 shows the structure of our distributed crawler system.

4. MEASUREMENT RESULTS FROM THE SINGLE CLIENT

We can learn about the status of the kad network from several aspects. First, we can learn the routing table size changing

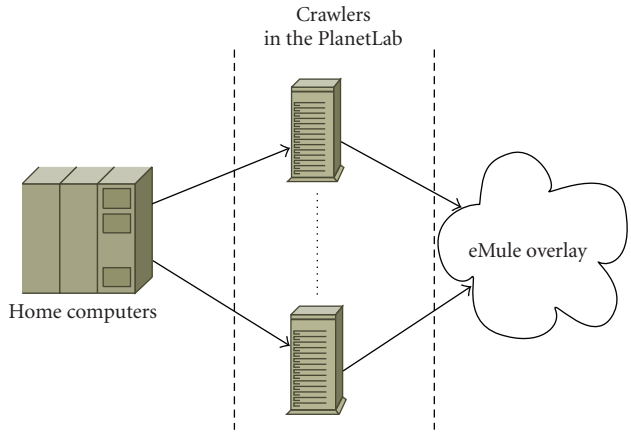


Figure 1: Routing table size.

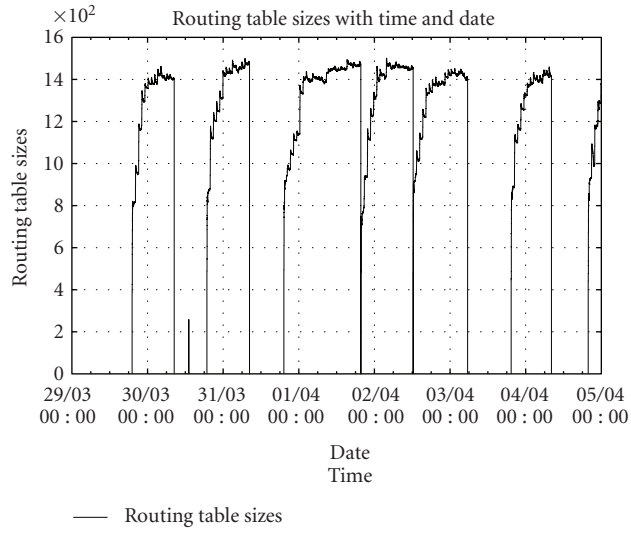


Figure 2: Routing table size.

with the time and the speed of the changing of the contacts in the routing table. When the peer received a request searching for a special node, we can check the message and learn that whether a searching request or publishing request will come at the next step. Among the searching node request, the distribution of the distance between the host and the target node is another aspect we would like to observe. The same observation can also be applied to the resource searching request and publishing request.

4.1. Routing table size

The data contains every record of adding or removing contacts, so we can learn how many contacts are in the routing table at a specific moment. We can also learn the trends of the changing of the routing table. Figure 2 shows the routing table size change with the time. Axis x is the time, and Axis y is the routing table size.

From Figure 2, we can see that in every online session of a user, the routing table size will come to a rather stable state

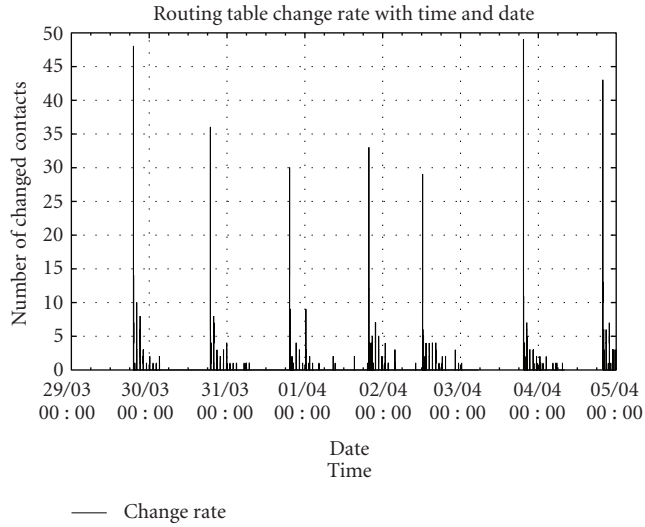


Figure 3: The changing rate of routing table.

at about 1400 contacts or so. The time a peer need to come to this stage is about six hours. Although users will connect to and disconnect from the network according their using style, most users' online session time is long enough to get their routing table into this stable state, and this forms the basis that the kademlia network needs to function normally.

4.2. Contacts change rate

Another question we would like to ask about the routing table is, at a specific moment, how many contacts are new in the routing table that they did not appear in the routing table five minutes ago? Or we can say how fast the contacts change in the routing table? Figure 3 shows the answer to this question. Every two adjacent sampling point in the figure has a distance of one minute.

Figure 3 tells us that the changing rate of the routing table is not very high. That means a contact in a peer's routing table is very likely to appear in the peer's routing table one minute later. If we observe this figure and Figure 2 together, we can also find that when a peer is just connecting to the kad network the changing rate is higher than stable state. This is because new contacts information are more likely to be filled into the routing table when it is rather empty, but when in the stable state, new contacts information cannot be written into the routing table unless some of the old contacts are detected offline and deleted from the routing table. This also tells us that when a client comes to a stable state, the possibility that many of the contacts in its routing table suddenly go offline is rather small, and thus the kademlia network can keep available in most of the time.

4.3. Nodes request distribution

The main purpose of the kademlia network is searching or publishing resource information. So when a peer received a message searching for a particular node, there are three cases

Table 2: The purpose of node searching.

Purpose	Quantities of node searching
Just searching for nodes	1504
Searching files or keywords	10758
Publishing files or keywords	61292

about the real intention of the peer at the other side of the network:

(i) The other peer is trying to search some peer whose ID is near a special ID, and use the searched contact information to fill its routing table. This happens when a peer finds its routing table does not contain enough contact information.

(ii) The other peer is trying to search some special file or keyword. The node searching process is used to find out the peers who are possible to have the information it wants.

(iii) The other peer is trying to publish information of a file or keyword.

Table 2 shows the distribution of the three cases. It illustrates that when a node searching request appears, most of the case the request is preparing for a publishing request. It is nearly about five times as many as the requests for file and keyword searching, this is because a peer needs to periodically publish all the information of his sharing files, but not all the resource it shared will be searched. Finally, we learn that the kademlia network can be rather resilient only through the resource publishing and searching, and a peer does not need many pure node searching requests to fill the routing table.

4.4. Nodes request distance distribution

To find out the character of the routing process, we can check distance between the host and the target. We use the number of same bits from the higher order to measure the distance. If the whole ID space is divided into 2^n subspaces, two IDs with the same n high-order bits will be in one of the same subspaces, but they will not be in the same subspaces if the whole ID space is divided into 2^{n+1} subspaces. So when this number is zero the distance between the two peers is very far in the logical space, and 128 means the two peers have the same ID, that is most likely they are actually the same peer. In Figure 4, we observed that most node searching requests ask this peer to search nodes which have about 10 high-order bits same as it. After the distance number comes greater than 20, the number of searching requests nearly drops to zero. We also notice that there are a few requests ask the peer to search for itself, that is, with a distance number of 128.

The node search process in the kad network usually has several attempts. The most searching requests indicate the situation of the first node search attempt. In such case, a peer begins this search rely on its local routing table only, and in most case it will find a contact with about 10 same high-order bits with any target. In the implementation of the kademlia

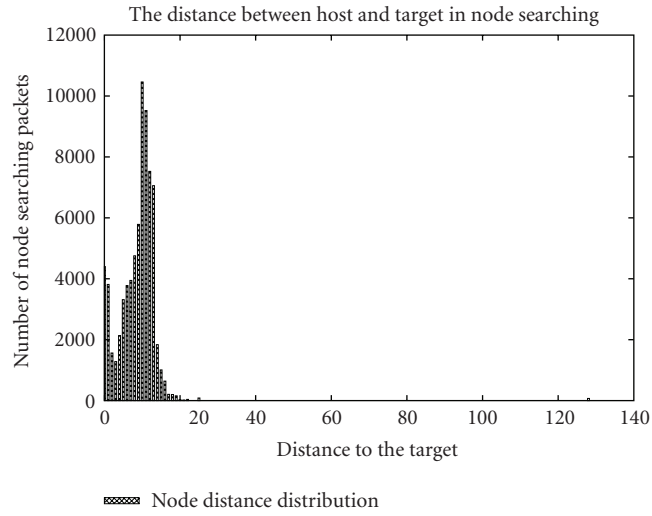


Figure 4: The distance distribution of nodes requests.

protocol, some parameters can be adjusted to balance the performance and the price. One of the parameters is the split depth, this means when a bucket depth is not enough, new contact will cause the splitting of the bucket in spite of its location, this implementation is some different from original kademlia [11]. Another is the size of a bucket. The above two parameters in the official eMule client implementation is 5 and 20. This means that in a normal eMule client has about 2^5 buckets full of contact information after it connect to the kad network for some time (several hours). So about 640 contacts are distributed in the ID space equably, when a peer begin a new search task, its very likely that it can find a contact with about 10 high bits same with the target. The requests number quickly drops to zero after the distance number greater than 20, this indicates most search attempts end here. This means when finding a node with a special ID, in the most case, we can find an ID with about 20 high-order bits same with it after searching the whole ID space. This reveals the density of the peers distributed in the whole kad network. We believe that the few requests with a distance number of 128 can be concluded as some rogue requests by other modified clients, and should be ignored.

4.5. Search request and publish request distance distribution

The method used here is quite the same as the measurement for the nodes request distance distribution; but the searching and publishing requests can only be sent to the peers whose ID are close enough to the target. This is called search tolerance in eMule, and normal eMule client will reject the requests of publishing or searching for some resource if the client's ID and the computed ID of the resource do not have 8 same high bits. So in Figure 5, we can see that it is obviously different from Figure 4 that when the number is smaller than 8, the number of requests can almost be ignored; and we have sufficient reasons to believe that the peers who send the requests are not normal eMule clients. The number of requests

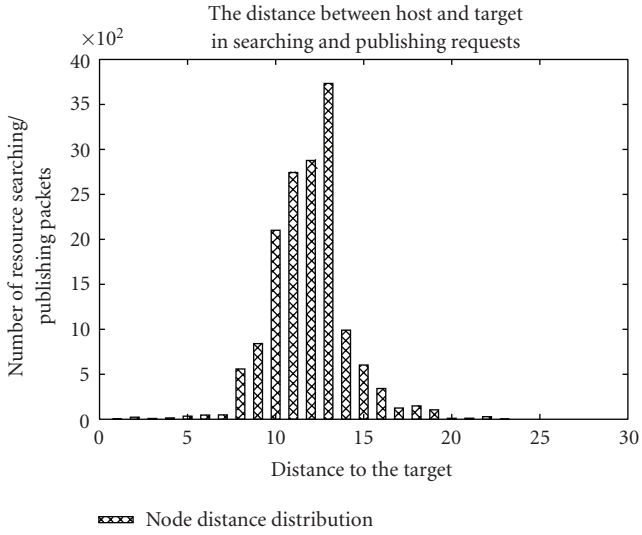


Figure 5: The distance distribution of searching and publishing requests.

drops zero after the number is larger than 25, this means it is very unlikely that a random-generated ID can have too many high-order bits same as some file or keyword ID.

4.6. Summary of the data from the single client

From the above data from single client, we can see that the kademlia network functions well in most of the time. Because the peers need to periodically publish the information of the resource they shared, the kademlia network kept well connected by the daily publishing and searching messages. We observed that a few clients are beyond normal behavior, but their impact to the whole kademlia network is rather small.

5. USER INFORMATION FROM THE CRAWLER

The instances of the crawler periodically return the user information they retrieved. The results they returned are separated by the part of the ID space. We observed the returned user information, and found that pollution in the user information continuously exists. Most of the false user contact information has a udp port number 53, which is the standard service port of DNS. Because all of the contact information from the crawlers are gathered from other peers' routing table, this means some of the peers in the structured overlay network have already got confused between normal eMule clients and DNS servers before our crawlers got those information from them. The crawlers can simply use a filter to remove all the incorrect results, but the confusion still happened in the routing tables of the other eMule clients without such kind of filter. We believe that this is a very important phenomenon and after studying the related RFC documents of the DNS [25, 26], we also believe that we have found the reason.

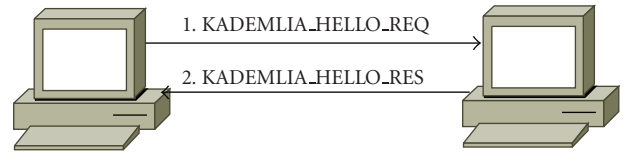


Figure 6: Correct heart-beat checking between two normal eMule clients.

5.1. Heart-beat checking mechanism in eMule

In a normal eMule client, after it received new contacts information and added them to its routing table, the new contacts will not be treated as alive until they are proved to be alive. This means the new contacts will only be used by the client (or peer) itself, such as searching and publishing resource. When other peers request for the information of users near to some ID, a normal eMule client will only tell them the information of living contacts. The way it proves whether other peers are alive is sending heart-beat packets to them, and recognizing the heart-beat response. In eMule kademlia protocol, every packet is composed by one byte of protocol code, one byte of operation code(opcode), and the rest are the content of the packet. Every kademlia protocol packet has a protocol code of OP_KADEMLIAHEADER(0xE4), and the heart-beat request and heart-beat response packet have opcodes of KADEMLIA_HELLO_REQ(0x10) and KADEMLIA_HELLO_RES(0x18), separately. This shows us that an eMule client can recognize a packet type by checking the first two bytes of the packet. Figure 6 shows how two normal eMule clients finish a heart-beat checking.

5.2. Confusion between eMule clients and DNS servers

We found many incorrect user contact information in the results from our crawler, this means that many eMule clients have the wrong information in their routing table, and they incorrectly think the information is from valid and alive eMule clients. Now, we can examine Figure 7 and see what happened when they use heart-beat packets to check the information. First, a heart-beat request packet is sent out using the ip and port information, but as we see, the target is a DNS server, not an eMule client, then the DNS server will try to explain the packet in the DNS protocol, and of course this will not succeed. But the way a DNS server treating an invalid DNS packet is not simply discarding it, it sends some packet back. The DNS server treats any packet sent to it as a DNS protocol packet, which has a message header, and the first two bytes of the header formed a query ID, this identifier is copied to the corresponding reply. The DNS server cannot interpret the incoming packet as a DNS packet, it then sends a reply whose first two bytes are the same as the incoming packet. Now that the eMule client receives the packet, it find that it is a valid eMule heart-beat request packet because the first two bytes remain unchanged, and the normal behavior of a eMule client after receiving a valid eMule heart-beat request packet is sending out a standard eMule heart-beating response packet. The DNS server will then try

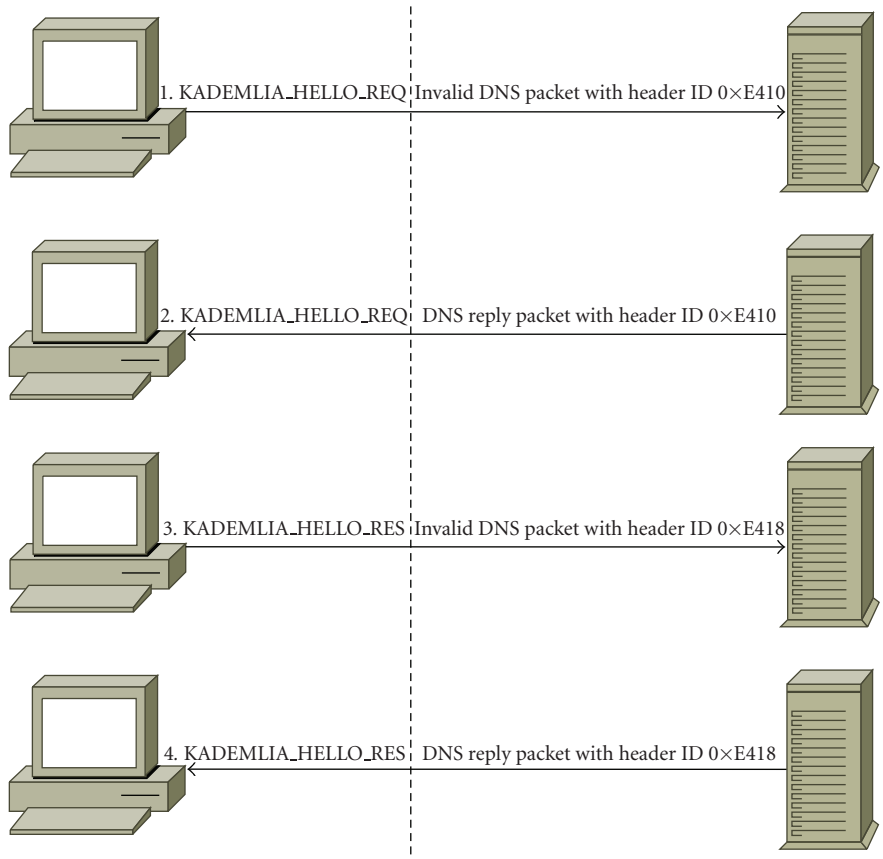


Figure 7: The eMule client is misguided when getting echo from dns servers.

to treat the packet as a DNS protocol packet only to finds that it fails again. In Figure 7, after the eMule client receives the second packet from the DNS server, that is, the message number four in the figure, the confusion begins. The eMule client will treat message number four in Figure 7 as a response for message number one, and it will believe that there is a valid and alive eMule client in the other side of the network.

Reference [27] has found that P2P systems can be used to launch a DDos attack, and it found two kinds of DDos threats to the structured overlay network in the Overnet. One of the ways is to publish false resource information to the peers, this is called index poisoning [28]. Index poisoning can be used to start a DDos attack by sending information about many popular files and telling other peers that the victim has the files, this will cause many downloading requests for the popular files to the victim, thus caused a DDos attack. The other way is directly polluting the routing tables of the peers, by sending lots of false IDs' information connected with the target host, many peers will try to contact with the target host. The contacting messages themselves can be a hazard to the target host if too many peers want to contact with it. Although the studies of [27, 28] are based on the Overnet, we can believe that it also works on the kademlia network in eMule; both of the attacks will stop after some time if the attacker does not fill false information to the kademlia network any more. But we can see the attack will be more serious if the target hosts provide services like DNS or other services which listen on udp ports and echo unrecognized packets, because in this situation the wrong information will be spread more wildly when more and more P2P clients believe that they are not wrong. The spreading of the wrong information will not stop even after the original attacker stopped polluting the nodes' routing tables. This is the reason why we continuously found DNS server contact information in the results returned by the crawlers. Other P2P systems can also be used to attack the above hosts if their heart-beat checking process is not carefully designed.

At the time of the above experiments, the most of the eMule clients in the internet have a version of 0.47a. In September 2006, eMule 0.47c was released, some of the code handling the kademlia network messages changed. Now it has a mechanism of recording all the requests it sent out in the last three minutes, the requests include the heartbeat requests, search requests, and other type of requests. If any type of "response" messages are received from the kademlia network, the new clients will check if it had corresponding request messages sent out, if not, the "response" is very suspicious, and the clients should discard it. We can see in this version that the clients have some resistance to the direct routing

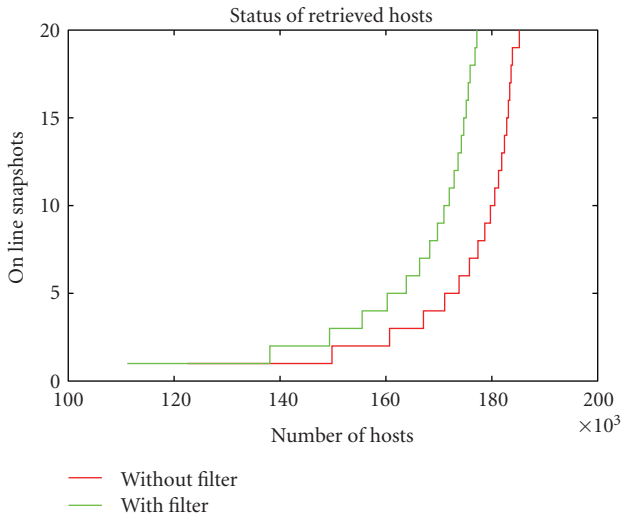


Figure 8: The results from crawlers with and without the filter.

table pollution made by malicious peers who continuously send "search node response" messages to other peers; but the threats are not completely solved. The index poisoning attack can still happen on the new clients, because it used "publish resource request" messages to perform the attack, not "response" message type. The routing table pollution can still be carried out by a passive way, this means that if any peers send search node request messages first, and malicious peers can return false contacts information. If the false contacts information look more believable to the normal peers, for example, the normal peers will believe that the false contacts as real contacts after they attempted a successful heart-beat checking. The DDos attack to the DNS servers we mentioned above will cause this type of confusion. This type of threat is still there, not solved.

5.3. *Improved validity checking of the retrieved information*

We improved the AI of the crawlers, and they will filter the wrong information. This filter includes checking the port of the contact information, contacts with port number 53 will be discarded. We have also filtered other suspicious ports. To have more confidence about the validity of the nodes, we used some test program to them. The test program will send a random-generated file ID to the node, and then search the same ID, a normal eMule client will return the file ID. Figure 8 shows the peers online distribution with and without the filter. Each time we finished crawling user information from a set of contacts, we get a snapshot of the status of peers, and we will use the snapshot to prepare for the next crawling. We selected 20 continuous snapshots for both crawlers with the filter and without the filter. We check the times that contacts appears in the snapshots. We can find crawlers using the filter will have a number of living peers about 5% less than that of crawlers without the filter in corresponding snapshots number. The false contact information is removed.

6. CONCLUSIONS

This paper makes some measurement study of the eMule overlay network from several aspects. We find that the routing table of a peer will go into a stationary state after a period of time, and the size of the routing table in stationary state will not fluctuate too much. We also find that the number of publishing is in a dominative position, so if we want to make some improvement to decrease the network cost, we can try to decrease the number of the publishing messages a peer issued. The structured overlay network keeps well connected with just the searching and publishing of resource. The distance between the target and host in node searching is rather further than that in file searching and publishing, this is mainly because the search tolerance in eMule makes peers do not like to search files or publish information at peers too far away from them, but the node searching is not restricted by the search tolerance. We retrieved many users' information and find a vulnerability in the overlay network, which is more dangerous than the normal routing table pollution. Although the eMule clients of the new version increased some resistance to the attacks, but the threats to the overlay network have not disappeared completely.

ACKNOWLEDGMENT

This work is supported by National Grand Fundamental Research 973 program of China under Grant number 2004CB318204.

REFERENCES

[1] B. Cohen, "Incentives build robustness in bitTorrent," in *Proceedings of the 1st Workshop on Economics of Peer-to-Peer Systems*, Berkeley, Calif, USA, June 2003.
[2] napster, http://www.napster.com/.
[3] gnutella, http://www.gnutella.com/.
[4] eMule, http://www.emule-project.net/.
[5] Y. Zhao, X. Hou, M. Yang, and Y. Dai, "Measurement study and application of social network in the maze P2P file-sharing system," in *Proceedings of the 1st International Conference on Scalable Information Systems (InfoScale '06)*, p. 57, ACM Press, Hong Kong, May 2006.
[6] Pplive, http://www.pplive.com/.
[7] D. P. Anderson, J. Cobb, E. Korpela, M. Lebofsky, and D. Werthimer, "SETI@home: an experiment in public-resource computing," *Communications of the ACM*, vol. 45, no. 11, pp. 56–61, 2002.
[8] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan, "Chord: a scalable peer-to-peer lookup service for internet applications," in *Proceedings of the International Conference on Applications, Technologies, Architectures, and Protocols for Computers Communications (SIGCOMM '01)*, pp. 149–160, ACM Press, San Diego, Calif, USA, August 2001.
[9] A. I. T. Rowstron and P. Druschel, "Pastry: scalable, decentralized object location, and routing for large-scale peer-to-peer systems," in *Proceedings of the IFIP/ACM International Conference on Distributed Systems Platforms Heidelberg (Middleware '01)*, pp. 329–350, Springer, Heidelberg, Germany, November 2001.

[10] B. Zhao, L. Huang, J. Stribling, S. Rhea, A. Joseph, and J. Kubiatowicz, "Tapestry: a resilient global-scale overlay for service deployment," *IEEE Journal on Selected Areas in Communications*, vol. 22, no. 1, pp. 41–53, 2004.

[11] P. Maymounkov and D. Mazieres, "Kademlia: a peer-to-peer information system based on the XOR metric," in *Proceedings of the 1st International Workshop on Peer-to-Peer Systems (IPTPS '02)*, pp. 53–62, Cambridge, Mass, USA, March 2002.

[12] S. Saroiu, P. K. Gummadi, and S. D. Gribble, "A measurement study of peer-to-peer file sharing systems," in *Multimedia Computing and Networking (MMCN '02)*, vol. 4673 of *Proceedings of SPIE*, pp. 156–170, San Jose, Calif, USA, January 2002.

[13] Y. Qiao and F. E. Bustamante, "Structured and unstructured overlays under the microscope: a measurement-based view of two P2P systems that people use," in *Proceedings of USENIX Annual Technical Conference*, pp. 341–355, Boston, Mass, USA, May-June 2006.

[14] Overnet, http://en.wikipedia.org/wiki/Overnet/.

[15] eDonkey, http://en.wikipedia.org/wiki/Edonkey2000.

[16] Q. Lv, P. Cao, E. Cohen, K. Li, and S. Shenker, "Search and replication in unstructured peer-to-peer networks," in *Proceedings of the 16th International Conference on Supercomputing (ICS '02)*, pp. 84–95, ACM Press, New York, NY, USA, June 2002.

[17] K. Sripanidkulchai, B. Maggs, and H. Zhang, "Efficient content location using interest-based locality in peer-to-peer systems," in *Proceedings of the 22nd Annual Joint Conference on the IEEE Computer and Communications Societies (INFOCOM '03)*, vol. 3, pp. 2166–2176, San Francisco, Calif, USA, March-April 2003.

[18] R. Bhagwan, S. Savage, and G. Voelker, "Understanding availability," in *Proceedings of the 2nd International Workshop on Peer-to-Peer Systems (IPTPS '03)*, pp. 256–267, Berkeley, Calif, USA, February 2003.

[19] J. Yang, H. Ma, W. Song, J. Cui, and C. Zhou, "Crawling the eDonkey network," in *Proceedings of the 5th International Conference on Grid and Cooperative Computing Workshops (GCCW '06)*, pp. 133–136, Hunan, China, October 2006.

[20] K. Tutschku, "A measurement-based traffic profile of the eDonkey filesharing service," in *Proceedings of the 5th anuual Passive and Active Measurement Workshop (PAM '04)*, pp. 12–21, Antibes Juanles-Pins, France, April 2004.

[21] D. Stutzbach and R. Rejaie, "Improving lookup performance over a widely-deployed DHT," in *Proceedings of the 25th IEEE Conference on Computer Communications (INFOCOM '06)*, pp. 1–12, Barcelona, Spain, April 2006.

[22] M. Zhou, Y. Dai, and X. Li, "A measurement study of the structured overlay network in P2P file-sharing applications," in *Proceedings of the 8th IEEE International Symposium on Multimedia (ISM '06)*, pp. 621–628, San Diego, Calif, USA, December 2006.

[23] B. Chun, D. Culler, T. Roscoe, et al., "PlanetLab: an overlay testbed for broad-coverage services," *ACM SIGCOMM Computer Communication Review*, vol. 33, no. 3, pp. 3–12, 2003.

[24] amule, http://www.amule.org/.

[25] Rfc1034, http://www.ietf.org/rfc/rfc1034.txt.

[26] Rfc1035, http://www.ietf.org/rfc/rfc1035.txt.

[27] N. Naoumov and K. Ross, "Exploiting P2P systems for DDoS attacks," in *Proceedings of the 1st International Conference on Scalable Information Systems (InfoScale '06)*, vol. 152, p. 47, ACM Press, Hong Kong, May-June 2006.

[28] J. Liang, N. Naoumov, and K. W. Ross, "The index poisoning attack in P2P file sharing systems," in *Proceedings of the 25th IEEE Conference on Computer Communications (INFOCOM '06)*, pp. 1–12, Barcelona, Spain, April 2006.

Hindawi Publishing Corporation
Advances in Multimedia
Volume 2007, Article ID 84150, 12 pages
doi:10.1155/2007/84150

Research Article

Analysis and Implementation of Gossip-Based P2P Streaming with Distributed Incentive Mechanisms for Peer Cooperation

Sachin Agarwal,[1] Jatinder Pal Singh,[1] and Shruti Dube[2]

[1] *Deutsche Telekom AG Laboratories, Ernst-Reuter-Platz 7, 10587 Berlin, Germany*
[2] *Department of Computer Science and Engineering, Indian Institute of Technology, Kanpur 208 016, India*

Received 30 January 2007; Revised 29 June 2007; Accepted 15 August 2007

Recommended by Shigang Chen

Peer-to-peer (P2P) systems are becoming a popular means of streaming audio and video content but they are prone to bandwidth starvation if selfish peers do not contribute bandwidth to other peers. We prove that an incentive mechanism can be created for a live streaming P2P protocol while preserving the asymptotic properties of randomized gossip-based streaming. In order to show the utility of our result, we adapt a distributed incentive scheme from P2P file storage literature to the live streaming scenario. We provide simulation results that confirm the ability to achieve a constant download rate (in time, per peer) that is needed for streaming applications on peers. The incentive scheme fairly differentiates peers' download rates according to the amount of useful bandwidth they contribute back to the P2P system, thus creating a powerful quality-of-service incentive for peers to contribute bandwidth to other peers. We propose a functional architecture and protocol format for a gossip-based streaming system with incentive mechanisms, and present evaluation data from a real implementation of a P2P streaming application.

1. INTRODUCTION

Peer-to-peer (P2P) applications are widely used to distribute media files over the Internet and are sometimes referred to as "file-sharing" applications. P2P applications for real-time audio and video streaming have also been proposed and implemented in various academic and commercial settings (see Section 2). These applications are becoming popular with end users who wish to be content providers but do not have high-bandwidth upstream connections to stream their videos to other users on the Internet.

All P2P content delivery systems work on the premise that the peers will share their resources in order to increase the total service capacity of the P2P system. In the case of file sharing and streaming, the resource is the upstream bandwidth each peer contributes to distribute content to other peers. Peers download parts (blocks) of the media file or stream from various other peers and then reassemble these blocks. Thus P2P systems derive bandwidth from the participating peers who operate independently of each other. A mechanism, that creates compelling incentives for all peers to contribute resources and thus guards against bandwidth starvation in the P2P system, is needed to sustain peer interest in sharing bandwidth.

Many implemented P2P protocols have incentive mechanisms that encourage peers to contribute resources and participate in this "block sharing." For example, the BitTorrent file-sharing application's tit-for-tat incentive mechanism [1] rewards peers who share more bandwidth by allocating better download rates to them.

Unlike file sharing, live video streams have a time-limited "value" of a stream block, because older stream blocks become obsolete as time progresses, with older blocks no longer being "live." Even when a prerecorded stream is multicast and *played back concurrently* on the peers as they download the stream, peers seek to download contiguous blocks instead of random blocks in order to be able to play back the stream in real time as it is being downloaded. Thus stream blocks cannot be downloaded in a random order and BitTorrent's tit-for-tat scheme of bartering random blocks is not directly applicable for the case of live P2P streaming.

Some P2P streaming protocols for dissemination of video content distribution are centered around tree-based overlays, as surveyed in Section 2. A difficulty with most tree-based protocols is that the concept of "equality" amongst peers is inherently missing owing to the hierarchical tree structure, and so the formulation of a distributed incentive scheme is nontrivial. Another approach is gossip-based P2P video

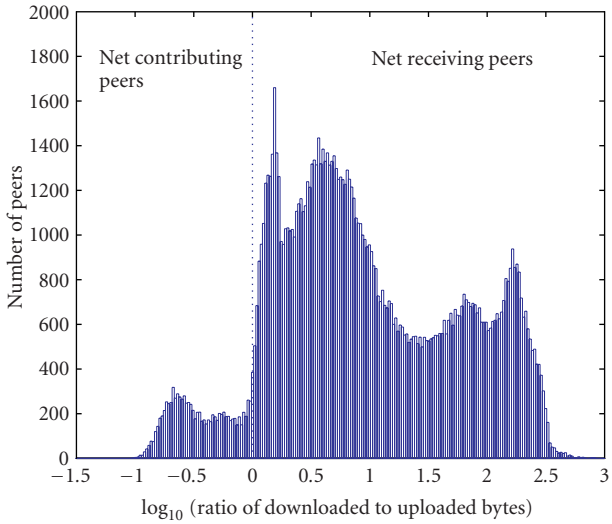


Figure 1: Histogram of the logarithm of the ratio of bytes sent to bytes received.

streaming, where stream blocks are "spread" amongst the peers using random gossip, as explained in Section 3. Creating incentive mechanisms for this class of protocols is one of the objectives of our work.

In order to further motivate our work, we obtained and analyzed log data from Roxbeam Media Networks (commercial offshoot of Cool Stream) to understand the overall distribution of peers' resource contributions to the P2P streaming effort. The particular log we analyzed was of a 4-hour baseball game streamed at 759 kbps video bitrate to about 120,000 unique IP addresses. The aim of this analysis was to highlight that many peers contribute very little in terms of bandwidth, and a few peers contribute a lot.

Figure 1 plots the distribution of the logarithm of the ratio of number of bytes downloaded to that uploaded by a peer. Most peers download more than they upload. Figure 1 clearly shows the inequality in P2P streaming systems: most peers contribute lesser than they download, and a few peers end up contributing the bulk of the bandwidth. We are motivated by this fact to develop an incentive system to reward cooperative, contributing peers in P2P streaming systems with better video streaming quality.

We provide a generic achievability argument for creating incentives to cooperate in gossip-based live P2P streaming. We do this by proving that after coupling a randomized optimistic unchoke component with an incentive mechanism, we can (still) distribute a stream block from the source to n peers in $O(\log n)$ time with high probability. This result is very significant because tree-based P2P streaming protocols also have logarithmic distribution times (because of the $O(\log n)$ tree height).

We employ a distributed incentive algorithm previously proposed for P2P file sharing and distributed file storage to demonstrate that this incentive mechanism is able to differentiate between peers on the basis of how much useful bandwidth each peer contributes, while still being in line with our analytical results for logarithmic completion time. Furthermore, we propose a functional architecture and protocol format for peers to implement and execute gossip-based streaming with incentive mechanisms. Finally, we provide simulation and real-implementation results to verify our claims.

We survey related work in overlay streaming multicast, gossip protocols, and distributed incentive mechanisms in Section 2. In Section 3, we elaborate on our model of a gossip-based overlay multicast protocol. The main result on the achievability of an incentive mechanism for gossip-based P2P streaming is provided in Section 4. We provide simulation results in Section 5 to verify the analytical results of Section 4. We then provide the functional architecture and protocol format for gossip streaming in Section 6. Finally, we provide performance results from implementation on a test bed in Section 7 and discuss some practical implementation aspects in Section 8. Conclusions and future work are presented in Section 9.

2. RELATED WORK

P2P systems are widely used to distribute content in the Internet, and it is estimated that a major portion of the bandwidth available on the consumer ISP networks carries P2P content [2]. It has been shown through analysis [3, 4], simulations, and measurements [5–7] that the P2P content delivery model scales gracefully with user demands in heterogeneous P2P networks. A majority of popular P2P systems are built around file sharing applications. These applications typically distribute stored and not live content, and the downloaded content is played back only after the entire file has been downloaded.

A large number of P2P streaming multicast solutions are based around setting up an overlay tree with the content source being the root and terminals being the other nodes of the tree [8–10]. This approach has the disadvantage of limited robustness against peer disconnections that disrupt stream reception on downstream neighbors. Many techniques [11] for alleviating this problem based on redundant data paths, multiple overlay trees, and multiple description coding have been proposed. We refer the interested reader to a good survey of P2P streaming in [12].

A comparison of the approaches and algorithms employed in various P2P streaming overlays can be found in [13, 14]. There have also been recent studies of commercial P2P streaming systems, for example [15–20], that study networking characteristics of some commercial P2P systems such as SopCast [21, 22], PPLive [23], Coolstreaming [24, 25] and Gridmedia [26].

Gossip protocols were first introduced as a means to achieve the lazy database replication [27]. The underlying architecture of BitTorrent [1] also uses a gossip-like randomized protocol to propagate data to peer hosts. The Pbcast protocol [28] uses the combination of a multicast tree and a gossip protocol to achieve robust content delivery. Recent work [29] confirms the utility of gossip-based protocols for media streaming applications. The Cool-stream technology [**?**] is a real-world implementation of such a gossip-based TV quality video stream delivery system.

3. GOSSIP-BASED P2P STREAMING

A gossip-based P2P streaming algorithm invokes a randomized gossip protocol on each stream block repeatedly to spread the blocks in the P2P network. Peers contact other peers randomly to spread the blocks through the network, much like an infectious virus spreads in a population or a rumor spreads in a crowd. This "gossiping" can be pull-based or push-based depending upon whether peers actively seek out the blocks they require (pull), or whether peers actively advertise (push) blocks to other peers.

For P2P streaming, the source peer only uploads blocks of the stream to randomly chosen peers. Unlike unicast, the source peer does not distribute a unique copy of the content to every recipient peer, and this characteristic underlines the P2P value: the source peer does not need a high-speed upload connection to simultaneously stream to a large population of recipient peers.

In the case of video streaming when recipients want to playback the received stream in real time (as it is being downloaded), recipient peers seek contiguous blocks of the stream so that the stream can be rebuilt from the downloaded blocks and played back instantaneously. We assume that each peer caches the downloaded stream and is capable of serving this stream to other peers. In case the stream is very long, only a certain stream window can be saved, but we discount this aspect in our analysis.

4. ANALYSIS

4.1. Incentives

The natural way of implementing an incentive mechanism is to make the bandwidth received by a peer a function of the bandwidth contributed by that peer. Ideally, the algorithm should be distributed with serving peers making decisions about how much bandwidth to allocate to other peers using local information only, without centralized control. We adopt the peer-wise proportional fairness scheme analyzed in [30, 31] to achieve this objective.

Let u_i be the total utility offered by peer i to the P2P network. At time t, peer i serves $u_{ij}(t)$ to peer j, computed on a peer-wise proportional basis, that is,

$$u_{ij}(t) = \frac{u_i}{\sum_{l=1}^{n} R_l(t) \sum_{k=0}^{t-1} u_{li}(k)} R_j(t) \sum_{k=0}^{t-1} u_{ji}(k), \qquad (1)$$

with the understanding that $0/0 = 0$ and $R_j(t)$ is set to 1 if peer j is requesting bandwidth from peer i, and 0 otherwise. (1) allocates peer i's offered utility u_i to all the other peers in the overlay requesting the utility based upon how much cumulative utility each peer has contributed to peer i in the past time.

The "utilities" in (1) can be the bandwidth (in terms of kbps, e.g.). In general, a peer i is free to assign any utility value to peer j depending on how peer i perceives the utility of peer j offered to itself (u_{ji}). For example, a noncooperative peer i may set all received utilities by setting u_{ji} to 0 for all j and then $u_{ij} = 0$, for all j according to (1), thus cutting off all contributions to other peers in the P2P network. Then the other peers receive no service from peer i and would decrease their contribution to peer i according to (1). Note that all peers only make local measurements of the utility they have received from other peers, thus making peer-wise proportional fairness a completely distributed algorithm.

In a live multimedia streaming application, peers gossip in order to discover the next useful (contiguous) block of the live stream amongst their peers. So even though a peer may be offering bandwidth to another peer, the utility of this bandwidth might be 0 in case the offering peer has no useful block to share with the other peer, as might be the case when new peers join the network. Thus per-stream fairness is not guaranteed by peer-wise proportional fairness. As we shall show later through simulations, the peer-wise proportional fairness algorithm is fair in the asymptotic sense (i.e., across multiple streams and over a long time).

The peer-wise proportional fairness criteria is biased toward peers who join the overlay streaming session early on and "build up credit" with other early peers. To allow newer peers to start downloading the stream, a fraction of the total utility of each peer called "optimistic unchoke" utility (borrowing a term from BitTorrent [1]) is offered to any requesting peer independent of the utility the requesting peer had offered previously. This optimistic unchoke utility also preserves the logarithmic completion time properties of gossip streaming, as we prove below.

4.2. Gossip under an incentive scheme

We model our P2P streaming system as a completely connected P2P network comprising n peers. For analysis, we assume that time is divided into discrete time slots (or rounds). Peer i can gossip with f_i other peers in one time slot (this is the fan-out of peer i, $f_i \geq 0$). Further, each peer assigns a finite fraction of f_i for uniform random gossip; this is the *optimistic unchoke* fraction δ_i of peer i.

We must show that the important property of a gossip protocol—in which each stream block is received with high probability by all peers in $O(\log n)$ time-slots—is valid with the incentive scheme, because our incentive scheme changes the *random* gossip model of classical gossip protocol analysis. This is important for comparing gossip-based streaming protocols to tree-based protocols, where the height of the tree ($\log n$) determines the worst case time before a stream block reaches every peer (for a balanced tree).

Formally, we seek to prove that under the assumption of all peers using a fraction of their fan-outs for optimistic unchoke, a stream block will reach all peers in $O(\log n)$ time slots with high probability. Without loss of generality, we consider the situation when the stream source has one block that it needs to spread to the other peers comprising the P2P network.

Let $I(t)$ denote the set of peers that have received the block at the end of t time slots (hence these peers are "infected"). Initially, only the stream source possesses the block, hence $I(0) = 1$. The number of peers that have not received the block after t time slots is denoted by $U(t) = n - I(t)$ (this quantity is the number of "uninfected" peers after time t).

We adopt the model of the block spreading from [32]. Specifically,

$$E\big(U(t+1) \mid I(t)\big) = U(t)\big(1-n^{-1}\big)^{\delta f I(t)} \quad (2)$$

$$\approx U(t)e^{-\delta f I(t)/n}. \quad (3)$$

As in [32], we neglect the fluctuation of $U(t+1)$ around its conditional expectation to get

$$U(t+1) = U(t)e^{-\delta f I(t)/n}. \quad (4)$$

Further, we define

$$S_n = \min\{t : I(t) = n\}, \quad (5)$$

so S_n denotes the number of time slots until everyone receives the stream block with high probability. Further, we define $\delta_m f_m = \min(\delta_i f_i)$ for all i. We now state and prove an important Lemma for proving an upper bound in the number of time slots S_n in Theorem 2. For two identical P2P networks α and β, let $U_\alpha(t)$ and $U_\beta(t)$ be the number of uninfected peers after time t in networks α and β, respectively.

Lemma 1. *For randomized gossip in two identical P2P networks α and β, if $U_\alpha(t) \le U_\beta(t)$, then $U_\alpha(t+k) \le U_\beta(t+k)$ for $k \ge 0$.*

Proof. Since $U_\alpha(t) \le U_\beta(t)$, we have $I_\alpha(t) \ge I_\beta(t)$, because in identical networks α and β, $U(t) + I(t) = n$ for all t. Applying these inequalities in (3), we get $E(U_\alpha(t+1) \mid I_\alpha(t)) \le E(U_\beta(t+1) \mid I_\beta(t))$. Or equivalently from (4),

$$U_\alpha(t+1) \le U_\beta(t+1). \quad (6)$$

An induction argument then proves the statement of the lemma. □

An incentive model with optimistic unchoke can be thought of as overlaying the incentive-based (nonrandom) gossip protocol on top of an altruistic optimistic unchoke gossip protocol. Lemma 1 asserts that the altruistic optimistic unchoke mechanism will only speed up due to the increased number of infected peers from the overlaid incentive-based gossip.

Theorem 2. *In probability,*

$$S_n \le \log_{\delta_m f_m + 1} n + \big(\delta_m f_m\big)^{-1} \log n + O(1) \quad (7)$$

as $n \to \infty$.

Proof. We provide a sketch of the proof here. The proof follows on almost identical lines to a similar result for randomized gossip in [32], we reproduce an adapted version here for completeness. We show that if all peers only run the minimal altruistic gossip with fan-out $\delta_m f_m$, then the statement of Theorem 2 is satisfied in equality. Lemma 1 ascertains the direction of the inequality when additional infections occur under an incentive scheme.

Specifically, define

$$x(t+1) = f\big[x(t)\big], \quad t \ge 0,$$
$$f(x) = 1 - (1-x)e^{-\delta_m f_m x}, \quad (8)$$

where $x(t) = I(t)/n$ so that $x(0) = n^{-1}$. In terms of $x(\cdot)$,

$$S_n = \min\{t : x(t) > 1 - n^{-1}\}. \quad (9)$$

So, $f(0) = 0$, $f'(0) = 1 + \delta_m f_m$, and $f'(1) = e^{-\delta_m f_m}$.

Meaning that $x(t)$ grows exponentially in the beginning and increases slowly with a rate of $\approx e^{-\delta_m f_m}$ when most peers are already infected.

Let $\epsilon = \epsilon(n) \to 0$ and

$$\tau_1 = \tau_1(n) = \min\{t : x(t) > \epsilon\},$$
$$\tau_2 = \tau_2(n) = \min\{t > \tau_1 : x(t) > 1 - \epsilon\}. \quad (10)$$

Then,

$$\tau_1 = \log_{1+\delta_m f_m} \frac{\epsilon}{n^{-1}},$$
$$S_n - \tau_2 \approx \big(\delta_m f_m\big)^{-1} \log \frac{\epsilon}{n^{-1}} \quad (11)$$

since $f(x) > x$ for $0 < x < 1$,

$$\tau_2 - \tau_1 = o\big[\tau_1 + S_n - \tau_2\big]. \quad (12)$$

This means that the rumor spreads much more rapidly when the number of infected peers is neither too large nor too small. Combining (11) and (12) yields the proof of the theorem assuming $\epsilon \to 0$. □

The optimistic unchoke component ensures that the gossip protocol completes disseminating the block in *at least* $O(\log n)$ time and overcomes the clique tendency of the peer-wise proportional fairness incentive mechanism of (1) that would partition the network into peers that exchange blocks only among themselves and completely "ignore" peers who join the stream later and/or have a lower upload rate.

5. SIMULATIONS

We provide simulation experiments to verify three important properties of incentives in gossip-based streaming. Firstly, we demonstrate that the number of time slots before a stream block reaches all peers in a large P2P streaming network is logarithmic in the network size. Secondly, we show that a constant rate is achieved at the peers, meaning that they can play the stream in real time as it is being downloaded. Thirdly, our simulations demonstrate that the peer-wise proportional fairness of (1) rewards peers according to their contributions to other peers. In addition, we show the asymptotic fairness properties of peer-wise proportional fairness in Section 5.1.

We created a discrete time simulator that implemented the push-based gossip protocol on n peers in a completely connected network, that is, each peer could route blocks to any other peer. Peers ran the distributed peer-wise proportional fairness algorithm (1) to compute the (weighted and nonuniform) probability of assigning a part of their fan-out to all other peers in each time slot. We simulated two classes of peers: those which contribute bandwidth back to the network and those that do not contribute anything back to the P2P streaming network.

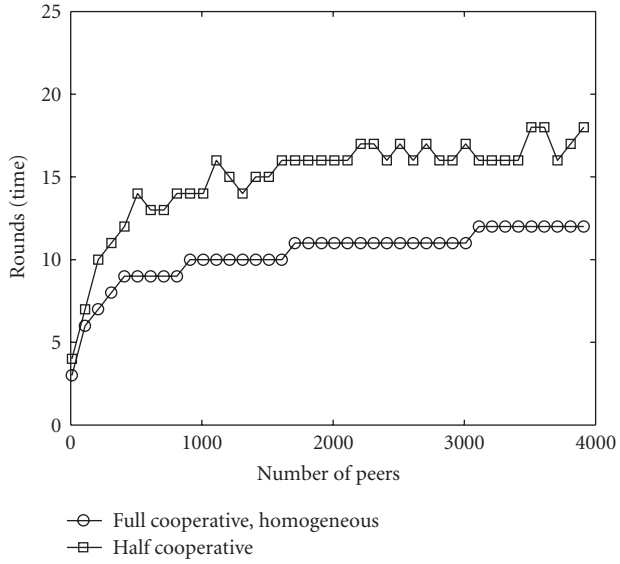


FIGURE 2: A stream block reaches all n peers in $O(\log n)$ time with the peer-wise proportional fairness incentive mechanism.

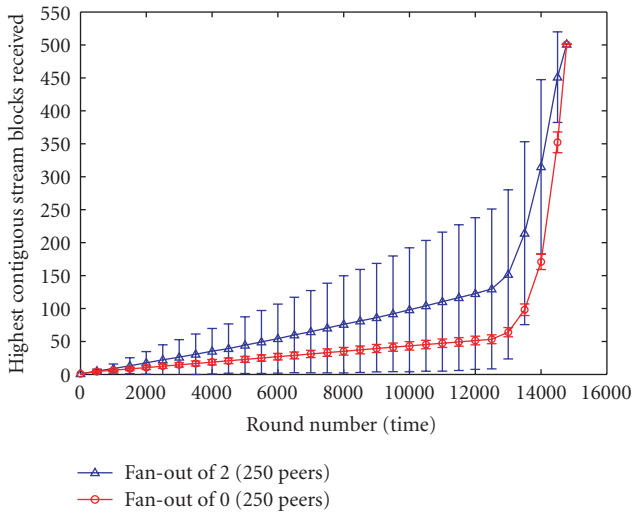


FIGURE 3: Optimistic unchoke fraction $\delta_i = 0.01$ for all i, 500 peers, 500 stream blocks. A constant streaming rate to peers is supported; and the peer-wise proportional fairness incentive algorithm differentiates peers according to their bandwidth contributions to the P2P network.

Figure 2 shows results for the number of time-slots (rounds) required to spread one block of a stream to all peers of the network with optimistic unchoke parameter $\delta_i = 0.01$ for all i. In the full-cooperative, homogeneous simulation of each peer had a fan-out of 3, meaning that each peer could support concurrent stream uploads to 3 other peers. In the "half-cooperative" mode, only half of the peers contributed to upload bandwidth. The plot confirms our analytical result for incentive-based gossip requiring $O(\log n)$ rounds.

Peers that contribute bandwidth back into the system were rewarded with higher-download rates, as Figure 3 indicates. In order to observe the streaming rate of a long stream,

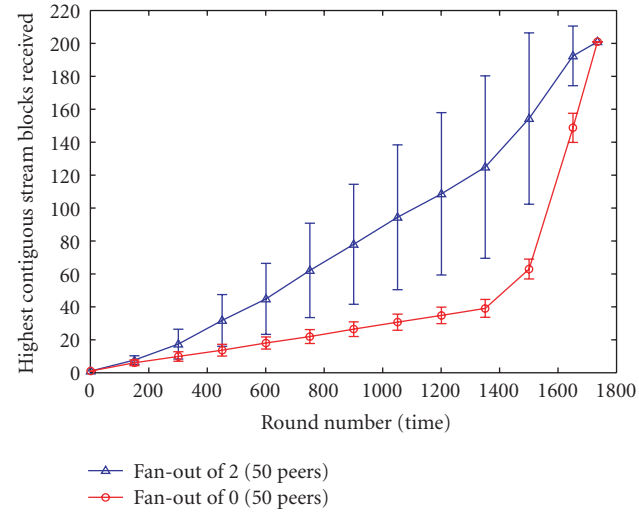


FIGURE 4: Optimistic unchoke fraction $\delta_i = 0.01$ for all i, 100 peers, 100 stream blocks.

our stream was divided into 500 stream blocks, each of which was gossiped in the P2P network. The knee in the graph indicates the time when most peers who were contributing more bandwidth (fan-out 2) to the system had completed the download and offered more bandwidth to all other peers. The distributed peer-wise proportional fairness algorithm correctly classifies peers according to how much they contribute back to the P2P network. Another important observation is that the graph is linear until the knee is reached, meaning that, on average, a constant download rate to peers is supported, which is of paramount importance for real-time playback of the video stream.

5.1. *Asymptotic properties of peer-wise proportional fairness*

The next few simulations shed light on the asymptotic properties of peer-wise proportional fairness. We simulated a P2P gossip streaming network with 100 peers, half of who contributed bandwidth (full cooperative, fan-out 2) while the other half did not contribute any bandwidth (noncooperative) to their peers.

As expected, peers who contributed bandwidth to the P2P system were rewarded with higher download rates, as Figure 4 indicates. But there was a large variance in the download rate amongst the full-cooperative (and to a lesser extent noncooperative) peers as indicated by the error bars of Figure 4. We therefore ran additional simulations in order to verify the asymptotic fairness of peer-wise proportional fairness across multiple stream downloads.

The bar graphs of Figures 5 and 6 show the average download time (rounds) taken by 50 full-cooperative and 50 noncooperative peers to receive the 100 stream blocks comprising a video stream. It is clear that, on average, all peers (full-cooperative or noncooperative) receive similar quality of service depending on how much they contribute back to

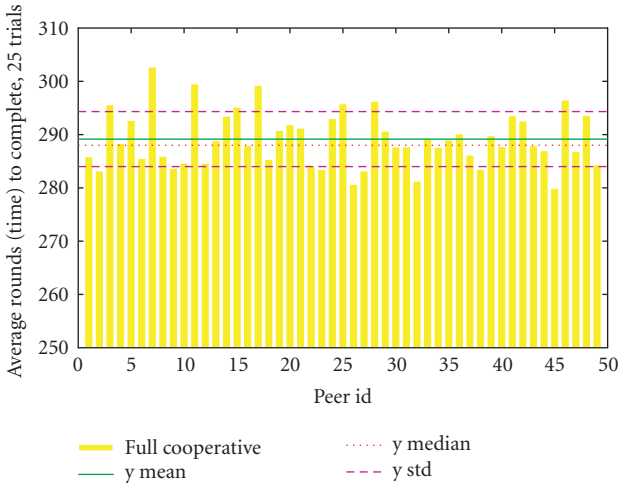


Figure 5: 50 full cooperative peers' download times (rounds), averaged over 25 streams.

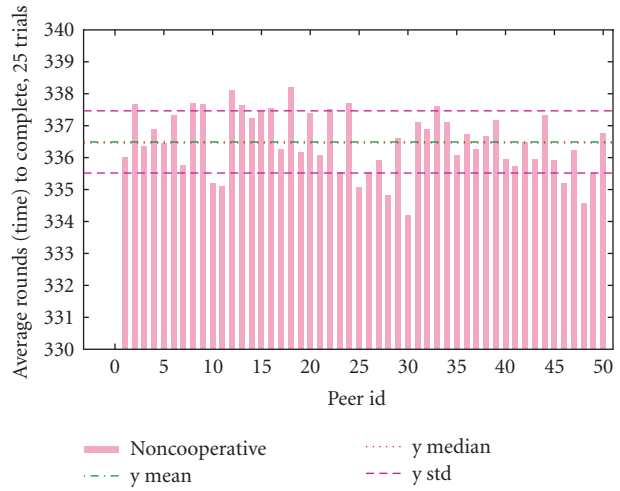


Figure 6: 50 full-noncooperative peers' download times (rounds), averaged over 25 streams.

the P2P system. Thus, the peer-wise proportional fairness scheme enables asymptotic fairness.

6. ARCHITECTURE AND PROTOCOL FORMAT

We next describe the architecture and functional components of the peers for gossip-based streaming. We also outline a protocol format that will be employed for implementation and evaluation in the next section.

6.1. Peer state space

Figure 7 shows the states of the peers as the incentive based P2P gossip streaming is executed by them in a distributed and asynchronous fashion.

(1) *Offline:* a peer in the offline state is not actively involved in downloading, uploading or otherwise participating in the media stream distribution.

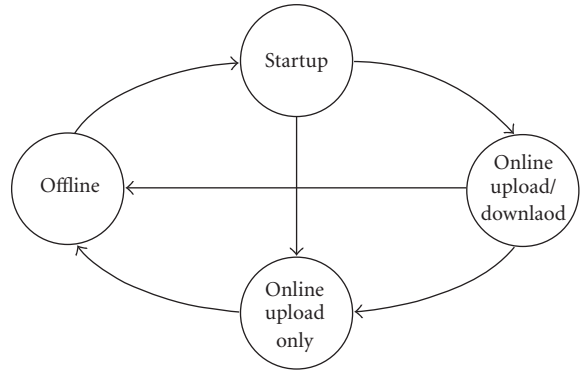


Figure 7: The state diagram of a peer.

(2) *Start-up:* a peer enters the start-up state when it joins the overlay P2P network. In case the peer is the source, it sets up the tracker that maintains a hostfile which contains the current list of all peers in the network. The nonsource peers on the other hand retrieve the hostfile from the tracker in the Start-up phase.
All peers fire-up a Listener component (Section 6.2) that listens to protocol messages. The streaming protocol parameters are also initialized during the Start-up phase. These include the gossip timeout, maximum block size, and upload bandwidth. The gossip timeout (ms) is the time between successive advertisement messages sent from a content uploading peer to a content downloading peer. Its value can be increased or decreased in order to slow or speed up the stream download rate, respectively, subject to network and content stream availability. The maximum block size (kB) is the size of the largest block transferred from one peer to another using the media streaming protocol.
The upload bandwidth is the upper limit on the amount of upload bandwidth dedicated to uploading stream blocks to requesting peers. The upper limit enforces a control over bandwidth that enables the users to run other applications without adversely affecting their online experience due to excessive utilization of the upload bandwidth of the network connections by the P2P streaming application.

(3) *Online upload/download:* following the completion of the startup phase, a peer enters the online upload/download state, during which the upload and download of data blocks happen. The source peer never enters this state as it does not download media content. A peer in online upload/download state can leave the overlay network and become offline.

(4) *Online upload only:* in this state a peer only uploads media content. This is the state acquired by the source after startup. All other peers will eventually complete downloading the stream (in case of a fixed-length media stream), thus ending up with a cached copy of the entire media stream file just like the source host. These peers may decide to altruistically act as sources and

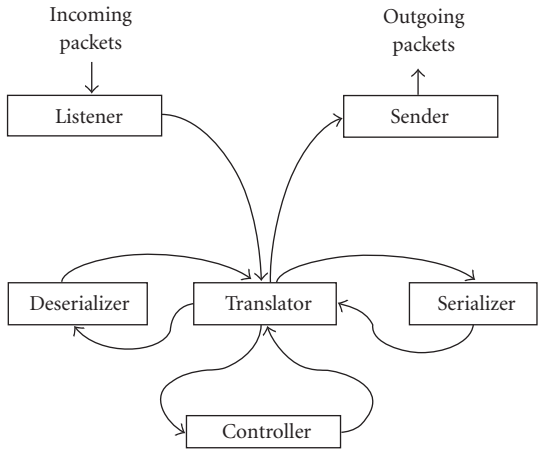


Figure 8: Functional components of a peer.

upload the stream to other requesting peers. The corresponding state of these peers is then online upload only.

The peers which act as content uploading peers or sources are called cooperative peers, whereas the peers which do not operate as content uploading peers are called noncooperative peers. The cooperative peers are able to achieve higher-download rates via the incentive-based mechanisms the implementation of which will be discussed in Section 6.4. A peer in online upload only state can leave the overlay network and enter the offline state.

6.2. Peer functional components

Figure 8 shows a schematic diagram of the functional components of a peer. The component listener listens on a specific port and receives transport layer packets from the media streaming application running on another peer: the payload of the packets being in a serialized suitable for transmission across the network, the listener passes the incoming packet's payload to a translator. The translator component deserializes the payload via the deserializer and stores it in a hash table which is then passed on to a controller. The translator is also responsible for converting a deserialized hash-table message understandable by the controller to a serialized message suitable for transmission in the payload of a packet. The serialization is performed via the serializer component. The serialized message is then transmitted by the component Sender to the listening port of another peer.

The controller comprises of a subcomponent called the Resource Broker that decides whether the media stream content should be sent to a soliciting peer. The Resource Broker makes the decision of sending based on the peer's available bandwidth and the incentive (credit) that the soliciting peer has earned by uploading to the peer.

6.3. Streaming operation

Figure 9 summarizes the operation of media streaming application for a nonsource peer. Push-based and pull-based gossip functionalities are depicted. A content downloading peer is said to be pulling a data block of the media data stream when it actively solicits the next contiguous block of the media data stream from a content uploading peer. The solicitation is performed via sending a request to send protocol message to a randomly chosen peer in every time slot. A content uploading peer is said to be pushing a data block when it actively advertises blocks of the media stream to content downloading peers via available messages. In general, the push-based or pull-based functionalities or a combination of the two may be employed for gossip-based media streaming.

Figure 10 depicts the operation of media streaming application for the source peer. The pull-based functionality is absent in this case since the source does not need to download content.

6.4. Realization of incentives

The choice of a peer for sending an available message (Figure 9) is made via an incentive-based mechanism. Each nonsource peer i maintains a vector U with elements designating the amount of data (in kB) received from each of the other peers. If the N peers are indexed, without loss of generality, as $0, 1, \ldots, N-1$, then the normalized vector

$$P = \frac{U}{\sum_{j=0}^{N-1} U[j]} \tag{13}$$

represents the probability mass function of a random number which represents the index of the peer to which available message is sent. Thus, the peer that contributed more earlier has a better chance of becoming a recipient of content.

Again, the Resource Broker in Figure 9 sends content to a peer according to the incentive-based approach. The size of a block sent to a peer k is ascertained as $P[k]B_{\max}$, where $B_{\max}$ is the maximum block size, as elaborated in Section 8.3.

7. IMPLEMENTATION AND EVALUATION

We implement the gossip streaming protocol as described in Section 6 and evaluate the streaming performance in a real test bed consisting of a small grid of computers networked via gigabit connectivity. We use a traffic shaping to limit the upload bandwidth assigned to the gossip streaming application on each peer. The scenario consists of 5 peers one of which acted as the content source. The source serves cached (prerecorded) audio or video files, although our implementation also supports live video capture via the java multimedia framework [33]. Peers use the implemented gossip streaming protocol in order to share and distribute blocks of the stream, with the objective of playing back the streamed file in real time (playback while the stream downloads). For the experiments, a 5.4 MB MPEG-2 video file is streamed from source peer. Figure 11 depicts the variation of time to download the file with the maximum block size. The gossip thread time-out is kept constant at 100 milliseconds and the upload bandwidth is fixed at 1200 kbps. As expected, the time to download the file decreases as the maximum block size was increased because the bigger block sizes resulted in

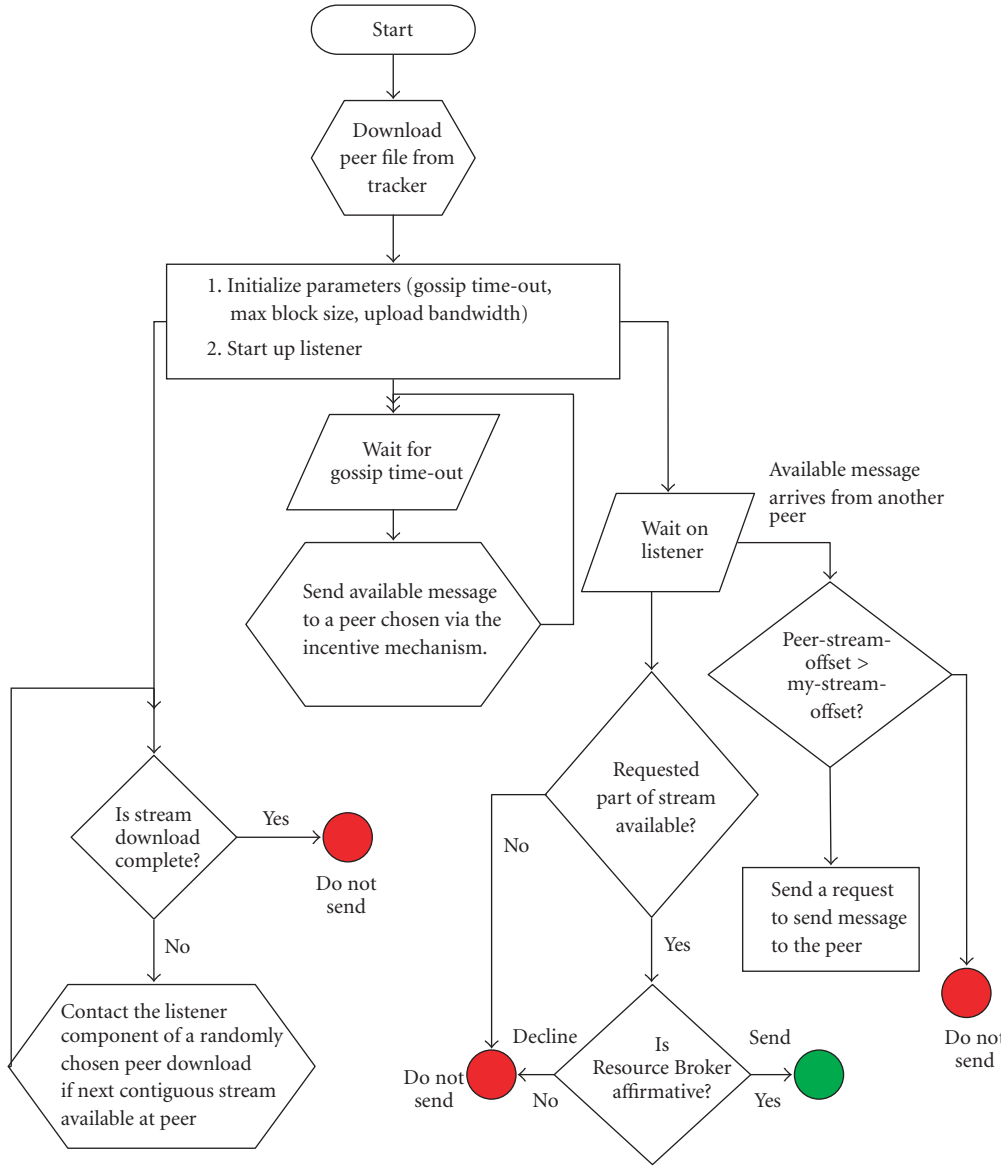


Figure 9: Streaming Operation of nonsource peer.

the same 5.4 MB being downloaded with smaller number of (bigger) blocks, thus requiring fewer gossip transactions. Figure 12 depicts the time to download the file as a function of the gossip thread time-out. The bandwidth is kept constant at 1200 kbps and the block size is fixed at 25 KB. It can be seen that the time to download the stream on a recipient peer grows linearly with the gossip thread time out because the number of requests (random probes) sent out decreased (per time), slowing down the gossip protocol and increasing the download completion time. Figure 13 depicts the variation of the time to download the file with the upload bandwidth. As expected, the download time decreases with increasing amounts of bandwidth available to serve peers on the P2P system. We measure the control overhead of our implemented protocol in Figure 14. The number of control data messages sent over the network remain fairly constant irrespective of the gossip thread time. A smaller gossip thread time-out only leads to a quicker completion of the stream download (as Figure 12), and the control overhead remains constant for a fixed stream download size (5.4 MB). This is an important scalability strength of our incentive-based gossip streaming protocol.

8. DISCUSSION

We briefly describe three practical improvements while implementing the incentive mechanism in a real peer-to-peer streaming system. First, we propose the use of a more realistic utility function rather than simply using the bandwidth. Second, propose a decay component in the utility function in order to make the implemented system adapt to changes in the peer-to-peer network fast. Third, we suggest varying

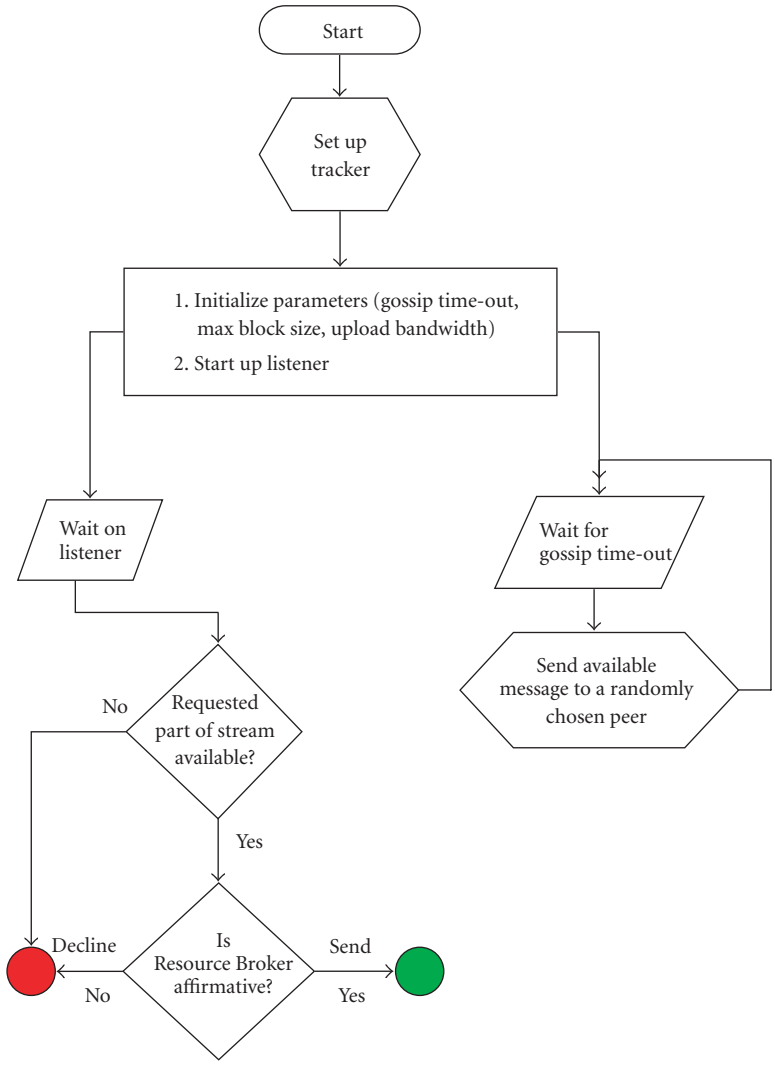


Figure 10: Streaming operation of the source.

block sizes as a supplementary measure to reward peers that contribute more bandwidth to the P2P system.

8.1. Realistic utility functions

In Sections 5, and 7 upload bandwidth was the primary barter quantity and a peer's utility to other peers was measured in terms of the upload bandwidth provided by the former to the latter. For example, the routing hop-distance (an indicator of delay and core network congestion), peer uptime (an indicator of peer reliability) can also be used by a peer while computing the utility of another peer. The modified utility function u'_{ij} in (1) can then be defined as

$$u'_{ij}(t) = \frac{u_{ij}(t) \cdot t_j}{h_{ji}}, \tag{14}$$

where u_{ij} is computed from (1), t_j is the time since peer i first communicated with peer j in the current session, and h_{ij} is the IP hop count as reported by the trace route IP routing tool.

Again, modifying the overlay characteristics by taking into account the underlying IP network and the up-time of peers breaks the "randomly connected" assumption of classical gossip analysis. By employing the optimistic unchoke argument of Section 4 we can still guarantee an $O(\log n)$ delivery time.

8.2. Exponential decay of utility

The expression for peer-wise proportional fairness provided in (1) may result in the distributed algorithm adapting slowly to peer churn. To see why this is case, consider peer i at time t that has received utility U_{ji} from peer i and U_{-j} from all other peers except peer j until time $t-1$. Then, the utility that $u_{ij}(t)$ assigns to peer j at time t is proportional to

$$u_{ij}(t) \propto \frac{U_{ji}}{U_{ji} + U_{-j}}. \tag{15}$$

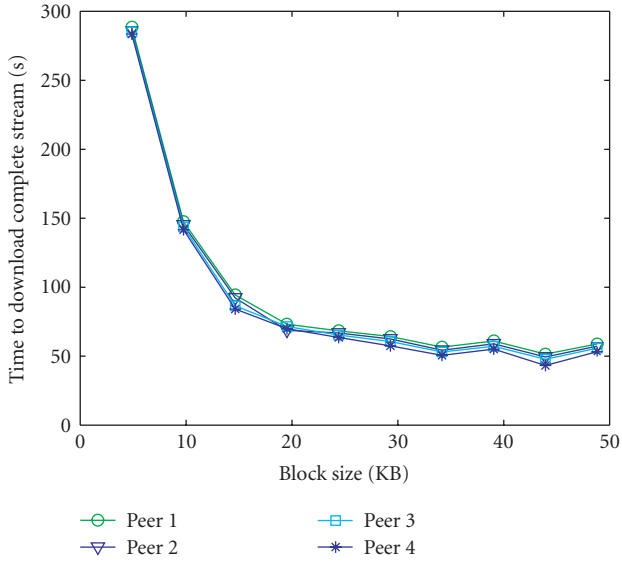


Figure 11: Time to download versus the block size.

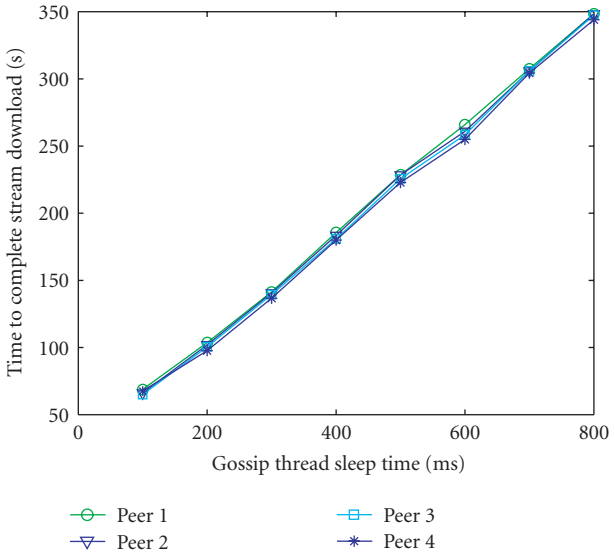


Figure 12: Time to download versus gossip thread time-out.

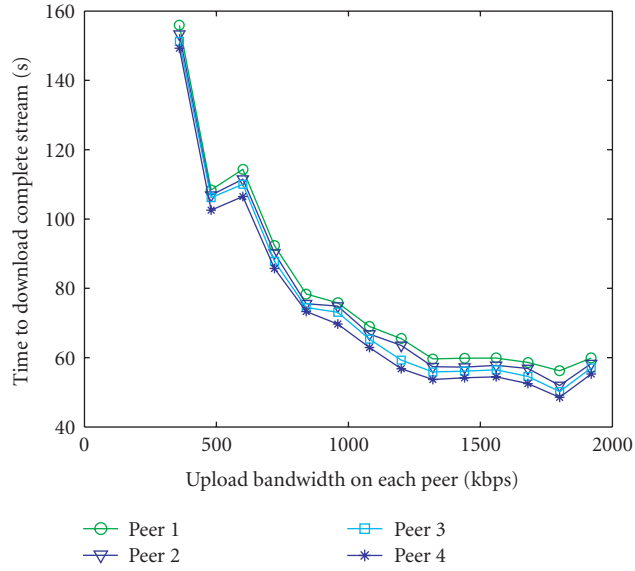


Figure 13: Time to download the complete stream (5.4 MB file) as a function of the upload bandwidth offered by each peer.

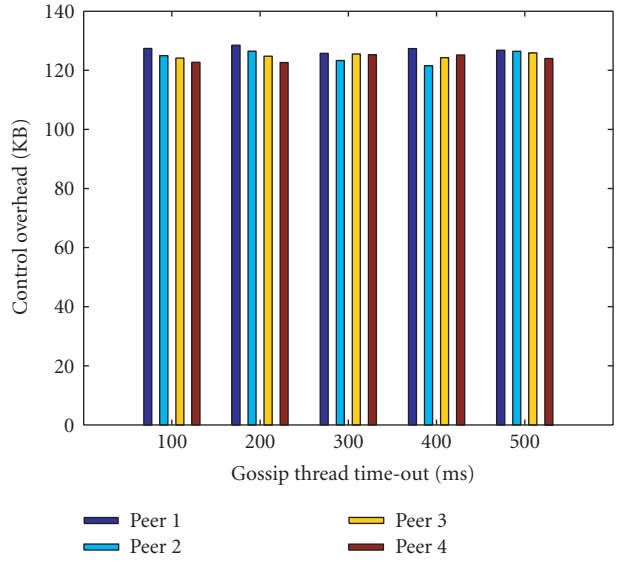


Figure 14: Control overhead remains constant across different gossip thread time-out.

Taking the derivative of $u_{ij}(t)$ with respect to U_{ji},

$$u'_{ij}(t) = \frac{U_{-j}}{(U_{ji} + U_{-j})^2}. \tag{16}$$

Since received utility is always nonnegative, the derivative $u'_{ij}(t)$ is nonnegative and upper-bounded by $1/U_{-j}$. This property makes $u_{ij}(t)$ adapt very slowly to changing peer characteristics after U_{-j} becomes sizeable (note that usually $U_{ji} \ll U_{-j}$). Thus while peer-wise-proportional fairness is asymptotically fair, it has slow dynamics in real systems.

In a real implementation, a decay mechanism to speed up the dynamics of the proposed method is incorporated in order to counter this problem. In particular, the utility values are decayed by a decay factor ρ ($0 \le \rho \le 1$) in each time slot (the time slot being an implementation-dependent parameter). Thus,

$$U(t+1) = \rho \cdot U(t). \tag{17}$$

This decay causes "historical" download information to become less relevant while recent download information becomes more relevant in the computation of utility for the incentive mechanism.

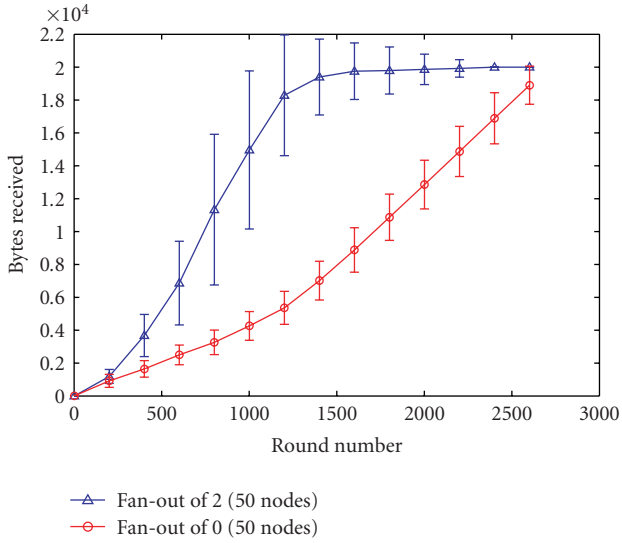


FIGURE 15: Simulation when stream block sizes are varied according to the incentive mechanism of (1).

8.3. *Stream block sizes as incentives*

A peer sending a stream block to a recipient peer can modulate the block size according to the latter's perceived utility as calculated on the sending peer (e.g., according to (1). Intuitively, a recipient peer that receives bigger blocks of the stream will download at a higher rate because fewer gossip transactions will be needed to download the whole stream.

We simulate these conditions in Figure 15. There are 100 peers, half of which contribute bandwidth while the other half do not contribute any bandwidth to the system. Peers push or upload stream blocks in sizes that are proportional to the utility of the recipient node calculated on the sending peer. Comparing Figure 15 to Figure 4, we observe that contributing peers achieve much higher rates as compared to their noncontributing counterparts.

9. CONCLUSIONS AND FUTURE WORK

Creating incentives for *live* video streaming is different than stored content incentives because the utility of the downloaded content is transient, with older content quickly becoming uninteresting to other peers, and hence unsuitable for barter. Conventional currency units (say dollars per downloaded MB) can be used as an instrument of barter, but the approach has the disadvantage of needing a centralized verification and transaction mechanism.

We have proved that by using an optimistic unchoke mechanism in a nonrandom gossip protocol we can achieve the important $O(\log n)$ time for dissemination of a stream block to all peers. Based on this result, an incentive mechanism to encourage peers to contribute resources such as bandwidth can be created while still preserving the logarithmic completion time property of gossip protocols.

We employed the distributed peer-wise proportional fairness algorithm for creating incentives for live video streaming. This algorithm achieves fairness in an asymptotic regime, as proved in [30, 31] and shown through simulations in Section 5.1. Short term fairness is not guaranteed however, as indicated from the error bars in Figure 3. Research in other fairness and incentive algorithms for P2P streaming remains an important part of future work.

Gossip protocols incur an extra factor of $\log n$ overhead as compared to tree-based P2P networks because the number of blocks transmitted in order to spread one block from the source to all other peers is $O(n \log n)$ in the Gossip-based approach as compared to $O(n)$ in the tree-based approach. The trade-off is between higher robustness (gossip protocols) and lower message overhead (tree-based protocols). Recent results [34] combine push- and pull-based gossip protocols to reduce the additional overhead factor to $O(n \log \log n)$. Our implementation uses such a combination of push- and pull-based gossip.

We have also delineated the functional components and protocol format that can be employed for the implementation of gossip-based streaming. Preliminary results from the implementation confirm the ability of achieving a constant throughput at the peers, a prime objective of any live streaming scheme. Experiments are underway to extend the results over a much larger test bed.

ACKNOWLEDGMENTS

The first author thanks Ari Trachtenberg, Moshe Laifenfeld, and Murat Alanyali for preliminary discussions on peer-wise proportional fairness. The first author also thanks Shruti Arun Gupta for editorial inputs. The authors sincerely thank the reviewers whose valuable suggestions helped in improving this work.

REFERENCES

[1] B. Cohen, "Bittorrent," http://bitconjurer.org/BitTorrent.
[2] J. Glasner, "P2P fuels global bandwidth binge," *Wired News*, April 2005.
[3] R. T. B. Ma, S. C. M. Lee, J. C. S. Lui, and D. K. Y. Yau, "A game theoretic approach to provide incentive and service differentiation in P2P networks," *ACM SIGMETRICS Performance Evaluation Review*, vol. 32, no. 1, pp. 189–198, 2004.
[4] Z. Ge, D. R. Figueiredo, S. Jaiswal, J. Kurose, and D. Towsley, "Modeling peer-peer file sharing systems," in *Proceedings of the 22nd IEEE Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM '03)*, vol. 3, pp. 2188–2198, San Francisco, Calif, USA, March-April 2003.
[5] S. Saroiu, P. K. Gummadi, and S. D. Gribble, "A measurement study of peer-to-peer file sharing systems," in *Multimedia Computing and Networking*, vol. 4673 of *Proceedings of SPIE*, pp. 156–170, San Jose, Calif, USA, January 2002.
[6] D. Qiu and R. Srikant, "Modeling and performance analysis of BitTorrent-like peer-to-peer networks," in *Proceedings of the Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications (ACM SIGCOMM '04)*, pp. 367–378, Portland, Ore, USA, August-September 2004.
[7] T. Ng, Y. Chu, S. Rao, K. Sripanidkulchai, and H. Zhang, "Measurement-based optimization techniques for bandwidth-demanding peer-to-peer systems," in *Proceedings of the 22nd*

Annual Joint Conference on the IEEE Computer and Communications Societies (INFOCOM '03), vol. 3, pp. 2199–2209, San Francisco, Calif, USA, March-April 2003.

[8] Carnegie Mellon University, "End system multicast," http://esm.cs.cmu.edu/.

[9] V. N. Padmanabhan, H. J. Wang, P. A. Chou, and K. Sripanidkulchai, "Distributing streaming media content using cooperative networking," in *Proceedings of the 12th International Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV '02)*, pp. 177–186, Miami, Fla, USA, May 2002.

[10] Y. Chawathe, "Scattercast: an adaptable broadcast distribution framework," *Multimedia Systems*, vol. 9, no. 1, pp. 104–118, 2003.

[11] J. Silber, S. Sahu, J. P. Singh, and Z. Liu, "Augmenting overlay trees for failure resiliency," in *Proceedings of the IEEE Global Telecommunications Conference (GLOBECOM '04)*, vol. 3, pp. 1525–1531, Dallas, Tex, USA, November-December 2004.

[12] C. L. Abad, W. Yurcik, and R. H. Campbell, "A survey and comparison of end-system overlay multicast solutions suitable for network-centric warfare," in *Battlespace Digitization and Network-Centric Systems IV*, vol. 5441 of *Proceedings of SPIE*, pp. 215–226, Orlando, Fla, USA, April 2004.

[13] E. K. Lua, J. Crowcroft, M. Pias, R. Sharma, and S. Lim, "A survey and comparison of peer-to-peer overlay network schemes," *IEEE Communications Surveys & Tutorials*, vol. 7, no. 2, pp. 72–93, 2005.

[14] N. Magharei, R. Rejaie, and Y. Guo, "Mesh or multiple-tree: a comparative study of live P2P streaming approaches," in *Proceedings of the 26th IEEE International Conference on Computer Communications (INFOCOM '07)*, pp. 1424–1432, Anchorage, Alaska, USA, May 2007.

[15] T. Silverston and O. Fourmaux, "P2P IPTV measurement: a comparison study," preprint, 2006.

[16] A. Ali, A. Mathur, and H. Zhang, "Measurement of commercial peer-to-peer live video streaming," in *The 1st International Workshop on Recent Advances in Peer-to-Peer Streaming in Conjunction with the 3rd International Conference on Quality of Service in Heterogeneous Wired/Wireless Networks*, Waterloo, Ontario, Canada, August 2006.

[17] X. Hei, C. Liang, J. Liang, Y. Liu, and K. W. Ross, "Insights into PPLive: a measurement study of a large-scale P2P IPTV system," in *Proceedings of the IPTV Workshop in Conjunction with the International World Wide Web Conference*, Edinburgh, UK, May 2006.

[18] X. Hei, C. Liang, J. Liang, Y. Liu, and K. W. Ross, "A measurement study of a large-scale P2P IPTV system," Tech. Rep., Department of Computer and Information Science, Polytechnic University, New York, NY, USA, 2006.

[19] X. Zhang, J. Liu, and B. Li, "On large-scale peer-to-peer live video distribution: coolstreaming and its preliminary experimental results," in *Proceedings of the IEEE International Workshop on Multimedia Signal Processing (MMSP '05)*, Shanghai, China, October-November 2005.

[20] M. Zhang, L. Zhao, Y. Tang, J.-G. Luo, and S.-Q. Yang, "Large-scale live media streaming over peer-to-peer networks through global internet," in *Proceedings of the ACM Workshop on Advances in Peer-to-Peer Multimedia Streaming (P2PMMS '05)*, pp. 21–28, Singapore, November 2005.

[21] SOPCast, "SOPCast," http://www.sopcast.org/.

[22] G. A. Fowler and S. McBride, "Newest export from China: pirated pay TV," *Wall Street Journal*, 2005.

[23] PPLive, "PPLive," http://www.pplive.com/.

[24] X. Zhang, J. Liu, B. Li, and T.-S. P. Yum, "CoolStreaming/DONet: a data-driven overlay network for peer-to-peer live media streaming," in *Proceedings 24th IEEE Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM '05)*, vol. 3, pp. 2102–2111, Miami, Fla, USA, March 2005.

[25] Coolstreaming, "Coolstreaming," http://www.coolstreaming.us/.

[26] Gridmedia, "Gridmedia," http://www.gridmedia.com.cn/.

[27] A. Demers, D. Greene, C. Hauser, et al., "Epidemic algorithms for replicated database maintenance," in *Proceedings of the 6th Annual ACM Symposium on Principles of Distributed Computing*, pp. 1–12, Vancouver, British Columbia, Canada, August 1987.

[28] K. P. Birman, M. Hayden, O. Ozkasap, Z. Xiao, M. Budiu, and Y. Minsky, "Bimodal multicast," *ACM Transactions on Computer Systems*, vol. 17, no. 2, pp. 41–88, 1999.

[29] S. Verma and W. T. Ooi, "Controlling gossip protocol infection pattern using adaptive fanout," in *Proceedings of the 25th IEEE International Conference on Distributed Computing Systems (ICDCS '05)*, pp. 665–674, Columbus, Ohio, USA, June 2005.

[30] X. Yang and G. de Veciana, "Service capacity of peer-to-peer networks," in *Proceedings of the 23rd Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM '04)*, vol. 4, pp. 2242–2252, Hong Kong, March 2004.

[31] S. Agarwal, M. Laifenfeld, A. Trachtenberg, and M. Alanyali, "Fast data access over asymmetric channels using fair and secure bandwidth sharing," in *Proceedings of the 26th IEEE International Conference on Distributed Computing Systems (ICDCS '06)*, p. 58, Lisboa, Portugal, July 2006.

[32] B. Pittel, "On spreading a rumor," *SIAM Journal on Applied Mathematics*, vol. 47, no. 1, pp. 213–223, 1987.

[33] Sun-Developer-Network, "Java multimedia framework," http://java.sun.com/products/java-media/jmf/.

[34] R. Karp, C. Schindelhauer, S. Shenker, and B. Vocking, "Randomized rumor spreading," in *Proceedings of the 41st Annual Symposium on Foundations of Computer Science (FOCS '00)*, pp. 565–574, Redondo Beach, Calif, USA, November 2000.

Hindawi Publishing Corporation
Advances in Multimedia
Volume 2007, Article ID 26070, 12 pages
doi:10.1155/2007/26070

Research Article

Enhanced P2P Services Providing Multimedia Content

E. Ardizzone, L. Gatani, M. La Cascia, G. Lo Re, and M. Ortolani

Dipartimento di Ingegneria Informatica (DINFO), Università degli Studi di Palermo, Viale delle Scienze, Ed. 6, 90128 Palermo, Italy

Received 30 January 2007; Accepted 28 August 2007

Recommended by Yi Cui

The retrieval facilities of most Peer-to-Peer (P2P) systems are limited to queries based on unique identifiers or small sets of keywords. Unfortunately, this approach is very inadequate and inefficient when a huge amount of multimedia resources is shared. To address this major limitation, we propose an original image and video sharing system, in which a user is able to interactively search interesting resources by means of content-based image and video retrieval techniques. In order to limit the network traffic load, maximizing the usefulness of each peer contacted in the query process, we also propose the adoption of an adaptive overlay routing algorithm, exploiting compact representations of the multimedia resources shared by each peer. Experimental results confirm the validity of the proposed approach, that is capable of dynamically adapting the network topology to peer interests, on the basis of query interactions among users.

1. INTRODUCTION

Peer-to-Peer (P2P) networks are distributed systems in which each node runs software with equivalent functionality, in order to operate without requiring central coordination [1]. The P2P paradigm has emerged in the past few years, mainly due to file sharing systems such as Napster [2] and Gnutella [3]. In the research community there has been an intense interest in designing and studying P2P systems. Due to their decentralization, these systems promise improved robustness and scalability.

It is well known that consumers are gathering more and more digital multimedia contents. Consumers capture contents using their digital cameras, digital camcorders, and mobile phones; and store them on different devices. They nowadays tend to store their data in such quantity, as it is becoming increasingly difficult for them to manage, to find and, in the end, to enjoy the videos and images they create.

While the amount of multimedia content keeps growing, locating and obtaining the desired resource has become a difficult task. Traditionally, P2P users request resources using keywords, or simply by searching for a specific file name pattern. This approach is insufficient when the collected data are huge or distributed, especially in the P2P environment. Namely, users might annotate the same file with different file names and keywords, making the data location process error-prone and user-dependent; moreover, artificial intelligence technologies are not mature enough to provide a complete automatic annotation solution bridging the semantic meanings and the low-level descriptors.

In this paper, we present a framework for sharing multimedia resources in a P2P network, exploiting an automatic content-based approach. The present work is based on previous research [4], and represents an extended version which includes a more detailed description of the resource selection mechanism, and a wider set of experiments.

With respect to current content-based image retrieval (CBIR) systems, we envisage the potential use of P2P networks in both scattering data storage and distributing workload of feature extraction and indexing. Through the realization of CBIR in P2P networks, enormous image collections can be managed without installing high-end equipment by the exploitation of individual user's contribution. Furthermore, we make use of the computational power of peers for image preprocessing and indexing in addition to data storage.

One of the most challenging problems related to data-sharing P2P systems is content localization. It determines whether the system resources can be efficiently used or not, affecting the scalability and robustness of P2P systems. Therefore, in order to effectively exploit the potential of CBIR in P2P networks, we also propose an adaptive mechanism for query routing that can well balance the storage overhead and the network load.

The rest of the paper is structured as follows. Section 2 presents both a brief review on the existing CBIR techniques, and the video descriptors chosen for representing multimedia resources; Section 3 illustrates the proposed approach to storing, managing, and retrieving multimedia content; some experimental results are presented in Section 4, and finally we conclude with summary remarks in Section 5.

2. RELATED WORK

Although the lookup of multimedia data in P2P networks represents a new, interesting research field known as CBP2PIR (content-based peer-to-peer image retrieval), to the best of our knowledge, only few works exist where this problem is addressed. In this section, we outline some ideas previously presented in relevant literature that are related to the key aspects of our work.

2.1. P2P for multimedia content management

In [5], the Firework Query Model for CBP2PIR is proposed. The main idea is to cluster peers with similar resources, using the set of feature vectors, as signature value of a peer, in order to measure similarity. The Firework Query Model exploits two classes of links (normal random links and privileged attractive links) in order to route queries. A query starts off as a Gnutella-like flooding query. If a peer deems the query too far away from the peers local cluster centroid, it will forward it via a random link, decreasing the query Time-To-Live (TTL). Otherwise, it will process the query, and forward it via all its attractive links without decreasing the TTL.

A similar CBIR scheme for P2P networks, based on compact peer data summaries, is presented in [6]. To obtain the compact representation of a peer's collection, a global clustering of the data is calculated in a distributed manner. After that, each peer publishes how many of its images fall into each cluster. These cluster frequencies are then used by the querying peer to contact only those peers that have the largest number of images present in one cluster given by the query.

In [7], the authors investigate a CBIR system with automated relevance feedback (ARF) using nonlinear Gaussian-shaped radial basis function and semisupervised self-organizing tree map clustering technique. The authors apply the CBIR system over P2P networks by grouping the peers into community neighborhoods according to common interest.

In [8], a different overlay setup technique is introduced, in order to cluster peers according to the semantic and feature-based characteristics of their multimedia content.

Finally, Wu et al. [9] propose a local adaptive routing algorithm that dynamically modify the network topology toward a small-world structure, using a learning scheme similar to that considered in this paper. However, they design their protocol with the aim of supporting an alternative model for peer-based Web search, where the scalability limitations of centralized search engines can be overcome via distributed Web crawling and searching.

Our work belongs to the same research area of the abovementioned proposals, but introduces a novel combination of video indexing and retrieval techniques, with a P2P adaptive routing strategy capable of leading to dynamically emerging small-world network communities.

2.2. Extracting information from multimedia content

Representing and describing digital image and video content has been an area of active research since mid 1990s, when this topic came into strong attention of both research and industry people working in the developing areas of CBIR, digital libraries, image and video coding, web query engines, and so on. The nature of the problem is well known: visual information is provided with intrinsic semantics that is very hard or impossible to make explicit by a manual description, such as a list of keywords or a full-text description. Thus, a number of content-based algorithms and techniques have been developed in the last decade (see [10] for a review) to make possible the automatic or semiautomatic extraction of low- or medium-level feature descriptors from images and videos, mainly in the areas of indexing and retrieval of image and video databases and of video coding.

CBIR video representation may be based on the decomposition of the video sequence into short video units named shots [11] or, more recently, into video objects [12]. Shot clustering has been proposed to represent video scenes. A shot cluster is considered a scene; a hierarchical scene transition graph, that is, a collection of directed graphs may be then used to model the whole video structure.

Simple shot content representations may be obtained through the description of few representative frames. A limited number of frames (said r-frames or key-frames) are selected from each shot, and each r-frame is therefore described in terms of its visual content, for example, through color and texture descriptors. Thus the description problem is reconducted to the extraction of static descriptors [11]. Motion activity may be also taken into account, for example, by computing motion features related to short sequences in which r-frames are embedded. In this way, a dynamic description of the r-frame may also be obtained.

The r-frame selection and the computation of visual and motion features may be performed in a number of ways. In the following, we will refer to images to indicate either single images, in the case of still image applications, or representative frames, that is, frames representing a subpart of a video sequence in the case of video applications.

In order to correctly classify and represent video contents it is fundamental to discover similarities in the extracted images; and the known useful features for this purpose are color and texture. In the last years, several color-based techniques have been proposed for video annotation (e.g., region-based dominant color descriptors [13], multiresolution histograms [14], and vector-quantized color histograms [15]). Several texture descriptors have also been proposed that try to mimic the human similarity concept, but they are normally useful only in classifying homogeneous texture. Generic images usually contain different kinds of texture, so that a global

texture descriptor hardly may describe the content of the whole image.

In this paper, we adopted HSV color histogram, edge density, motion magnitude, and motion direction histogram to represent r-frames. These features have been chosen as to lead to satisfactory results in previous work [16].

3. THE PROPOSED APPROACH

The technique we propose in this paper tries to improve the scalability and efficiency of the resource discovery in the unstructured P2P environment, through an adaptive routing algorithm which suppresses flooding.

Queries issued by a user are routed to neighbor peers in the overlay network, in order to find resources that satisfy them. Initially the network has a random, unstructured topology (each peer is assigned a number of neighbors randomly chosen), and queries are forwarded as in the scoped flood model. However, we adopt an approach that dynamically selects the neighbors to which a query has to be sent or forwarded. The selection process is performed with the aim to detect peers that with high probability share resources satisfying the query. The selection is driven by an adaptive learning algorithm by which each peer exploits the results of previous interactions with its neighbors to build and refine a model (profile) of the other peers, describing their interests and contents. Each peer is characterized by one (or several) general interest and shares resources according to its interest. The characteristics of each peer are summarized in a peer profile. When an agent needs to forward a query, it compares the query with its known profiles, in order to rank all known peers and select the best suited to return good response. The network topology (i.e., the actual set of peers that are neighbors in the overlay) is then dynamically modified on the basis of the learned contexts and the current information needs, and the query is consequently routed according to the predicted match with other peers' resources.

3.1. Multimedia content representation

As outlined in Section 2, the choice of r-frames is a crucial task in automatic video annotation. In this work, we adopted a simplified technique of nonlinear time sampling, based on the comparison of a cumulative difference of frame brightness values with a threshold, whose value has to be tuned experimentally [16].

In our approach, a video descriptor is structured in a hierarchical way. At the highest level, this descriptor simply consists of references to the shot descriptor for each shot belonging to the video; each of those shot descriptors, in turn, consists of: (i) the shot duration (in seconds), (ii) the number of r-frames contained in the shot, and (iii) a reference to the r-frame descriptor for each r-frame belonging to the shot. R-frames (or still images) are globally described, and the relative visual descriptor consists of attributes of both static and motion-based kind. The former kind is based on texture and color, whereas the latter is based on the optical flow field of the r-frame; their computation involves considering a few frames before and after the r-frame.

A simple but effective method [16], based on a 3-dimensional quantized color histogram in the "Hue-Saturation-Value" (HSV) color space and a Euclidean metric, is used here to compare the query image to images contained in the database. The HSV quantization needed to compute a discrete color histogram is done taking into account that hue is the perceptually most significant feature. Thus a finest quantization has been used for hue, allowing for 18 steps, whilst only 3 levels are allowed for saturation and value. In such a way, we obtain a 162 ($18 \times 3 \times 3$) bins HSV histogram, that may be easily represented by a 162×1 vector.

The texture features we propose are related to coarseness, directionality, and position of texture within the image. All these features are based on edge density measures. Edge density is directly related to coarseness, directionality is addressed by repeating the edge measure for different directions, and spatial position is taken into account by a simple partitioning of the r-frame. In particular, we first subdivide the r-frame into four equal regions. For each region, we compute the edge maps through directional masks, respectively, aligned along the directions 0, 45, 90, and 135 degrees. Values of edge map exceeding a fixed threshold are considered edge pixels. The threshold value has been determined experimentally. The ratio between the number of edge pixels and the total number of pixels is the edge density. Since we determine 4 edge density values for each region, we have a 16×1 texture-based vector.

Motion-based descriptors are based on the optical flow field [17] of the r-frame, and their computation involves considering a few frames before and after the r-frame. We used a gradient-based technique and the second-order derivatives to measure optical flow [16]. The basic measurements are integrated using a global smoothness constraint. This technique allows to obtain a dense and sufficiently precise flow field at a reasonable computational cost. Once the optical flow field is computed, we need a method able to code the associated information in a form adequate for content description. First, we segment the flow field into four equal region; for each region we then compute motion based features. The splitting was performed to preserve spatially related information that are not integrated in the computed features. In conclusion, the adopted motion descriptors are a measure of the average motion magnitude in the considered region, and a normalized 18 bins histogram of motion vectors directions.

In summary, the visual descriptor of an r-frame, computed automatically by the system, is a 254-dimensional vector $\underline{x} = [\,\underline{c}\ \underline{t}\ \underline{m}\ \underline{d}\,]$ where $\underline{c}$ is a 162-dimensional vector representing the global HSV color histogram and $\underline{t} = [\,\underline{t_{tl}}\ \underline{t_{tr}}\ \underline{t_{bl}}\ \underline{t_{br}}\,]$ is a 16-dimensional vector representing the edge density computed, respectively, over the top-left, top-right, bottom-left, and bottom-right quadrants of the r-frame. Similarly, $\underline{m} = [\,m_{tl}\ m_{tr}\ m_{bl}\ m_{br}\,]$ and $\underline{d} = [\,\underline{d_{tl}}\ \underline{d_{tr}}\ \underline{d_{bl}}\ \underline{d_{br}}\,]$ are a 4-dimensional vector and a 72-dimensional vector containing, respectively, the average motion magnitudes and the 18 bins motion vectors direction histograms computed over the four regions as above.

3.2. The adaptive search algorithm

Since our goal is to allow peers to form communities in a fully distributed way, they should find new peers and evaluate their quality in relation to their own interests. In our system, we follow an approach similar to the one presented in [9]. When a peer enters the network for the first time, the bootstrap protocol returns the address of some existing peers to get started. The new peer can then discover other nodes through these known peers. In the proposed system, a peer would discover new peers through its current neighbors, during the normal handling of queries and responses. Each peer maintains a fixed number of slots for known peers. This number can vary among peers depending on their available memory (a peer must properly prune other peers' information when needed). For each known peer, a profile which concisely describes the shared resources is stored. The actual set of neighbors, that is, those to whom queries are sent, is selected dynamically for each query at time step t among all the known peers. In particular, when a peer receives a locally generated query, it compares the query with its stored profiles. Each peer applies a simple ranking algorithm for dynamically selecting peers to which query must be sent. The maximum number of selected peers depends on peer bandwidth and computational power to process neighbor data. Network connectivity is not negatively affected by dynamic neighbor selection, since each peer also stores references to other potential neighbors that may be used to replace previously selected ones, in case they are no longer available; nevertheless this is not per se a guarantee against network partitioning.

The selection mechanism tends to reinforce connections among peers with similar content; such nodes are thus clustered together in the overlay network, whereas fewer intercluster connections are maintained. This scenario resembles the typical small-world configuration [18], where each node in a network is reachable in a limited number of hops; namely, peers that share common interests should be directly connected, whereas peers sending queries out of their interest area will have their messages traverse several clusters before reaching the potential recipient. This is favorable in terms of effectiveness of the search procedure, but is known to be prone to network partitioning as consequence of the potential disconnection of one of the peers that act as hubs among clusters. A peer might thus not be able to get a successful query hit although the desired content is in fact available in the network, because it may be unreachable due to partitioning.

We propose here a heuristic for lessening the impact of this issue. Besides relying on the neighbors whose profiles show a commonality of interests, peers participating to our framework will additionally choose to forward queries to randomly selected nodes according to a small prefixed probability P_1. In order to do this, they must have previously stored profiles of all peers that they may have come in contact with, regardless of the similarity metric; this mechanism will guarantee higher variance to the profile list by allowing also "unlikely" peers to be selected once in a while. Moreover if, during the query forwarding step, the ranking procedure cannot return any peer with a sufficiently high similarity measure (according to a predefined threshold), a peer will again choose a random neighbor, with a probability P_2, where $P_2 > P_1$.

Although no strict guarantee can be provided against network partitioning, careful choice of the above-mentioned thresholds will make this phenomenon unlikely to occur.

Each peer profile maintains a concise representation of the shared resources, by the adoption of different techniques for textual and visual contents. In particular, the system adopts a simple taxonomy and Bloom filters [19] to build a binary vector that represents the textual contents. As regards visual resources, after the meaningful features have been extracted from the image database, each peer will work on extracting representative information that may succinctly describe its whole content. Finally, the set of cluster representatives may be used as a sort of "signature" for the content of each peer and we use their vectorial representation as reported at the end of Section 3.1.

Our system supports a basic query language (where a query string is interpreted as a conjunction of keys) for textual information retrieval, while a standard "query-by-example" approach is exploited to search the image database.When asked with a query, the system looks up the information in its profile database (using the selection mechanism described in Section 3.3) in order to obtain a list of candidate peers that might store data matching the query. When a peer receives a query from another peer, it checks its local repository in order to locate the resources that better match with the desired content. In particular, textual resources are searched using a standard keyword-based technique, while the visual similarity between two images is computed by means of a weighted sum of normalized Euclidean distances, as already presented in [16]. In order to normalize the distances, we estimate a probability distribution for the Euclidean distances of each visual feature (color, texture, motion), comparing each r-frame in a training database with all the others. These distributions are then used to normalize all the distances to the range [0,1]. The normalization is needed to make the use of a weighted sum of distances meaningful. Furthermore, a peer that has received a query can forward it to those neighbors whose profiles match the query. To this aim, the peer uses the same selection algorithm applied to locally generated queries (note that the peer automatically excludes both the peer that has forwarded the query, and the peer that has generated the query). In order to prevent potential DoS attacks which exploit the response system, we impose that a peer replies to a forwarded query sending the response to the neighbor that has forwarded the query, and not directly to the originating peer. To limit congestion and loops in the network, queries contain a TTL, which is decreased at each forward, and queries will not be forwarded when TTL reaches 0. When a peer receives the responses for a locally generated query, it can start the actual resource downloading. Moreover, if a peer that has sent a response is not yet included in the list of known peers, a profile request is generated. For this request, the two peers contact each other directly. When the message containing the profile will arrive, the new peer will be inserted among the set of

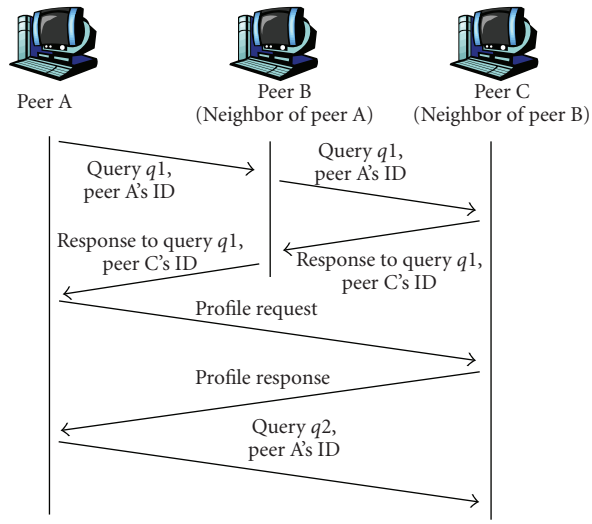


Figure 1: The process of neighbor discovery.

known peers and its features will be considered in order to select actual neighbors for the following queries (see Figure 1). The maintained profiles are continuously updated according to the peer interactions during the normal system functioning (i.e., matches between queries and responses). Moreover, a peer can directly request a more up-to-date profile if necessary.

3.3. The selection mechanism

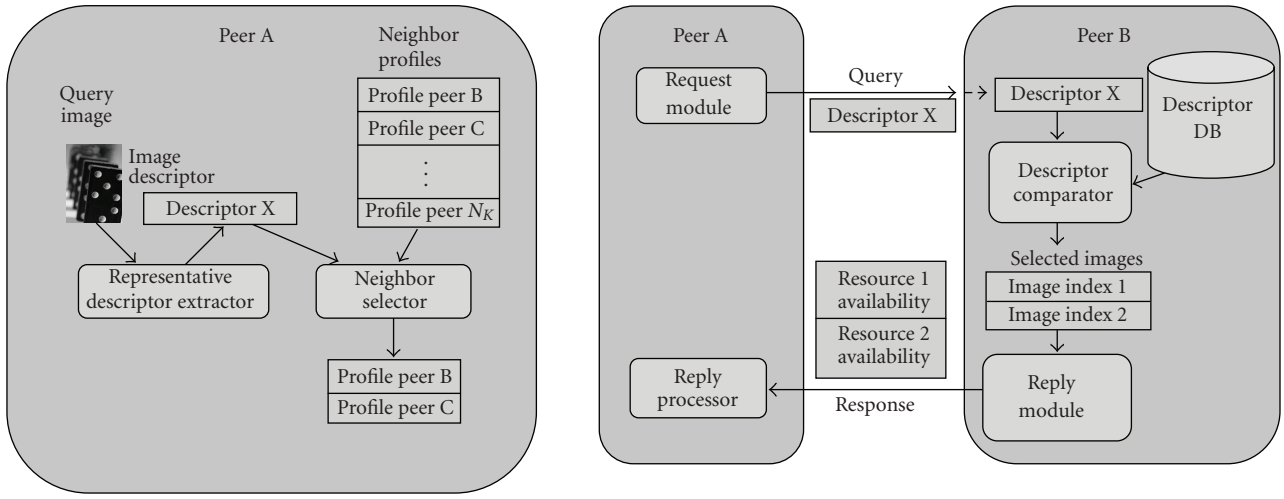
As already mentioned, our system supports different query languages for textual information retrieval, and for image and video retrieval, respectively. In particular, for the latter kind of resources we employ a query-by-example paradigm for color and texture features and a direct query paradigm for motion features. In practice, the user presents to the system an image similar to the one she is looking for and, in the case of video query, provides information about motion (e.g., a zoom-in shot has a particular motion direction histogram, an almost-static shot has a very low motion magnitude, etc.). The technique we used for querying has already been presented in [16] and makes use of a weighted sum of normalized Euclidean distances. More specifically, in order to normalize those distances, we estimate a probability distribution for the Euclidean distances of each visual feature (color, texture, motion), comparing each r-frame in a training database with all the others. These distributions are then used to normalize all the distances to the range [0-1].

The similarity between the current query and the general interests of each peer is also managed differently on the basis of the kind of searched resource. Similarity between textual resources (as well as textual annotations and high-level descriptors associated to multimedia resources) is evaluated exploiting a standard technique for textual retrieval. As regards visual resources, the peer computes the distance to each cluster representative and chooses the closest ones as possible matches. Furthermore, if the resources are opportunely indexed, the system can also exploit the representation of the resources by means of Bloom filters [19] which are maintained into the peer profiles. This way, it is possible to check, with high probability, if a given resource belongs to the resource set shared by a peer. This approach enhances the topological properties of the emergent overlay network and it is very useful in those applications where resources are uniquely characterized by an identifier or are semantically annotated.

The selection mechanism takes primarily into account the experience that peers acquire during their normal interactions: each newly available piece of information is opportunely elaborated and exploited to enrich the system knowledge. Each peer profile maintains a concise representation of the shared resources, by the adoption of different techniques for textual and visual contents. In particular, the system adopts simple taxonomies and Bloom filters to build a binary vector that represents the textual contents. As regards visual resources, after the meaningful features have been extracted from the image database as described in Section 2.2, each peer will work on extracting representative information that may succinctly describe its whole content.

Once the contents have been properly represented as vectors in the search space, our implementation makes use of clustering techniques from literature through which each peer will roughly partition its data space into separate regions that represent different groups of related images. Specifically, we employ the well-known k-means clustering method presented for the first time in [20], whose underlying idea consists in assigning each data element to one cluster by means of a similarity function, which is often based on the Euclidean distance as a metric; each cluster is then represented by a prototype, or cluster representative, typically computed as the cluster centroid. The basic formulation of the algorithm assumes that the number of clusters is known in advance, which may be a too tight constraint for our present scenario; however, this requirement may be partially loosened with the use of controlled iterations and of a cluster validity assessment technique [21, 22]. Furthermore, in order to cope with the stream of continuously incoming data, we use a variation on the basic k-means algorithm that allows online updating of the computed clusters. Without delving into too much detail, our method builds upon the ideas presented in [23] and generalizes them from the binary case to multiclass classification. Finally, the set of cluster representatives may be used as a sort of "signature" for the content of each peer and will be spread to all nodes of the P2P network in order for them to be able to perform their searches. Another node that is presented with a request for a new element and needs to find out a list of possible owner candidates will compute the distance to each cluster representative and choose the closest ones as possible matches. It is worth noting that, while all processing is performed locally, manipulated objects exist in a globally defined vector space; hence all feature vectors, as well as all cluster centroids, are globally comparable; however, clusters are not required to have a global semantic validity as they are only used to compute relative distances.

After the peer has extracted representative descriptors for the content to be searched, it will compare them with its neighbors' profiles in order to find the best matches.



(a) The local selection process

(b) Resource selection on remote node through descriptors

Figure 2: The resource selection mechanism.

Table 1: Selection criteria.

Parameter	Description	Weight
$\overline{R}_t$	Current estimate of the contact	α
R_{t-1}	Old reliability value of the contact (according to past history)	$1-\alpha$
R_t	New reliability value used to rank contacts	—
J	Percentage of contact interests with respect to query topics	β
S	Percentage of successes provided by the contact	γ
B	Result of membership test (produced by Bloom filter), if applicable	δ
Q	Capability summarization of the contact (bandwidth, CPU, storage, etc.)	ϵ
C	Connection characteristic summarization of the contact	ζ

A basic criterion for conveniently selecting peers satisfying a given request would exploit the experience of past interactions among peers, thus giving a good indication about the probability that a contact could directly provide the resources searched. We enhance this criterion by adopting a further mechanism capable of singling out peers that, although not directly owning the desired resources, can provide good references to the resource owners. It is worth noting that while the first criterion, based on the commonality of interests, tries to increase the overlay network clusterization by the creation of intracluster links, the second one typically sets links between different clusters, providing a quick access to peers that are close to several resources. Furthermore, the selection mechanism considers some additional criteria, in terms of peer capabilities (bandwidth, CPU, storage, etc.) and end-to-end latency, in order to take into account the topological characteristics of the peer community (thus reducing the mismatch between the overlay and the real topology). Figure 2 visually describes the selection process, showing both the operations performed locally by a peer, and the interactions between that peer and a possible selected neighbor; this selected node would be the one that provides proper references in order to decide on the eventual request of resources.

In order to get a deeper understanding of the process just outlined, we can consider a formalized version. Each peer stores a parameter, R, associated to each of its contacts that provides a measure of that contact's reliability. The parameter value is related to the interactions in the peer community and it changes according to the criteria previously described (see also Table 1). Each single criterion gives a partial value. These partial values are then jointly considered by means of a weighted average (1) that produces an estimate of the overall reliability for the situation at the current timestep t. Indicating this estimate by $\overline{R}_t$, this yields:

$$\overline{R}_t = \beta \cdot J + \gamma \cdot S + \delta \cdot B + \epsilon \cdot Q + \zeta \cdot (1 - C), \tag{1}$$

where

$$\begin{aligned} \beta + \gamma + \delta + \epsilon + \zeta &= 1, \\ 0 \le \beta, \gamma, \delta, \epsilon, \zeta &\le 1. \end{aligned} \tag{2}$$

The current estimate $\overline{R}_t$ is finally combined with the old estimate R_{t-1}, generating the new value R_t for the reliability parameter. In order to smooth the results of the selection process, a kind of time window is employed to balance new information against past experience. The new estimate is then formally computed by the formula:

$$R_t = \alpha \cdot \overline{R}_t + (1 - \alpha) \cdot R_{t-1}, \tag{3}$$

where

$$0 \leq \alpha \leq 1, \quad \alpha \ll (1-\alpha). \tag{4}$$

The R_t value is then exploited to rank all the known peers, according to the estimated reliability. In order to establish a balance between the exploration and exploitation of the search space, the algorithm in the early steps can select peers different from the top ones. This random search behavior is characterized by a probability of choosing no optimal contacts:

$$P = \exp\left(\frac{\delta R_t}{T}\right), \tag{5}$$

where, using an approach similar to that adopted in the "simulated annealing" searching technique [24], δR_t represents the decrease in the reliability value, and the "temperature" T is a control parameter.

4. EXPERIMENTAL EVALUATION

The experimental results presented in this section are aimed to validate the two main different aspects dealt with in the paper: namely, the adaptive routing mechanism adopted by the P2P protocol and the image video descriptors proposed for multimedia content retrieval.

Regarding the first topic, the key idea of our approach is that a clustered overlay topology (where peers with shared interests and domains tend to form strongly connected communities) can spontaneously emerge by means of an intelligent peer collaboration. In particular, the adoption of our adaptive mechanism, based on a reinforcement learning scheme, should better cope with highly dynamic P2P communities, leading to topologies with small-world properties [18]. In such a topology, a flood-based routing mechanism (with limited scope) is well suited, since it allows any two peers to reach each other via a short path, while granting higher communication efficiency within clustered peer communities. Furthermore, the proposed approach should take advantage from the adaptive overlay rearrangement, in order to well cope with high node volatility and massive node disconnections.

Regarding the issue of multimedia content representation, we aim to validate the effectiveness of the adopted visual descriptors, in order to confirm the expected retrieval capabilities of our approach. In particular, the experimental evaluation should prove that the visual descriptors we adopted can encode the visual content reasonably well.

4.1. Experimental settings

We decided to use simulation to investigate the properties of the proposed approach. This is mainly due to the difficulty to obtain a precise and comprehensive snapshot of actually deployed P2P networks [25–27], whose behavior is complicated by the complex dynamics in overlay connections and peer lifetimes. Simulation of P2P networks can instead provide a straightforward and effective tool to conduct thorough analysis of their characteristics. In order to study the behavior of peer interactions in our system, we designed and implemented a simulator (see, also, [28]) that allows to model synthetic peer topologies, the shared resources, and the issued queries. The main goal of our simulator is to analyze the effectiveness of our routing protocol and the topology properties of emergent peer networks. As observed in [29], unstructured P2P systems are characterized by high temporal locality of queries (i.e., with high probability a single peer issues similar queries over time). Therefore, in the simulations carried out, each peer belongs to one or more groups of interest, according to the topics of owned resources and issued query. In order to better investigate how the adaptive mechanism proposed can support efficient resource searching, we consider that each peer generates queries belonging, with high probability, to one of the topics (however, a smaller number of queries is generated on a randomly selected topics). It is also worth noting that peers can have interests that partially overlap each other, and that each resource can be replicated on several peers.

Let $G = (V, E)$ denote a connected graph modeling a communications network, $N = |V|$ the cardinality of the set of vertices, and $d(i, j)$ the length (in hops) of the shortest path between two vertices i and j in V. For the experimental analysis of emergent topological properties, we consider two network metrics, the clustering coefficient, $C(G)$, and the characteristic path length, $L(G)$, that well characterize the topological properties of dynamic networks.

The characteristic path length $L(G)$ is defined as the number of edges in the shortest path between two vertices, averaged overall pairs of vertices. To define the clustering coefficient $C(G)$, suppose that a vertex $v \in V$ has k_v neighbors; then at most $k_v(k_v - 1)/2$ edges can exist between them (this occurs when every neighbor of v is connected at every other neighbor of v). Let C_v, the local clustering coefficient of v, denote the fraction of these allowable edges that actually exist. The (overall) clustering coefficient is then defined as the average of C_v over all v. While L measures the typical separation between two vertices in the graph (a global property), C measures the cliquishness (degree of compactness) of a typical neighborhood (a local property). Since in our simulations it is possible that the network is not always strongly connected, we adopt an alternative definition ($L'(G)$) for the characteristic path length, using the harmonic mean of shortest paths that can be computed irrespective of whether the network is connected:

$$L'(G) = \left(\frac{N}{N-1}\sum_{i,j\in V} d(i,j)^{-1}\right)^{-1}. \tag{6}$$

We also compute the ratio C/L' that gives a good insight of the overall topological properties: high values are associated with networks that present both a strong clusterization, and a low average separation between nodes. To compute these metrics, the simulator takes a snapshot of the network for each time step: C and L' are then computed in the directed graph which models the overlay network, based on each peer's actual neighbors.

Furthermore, in order to quantify the efficiency of the approach proposed, three further metrics are adopted: the query hit rate, HR, that represents the percentage of queries successfully replied, the query coverage rate CR, that

represents the average number of nodes reached by a query, and the node message-load ML, that represents the average number of messages that a node has to process during a single time step.

Finally, to assess the effectiveness of the adopted multimedia content descriptor in the context of content-based retrieval, we use a normalized version of precision and recall that embodies the position in which relevant items appear in the retrieval list [30]. All the tests are performed using a database containing about 1500 r-frames obtained from about 500 shots. We consider 20 r-frames randomically chosen and we evaluate for each one of them the system response to a query by example. Recall and precision measurements require to determine which r-frames are relevant with respect to a posed query, but stating relevance is a very subjective task as also noted in [31]. To overcome this problem we adopt a subjective criterion: *candidate-to-relevance* r-frames for each query are determined by four different people (not including the authors) and a r-frame is considered as relevant if at least three people chose it. Once known the *correct* query result we are able to evaluate performances. Fixed to n the number of images to be retrieved, for each query we perform the following measures:

(i) AVRR, the average rank of all relevant, retrieved images;
(ii) IAVRR, the ideal average rank, that is, when all the relevant images are ranked at the top;
(iii) MT, the number of relevant images that are missed;
(iv) AVRR/IAVRR

In particular, let I denote the number of relevant images among the n retrieved, ρ_r the rank of the rth relevant retrieved image, and T the total number of images relevant to the posed query. Then we can state the following definitions:

$$\text{AVRR} = \frac{1}{I}\sum_{r=1}^{I}\rho_r; \qquad \text{IAVRR} = \frac{T}{2}; \qquad \text{MT} = \frac{I}{T}. \tag{7}$$

Note that these measures depend on n, the number of retrieved images: in our experiments we perform each query twice, one for $n = 32$ and one for $n = 64$.

The visual descriptors we adopted, despite their compactness and the availability of simple algorithms to compute, have been proved to encode the visual content reasonably well. In particular, previous experiments [16] showed that visual descriptors are adequate in most cases if the image collections are not too large (less than 10,000 images). For larger image collection, when query results obtained using only visual descriptor tends to become unreliable, the use of textual information greatly improve the results.

Further investigation on our CBP2PIR system using both textual, and visual data showed very promising retrieval capability, confirming the feasibility of our searching method for feature vectors derived from multimedia resources.

4.2. Experimental results

In order to thoroughly evaluate the proposed approach, we preliminarily investigated the effectiveness of the techniques adopted for multimedia content representation. Then, we performed extensive simulations on the P2P routing algorithm, considering several scenarios, each of them characterized by the variation of one significant simulation parameter. For each simulation, the aim is to study how network properties and searching performance change when the parameter value is varied. Finally, we analyzed the impact of dynamic changes in the peer communities, in order to test the robustness of the algorithm against such events. Since the initial random topology can affect the final results, for each simulation, we perform several independent simulations, averaging across all the results.

Table 2: Average query results for twenty test queries by color and texture, with $n = 32$ and $n = 64$.

	Metric	Color	Texture
$n = 32$	AVRR	6.935	11.124
	IAVRR	5.501	10.836
	AVRR/IAVRR	1.270	1.063
	MT	0.854	0.474
$n = 64$	AVRR	10.074	21.996
	IAVRR	5.507	10.834
	AVRR/IAVRR	1.736	2.083
	MT	0.912	0.709

4.2.1. Studying the effectiveness of multimedia content descriptors

In order to assess the retrieval capabilities of the visual descriptors, we carried out several experiments, under different assumptions. As already mentioned, extensive results evaluating the retrieval performance based only on visual information have been reported [16]; in this paper we present some recall and precision measurements for the improved CBP2PIR framework. In particular, we report in Table 2 the average values of AVRR, IAVRR, AVRR/IAVRR, and MT related to 20 test queries (both color-based, and texture-based), respectively for $n = 32$ and $n = 64$. Results show that color indexing exhibits a distinctly good behavior.

4.2.2. Studying the P2P searching mechanism

For the sake of brevity, we can only present some representative results; a detailed performance evaluation of the proposed searching approach can be found in [28], confirming the idea that adaptive routing can properly work and that small-world network topologies can emerge spontaneously from local interactions between peers, structuring the overlay in such a way that it is possible both to locate information stored at any random node by only a small number of hops and to find quality resources quickly and even under heavy demands. In order to illustrate how the proposed approach can significantly improve searching performance in P2P communities, we only report in Table 3 a comparison between the results obtained in the same experimental conditions by the adoption, respectively, of the algorithm proposed, a flooding scheme, and a random-walk searching

Table 3: Comparison between adaptive routing, flooding, and random-walk.

Searching scheme	*Sim*	*C*	*L′*	*C/L′*	HR (%)	CR	ML
Adaptive routing	Sim1	0.184750	7.442001	0.024825	0.615112	28.193076	0.252483
	Sim2	0.185430	7.224156	0.025668	0.637082	28.308287	0.274244
	Sim3	0.188139	7.734175	0.024326	0.610095	27.471692	0.224489
Flooding	Sim1	0.004210	5.397369	0.000780	0.472454	22.459130	0.226549
	Sim2	0.004732	5.417398	0.000873	0.439865	22.828022	0.262706
	Sim3	0.003982	5.398215	0.000737	0.457117	22.408658	0.215307
Random-walk	Sim1	0.002037	69.540365	0.000029	0.481486	33.870043	0.703346
	Sim2	0.002096	69.670214	0.000030	0.486615	32.933378	0.805104
	Sim3	0.002148	70.736614	0.000030	0.480732	32.168622	1.019697

Table 4: Evolution of topological metrics.

		Initial topology			Final topology		
Searching scheme	*Sim*	*C*	*L′*	*C/L′*	*C*	*L′*	*C/L′*
Adaptive routing	Sim1	0.004210	5.397369	0.000780	0.184750	7.442001	0.024825
	Sim2	0.004732	5.417398	0.000873	0.185430	7.224156	0.025668
	Sim3	0.003982	5.398215	0.000737	0.188139	7.734175	0.024326
Flooding	Sim1	0.004210	5.397369	0.000780	0.004210	5.397369	0.000780
	Sim2	0.004732	5.417398	0.000873	0.004732	5.417398	0.000873
	Sim3	0.003982	5.398215	0.000737	0.003982	5.398215	0.000737

mechanism. All the values related to clustering coefficient and characteristics path length are measured at the final time step of the simulation. It is worth noting that the adaptive algorithm performs significantly better both in term of success rate, and as regard the message load imposed on the network. Table 4 shows the C and L' values both at the beginning, and at the end of the simulation, for the algorithm proposed and for flood-based search. It is straightforward to note that while flooding does not affect the topological properties, the approach proposed, although it starts from the same initial graph, achieves a final configuration where the topological metrics assume values that are typical of small-world networks.

4.2.3. *Studying the impact of dynamic changes in the peer community*

We analyze here the behavior of the system when the peer community is subject to a large amount of connection and disconnection events. All the simulations are carried out starting from the same initial conditions (considering 5000 timesteps for each execution). Since the considered scenarios are qualified by the event sequences, which depend from the adopted event models, in the following we present the results according to this categorization. Simulation results in Figure 3 are characterized both by a dynamic community, and a fixed number of participants; the peer number is maintained constant, since each connection is associated to a corresponding disconnection, according to a Pareto distribution for the model of events.

The following simulations consider a dynamic community with a variable number of participants. As reported in Table 5, six different cases are considered, each characterized by a synthetically predefined sequence of events: we believe that these choices well describe the plausible evolutions of P2P communities. For each sequence, the number of peers at the initial and final steps and the total amount used in each simulation are also reported in Table 5.

For the sake of brevity, we only report the plots relative to the two most challenging cases, corresponding to the 3rd and 6th event sequence. A very high activity distinguishes the third synthetic sequence (see Figure 4), where many new connections and disconnections raise the peer expansion to the 65% of their initial value. Similarly, a relevant variability qualifies the sixth sequence (see Figure 5) which is also characterized by an unbalanced ratio between few peer additions and many disconnections. This behavior leads to a 37% reduction of the initial peer population.

As all the gathered data show, both algorithm properties (capability of evolving toward a small-world topology and efficiency) are minimally affected by the dynamic events, also when those involve large portions of the peer community (see Figures 4 and 5), thus confirming the robustness of the algorithm.

5. CONCLUSION

This paper presented a CBIR approach to information retrieval in P2P networks, that relies on an adaptive technique for routing queries and is specifically targeted to multimedia content search. The main motivation behind our work is that the huge amounts of data, their peculiar nature, and, finally, the lack of a centralized index make it particularly difficult to pursue the goal of efficiency in this kind of systems. Our approach employs a decentralized architecture which fully exploits the storage and computation capability of computers

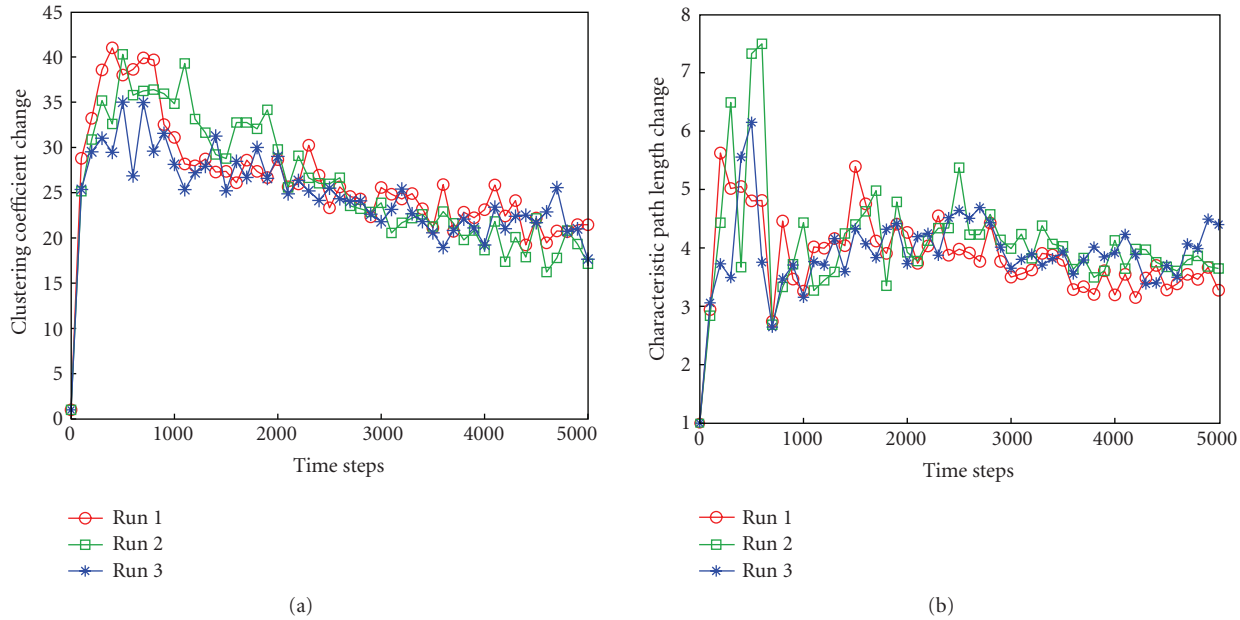


Figure 3: Relative variation of topological metrics using Pareto event model.

Table 5: Synthetic sequences for dynamic evolution of P2P communities.

Seq.	Initial peer no.	Final peer no.	Total peer no.	Kind of dynamic operations	Degree of dynamic evolution
1st	500	550	550	Only connections	Low increase
2nd	500	575	611	Connections and disconnections	Low increase
3rd	500	825	890	Connections and disconnections	High increase
4th	500	450	500	Only disconnections	Low decrease
5th	500	445	575	Connections and disconnections	Low decrease
6th	500	315	530	Connections and disconnections	High decrease

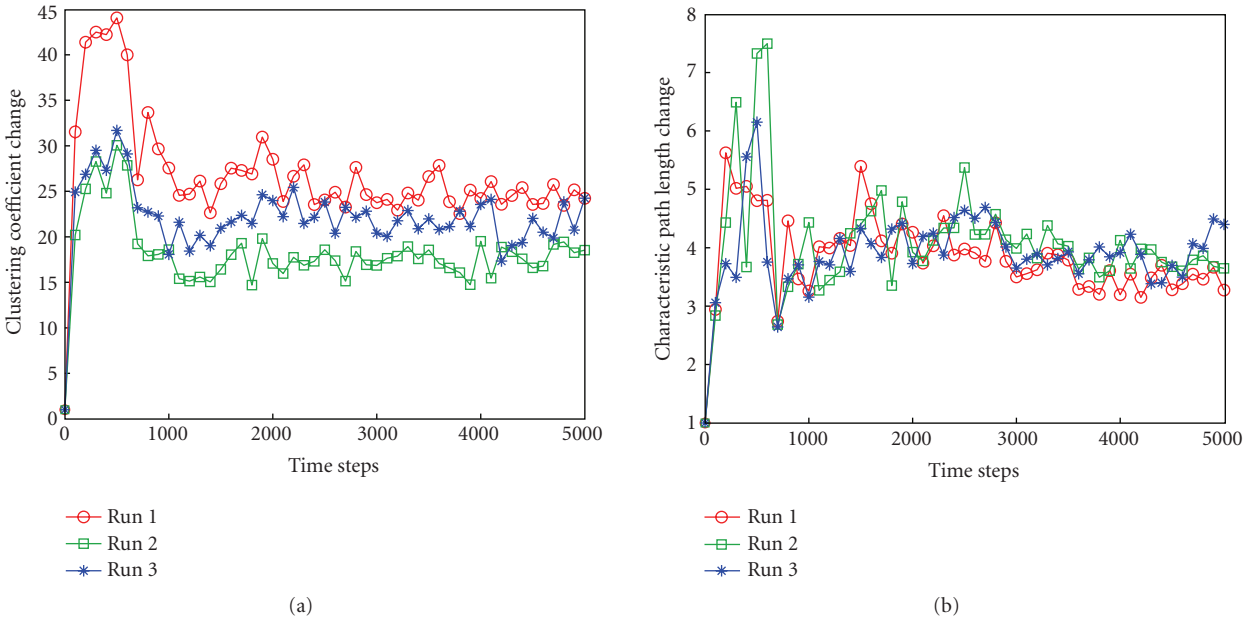


Figure 4: Relative variation of topological metrics using the 3rd sequence of events.

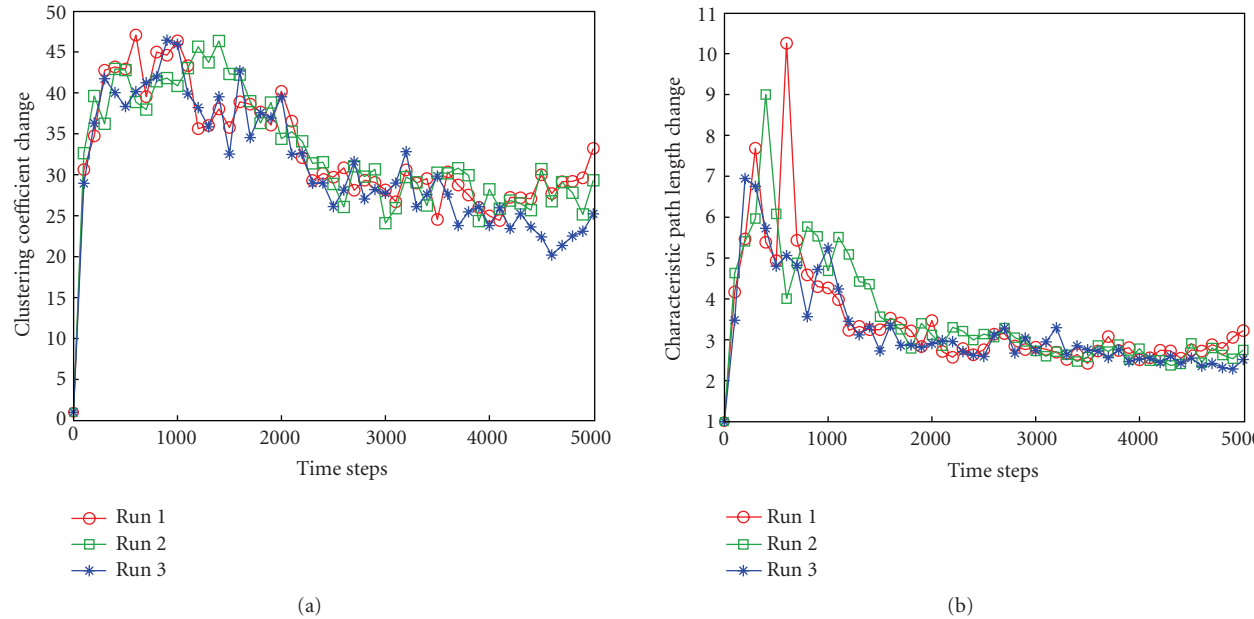


FIGURE 5: Relative variation of topological metrics using the 6th sequence of events.

in the Internet and broadcasts queries throughout the network using an adaptive routing strategy that dynamically performs local topology adaptations. Modifications in the routing structure are driven by query interactions among neighbors in order to spontaneously create communities of peers that share similar interests; moreover, a small-world network structure can emerge spontaneously thanks to those local interactions. Network traffic cost, and the query efficiency are thus significantly improved as is confirmed by our preliminary experiments.

REFERENCES

[1] D. Milojicic, V. Kalogeraki, R. Lukose, et al., "Peer-to-peer computing," Tech. Rep. HPL-2002-57, HP Labs, Palo Alto, Calif, USA, 2002.

[2] C-Net News, "Napster among fastest-growing Net technologies," 2000.

[3] Limewire, "The Gnutella protocol specification (ver. 0.4)," http://www9.limewire.com/developer/gnutella_protocol_0.4.pdf, 2001.

[4] E. Ardizzone, L. Gatani, M. La Cascia, G. Lo Re, and M. Ortolani, "Enhanced P2P services providing multimedia content," in *Proceedings of the 8th IEEE International Symposium on Multimedia (ISM '06)*, pp. 637–646, San Diego, Calif, USA, December 2006.

[5] I. King, C. H. Ng, and K. C. Sia, "Distributed content-based visual information retrieval system on peer-to-peer networks," *ACM Transactions on Information Systems*, vol. 22, no. 3, pp. 477–501, 2004.

[6] W. Müller and A. Henrich, "Fast retrieval of high-dimensional feature vectors in P2P networks using compact peer data summaries," in *Proceedings of the 5th ACM SIGMM International Workshop on Multimedia Information Retrieval (MIR '03)*, pp. 79–86, Berkeley, Calif, USA, November 2003.

[7] I. Lee and L. Guan, "Content-based image retrieval with automated relevance feedback over distributed peer-to-peer network," in *Proceedings of the IEEE International Symposium on Circuits and Systems (ISCAS '04)*, vol. 2, pp. 5–8, Vancouver, British Columbia, Canada, May 2004.

[8] M. Kacimi, K. Yétongnon, Y. Ma, and R. Chbeir, "HON-P2P: a cluster-based hybrid overlay network for multimedia object management," in *Proceedings of the 11th International Conference on Parallel and Distributed Systems (ICPADS '05)*, vol. 1, pp. 578–584, Fukuoka, Japan, July 2005.

[9] L.-S. Wu, R. Akavipat, and F. Menczer, "6S: distributing crawling and searching across Web peers," in *Proceedings of the IASTED International Conference on Web Technologies, Applications, and Services (WTAS '05)*, pp. 159–164, Calgary, Alberta, Canada, July 2005.

[10] R. Datta, J. Li, and J. Z. Wang, "Content-based image retrieval: approaches and trends of the new age," in *Proceedings of the 7th ACM SIGMM International Workshop on Multimedia Information Retrieval (MIR '05)*, pp. 253–262, Singapore, November 2005.

[11] A. Del Bimbo, *Visual Information Retrieval*, Academic Press, New York, NY, USA, 1999.

[12] B. Günsel, A. M. Tekalp, and P. J. L. van Beek, "Content-based access to video objects: temporal segmentation, visual summarization, and feature extraction," *Signal Processing*, vol. 66, no. 2, pp. 261–280, 1998.

[13] Y. Deng, B. S. Manjunath, C. Kenney, M. S. Moore, and H. Shin, "An efficient color representation for image retrieval," *IEEE Transactions on Image Processing*, vol. 10, no. 1, pp. 140–147, 2001.

[14] E. Hadjidemetriou, M. D. Grossberg, and S. K. Nayar, "Multiresolution histograms and their use for recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 26, no. 7, pp. 831–847, 2004.

[15] S. Jeong, C. S. Won, and R. M. Gray, "Image retrieval using color histograms generated by Gauss mixture vector

quantization," *Computer Vision and Image Understanding*, vol. 94, no. 1–3, pp. 44–66, 2004.

[16] E. Ardizzone and M. La Cascia, "Automatic video database indexing and retrieval," *Multimedia Tools and Applications*, vol. 4, no. 1, pp. 29–56, 1997.

[17] B. K. P. Horn and B. G. Schunck, "Determining optical flow," *Artificial Intelligence*, vol. 17, no. 1–3, pp. 185–203, 1981.

[18] D. J. Watts and S. H. Strogatz, "Collective dynamics of 'small-world' networks," *Nature*, vol. 393, no. 6684, pp. 440–442, 1998.

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

[20] J. MacQueen, "Some methods for classification and analysis of multivariate data," in *Proceedings of 5th Berkeley Symposium on Mathematical Statistics and Probability*, vol. 1, pp. 281–297, Berkeley, Calif, USA, 1967.

[21] M. Halkidi, Y. Batistakis, and M. Vazirgiannis, "On clustering validation techniques," *Journal of Intelligent Information Systems*, vol. 17, no. 2-3, pp. 107–145, 2001.

[22] N. R. Pal and J. C. Bezdek, "On cluster validity for the fuzzy c-means model," *IEEE Transactions on Fuzzy Systems*, vol. 3, no. 3, pp. 370–379, 1995.

[23] C. Ordonez, "Clustering binary data streams with K-means," in *Proceedings of the 8th ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery (DMKD '03)*, pp. 12–19, San Diego, Calif, USA, June 2003.

[24] S. Kirkpatrick, C. D. Gelatt Jr., and M. Vecchi, "Optimization by simulated annealing," *Science*, vol. 220, no. 4598, pp. 671–680, 1983.

[25] M. Ripeanu, A. Iamnitchi, and I. Foster, "Mapping the Gnutella network," *IEEE Internet Computing*, vol. 6, no. 1, pp. 50–57, 2002.

[26] S. Saroiu, P. K. Gummadi, and S. D. Gribble, "Measurement study of peer-to-peer file sharing systems," in *Multimedia Computing and Networking*, vol. 4673 of *Proceedings of SPIE*, pp. 156–170, San Jose, Calif, USA, January 2002.

[27] S. Sen and J. Wang, "Analyzing peer-to-peer traffic across large networks," *IEEE/ACM Transactions on Networking*, vol. 12, no. 2, pp. 219–232, 2004.

[28] L. Gatani, G. Lo Re, and L. Noto, "Efficient query routing in peer-to-peer networks," in *Proceedings of the 3rd International Conference on Information Technology: Research and Education (ITRE '05)*, pp. 393–397, Hsinchu, Taiwan, June 2005.

[29] E. P. Markatos, "Tracing a large-scale peer-to-peer system: an hour in the life of Gnutella," in *Proceedings of the 2nd IEEE/ACM International Symposium on Cluster Computing and the Grid (CCGrid '02)*, pp. 65–74, Berlin, Germany, May 2002.

[30] C. Faloutsos, R. Barber, M. Flickner, et al., "Efficient and effective querying by image content," *Journal of Intelligent Information Systems*, vol. 3, no. 3-4, pp. 231–262, 1994.

[31] R. W. Picard, "Computer learning of subjectivity," *ACM Computing Surveys*, vol. 27, no. 4, pp. 621–623, 1995.

Hindawi Publishing Corporation
Advances in Multimedia
Volume 2007, Article ID 64938, 10 pages
doi:10.1155/2007/64938

Research Article

A Hybrid Query Scheme to Speed Up Queries in Unstructured Peer-to-Peer Networks

Zhan Zhang, Yong Tang, Shigang Chen, and Ying Jian

Department of Computer and Information Science and Engineering, University of Florida, Gainesville, FL 32611-6120, USA

Received 31 January 2007; Accepted 6 June 2007

Recommended by Ben Y. Zhao

Unstructured peer-to-peer networks have gained a lot of popularity due to their resilience to network dynamics. The core operation in such networks is to efficiently locate resources. However, existing query schemes, for example, flooding, random walks, and interest-based shortcut suffer various problems in reducing communication overhead and in shortening response time. In this paper, we study the possible problems in the existing approaches and propose a new hybrid query scheme, which mixes intercluster queries and intracluster queries. Specifically, the proposed scheme works by efficiently locating the clusters, sharing similar interests with intercluster queries, and then exhaustively searching the nodes in the found clusters with intracluster queries. To facilitate the scheme, we propose a clustering algorithm to cluster nodes that share similar interests, and a labeling algorithm to explicitly capture the clusters in the underlying overlays. As demonstrated by extensive simulations, our new query scheme can improve the system performance significantly by achieving a better tradeoff among communication overhead, response time, and ability to locate more resources.

1. INTRODUCTION

Peer-to-peer networks surged in popularity in recent years. The core operations in most peer-to-peer networks is to efficiently locate data items, in which the fundamental challenges are to achieve faster response time, smaller network diameter, stronger ability of locating more resources, and better resilience to network dynamics.

Structured P2P networks have been proposed by many researchers [1–7], in which distributed hash tables (DHTs) are used to provide data location management in a strictly structured way. Whenever a node joins/leaves the overlay, a number of nodes need to update their routing tables to preserve desirable properties for fast lookup. While structured P2P networks can offer better performance in response time and communication overhead for query procedures, they suffer from the large overhead for overlay maintenance due to network dynamics.

Unstructured P2P networks such as Gnutella rely on a random process, in which nodes are interconnected in a random manner. The randomness offers high resilience to the network dynamics. However, basic unstructured networks rely on flooding [8] for users' queries, which is expensive in computation and communication overhead. Consequently, scalability has always been a major weakness for unstructured networks [9]. Even with the use of super nodes in Morpheus [10] and KaZaA [11], the traffic is still high, and even exceeds web traffic.

Searching through random walks is proposed in [12–14], in which incoming queries are forwarded to the neighbor that is chosen randomly. In random walks, there is typically no preference for a query to visit the most possible nodes maintaining the needed data, resulting in long response time.

Interest-based shortcut [15] exploits the locality of interests among different nodes. In this approach, a peer learns its shortcuts by flooding or passively observing its own traffic. A peer ranks its shortcuts in a list and locates content by sequentially asking all of the shortcuts on the list from the top until content is found. The basic principle behind this approach is that a node tends to revisit accessed nodes again since it was interested in the data items from these nodes before. The concept of interest similarity is vague and it is difficult to make a subtle, quantitative definition based on it. In addition, it may cause new problems as discussed later.

In this paper, we take the unstructured approach, and propose a new query scheme to address these problems. The main contributions of the paper include the following.

(i) *We define a metric, independent of any global information, to measure the interest similarity between nodes.*

Based on the metric, we propose a clustering algorithm to cluster nodes sharing similar interests with small overhead, and fast convergence.

(ii) *We propose a distributed labeling algorithm to explicitly capture the borders of clusters without any extra communication overhead.*

(iii) *We propose a new query scheme, which is able to deliver a better tradeoff among response time, communication overhead, and the ability to locate more resources by mixing intercluster queries and intracluster queries.*

The rest of the paper is organized as follows. Section 2 reviews the possible problems in prior approaches. Section 3 gives the overview of our scheme. Section 4 defines the interest similarity and proposes a light-weight algorithm to cluster nodes within the same interest group and a labeling algorithm to explicitly border the clusters. Section 5 introduces our query scheme in detail. Section 6 evaluates the scheme with extensive simulations. Section 7 draws the conclusion.

2. INEFFICIENCY IN PRIOR WORKS

Small communication overhead and short response time are the two main concerns in designing efficient query schemes in peer-to-peer networks. However, current approaches suffer various problems in achieving a better tradeoff between them due to the blindness in searching procedures.

Flooding

Flooding [8, 16] is a popular query scheme to search a data item in fully unstructured P2P networks such as Gnutella. While flooding is simple and robust, its communication overhead, that is, the number of messages, increases exponentially with the hop number. In addition, most of these messages visit the node that has been searched in the same query, and they can be regarded as duplicate messages. Consequently, communication overhead and scalability are always the main problems in this approach [9, 17].

Random walks

Random walks [12–14, 18] rely on query messages randomly selecting their next hops among neighbors with equal probabilities to reduce the communication overhead. A query may have to go through many hops before it successfully locates the queried data item. Consequently, this approach takes a long time to locate queried data items. If the networks are well clustered (nodes with similar interests are densely connected), it is expected that the query latency can be reduced significantly. However, it is not true, because the number of hops escaping out of the cluster decreases exponentially with the ratio r of the number of intercluster edge to the number intracluster edges, as shown in Figure 1.

In the case of a network with a small value of r, for example, $r < 0.01$, if queried data items are in different clusters from the source node, a query message has to walk a long distance to be able to traverse the cluster border and locate the queried data items. In the case of a network with a large value of r, for example, $r > 0.1$, query requests may escape out of the original cluster within a small number of hops, resulting in a long response time if the queried data is in the original cluster. These observations are also demonstrated by our simulations in Section 6. Consequently, random walks may suffer long response time regardless of the network having been well clustered or not.

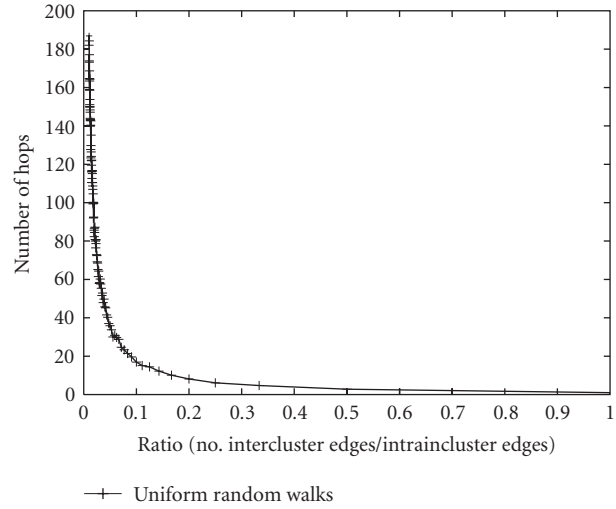


Figure 1: The number of hops of random walks escaping out of the cluster decreases exponentially with the ratio of number of intercluster edges to the number of intracluster edges.

Interest-based shortcut

Interest-based shortcut, for example, [15], tries to avoid the blindness in random walks by favoring nodes sharing similar interests with the source, which can be regarded as a variation of Markov random walks, biased towards some specific nodes. Markov random walks may accelerate the query process to some extent in some cases. However, it causes new problems. Suppose that nodes in an interest group have formed a cluster, and query messages can be artificially confined in this specific cluster. In the sense of nodes in the cluster share similar interests, any of them possibly maintains the queried data. Thus, the query procedure should shorten the covering time of the whole cluster instead of the hitting time of some specific nodes in it. However, due to the bias in selecting next hop in Markov random walks, it tends to keep visiting some specific nodes, resulting in less distinct nodes being covered comparing to uniform random walks, as illustrated by Figure 2. Consequently, Markov random walks work worse than uniform random walks if both of them can be confined in specific clusters.

3. SYSTEM OVERVIEW

Researchers [15, 19] have found many peer-to-peer networks exhibit small-world topology, and most of queried data items are offered by the nodes, which share similar interest with the source node.

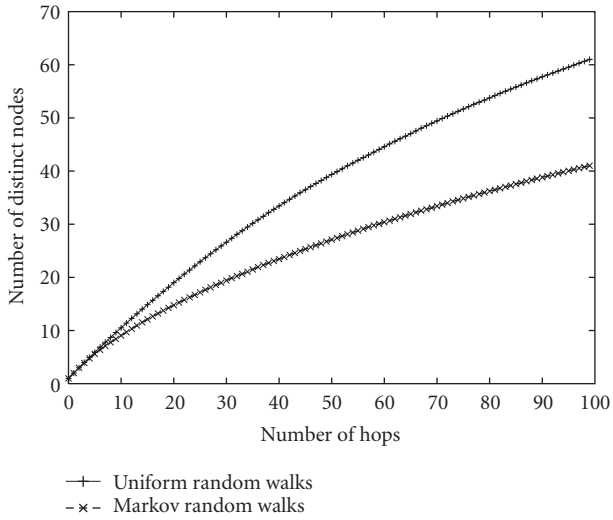


Figure 2: Markov random walks discover a smaller number of nodes than uniform random walks do.

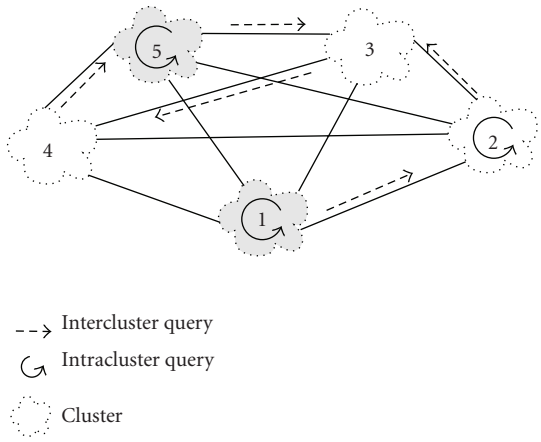


Figure 3: A query scheme mixing intercluster queries and intracluster queries (the nodes in the grey clusters fall into the same interest group).

Intuitively, the nodes sharing similar interests with the source node should have higher priority to be searched than others. Practically, there are two challenges in designing such a query scheme. The first one is how to cluster nodes with similar interests based on the small-world property of P2P networks. By saying "similar interests," we actually mean that two nodes are interested in a common set of data items. Thus, the number of common accessed data items can serve as a metric to measure the interest similarity between two nodes. A clustering algorithm based on the metric can be easily designed to densely connect the nodes in the same interest group. Moreover, each node u can explicitly pick up a set of *intercluster* neighbors that have different interests from u, and a set of *intracluster* neighbors that share similar interests with u. Take Figure 3 as an example. The network consists of 5 clusters, and nodes in the same cluster fall into the same interest group. Note that there exists an interest group consisting of two clusters, 1 and 5.

Suppose the network has been well clustered, and each node explicitly maintains a set of intercluster and intracluster neighbors. The second challenge is how to fast locate the clusters that share similar interests with the source node and how to exhaustively search nodes in the found clusters if the queried data items are in the source node's interest group. We introduce two types of queries, *intercluster* queries and *intracluster* queries. The intercluster queries are for the purpose of discovering the clusters that share similar interests with the source node, and they are issued by source node, carry the interest information, and only travel on intercluster neighbors. It can be expected that clusters sharing similar interests with the source node can be located quickly, because the number of cluster is much smaller comparing to the network size, and intercluster queries only travel among different clusters. The intracluster queries are spawned by intercluster queries when a cluster sharing similar interest with the source node is hit. An intracluster query thoroughly go through nodes in the found cluster, where it is spawned, by only traveling on intracluster neighbors. How to estimate to what extent a cluster has been searched will be discussed later. Note that intercluster queries and intracluster queries can be easily implemented if each node explicitly knows the types of its neighbors.

Occasionally, queried data may be out of the source node's interest group, and possibly maintained by a cluster(s) with different interests. This problem is addressed by blind search: intercluster messages randomly spawning intracluster messages when hitting clusters with different interests.

For example, in Figure 3, first inter-queries are initiated by a node in cluster 1, which travel among different clusters. By the interest information carried in the inter-queries, cluster 5 is found to share similar interests when it is hit, and an intracluster query is spawned, which then will exhaustively search the nodes in it. In addition, an intracluster query is spawned in cluster 2 by intercluster queries to support blind search.

4. CLUSTERING ALGORITHM

4.1. Measuring the interest similarity between two nodes

Cluster is generally formed by connecting nodes with similar interests in a network. Thus, we start our discussion with the definition of interest similarity between two nodes in P2P networks.

If node u and node v share similar interests, then it is very likely that they have accessed same data items more or less previously. Therefore, the size of the common subset of accessed data items can serve as a metric to measure to what extent the interests of two nodes are similar.

However, each node may offer hundreds of data items, and hence, there may exist a large number of data items even in a small network. As a result, only if u and v have visited a large number of data items, respectively, they are able to show some degree of similarity. An alternative to evaluate

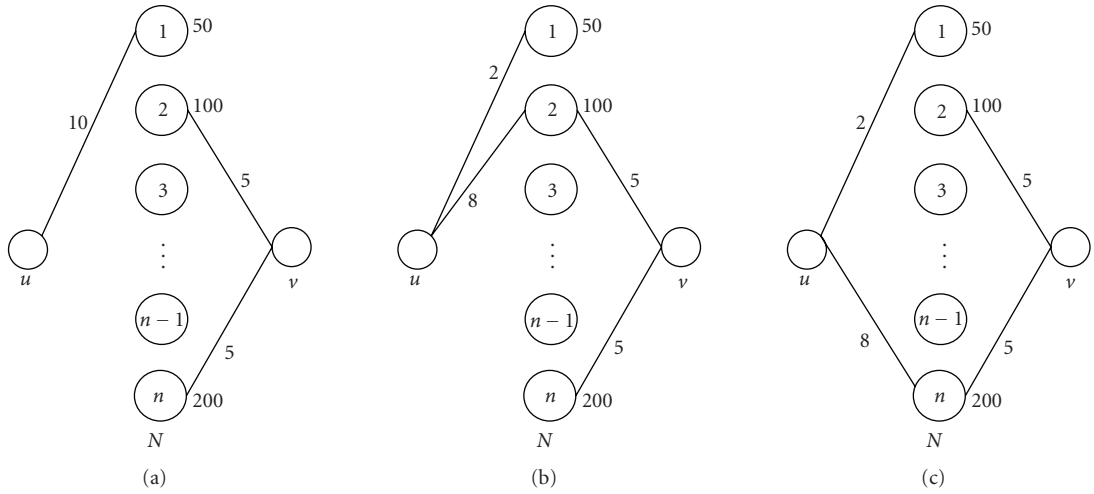


Figure 4: Interest similarity between nodes. The number of data items in 1, 2, and n is 50, 100, and 200, respectively. (a) No common visited nodes (different interests), (b) u and v have visited node 2 (100 data items) with 8 and 5 times, respectively (a certain level of similar interests), (c) u and v have visited node n (200 data items) with 8 and 5 times, respectively (falls in between).

the interest similarity is by the number of common accessed nodes, which may enable a clustering algorithm to converge faster than the former approach. The problem in this approach is that two nodes visiting a common node does not indicate they have similar interests, because a node may offer data items belonging to multiple interest groups. For instance, a user u may offer resources for two groups: a number of mp3 music files for one group, and a number of research literatures in P2P networks for the other group. It is possible that two nodes that have visited u may be interested in data items in different interest group. Thus, we have to address the discrepancy between the common set of accessed nodes and the common set of visited data items.

Suppose that there are n nodes $N = \{1, 2, \ldots, n\}$ in the whole P2P network. Suppose a node i offers a number of data items to others. It categorizes (maps) all of these data items into α_i different categories, denoted as $C^i = \{c_1^i, c_2^i, \ldots, c_{\alpha_i}^i\}$. Suppose a data item x in i is mapped to a category $c^i(x)$, where $c^i(x) \in C^i$. How to categorize the data items is determined by the node i independently. For instance, node i may classify music files as category 1, while another node may classify music files as category 2. On the other hand, node i may fall into multiple interest groups, denoted as $G^i = \{g_1^i, g_2^i, \ldots, g_{\beta_i}^i\}$.

For a node u, the access history with respect to each of its interest groups, for example, g_1^u, can be specified by a set of data items x, denoted as $(i, c^i(x))$, where i represents the node offering the data item, and $c^i(x)$ is the category in C^i defined by i. If two nodes u and v share "similar interests," for example, $g_1^u \approx g_2^v$, their histories for g_1^u and g_2^v tend to consist of a common set of $(i, c^i(x))$.

Note that in the above definitions, each node determines its interest groups and categories independently, indicating a node need not maintain any global information.

For easy explanation, we study a basic approach by assuming each user only falls into one interest group, and offers one category of data items. In this scenario, the access history can be represented by the accessed nodes alone. This approach can be easily extended to the multicategories and multigroups based on the definitions above.

One node u may access another node multiple times for different data items, and hence, the access history of node u can be represented by a vector $V^u = (v_1^u, \ldots, v_n^u)$,[1] where v_x^u represents the number of times u has visited node x. To cancel out the number of queries a node has issued, the access vector V^u is normalized to the *frequency vector* F^u, in which the ith element in F^u is denoted as f_i^u, computed by V^u as $f_i^u = v_i^u / \sum_{j \in N} v_j^u$, representing the frequency of the corresponding node i having been accessed.

Note that the value in F^u falls into the range $[0, 1]$. If u has never accessed node i, the corresponding element f_i^u is equal to 0. The summation of all elements is equal to 1.

Furthermore, if the number of data items in node i, denoted as d_i, is large, the chance that two nodes u and v have visited common data items in i may be small even if both of them have visited i multiple times. As an example, in the middle and right parts in Figure 4, u and v have visited one common node. But u and v in the middle part have more chance of having visited common data items because the number of data items in node 2 is half of that in node n. To account for this issue, we introduce a weighted diagonal matrix W with (i, i)th value $w_{i,i}$ equal to $1/d_i$. It represents the probability of both u and v visiting a common data items, if both of them visit i once.

Now we define the following metric to evaluate the interest similarity between two nodes:

$$A^{u,v} = F^{u^T} W F^v = \sum_i f_i^u f_i^v \frac{1}{d_i}. \quad (1)$$

[1] The real size of the data structure maintaining V^u is much smaller than the network size n, and can be fixed to only record the nodes accessed most frequently by u.

Take Figure 4 (middle) as an example:

$$\begin{aligned} A^{u,v} &= F^{u\,T} W F^{v} \\ &= \begin{pmatrix} .2 \\ .8 \\ 0 \\ \cdot \\ \cdot \\ 0 \end{pmatrix}^{T} \begin{pmatrix} .02 & 0 & 0 & 0 & 0 & 0 \\ 0 & .01 & 0 & 0 & 0 & 0 \\ 0 & 0 & d_3 & 0 & 0 & 0 \\ 0 & 0 & 0 & \cdot & 0 & 0 \\ 0 & 0 & 0 & 0 & \cdot & 0 \\ 0 & 0 & 0 & 0 & 0 & .005 \end{pmatrix} \begin{pmatrix} 0 \\ .5 \\ 0 \\ \cdot \\ \cdot \\ .5 \end{pmatrix} \\ &= 0.004. \end{aligned} \tag{2}$$

Similarly, the interest similarity in Figure 4(c) (right) is 0.002, which means nodes in the middle example show more interest similarity.

If we view f_i^u and f_i^v as the probabilities of nodes u and v visiting node i, and $1/d_i$ as the probability of u and v visiting a common data items if both of them visit i, then the summation $A^{u,v}$ can be used to predict the probability that both u and v will visit a common data item in their future queries.

Note that if a node i has not been visited by both u and v, then $f_i^u = 0$ and/or $f_i^v = 0$, indicating that a node does not need to maintain any information of the nodes it has never visited. As we have discussed, the size of the vector V^u is fixed. Hence, both the storage for access history and the computation overhead for $A^{u,v}$ are constant.

Our definition is advantageous in manyfold. First, nodes u and v need not maintain any global information to compute $A^{u,v}$. Second, the frequency vector cancels out the effect of the number of queries that a node has issued, enabling a clustering algorithm based on this definition to converge fast with the average number of queries. Third, the definition prefers the nodes with good properties. For instance, if two nodes i, j maintaining the same data items can be accessed by 1 Mp/s and 56 Kbp/s, respectively, i obviously will be accessed more often, resulting in a larger value of f_i^u and f_i^v. Fourth, it reduces the impact of the possible discrepancy in the category definitions by nodes. For instance, if a category in node i is poorly defined, and consists of data items belonging to various interest groups, then the category will be seldom accessed comparing to its size, resulting in a small value of $f_i^u f_i^v (1/d_i)$.

4.2. Clustering nodes with similar interests

Given the metric to evaluate the interest similarity between two nodes, we propose a light-weight clustering algorithm to connect nodes sharing similar interests.

In our strategy, each node i maintains a list L with limited size, for example, 30, to record the nodes that possibly share same interests with itself. Each time a query message is processed, the similarity between the querying node itself and the node that owns the data items is computed. The overhead of similarity computation is fixed given that the size of the list L is predefined. The newly obtained interest similarity, and the corresponding node's address are inserted into the list L. If the list is full, the stored neighbor with the lowest interest similarity is dropped.

By assuming that interests of nodes will not shift in a limited time frame, the nodes collected in L possibly fall into the same interest group as i, and will serve as candidates of its intracluster neighbors.

4.3. Bounding clusters

Although a small-world topology can be formed along with queries by the above clustering algorithm, existing query schemes, for example, random walks, can only benefit marginally from it as we discussed in Section 3.

To exploit the characteristics of the small-world topology, our approach is to explicitly capture the clusters in the underlying topology by each node i maintaining a set of intercluster neighbors in other interest groups, and intracluster neighbors in its own interest group.

For intercluster neighbors, a node i can learn them easily. For example, i can issue a certain number of random-walk messages only traveling on other nodes' intercluster neighbors, and choose the nodes hit by the messages as its intercluster neighbors. Note that the list L collects candidates of its intracluster neighbors and should not overlap with the set of intercluster neighbors.

Thus, it is of the most importance to learn the intracluster neighbors, which can be selected from the nodes collected in the list L. The purpose of intracluster neighbors is to confine intracluster queries within a specific interest group. Two nodes falsely regarded as intracluster neighbors may create a dramatic impact because an intracluster query may traverse to another cluster with different interests. On the contrast, two nodes i and k that are falsely regarded as intercluster neighbors will only have limited impact, because i and k may be connected by other intermediate intracluster neighbors j. In addition, the chance of i and k falling into the same cluster tends to become larger along with query procedures if they are in the same interest group.

Based on this observation, we propose a labeling algorithm, which ensures that if a link (i, j) is labeled as an intracluster edge, then i and j are in the same interest group with high probability.

For a node i, we normalize the interest similarity of its neighbors j in L as follows, where k is the neighbor of i:

$$p_{i,j} = \frac{A^{i,j}}{\sum_k A^{i,k}}. \tag{3}$$

If a matrix P is organized such that its i, jth element is $p_{i,j}$, then the rows in P sum to 1 as the matrix P is row stochastic. Intuitively, $p_{i,j}$ can be viewed as the transition probabilities for the Markov random walk.

The transition probability $p_{i,j}$ can serve as a good metric to determine whether i and j are in the same interest group or not by introducing a threshold, denoted as T, as a lower bound. T can be set as a relatively larger value, because the false negative has limited impact as discussed. Suppose there are α neighbors in L that are possibly in the same interest group with i. T can be set as $1/\alpha$. Note that $p_{i,j}$ and T are computed by node i locally. Thus, the labeling algorithm does not involve any extra communication.

5. A HYBRID QUERY SCHEME

5.1. Mixing intercluster queries and intracluster queries

By explicitly capturing the cluster structures in the underlying network, we can formally define the following three types of query messages.

The first one is called *l-query message*, which is a special type of intercluster message only traveling on intercluster neighbors. The purpose of it is to quickly locate the clusters that may share similar interests with the source node and disperse intra-queries among different clusters. Messages in this type are issued by the source node, and walk among different clusters randomly. Moreover, if the queried data is in the source node's interest group, l-query messages should piggyback the source node's frequency vector, such that nodes hit by the messages can determine whether their clusters share similar interests with the source node.

The second one is called *s-query message*, which is a special type of intracluster message confining itself within a specific cluster by doing uniform random walks only on intracluster neighbors. s-query messages are only spawned/issued in the clusters that share similar interests with the source node. The purpose of it is to exhaustively search nodes that fall into the same interest group as the original node. Messages in this type are spawned by l-query messages when clusters sharing similar interests are hit.

Now the problem is how a node estimates to what extent the cluster has been covered. If the cluster has been well covered, the s-query message should be discarded in order to reduce the query overhead. Accurately estimating the covering time of a cluster is difficult and resource-consuming in a distributed system. Heuristically, if the message has been consecutively hitting a certain number, denoted as h, of nodes that have been visited before, it indicates that most of nodes have been covered, and hence, the message should be discarded. This information can be maintained by a counter in each s-query message.

The last one is called *b-query message*, which is also a special type of intracluster message similar to s-query message. But b-query messages may be spawned in clusters that have different interests. The purpose of it is to support blind search, because occasionally, the queried data may be out of the source node's interest group. The chance that the queried data item in a cluster has different interests is very small. Thus, once a b-query message hits a node that has been visited by intracluster messages before, the message is discarded in order to reduce the query overhead.

To control the communication overhead, the total number of concurrent query messages has to be limited. The source node needs to count the number of l-query, s-query, and b-query messages, denoted as m_l, m_s, and m_b, respectively. The overhead to maintain the counter is negligible given that the counter needs to be updated only if s-query message or b-query message is spawned or discarded. Only if the summation of m_l, m_s, and m_b is smaller than a certain number, denoted as m, a new b-query message can be spawned to support blind search. s-query messages can be spawned without restriction, and thus, the total number of concurrent messages may be larger than m temporarily. In addition, all messages need to periodically check the status of the source node so that they can stop if the query has been successfully returned.

With these three types of messages, a query scheme is designed as follows.

Initialization

To initiate a query request, a node u issues a number m_l of l-query messages. If the queried data item falls in u's interest group, l-query messages carry the source's frequency vector, and a certain number m_s of s-query messages are issued to exhaustively search its own cluster. Otherwise, a b-query message is issued in the meantime.

Receiving an l-query message

In the case of a node u receiving an l-query message, it calculates the interest similarity with the source node. If u shares similar interests, for example, the similarity is larger than a small value, it spawns a new s-query message and update m_s maintained by the source node. Otherwise, a new b-query message is spawned if the node has not been hit by s-query messages and b-query messages, and $m_l + m_s + m_b < m$. Finally, node u forwards the received message to a randomly selected intercluster neighbor.

Receiving an s-query message

In the case of a node u receiving an s-query message, if u has been hit by s-query messages or b-query messages, it increases the counter in the message by 1. Otherwise, it resets the counter to 0. Next, if the counter is larger than the threshold h, the node discards the message and notifies the source node to update the counter m_s. Otherwise, it forwards the message to a randomly selected intracluster neighbor.

Receiving a b-query message

In the case of a node u receiving a b-query message, if it has been hit by messages in s-query messages and b-query messages, the node discards the message and notifies the source node to update the counter m_b. Otherwise, it forwards the message to a randomly selected intracluster neighbor.

Our scheme can be considered to be *stateful*, in which if the same queries are reissued multiple times, less intracluster queries will be spawned in the clusters that have been well searched, and in contrast, more intracluster queries will be spawned in the less-searched clusters, resulting in stronger ability to discover more resources/replicas.

5.2. Reducing the communication overhead

By mixing intercluster and intracluster queries, it can be expected that the system performance can be improved significantly. Because the access vector V^u of a node u can be fixed

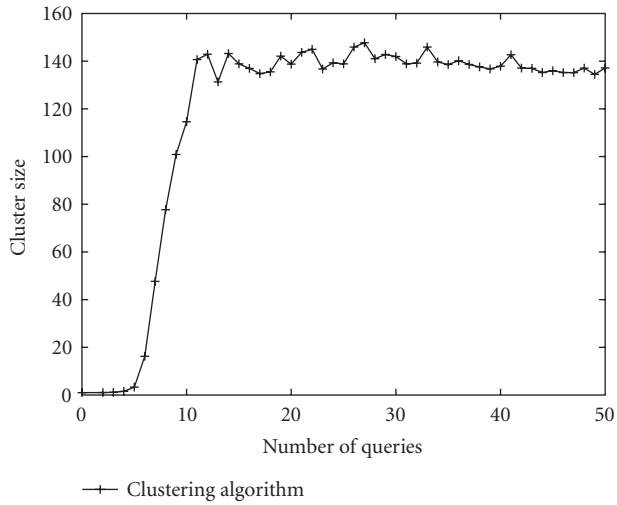


Figure 5: The effect of the average number of queries on the cluster size.

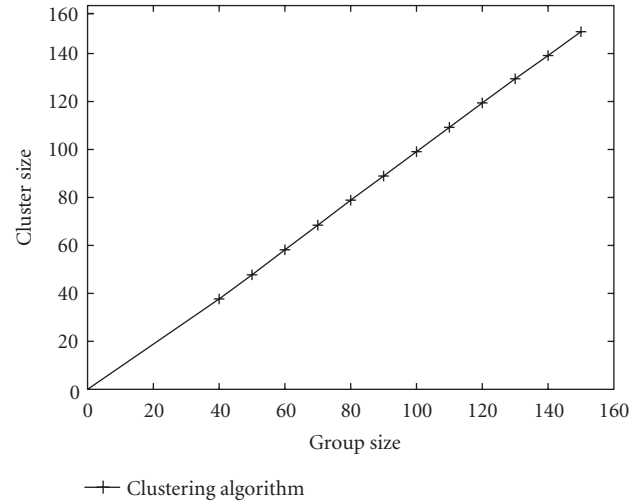


Figure 6: The interest association is a good metric to estimate interest similarity.

to a small size, the extra overhead will not increase largely (only l-query messages need to carry the frequency vector).

Moreover, l-query messages only travel among different clusters, and the number of clusters, especially in a well-clustered network, is much smaller comparing to the real network size. It can be expected that most of clusters can be covered by l-query messages within a small number of hops. Thus, l-query messages can remove the frequency vector from the payloads after a certain number of hops. In the meantime, a source node can specify one l-query message to keep the frequency vector for the case that some clusters sharing similar interests with the source node have not been discovered after the specified number of hops.

6. SIMULATION

In this section, the performance of the proposed clustering algorithm and query scheme is studied by simulations. If not explicitly defined, the default number of nodes is 10 000, and each node maintains 1 000 data items, which are randomly generated. The number of nodes in each interest group is 150, the average number of queries issued by each node is 30, and the threshold h is equal to 10. We set m to be 32, and m_l to be 16. Moreover, the probability of a node incorrectly classifying its queries or data items is 0.1. We also simulate other scenarios, in which nodes classify its queries or data items with various probabilities. The simulation results are similar, except that it takes a few more queries for the converge of the clustering algorithm.

We compare our scheme to random walks, in which a source node issues 32 random walk messages in each query, correspondingly. We also have compared our scheme to flooding schemes. As expected, we observe the flooding schemes suffer from very large communication overhead.

In Figures 7–10, the legend "Uniform random walks (0)"/"Uniform random walks (1)" denotes that the queried data items are out of/in the source node's interest group in the uniform random walks query scheme, and similarly "Inter/intra (0)"/"Inter/intra (1)" denotes that the queries are out of/in the source node's interest group in the proposed scheme.

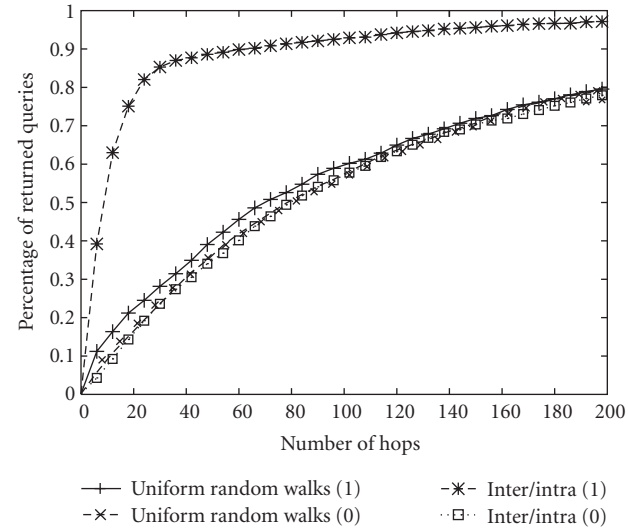


Figure 7: The percentage of returned query within a specific hop number.

First, we study the effectiveness of the metric measuring interest similarity and the clustering algorithm. In Figure 5, it is observed that when the average query number is larger than 10, the algorithm reaches a stable state and almost all nodes in the same interest group form a single cluster. It indicates that our algorithm converges fast with the average number of queries, which is especially useful in P2P networks, where nodes tend to join/leave the system more frequently.

By Figure 6, it can be observed that the average number of nodes in a cluster is almost the same as group size, demonstrating that $A^{u,v}$ can effectively measure the nodes' interest similarity.

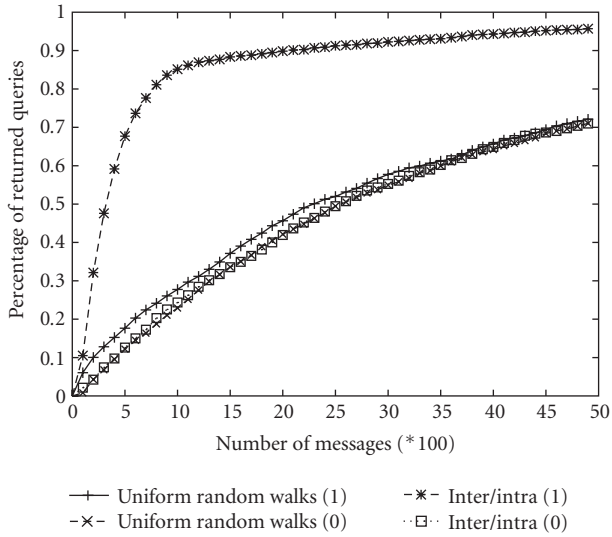


Figure 8: The percentage of returned query within a specific message number.

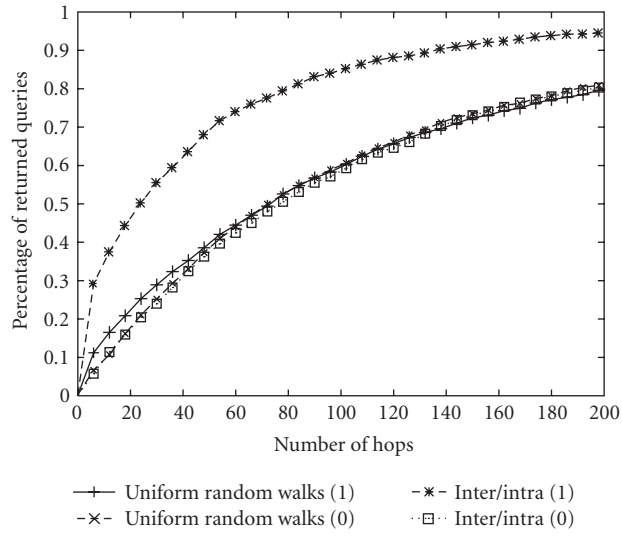


Figure 9: The percentage of returned query within a specific hop number in a less-clustered network.

Second, we study the performance of our scheme with respect to query latency and communication overhead.

In Figure 7, it is observed that if the queried data items fall into the original node's interest group, the number of hops needed for the majority of the queries is significantly reduced to about 20, while in the uniform random walks, it takes a much longer time. Correspondingly, the number of messages is also much smaller in our scheme than that in random walks, as shown in Figure 8. The figures also show that if the queried data are out of the source node's interest group, the performance of our scheme is similar to uniform random walks. Note that a longer response time is acceptable since only a few queries will be out of source node's interest group in many P2P networks. In addition, these two figures also demonstrate that random walks for queries in the source node's interest group can only benefit marginally from the underlying clustered topology, for example, only a little larger percentage of them can be returned than those out of source node's interest group within the same number of hops (messages).

We also have studied the performance of a network, in which each group consists of multiple different clusters, as shown in Figures 9 and 10. The results show the similar trends, which keeps true with respect to all other metrics that will be studied later. Moreover, comparing Figure 7 with Figure 9, and Figure 8 with Figure 10, it can be observed that the performance of random walks in two different (well-clustered and poor-clustered) networks is similar, which further verifies our argument in Section 3.

As have been observed, when the queried data items are in the source nodes' interest group, our scheme works much better than random walks. The reason behind it is that our scheme can discover more distinct nodes in the source nodes' interest groups within the same number of messages or hops, as shown in Figures 11 and 12. In these figures, it can be observed that within the first 1 000 messages or 30 hops, more

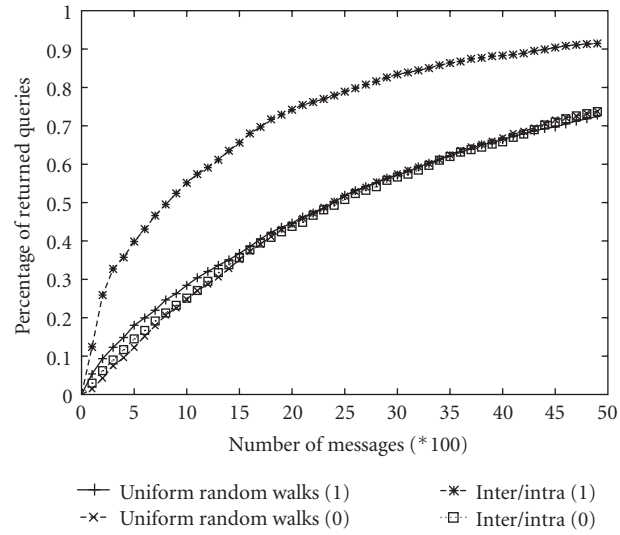


Figure 10: The percentage of returned query within a specific message number in a less-clustered network.

than 120 nodes in the source node's interest group have been searched by query messages. Consequently, the majority of queried data falling into the node's interest group can be found with smaller overhead and shorter latency. It also indicates that our scheme has a strong ability to locate more replicas since it can discover a much larger number of nodes sharing similar interests.

Occasionally, the queried data item may be maintained by nodes in other interest groups, or classified into wrong interest group by source node. In the former case, l-queries will not carry any interest information, but in the latter case, l-queries will carry wrong interest information. In both cases, the efficiency of our query scheme can be evaluated by the

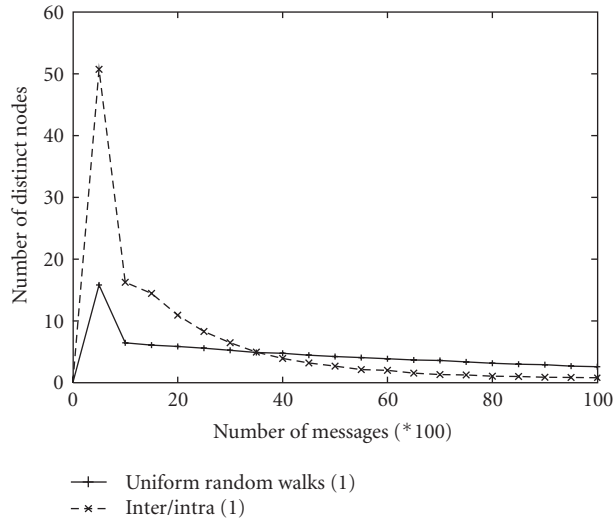


Figure 11: The number of distinct nodes discovered in the same group within a certain message range.

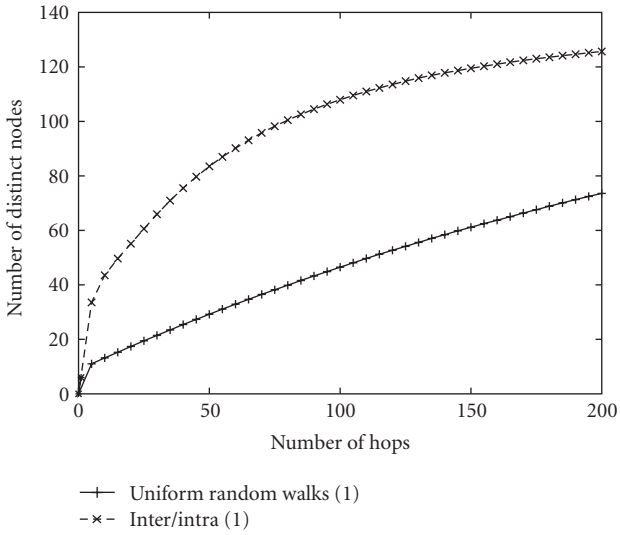


Figure 12: The number of distinct nodes discovered in the same group within a specific hop number.

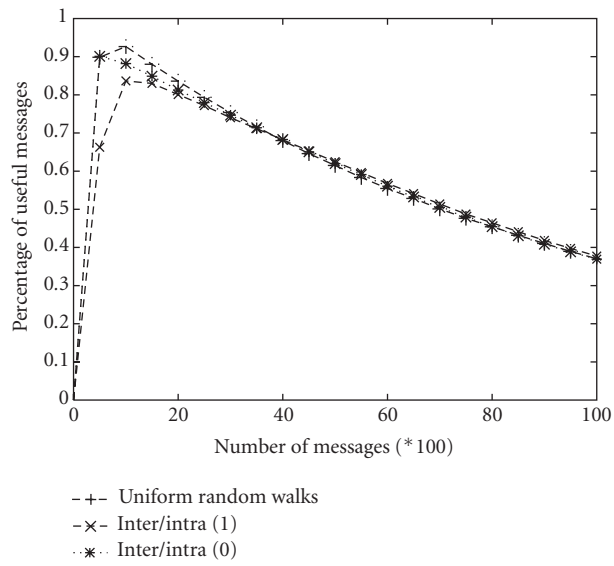


Figure 13: The percentage of messages discovering distinct nodes within a certain message range.

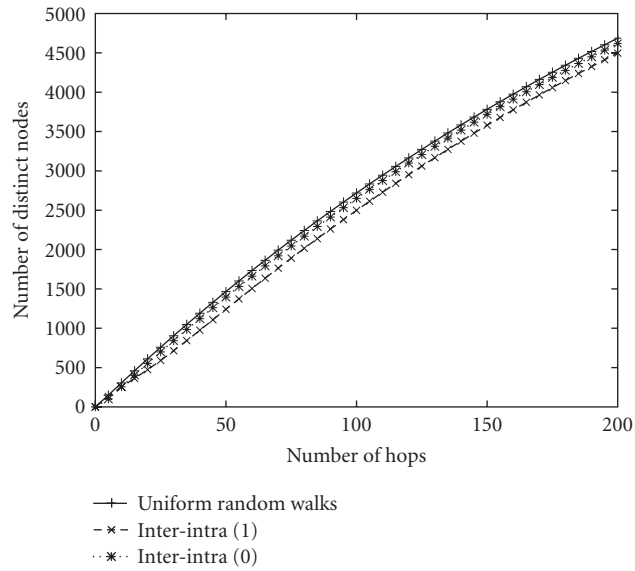


Figure 14: The total number of distinct nodes discovered within a specific hop number.

number of distinct nodes discovered by queries, including those out of the source node's interest group, within a certain number of messages and hops. Note that whether the queries are in or out of the original node's interest group makes no difference to random works. Figures 13 and 14 show that in the first 1 000 messages, if the queries carry interest information, fewer distinct nodes can be searched in our scheme. The reason is that *s*-query messages mistakenly exhaustively search the nodes in the clusters that share "similar" interests in the beginning, which has been demonstrated by our previous simulations. Consequently, the number of *b*-query messages is limited. Along with the increment of the number of messages/hops, our scheme works similar to the uniform random walks. It is because after most of nodes sharing similar interests are covered, more *b*-query messages will be spawned to search clusters with different interests, which are able to discover more distinct nodes. In addition, by the figures, if the queries carry no interest information, our scheme works similar to uniform random walks.

7. CONCLUSION

In this paper, we strictly define the metric to measure the interest similarity between nodes. A distributed clustering algorithm has been also presented, which gives P2P networks better resilience to network dynamics. We propose an algorithm to explicitly capture clusters in the underlying networks

without any extra communications. A query scheme mixing intercluster and intracluster queries has been designed for unstructured P2P networks. It can achieve a better tradeoff among communication overhead, response time, and the ability to locate more resources (replicas). The performance of the algorithms has been demonstrated by simulations.

REFERENCES

[1] I. Stoica, R. Morris, D. Liben-Nowell, et al., "Chord: a scalable peer-to-peer lookup protocol for Internet applications," *IEEE/ACM Transactions on Networking*, vol. 11, no. 1, pp. 17–32, 2003.

[2] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker, "A scalable content-addressable network," in *Proceedings of the Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications (SIGCOMM '01)*, pp. 161–172, ACM Press, San Diego, Calif, USA, August 2001.

[3] B. Y. Zhao, L. Huang, J. Stribling, S. C. Rhea, A. D. Joseph, and J. D. Kubiatowicz, "Tapestry: a resilient global-scale overlay for service deployment," *IEEE Journal on Selected Areas in Communications*, vol. 22, no. 1, pp. 41–53, 2004.

[4] A. I. T. Rowstron and P. Druschel, "Pastry: scalable, decentralized object location, and routing for large-scale peer-to-peer systems," in *Proceedings of IFIP/ACM International Conference on Distributed Systems Platforms (Middleware '01)*, pp. 329–350, Heidelberg, Germany, November 2001.

[5] C. G. Plaxton, R. Rajaraman, and A. W. Richa, "Accessing nearby copies of replicated objects in a distributed environment," *Theory of Computing Systems*, vol. 32, no. 3, pp. 241–280, 1999.

[6] D. Malkhi, M. Naor, and D. Ratajczak, "Viceroy: a scalable and dynamic emulation of the butterfly," in *Proceedings of the 21st Annual Symposium on Principles of Distributed Computing (PODC '02)*, pp. 183–192, ACM Press, Monterey, Calif, USA, July 2002.

[7] A. Kumar, S. Merugu, J. Xu, and X. Yu, "Ulysses: a robust, low-diameter, low-latency peer-to-peer network," in *Proceedings of the 11th IEEE International Conference on Network Protocols (ICNP '03)*, pp. 258–267, Atlanta, Ga, USA, November 2003.

[8] N. Chang and M. Liu, "Revisiting the TTL-based controlled flooding search: optimality and randomization," in *Proceedings of the 10th Annual International Conference on Mobile Computing and Networking (MOBICOM '04)*, pp. 85–99, Philadelphia, Pa, USA, September-October 2004.

[9] J. Ritter, 2001, Why Gnutella can't scale. no, really. http://www.darkridge.com/~jpr5/doc/gnutella.html.

[10] Morpheus, "Morpheus file sharing system," 2002, http://www.musiccity.com/.

[11] KaZaA, "Kazaa file sharing network," 2002, http://www.kazaa.com/.

[12] Y. Chawathe, S. Ratnasamy, L. Breslau, N. Lanham, and S. Shenker, "Making gnutella-like P2P systems scalable," in *Proceedings of the Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications (SIGCOMM '03)*, pp. 407–418, ACM Press, Karlsruhe, Germany, August 2003.

[13] C. Gkantsidis, M. Mihail, and A. Saberi, "Random walks in peer-to-peer networks," in *Proceedings of the 23rd Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM '04)*, vol. 1, pp. 120–130, Hong Kong, March 2004.

[14] Q. Lv, P. Cao, E. Cohen, K. Li, and S. Shenker, "Search and replication in unstructured peer-to-peer networks," in *Proceedings of the 16th International Conference on Supercomputing (ICS '02)*, pp. 84–95, ACM Press, New York, NY, USA, June 2002.

[15] K. Sripanidkulchai, B. Maggs, and H. Zhang, "Efficient content location using interest-based locality in peer-to-peer systems," in *Proceedings of the 22nd Annual Joint Conference on the IEEE Computer and Communications Societies (INFOCOM '03)*, vol. 3, pp. 2166–2176, San Francisco, Calif, USA, March-April 2003.

[16] N. B. Chang and M. Liu, "Optimal controlled flooding search in a large wireless network," in *Proceedings of the 3rd International Symposium on Modeling and Optimization in Mobile, Ad Hoc, and Wireless Networks (WiOpt '05)*, pp. 229–237, Trentino, Italy, April 2005.

[17] K. Sripanidkulchai, "The popularity of Gnutella queries and its implications on scalability," February 2001, http://www.cs.cmu.edu/~kunwadee/research/p2p/gnutella.html.

[18] C. Gkantsidis, M. Mihail, and A. Saberi, "Hybrid search schemes for unstructured peer-to-peer networks," in *Proceedings of the 24th Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM '05)*, vol. 3, pp. 1526–1537, Miami, Fla, USA, March 2005.

[19] A. Iamnitchi, M. Ripeanu, and I. Foster, "Small-world file-sharing communities," in *Proceedings of the 23rd Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM '04)*, vol. 2, pp. 952–963, Hong Kong, March 2004.

Hindawi Publishing Corporation
Advances in Multimedia
Volume 2007, Article ID 92485, 14 pages
doi:10.1155/2007/92485

Research Article

Globally Decoupled Reputations for Large Distributed Networks

Gayatri Swamynathan, Ben Y. Zhao, Kevin C. Almeroth, and Haitao Zheng

Department of Computer Science, University of California, Santa Barbara, CA 93106, USA

Received 8 March 2007; Accepted 7 June 2007

Recommended by Shigang Chen

Reputation systems help establish social control in peer-to-peer networks. To be truly effective, however, a reputation system should counter attacks that compromise the reliability of user ratings. Existing reputation approaches either average a peer's lifetime ratings or account for rating credibility by weighing each piece of feedback by the reputation of its source. While these systems improve cooperation in a P2P network, they are extremely vulnerable to unfair ratings attacks. In this paper, we recommend that reputation systems decouple a peer's *service provider* reputation from its *service recommender* reputation, thereby, making reputations more resistant to tampering. We propose a scalable approach to system-wide decoupled service and feedback reputations and demonstrate the effectiveness of our model against previous nondecoupled reputation approaches. Our results indicate that decoupled approache significantly improves reputation accuracy, resulting in more successful transactions. Furthermore, we demonstrate the effectiveness and scalability of our decoupled approach as compared to PeerTrust, an alternative mechanism proposed for decoupled reputations. Our results are compiled from comprehensive logs collected from Maze, a large file-sharing system with over 1.4 million users supporting searches on 226TB of data.

1. INTRODUCTION

The explosive growth of Internet connections in the last decade has resulted in an increase in the use and popularity of online peer-to-peer (P2P) communities. While the growth of these communities advances distributed applications like file transfer, remote storage, and computation, it is becoming increasingly critical to manage trust relationships in order to improve the performance of these applications. Large-scale P2P communities, like Gnutella [1], rely on cooperation among network peers. These communities, however, neither enforce cooperation nor centrally control peers. Peers are anonymous and self-interested, behaving only in their best interests. While this open nature of a P2P community is increasing the number of participating peers, it also makes these communities extremely difficult to police and vulnerable to attacks, thereby reducing the performance of the network.

As a popular P2P network, Gnutella is susceptible to a variety of attacks. One common attack is "whitewashing," where a free-riding node repeatedly joins the network under a new identity in order to avoid the penalties imposed on free-riders. A more serious attack is when dishonest peers distribute viruses and Trojan horses hidden as files. The *VBS.Gnutella* worm, for example, stores Trojan-horse executable files on network peers [2]. Meanwhile, a Gnutella worm called *Mandragora* registers itself as an active peer in the network, and provides a renamed copy of itself for download in response to intercepted queries [3]. Finally, dishonest peers often pass corrupted or blank files as legitimate content.

In order to reduce transaction risks and improve performance, P2P networks need to motivate cooperation and honest participation within their networks. Reputation systems help address this need by establishing a trust mechanism that enables peers to decide who to trust before undertaking a transaction. The predictive power of reputation assumes that a peer's past behavior is indicative of its future behavior. Feedback from all the peers that have previously interacted with a peer are aggregated to compute the first peer's reputation. Such a reputation mechanism, consequently, enables a community to police itself in order to establish social control.

A large amount of literature confirms the fact that reputation systems are an effective means of social control [3–9]. Within the bounds of their assumptions, these systems

demonstrate the ability to significantly reduce the number of malicious transactions and improve cooperation in a network. Despite these proposals, designing a robust and reliable reputation system is still largely an open challenge. The trust model used in existing reputation systems is extremely vulnerable to misleading or unfair feedback.

The challenge, therefore, lies in building a reputation system that is effective despite attacks that compromise the reliability of feedback. Most existing reputation systems either average a peer's lifetime ratings, resulting in the "increased trust by increased volume" vulnerability, or correlate service trust to imply feedback trust. The latter assumes that peers reputed to provide trustworthy service, in general, provide trustworthy feedback. While useful as a simple defense, such an assumption can easily fail or be manipulated. For example, colluding nodes can offer honest service for the express purpose of boosting their reputations so they can badmouth other peers. Countering false feedback attacks is a critical challenge that needs to be addressed to improve the performance of reputation systems, and consequently, peer-to-peer networks.

With this research challenge in mind, this paper offers three contributions. First, we discuss the various types of false feedback (or rating) attacks that can compromise existing reputation systems. We then discuss different reputation-based trust approaches that have been proposed for P2P networks and analyze their ability to overcome these false rating attacks. Next, we describe our scalable globally decoupled reputation model, which decouples each per-user reputation rating into a service rating and a feedback rating. The credibility of a peer's feedback in our model is weighed by its reputation as a service recommender, and not as a service provider. Finally, we use extensive evaluations to compare our trust model against some of the existing approaches, including the approach of averaging lifetime ratings, the coupled trust approach, and PeerTrust's personalized similarity trust metric, a system that employs a similar decoupled trust policy [10]. Our simulations show that decoupled trust models provide significantly more accurate reputations by detecting and isolating malicious peers. We also demonstrate the effectiveness and scalability of our decoupled system on peer traces gathered from *Maze*, a large scale peer-to-peer file-sharing system.

The remainder of the paper is organized as follows. We begin by identifying related work in Section 2. In Section 3, we describe our decoupled trust model and present our reputation system. Next, we classify unfair ratings attacks and discuss some reputation-based approaches employed to counter these attacks in Section 4. Our experimental evaluations in Section 5 perform detailed comparisons of our proposed trust model with respect to the different reputation models and different unfair ratings attacks discussed in Section 4. We evaluate our system using trace-driven simulations from the Maze file-sharing system in Section 6. Furthermore, we employ the Maze logs to compare the PeerTrust decoupled algorithm and our decoupling approach and highlight some vulnerabilities inherent to PeerTrust. Finally, we discuss implications of our system in Section 7 and conclude in Section 8.

2. RELATED WORK

Significant prior work has shown that reputation systems, if reliable, can effectively motivate trustworthiness and cooperation [3, 5, 7–13]. Reputation systems can build trust models using two approaches. One approach is to use only firsthand information to evaluate peers. While highly reliable, a firsthand-only approach does not employ all reputation information available in the network, making it highly inefficient and unscalable. Firsthand information proves sufficient if a peer locates honest service providers with which it repeatedly transacts [14].

Almost all reputation systems use global information, that is, peers aggregate opinions of all other peers that have interacted with them in the past. While global reputations are efficient and help to quickly detect misbehavior in the system, they are vulnerable to false ratings and collusion. One technique that incorporates global information is a simple averaging or summarizing of ratings. EBay, the largest online auction site, uses a reputation-based trust scheme where, after each transaction, buyers and sellers rate each other using the *Feedback Forum* [15]. Because EBay uses a central authority to manage all communication and coordination between peers, it essentially eliminates much of the complexity that exists in a decentralized system.

Simple summarizing schemes, in general, are highly vulnerable to malicious participants that increase their transaction volume to hide frequent misbehavior. A peer could increase its trust value by increasing its transaction volume, thereby hiding the fact that it frequently misbehaves at a certain rate. For example, a peer could undertake a thousand good transactions of low value (say, worth $1) and use the accumulated good reputation towards one dishonest transaction of high value (say, worth $1000). Additionally, if all ratings are given an equal weight, Sybil attacks and collusion are encouraged.

Therefore, to incorporate global information effectively, trust models must account for the credibility of the service raters using different trust propagation mechanisms. Examples include *transitive* trust, *coupled* trust, or *decoupled* trust. The underlying principle of *transitive trust* is if A trusts B and B trusts C, then it is likely that A trusts C. Reputation systems employ such web-of-trust chains to establish and propagate trust among peers. In general, longer chains imply greater risk of encountering a malicious link. Schemes like weighing ratings of a transitive chain by the reputation of the least reputed peer in the chain [16], employing a node distrust table [6], and using pretrusted peers [8] have been proposed.

Coupled trust approaches assume that peers reputed to provide trustworthy service, in general, will be likely to provide trustworthy feedback. We cite two well-known examples of reputation systems that rely on the correlated trust assumption. Aberer and Despotovic propose a decentralized reputation system for P2P networks where data is stored on a P-Grid [5]. Their system assumes that most network peers are honest, and reputations in the system are expressed as complaints. EigenTrust is a reputation system for P2P networks designed to combat the spread of fake files [8]. Each

peer is associated with a global trust value that reflects the experiences of all other peers with it. These values are used as a metric of reliability when choosing download sources.

Alternatively, some systems make use of *decoupled trust*, which involves using separate metrics to evaluate service trust and feedback trust [10, 11]. To decouple feedback trust from service trust in PeerTrust [10, 17], peers use a *personalized similarity measure* to more heavily weigh opinions of peers who have provided similar ratings for a common set of past partners. In a large P2P system, however, finding a statistically significant set of such past partners is a challenge. Peers are likely to make choices among a set of candidates for which there is no information. In Section 6.3, we empirically demonstrate how this lack of trust data information can impact the effectiveness and scalability of PeerTrust in its computation of trust values.

Confidant, another decoupled trust mechanism, attacks the problem of false ratings using a Bayesian approach in a mobile ad hoc network [11]. They distinguish between reputation, which they define as how well a node behaves in routing, and trust, which is how well it behaves in the reputation system. A node distributes only firsthand information to other nodes and only accepts other firsthand information if those opinions are similar to its own opinion. Compared to Confidant, where a node's referral is interpreted subjectively per node, our proposal produces a system-wide referrer rating per node.

Reputation ratings are normally associated with peers. But in some cases, a reputation rating is associated with a resource, for example, a file in a file-sharing P2P network [18]. Damiani et al. present a detailed discussion on the advantages and disadvantages of employing a pure resource-based or peer-based reputation and propose combining both reputations [3]. Storage overheads are substantially higher because the number of resources in any system tends to be significantly larger than the number of peers. Also, it is often not possible for a single resource to be widespread enough to have a sufficient number of raters for it. Credence [18] overcomes this problem by employing a web-of-trust in the absence of direct observations.

3. OUR SCALABLE DECOUPLED TRUST MODEL

We now discuss our decoupled reputation framework and describe the trust model used to update reputation ratings.

3.1. Reputation propagation framework

Our reputation system associates two sets of reputation ratings with each peer: an aggregated service rating (SR) and an aggregated feedback rating (FR). The service rating indicates a peer's trustworthiness as a service provider, for example, a peer's overall file-sharing behavior in a P2P file-sharing system. The feedback rating of a peer indicates its overall trustworthiness as a service recommender. Additionally, each peer maintains a list of peers that has rated it and the ratings provided by them. Service reputations are normalized values between 0.0 and 1.0 with 1.0 indicating a perfect service reputation. Similarly, feedback reputations are normalized values that range from 0.0 and 1.0 with 1.0 indicating a perfect rater. Initially, SR and FR are set to 1.0 for all peers.

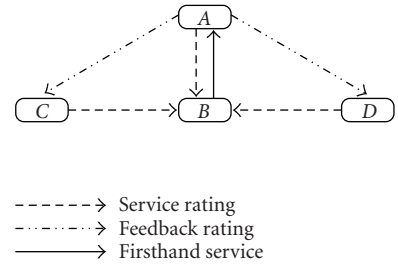


Figure 1: Decoupling service and feedback reputation: after interacting with B, peer A modifies B's service reputation and also modifies the feedback reputations of B's previous raters, C and D.

Consider a peer, A, that queries for a file. In order to make a decision about which responding peer with which to transact, A chooses the peer with the highest aggregated service rating. While this choice can result in an unbalanced load distribution in the network, a probabilistic approach can be employed to distribute load [8]. After finishing a transaction with service provider B, A provides B with either a rating of 0 (unsatisfactory) or 1 (satisfactory), depending on the outcome. This rating is weighted by $\mathrm{FR}(A)$, that is, the feedback rating of A. This weighting implies that A needs to be wellreputed as a feedback provider in order for its opinions to have an effect on B's service reputation. In other words, feedback from peers with higher feedback trust ratings will have more impact than those with lower feedback ratings. While we currently employ binary ratings to rate transaction outcomes, our design framework works for complex ratings schemes as well. For example, subjective ratings like Very Good, Good, OK, Bad, Very Bad can be mapped to quantitative rating values.

At the end of the transaction, A also needs to send feedback rating updates to all peers that had rated B earlier. A provides a rating of 1 to all peers that rated B with a value consistent with A's firsthand experience. In the case where the outcome of A's transaction with B did not match with a prior service rating, A generates a feedback rating of 0 to the originator of the rating. This rating is in turn weighted by A's feedback rating. This process is shown in Figure 1, where A interacts with B, updates B's service reputation, and updates the feedback ratings of C and D, who contributed to B's service reputation.

Feedback reputations pose additional storage and management overhead. By maintaining reputation information for only a window of recent transactions, the amount of overheads can be controlled. Our experiments in Section 5 demonstrate that the increase in reputation accuracy justifies this additional storage overhead.

We do not explicitly address storage and communication issues in our model, since they are largely orthogonal to our problem of decoupling reputation. Our trust model would

work with a number of different storage models. For example, peers can compute and maintain their reputations in a self-storing model [9]. While digital signatures and time stamps need to be built into the reputation system to ensure validity and integrity of the reputation data, such local storage schemes ensure data availability, thus eliminating the problem of a reputation storer being offline while the target is online. Alternatively, peer reputations can be stored and computed independently by third parties using distributed hash table- (DHT-) based approaches [5, 8]. The dissemination of reputations can leverage the communication protocol of the peer-to-peer network [3] or employ lookup schemes for structured storage [10].

3.2. The trust model

In this section, we formalize our trust metric for updating service and feedback reputations.

Our trust model is based on peer evaluation. Each peer observes two kinds of peer evaluations: *service rating*(s) after it serves as a service provider and *feedback rating*(s) after it provides service evaluations. For each transaction between two peers (i, j), where i represents the service provider and j represents the service requester, we define the following terms:

(i) *transaction rating* ($s_{i,j}$): j's firsthand observation of i's service for their most recent transaction (0: unsatisfactory, 1: satisfactory);
(ii) *transaction set*, T_i: let $|T_i|$ represent the cardinality of the set of peers that have transacted with peer i;
(iii) *peer i's average service rating*, $\mathrm{SR}(i)$: the life-time average rating of i based on peer evaluation;
(iv) *peer j's average feedback rating*, $\mathrm{FR}(j)$: the life-time average rating on j's feedback. In other words, this value is the average credibility of j's feedback on service satisfaction.

For each peer, i, its average service rating is an average of all the firsthand observations provided by peers with which it transacted, weighted by their individual feedback credibility. That is,

$$\mathrm{SR}(i) = \frac{1}{|T_i|} \cdot \sum_{j \in T_i} s_{i,j} \cdot \mathrm{FR}(j). \tag{1}$$

Like the service rating, each peer's feedback credibility is based on peer evaluation. In particular, at the end of each transaction (i, j), the service requester, j, not only sends feedback on i's service, but also updates feedback ratings of all peers that have previously transacted and rated i.

We define $f_{k,j}$ as the feedback rating of k by j based on j's current transaction with i and k's most recent transaction with i. If the service ratings are consistent, j rates k's feedback helpful or malicious. That is,

$$f_{k,j} = \begin{cases} 1, & s_{i,j} = s_{i,k}, \\ 0, & s_{i,j} \neq s_{i,k}. \end{cases} \tag{2}$$

The overall (average) feedback rating of k is the average of the most recent feedback ratings from its peers, weighed by their feedback credibility. The derivation is as follows:

$$\mathrm{FR}(k) = \frac{1}{|T_k^f|} \cdot \sum_{j \in T_k^f} f_{k,j} \cdot \mathrm{FR}(j), \tag{3}$$

where T_k^f represents the set of peers that have provided feedback ratings for k.

In this way, we incorporate peer evaluations on both the service provider and the requester in the computation of trust values making them robust to peer maliciousness.

3.3. Handling dynamic peer personalities

In our proposed trust metric, a peer's service or feedback rating is an aggregation of all the firsthand service or feedback observations it has received in its lifetime. This long-term aggregation makes our reputation system slow to react to changes in a peer's "personality." A peer can establish a good reputation in order to behave badly, without the results significantly impacting its reputation. In addition, honest peers can be subverted at any time by attackers and begin behaving badly. Therefore, peer reputations must be representative of more recent behavior rather than old ratings. To address the issue of dynamic behavior, we employ a simple window-based adaptation of our metric, a technique similar to that of PeerTrust [10].

We define the following:

(a) *peer i's reference service rating*, $\mathrm{SR}_{\mathrm{ref}}(i)$: i's service rating averaged over the set of M recent transactions.[1] That is,

$$\mathrm{SR}_{\mathrm{ref}}(i) = \frac{1}{M} \sum_{m=1}^{M} \mathrm{SR}_m(i), \tag{4}$$

where $\mathrm{SR}_m(i)$ represents i's average service rating after the $(M - m)$th most recent transactions provided by i. $\mathrm{SR}_{\mathrm{ref}}(i)$ provides a guideline to regulate i's dynamic behavior. In particular, the following actions are performed (in order) after each transaction with i:

$$\mathrm{SR}(i) = \frac{1}{|T_i|} \cdot \sum_{j \in T_i} s_{i,j} \cdot \mathrm{FR}(j)$$

{calculate average service rating},

$$\mathrm{SR}_{\mathrm{ref}}(i) = \frac{1}{M} \sum_{m=1}^{M} \mathrm{SR}_m(i)$$

{calculate reference service rating},

$$\mathrm{SR}(i) = \begin{cases} \mathrm{SR}_{\mathrm{ref}}(i), & \mathrm{SR}(i) - \mathrm{SR}_{\mathrm{ref}}(i) > \epsilon \\ \mathrm{SR}(i), & \text{otherwise} \end{cases}$$

[1] M is a design parameter. We assume that, in our system, transactions occur periodically. Hence, M also directly relates to the length of the average time window.

$$\{\text{check whether performance has dropped recently}\},$$
$$\mathrm{SR}_{m+1}(i) = \mathrm{SR}_m(i), \quad m = 1 \cdots M - 1$$
$$\{\text{update ratings over } M \text{ recent transactions}\},$$
$$\mathrm{SR}_1(i) = \mathrm{SR}(i); \tag{5}$$

(b) *peer j's reference feedback rating*, $\mathrm{FR}_{\mathrm{ref}}(j)$: j's service rating averaged over the set of M recent feedback ratings provided to j. $\mathrm{FR}_{\mathrm{ref}}(j)$ is derived in a manner similar to $\mathrm{SR}_{\mathrm{ref}}(i)$.

In (3.3), if $\mathrm{SR}_{\mathrm{ref}}(i)$ is smaller than $\mathrm{SR}(i)$ by a specified threshold ϵ, it means that a peer's performance has dropped significantly within some recent time frame. In this case, $\mathrm{SR}_{\mathrm{ref}}(i)$ is assigned as the peer's new reputation rating. This time-based adaptation allows reputations to be more sensitive to recent behavior and helps our system to quickly adapt to changes in peer behavior. This approach proves particularly adept in counteracting oscillating peers who alternate between honest and dishonest behavior in order to build and abuse good reputations.

4. SAFEGUARDING REPUTATIONS

The reliability of reputation systems is compromised by a variety of attacks. In this section, we discuss some important attacks to reputations, namely, unfair ratings attacks, dynamic peer personalities, and collusion. We also discuss three statistical-based approaches, namely, conventional, coupled, and decoupled PeerTrust PSM, used to safeguard reputation systems. Our experimental evaluations in Section 5 offer a detailed comparison of our proposed trust model with the three reputation models and with respect to the different reputation attacks discussed in this section.

4.1. Attacks to reputation systems

(i) *Unfair ratings.* An *honest* peer is one that is honest in providing service recommendations to other peers. A *malicious* peer, on the other hand, tries to subvert a system by falsely rating a bad transaction as good, and vice versa. This behavior could be due to jealousy, competition, or other malicious reasons. A malicious peer with a static personality (e.g., always behaving badly) is easily detected in the network. A *strategic* peer, on the other hand, is a malicious peer that may choose to behave honestly with some probability, in order to confuse other peers and cheat the reputation system.

(ii) *Dynamic (oscillating) peer personalities.* Some peers can exhibit a dynamic personality, that is, switching between an honest and dishonest behavior. Behavior changes can be based on the type or value of the transaction or the party involved at the other end. Reputation *milkers*, or *oscillating* peers, are one type of peer personality that builds a good reputation and then takes advantage of it to do harm.

(iii) *Collusion.* Dellarocas identifies four scenarios in which peers can intentionally try to "rig the system," resulting in biased reputation estimates [19]. In *ballot stuffing*, a colluding group inflates the colluder's reputation which then allows the colluder to use its good reputation towards other malicious motives. Similarly, in *badmouthing*, a malicious collective conspires against one or more peers in the network by assigning unfairly low ratings to the target peers, thereby, hurting their reputation. Finally, positive (and negative) *discrimination* arises when peers provide good (and poor) service to a few targeted peers. Controlled anonymity has been shown to avoid badmouthing and negative discrimination, while cluster filtering can be used to reduce ballot stuffing and positive discrimination [19].

(iv) *Sybil attacks.* Douceur has shown that unless there is a centrally trusted party, it is impractical to establish distinct identities (i.e., one identity for one entity) in a large-scale decentralized network [20]. There are no certificate authorities in a P2P network, and peers are free to generate their own identities. This availability of free identities results in the *whitewashing* attack, where a free-riding malicious peer rejoins the network under a new identity to avoid imposed penalties on its behavior. A peer can also generate a large number of identities or "Sybils" to maliciously increase the reputation of one or more of its identities. *Sybil-proofing* reputation systems is an open challenge to current reputation systems [21].

Current reputation schemes are highly vulnerable to tampering via ratings attacks including the above-mentioned unfair ratings attacks, collusion, and Sybil attacks. These vulnerabilities limit the reliability of reputations in predicting a peer's trustworthiness. By decoupling each per-peer reputation rating into a service rating and a feedback rating, a peer is accountable for its behavior both as a service provider and service recommender. Assuming that a majority of network peers are honest in nature, malicious peers that provide poor ratings to honest peers will have little agreement with the network as a whole. Their feedback credibility, consequently, will be low. In order to cheat the reputation system, an intelligent (or strategic) malicious peer may occasionally concur with the network majority and rate an honest peer correctly, thereby, getting itself a good feedback reputation. But, by ensuring that a good feedback and service reputation is difficult to gain and easy to lose, our decoupled trust approach forces a strategic malicious peer (or an oscillating peer) to constantly rebuild its reputation rating. A malicious peer spends a greater amount of time rebuilding its reputation rather than performing malicious transactions. In general, a greater number of feedback disagreements with honest peers results in a more rapid decline in maliciously acquired reputations. This discourages peers from providing incorrect service and feedback reputation ratings.

Similarly, a collusive group will give good ratings to peers within the group and false ratings to the outside network. Even one transaction with an honest peer, however, can bring down the service reputation of the malicious service provider and feedback reputations of the collusive group. Honest peers not only rate a colluding peer poorly for bad service, but also rate other colluding peers poorly for their incorrect feedback. A greater number of interactions with honest peers outside the colluding group results in a more rapid decline of reputations within the group. Additionally, by ensuring that a good reputation is difficult to gain and easy to lose, a malicious peer spends a greater amount of time colluding and

rebuilding its reputation rating rather than performing malicious transactions. Therefore, by reducing the productivity of unfair raters and colluders, our decoupled trust mechanism is able to curtail ratings attacks.

4.2. Reputation-based trust models

We now identify three statistical reputation-based trust approaches and discuss the effectiveness of these approaches in the presence of false feedback attacks. We will perform a detailed comparison of our proposed trust model with respect to these reputation models in our evaluation.

(i) *Conventional approach.* This approach is the simple technique of averaging service ratings in order to measure the trustworthiness of peers [7, 15]. The service trust rating of peer, i, denoted by $\mathrm{SR}_{\mathrm{conv}}(i)$, for the conventional approach, is derived as

$$\mathrm{SR}_{\mathrm{conv}}(i) = \frac{1}{|T_i|} \cdot \sum_{j \in T_i} s_{i,j}. \tag{6}$$

Here, $s_{i,j}$ is the rating given by peer j to i, valued at 0 (unsatisfactory) or 1 (satisfactory). T_i indicates the set of nodes that have previously transacted with i.

A simple averaging approach is flawed in several respects. A peer can easily increase its transaction volume to hide an occasional, or even, frequent misbehavior. Incremental ratings do not affect a peer once it has established a good reputation, thereby, giving a peer little incentive to behave honestly. For such a scheme, *decaying* a peer's reputation is important, that is, a peer's reputation should be representative of recent behavior rather than old behavior.

(ii) *Coupled approach.* This approach weighs the credibility of a peer's feedback by its reputation as a service provider [5, 8]. The service trust rating of peer i, denoted by $\mathrm{SR}_{\mathrm{coupled}}(i)$, for the coupled approach, is derived as

$$\mathrm{SR}_{\mathrm{coupled}}(i) = \frac{1}{|T_i|} \cdot \sum_{j \in T_i} s_{i,j} \cdot \mathrm{SR}_{\mathrm{coupled}}(j). \tag{7}$$

By taking into account the credibility of the feedback source, a coupled approach performs better than the simple averaging approach. However, a good service provider cannot be assumed to always provide good recommendations, and a bad service provider cannot be assumed to always provide bad recommendations. A peer providing honest service may provide false feedback about other peers' service due to jealousy or competition.

(iii) *PeerTrust personalized similarity measure.* PeerTrust PSM decouples feedback trust from service trust [10, 17]. PeerTrust uses a personalized similarity measure to more heavily weigh opinions of peers who have provided similar ratings for a common set of past partners. Each peer, x, in PeerTrust maintains a local copy of all feedback provided by the peer. This information is accessed up by a peer, y, that wishes to evaluate its feedback similarity with x. The root mean square or standard deviation (dissimilarity) of x's and y's feedbacks is used to compute their feedback similarity. While statistically hard to find a significant set of such overlapping past partners in a large-scale network, PeerTrust is reasonably robust compared to other approaches in handling unfair attacks and peer collusion.

In the following sections, we perform two sets of detailed experiments to evaluate the effectiveness, benefits, and scalability of our decoupled trust approach. The first set of experiments compares the effectiveness and benefits of our decoupled approach with the afore-mentioned approaches for reputation modeling, namely, conventional averaging, coupled service and feedback trust approach, and PeerTrust PSM. The second set of experiments, performed on Maze transaction logs, demonstrates the effectiveness and scalability of our reputation system on peers in that system. We also employ the Maze logs to compare the PeerTrust decoupled algorithm to our system-wide decoupling approach and highlight some of the vulnerabilities inherent in PeerTrust.

5. EXPERIMENTAL EVALUATION

We evaluate our decoupled trust model using a number of simulated and trace-driven experiments. In the following section, we describe our simulation setup, and examine the accuracy of our system in a variety of environments ranging from a network with malicious peers, strategic peers, peers with oscillating behavior, and colluding peers.

5.1. Simulation setup

We have implemented our simulator in *C* using tools included with the Stanford Graph-Base (SGB) [22]. We use graphs in the SGB platform to represent members of a P2P community. For our simulations, we use graphs of 100 to 5000 peers generated from the GT-ITM topology generator [23]. Table 1 summarizes the main parameters related to the peer model and the simulation. Our results are generated from a simulated community of 64 to 100 peers. We also run our experiments on a community of 5000 peers but observed no qualitative difference in the results.

Our network simulations proceed in cycles. We assume, for simplicity, that every peer in the network makes one transaction in each query cycle. With the help of the trust-based selection scheme, the peer requesting the service initiates a transaction with the peer that has the highest trust value. At the end of the transaction, the requesting peer gives the service provider peer a rating of either 0, indicating a bad transaction, or 1, indicating a satisfactory transaction.

The peer model includes the five types of behavior patterns discussed in section, namely, *honest*, *malicious*, *strategic*, *oscillating*, and *colluding*. Our experiments illustrate the effectiveness of our model against all these types of attacks and also show better performance when compared to the three existing reputation-based trust approaches, namely, the *conventional* approach of averaging lifetime ratings, the *coupled* approach of weighing feedback by the service reputation of its source, and PeerTrust's *personalized similarity measure.*

5.2. Metrics and experiments

In the first set of experiments, we employ three metrics in our simulations, namely, transaction success rate, trust

Table 1: Simulation parameters.

	Parameter	Value range	Default value
Peer model	Number of peers in the network	50–5000	100
	Percentage of honest peers	0–100	75
	Percentage of malicious peers	0–100	25
	Percentage of strategic peers	0–100	25
	Percentage of oscillating peers	0–100	5
	Percentage of colluding peers	0–25	10
	Percentage of peers responding to a query request	0–20	5
Simulation	Number of query cycles in one experiment	100–10000	100
	Number of experiments over which results are averaged	5	5

computation error, and computed reputation rating. First, we define the *transaction success rate* as the ratio of the number of successful transactions over the total number of transactions in the system. This metric enables us to illustrate the benefit of a trust-based peer selection scheme over a random peer selection process.

Second, we define *trust computation error* [10] to evaluate the effectiveness of our trust model against malicious and strategic peers. The trust computation error is the root mean square (RMS) of the computed trust value for all peers and the actual probability of those peers performing a satisfactory transaction, that is, 1 for good peers, and 0 for malicious peers. The RMS error is a value that ranges from 0 to 1. A lower RMS error indicates a more accurate reflection of a peer's trustworthiness.

Finally, in order to evaluate the effectiveness of our system against peer oscillation and collusion, we measure the *average reputation value* of the oscillating and colluding peers as transactions are being performed by them in the network. Periodically, measuring the reputation values enables us to measure the increase and decrease in reputation values for these peers as they perform malicious, collusive, and honest transactions.

5.3. Simulation results

We now present the results of our experiments and demonstrate the effectiveness of our decoupled trust model as compared to other reputation models. Data points in our figures represent an average of results from five randomized runs.

5.3.1. Benefit of trust-based peer selection

The first experiment demonstrates the benefits of a trust-based reputation scheme in a peer-to-peer system. A transaction is deemed successful if, at the end of the transaction, the service provider is given a rating of 1. We define the transaction success rate to be the ratio of the number of successful transactions over the total number of transactions. The experiment proceeds by having honest peers repeatedly initiate transactions. This method ensures completely honest feedback in the reputation system. As seen in Figure 2, without a trust model, there is a 50% probability of a transaction being satisfactory. When any type of trust model is used, however, a much higher success rate is achieved. All of the models have a success rate close to 100%. Clearly, a peer community with a higher transaction success rate is more productive. By avoiding transactions with untrustworthy peers, the number of unsatisfactory transactions is reduced. This experiment, therefore, indicates that having any kind of trust model in place provides a significant benefit for a peer-to-peer system.

5.3.2. Effectiveness against malicious behavior

In this experiment, our objective is to evaluate the effectiveness of our decoupled model against malicious peers as compared to other trust models. A malicious peer is defined as one that provides dishonest service and dishonest feedback at all times. We perform the evaluations after conducting 6,400 transactions over 100 peers, that is, an average of 100 transactions per peer. The percentage of malicious peers varies from 10% to 70% and the trust computation error is measured. We define the trust computation error [10] as the root mean square (RMS) of the computed trust value for all peers and the actual probability of those peers performing a satisfactory transaction, that is, 1 for good peers and 0 for malicious peers.

As Figure 3 shows, the conventional approach of averaging ratings is not effective against malicious behavior. This result occurs because the approach does not consider the credibility of the feedback provider and is particularly vulnerable to unfair ratings. We note that the correlated trust approach performs well when a small percentage of network peers is malicious. In the correlated approach, ratings assigned to the service provider at the end of a transaction are weighed by the service rating of the rater. That is, the feedback from those peers with higher service ratings will be weighed more than those with lower service ratings. When malicious peers exhibit static personalities, the assumption that a dishonest peer will provide dishonest feedback holds true. Hence, traditional reputation models that correlate service and feedback trust work well. Malicious peers are easily detected and avoided, resulting in a low trust computation error.

As malicious peers become the majority (>50%), however, they begin to overwhelm honest nodes, resulting in a significant increase in the trust computation error. Both our decoupled approach and the PeerTrust PSM models exhibit

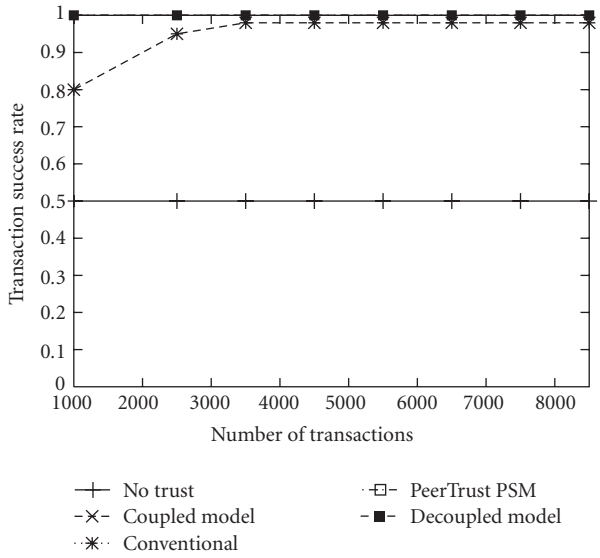


Figure 2: The benefit of a trust-based peer selection scheme.

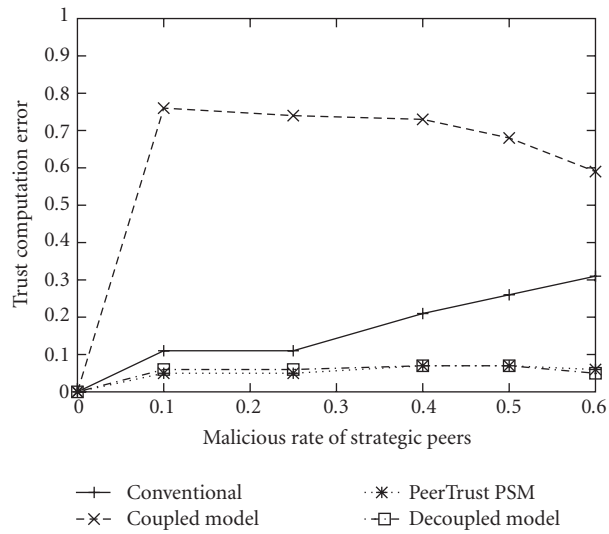


Figure 4: Trust computation error in a network with varying malicious rate of strategic peers.

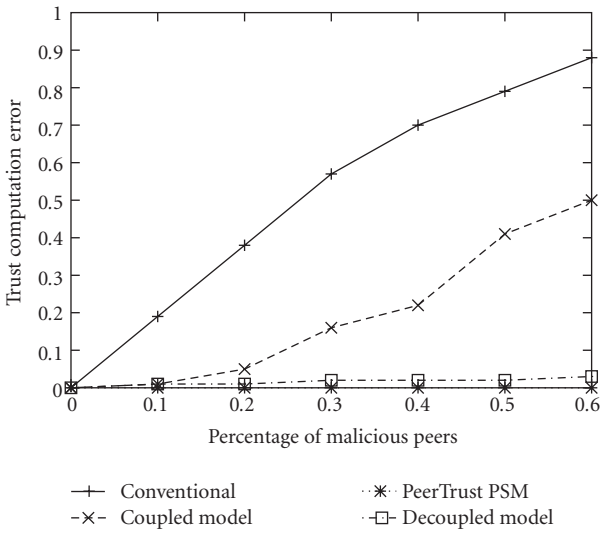


Figure 3: Trust computation error in a network with a varying percentage of malicious peers.

a trust computation error of nearly 0. This result occurs because both models are sensitive to the credibility of the feedback source. Again, as the malicious peers become the majority (> 50%), there is a slight increase in the trust computation error with our decoupled approach. This result demonstrates the natural collusion between dishonest nodes when they form a network majority.

5.3.3. Effectiveness against strategic behavior

The objective of this experiment is to evaluate the benefits of decoupling service and feedback trust compared to the correlated trust approach and the conventional approach. We introduce strategic behavior in this experiment. Strategic peers are malicious peers that try to "rig the system" by providing honest feedback, instead of dishonest feedback, in some cases. We use a network with 25% strategic peers and 75% honest peers. We vary the percent of maliciousness, X, such that a strategic peer will act maliciously, for service and feedback, for 10% to 100% of the total number of its transactions. It will provide honest feedback for the rest of its transactions. For this experiment, we define the trust computation error as the root mean square of the computed trust value for all peers and the actual probability of those peers performing a satisfactory transaction, that is, 1 for good peers, and 1-X for malicious peers.

A number of interesting observations can be made from Figure 4. Our decoupled approach and the PeerTrust PSM significantly outperform the conventional and the correlated trust approaches by reducing the number of malicious transactions and having a low trust computation error. By weighing service ratings with the credibility of the feedback source, both these approaches are able to detect strategic behavior. As the rate of malicious transactions by malicious peers increases beyond 60%, the trust computation error becomes nearly 0. An increase in the rate of malicious transactions results in a corresponding increase in the trust computation error for the conventional approach of averaging ratings. As this approach does not weigh the credibility of the feedback provider, it is unable to counter the strategic peers.

In the correlated approach, feedback from those peers with higher service ratings will weigh more than those with lower service ratings. By giving bad service but occasional good feedback, strategic peers are able to take advantage of the correlated trust assumption and fool the system. We also note that once the rate of malicious transactions increases beyond 40%, the trust computation error for the correlated model decreases. This result occurs because malicious peers are dishonest in providing service and feedback for more than 40% of their transactions, and hence confuse the system

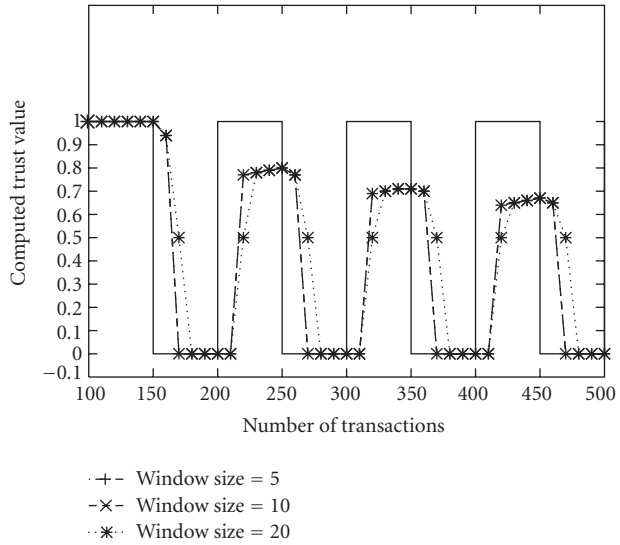


Figure 5: Effectiveness of our decoupled trust model against oscillating peer personalities.

less often. The coupled model, nonetheless, performs poorly in the presence of strategic peers. On the other hand, our decoupled model associates two ratings with each peer, one for its role as a service provider, and the other for its role as a service recommender. It is able to correctly identify peers as malicious service providers or sources of malicious feedback.

5.3.4. Effectiveness against oscillating behavior

Once a peer has established a good reputation in the network, it can abuse it by cheating occasionally. Honest peers can be subverted at any time and begin to behave badly. In order to motivate peers to perform honestly at all times, a peer's reputation must be representative of more recent behavior rather than an old established reputation. To address this issue, we calculate two reputation ratings for each peer. The first reputation rating is calculated over all the transactions undertaken by a peer, while the other is calculated over a subset of the ratings acquired by that peer in a recent window of transactions. As explained in (3.3), if the latter reputation value is smaller that the first value by a certain threshold, we assign it as the peer's new reputation rating.

The objective of this experiment is to illustrate the effectiveness of our decoupled approach against oscillating peer personalities. An oscillating peer is a malicious peer that builds and then abuses its reputation periodically. We simulate a community of 100 peers, with one oscillating peer and the rest completely honest. Each peer performs an average of 500 transactions. For this experiment, the oscillating peer is simulated to change its behavior every 50 transactions and we periodically measure the trust value of that peer as computed by an honest peer. Additionally, we vary the window size in our experiment in order to understand its effect on the computed trust value.

A number of interesting observations can be made from Figure 5. First, by employing our approach, a peer's reputation rating can drop quickly, but is hard to rebuild afterward. Second, a smaller window results in a more rapid reputation decay as compared to a larger window. With a large window, a peer is still able to take advantage of its available reputation and conduct malicious transactions. Performing continuous malicious transactions, however, results in an eventual breakdown of that peer's reputation. We also observe from the figure that a peer can never build its reputation on previous high levels once it cycles through periods of building and abusing its reputation.

5.3.5. Effectiveness against collusion

Service requesters and/or providers can intentionally "rig the system," resulting in biased reputation estimates [19]. In *ballot stuffing*, a colluding group inflates the reputations of its members, who can then leverage their good reputations for attacks. Similarly, in *badmouthing*, a malicious collective conspires against one or more participants in the network by assigning unfairly low ratings to them and, thereby, bringing down their reputation.

We ran experiments to observe the effectiveness of our model against a ballot stuffing type of collusion. The objective of the first experiment is to observe the reputation of a colluding group as computed by an honest peer. A colluding peer is a malicious peer that provides dishonest service and feedback at all times. When transacting with each other, however, two colluding peers boost each other's reputation rating. For this experiment, we use a network of 100 peers with 10% colluders and 90% honest peers. We observe similar results for a higher percentage of colluding peers. The experiment proceeds with each peer randomly performing transactions with other peers. We vary the number of transactions from 100, that is, an average of 1 transaction per peer, to 10 000, that is, an average of 100 transactions per peer. Colluding peers collude with one another at a fixed rate of 10%, 50%, or 80% of their total transactions. We monitor the average reputation of the colluding group per query cycle.

As seen in Figure 6, the service reputation rating of the colluding group declines rapidly in the presence of honest peers. In the decoupled approach, honest peers not only rate a colluding peer poorly for bad service, but also rate other colluding peers poorly for their incorrect feedbacks.

The objective of our next experiment is to determine the effectiveness of our decoupled approach against varying rates of collusion. We use a network of 100 peers with 10% colluders and 90% honest peers. The experiment proceeds as each peer randomly performs transactions with other peers. We vary the probability, X, that a colluding peer undertakes a collusive transaction. A collusive transaction implies that both the colluding requester and provider boost each other's service and feedback reputations, respectively. A colluding peer will behave dishonestly, in feedback and service, for any noncollusive transaction. We measure the average service reputation of the colluding group at the end of all the transactions.

As seen in Figure 7, a decoupled approach is more sensitive to peer collusion compared to a coupled approach. A

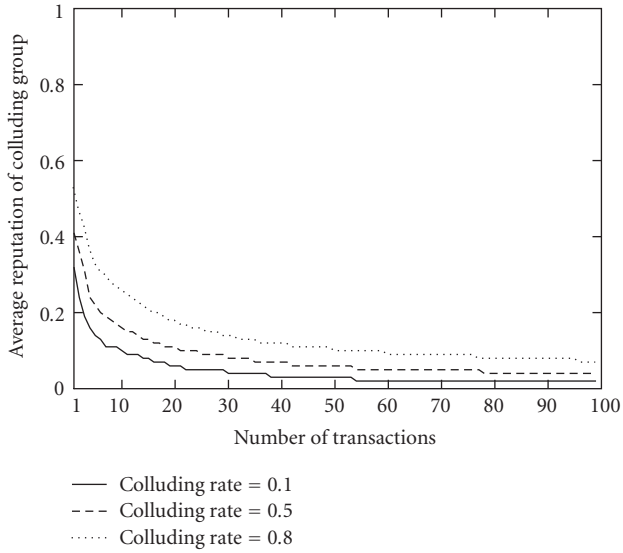


Figure 6: Effectiveness of our decoupled trust model against collusion.

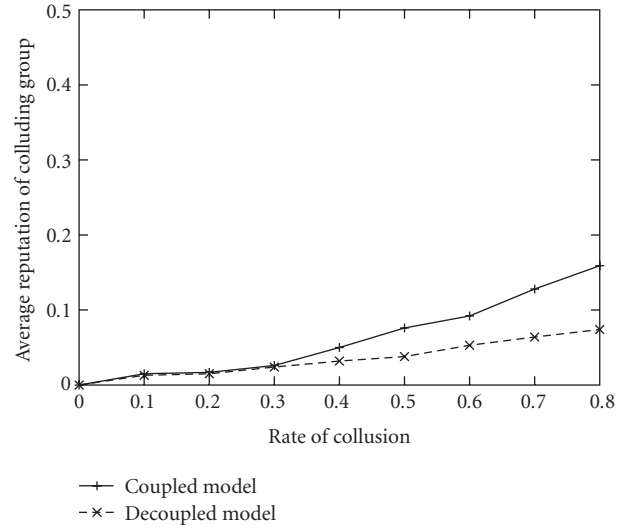


Figure 7: Effectiveness of our decoupled trust model with varying rates of collusion.

collusive rate of about 90% results in an average rating of less than 0.1 for the group. With the coupled approach, on the other hand, the colluding group is able to maintain about twice that rating for the same collusive rate. The assumption of static personalities holds true for low collusive rates as colluding peers fool the system to a lesser extent. However, as the collusive rate increases, a colluding peer exhibits a more dynamic personality, which the coupled approach is unable to handle. Also, a higher rate of collusion implies that a colluding peer spends a greater amount of time colluding and rebuilding its reputation rating rather than performing malicious transactions with honest peers. Clearly, our decoupled trust approach results in less productivity for colluding peers as compared to the coupled approach.

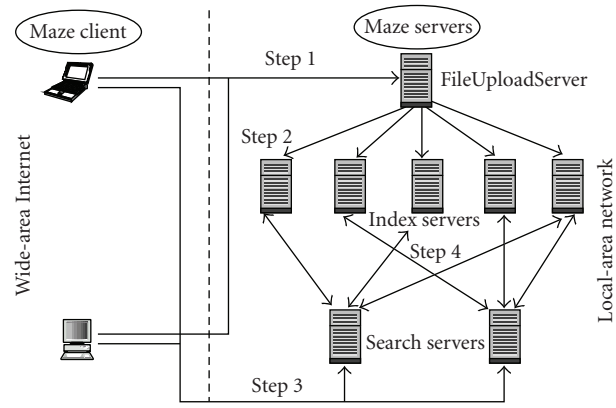


Figure 8: Architecture of the Maze file-sharing network.

6. TRACE-DRIVEN EXPERIMENTS

The previous section described our results conducted on a simulated peer community. In this section, we use transaction logs from *Maze*, a large deployed P2P file-sharing network, to drive our simulated experiments. We give a brief background on Maze [24] before presenting our results.

Maze is a popular Napster-like peer-to-peer network designed, implemented, and deployed by an academic research team at Peking University, Beijing, China. Maze is currently deployed across a large number of hosts inside China's internal network. Maze currently includes a user population of 1.4 million users and supports searches on 140 million files, 20 million of which are unique, totaling over 226TB of data. At any given time, there are over 50 000 users online, and over 1.3 million file transfers per day [24].

Maze uses a simple centralized architecture where metadata for all user files is stored on a set of central index servers. As shown in Figure 8, clients forward metadata to a FileUploadServer, which is then forwarded to index servers. Other clients issue queries to search servers. Because all transactions go through central servers managed by the team at Peking University, they monitor and log all users, metadata, and transaction records.

We perform our experiments using a sequence of transaction logs gathered from February 19, 2005 to March 24, 2005. While this log includes more than 32 million file transfers, we limit our analysis to a truncated data set that includes transactions conducted between the first 5000 users. On average, each peer performs 15 transactions.

The format of the Maze transaction logs is presented in Table 2. *UIDc* and *UIDs* refer to the file requester and provider, respectively. The *GlobalTime* and *downloadSize* fields refer to the transfer end time and the transfer size for a given session. If *downloadSize* is less than *totalSize*, it implies an incomplete transfer. *End/start* is the session time. Related work has shown that significant amount of colluding behavior can be observed from Maze transactions logs [24]. While we cannot conclusively determine the full extent of colluding behavior, we take a pessimistic approach, and define a file transfer (transaction) as failed (or malicious) if

Table 2: Maze transaction logs.

UIDc UIDs	GlobalTime	Download size	Totalsize	End/start
799 141532 302	1109 606 402	2959 816	2959 816	42
799 141532 302	1109 606 402	9240 638	9240 638	97
572 989318 395	1109 606 402	600 000	222 119 423	1
572 989318 395	1109 606 402	600 000	161 026 413	8
621 848435 436	1109 606 403	600 000	165 177 856	287
841 802538 283	1109 606 403	7491 756	7491 756	105
655 843791 333	1109 606 404	1684 920	1864 920	548
805 000278 815	1109 606 404	1800 000	3514 636	522
295 726634 900	1109 606 404	600 000	13 258 424	259 237

the *downloadsize* field equals zero or is less than half of the *totalsize* field.

We now evaluate the effectiveness of our reputation-based trust model against malicious and strategic peers in Maze. We employ the trust computation error metric to evaluate our model.

6.1. Effectiveness against malicious behavior

In this experiment, our objective is to evaluate the effectiveness of our decoupled approach against malicious peers in Maze. A malicious peer always behaves dishonestly when providing feedback and service. We identify a malicious peer as one that has failed in more than X% of its total transactions as a service provider. The extreme case is when X is 100%, that is, when a peer is defined as malicious only if it has failed in every single transaction. The total percentage of malicious peers, in this worst case scenario, is about 40% of all the peers.

We vary the percentage of malicious peers in the community by varying X, from 5% to 40%. The experiment proceeds as each peer randomly performs transactions with another peer. At the end of about 75 000 transactions covering 5000 peers, an honest peer is chosen to evaluate the trustworthiness of all peers and the trust computation error is measured.

As seen in Figure 9, when a small percentage of network peers malicious, the correlated trust approach performs as well as our decoupled ratings approach. In the correlated approach, ratings assigned to the service provider at the end of a transaction are weighed by the service rating of the rater. Hence, when peers exhibit static personalities, the assumption that a dishonest peer will provide dishonest feedback holds true and reputation models that correlate service and feedback trust work well. We note that these observations made from the Maze transaction logs are similar to the results observed in our simulated community. The trust computation error in Figure 3, however, is lower than in Figure 9 since there are more transactions per peer in the former experiment. This result confirms that the performance of a reputation system increases with an increasing number of transactions in the community. Fewer transactions per peer results in less accurate trust ratings.

This experiment also shows that when a peer community like Maze employs a trust management scheme, it observes a higher transaction success rate and greater productivity.

6.2. Effectiveness against strategic behavior

We now discuss the impact of strategic peers on trust computation. The objective of this experiment is to evaluate the effectiveness of our decoupled approach against strategic peers in Maze. A strategic peer is a malicious peer that tries to fool the reputation system by behaving honestly in some of its transactions. We use a network with 40% strategic peers and 60% honest peers. We vary the rate, X, that a strategic peer will act maliciously, for service and feedback, from 10% to 100%.

The experiment proceeds as each peer performs random transactions with other peers. At the end of about 75 000 transactions over 5000 peers, an honest peer is chosen to evaluate the trustworthiness of all peers. We define the trust computation error as the root mean square of the computed trust value for all peers and the actual probability of those peers performing a satisfactory transaction, that is, 1 for good peers, and 1-X for malicious peers.

Figure 10 clearly indicates that our decoupled approach outperforms the correlated trust approach by having a lower trust computation error. Our approach is able to correctly identify peers as malicious service providers or malicious recommenders. The trust computation error is nearly 0 in most cases. On the other hand, strategic peers are able to take advantage of the correlated trust assumption and fool the coupled model. Once the rate of malicious transactions increases beyond 50%, however, the trust computation error for the correlated model decreases. This result occurs because malicious peers become more predictable when they are dishonest in more transactions than honest.

The last two experiments evaluated the effectiveness of our decoupled trust model against malicious and strategic peers in Maze. The goal of our next set of experiments is to employ the complete Maze dataset to compare the PeerTrust decoupled algorithm with our globally decoupled trust model and highlight some of the vulnerabilities inherent in PeerTrust.

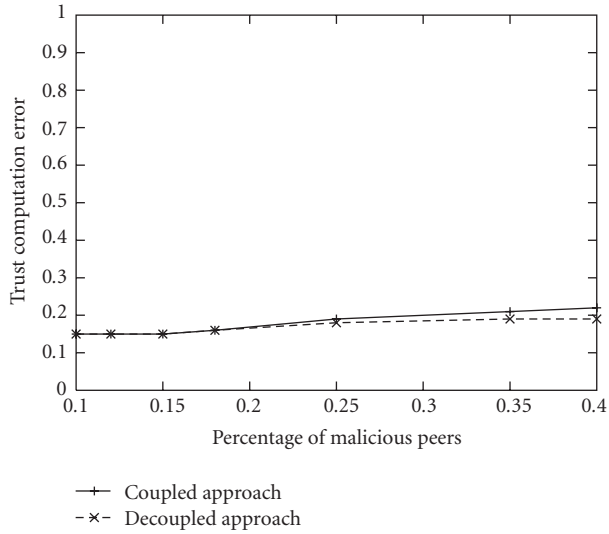


Figure 9: Trust computation error with respect to percentage of malicious peers.

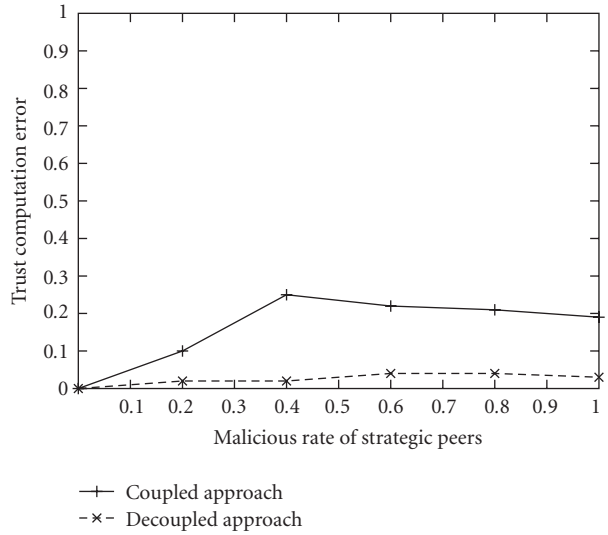


Figure 10: Trust computation error in a network with a varying malicious rate of strategic peers.

6.3. Comparison with PeerTrust's personalized similarity metric

An alternate mechanism proposed for decoupled reputations is that of PeerTrust. PeerTrust proposes the use of a personalized similarity measure (*PeerTrust PSM*) to more heavily weigh opinions of peers who have provided similar ratings for a common set of past partners. To measure the feedback credibility of any peer, x, a peer, y, computes the feedback similarity between y and x over the common set of peers with which they have interacted in the past.

While this personalized credibility provides PeerTrust with its flexibility and robustness against maliciousness, it is limited by the availability of trust data required for computing reputations. In this section, we empirically demonstrate two vulnerabilities: (a) PeerTrust's vulnerability to an overlapping set of past partners, and (b) network churn's effect on the availability of data needed for trust computations and, consequently, impact of the effectiveness and scalability of PeerTrust in large distributed P2P systems. The following experiments employ transaction logs from Maze, a P2P file-sharing network deployed in China, to compare the performance of PeerTrust PSM and our decoupled approach in overcoming the abovementioned vulnerabilities.

6.3.1. Overlapping set of past partners

In large P2P systems, finding a statistically significant set of overlapping partners can be challenging. In general, a greater number of common past partners will result in an increasingly accurate personalized feedback similarity measure for PeerTrust. But an absence of such common past partners can result in peers making arbitrary decisions among a set of candidates for which there is no information.

In the first experiment, we randomly sample 15.5 million unique transaction Maze peers. As Figure 11 illustrates, 92% of the pairs do not share even a single common partner. Approximately 4% of the pairs share only one common past partner and 2% of pairs share two common past partners. Clearly, there is not enough experience with a breadth of peers for an accurate measure of PeerTrust PSM.

6.3.2. Network churn

The second limitation of PeerTrust PSM, and of reputation systems in general, is their vulnerability to network churn. Each peer, x, in the PeerTrust algorithm maintains a local copy of all feedback provided by it. This information is accessed by a peer, y, that wishes to evaluate its feedback similarity with peer x. High peer turnover (or churn) in P2P systems impacts the availability of trust information needed to dynamically compute feedback credibilities. PeerTrust proposes an approximate trust calculation algorithm (PSM/ATC) where cached service trust and feedback similarity values are stored by each peer for reference in future transactions. However, such cached copies cannot be generated and maintained for every peer in the network.

In the second experiment, we evaluate the accuracy of the PeerTrust algorithm and our decoupled algorithm in computing trust while experiencing churn as modeled by the Gnutella churn trace. We conduct 13,517 transactions (one transaction per Gnutella peer) over a 60-hour interval (the time interval for the Gnutella trace logs). We map transaction histories of 13,517 random Maze peers to the Gnutella peers. Each time a Gnutella requester, x, wishes to evaluate a provider, y, it searches for local feedback data stored by y's previous transaction partner set, Z. If $z \in Z$ is unavailable at the transaction time, then x is unable to compute its credibility similarity with z. For this experiment, we assume that x does not hold a locally cached credibility measure for z.

We define the *percentage of successful computations* for a given provider, y, as the ratio of the number of y's past transaction partners online at the given time with respect to

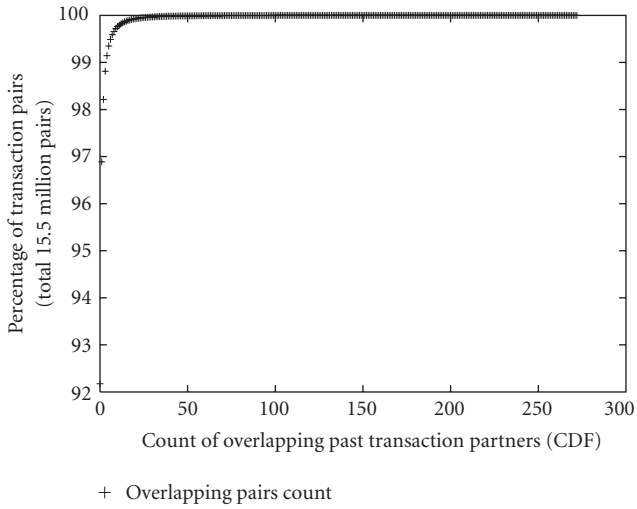


FIGURE 11: Vulnerability of PeerTrust algorithm to overlapping past transaction histories.

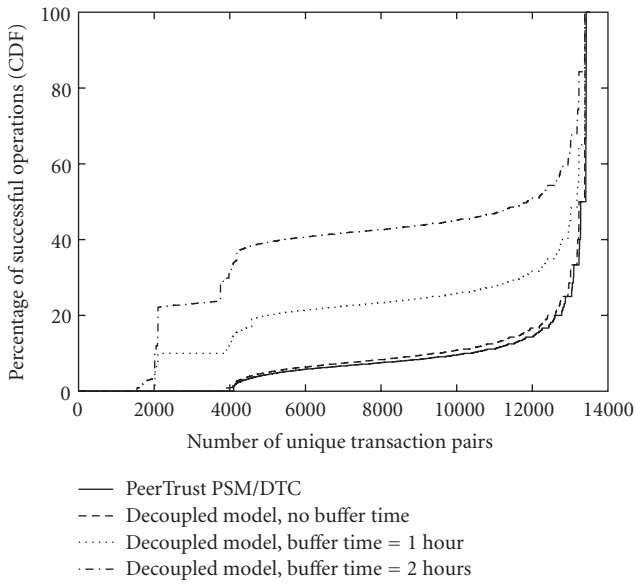


FIGURE 12: Performance of PeerTrust and our decoupled approach in overcoming network churn.

the total number of y's past transaction partners required to compute y's true trust value as observed by requester x. As Figure 12 illustrates, 30% of transactions employing the PeerTrust algorithm have 0 successful trust computations. This implies that requesters in 30% of the cases found no past partners online for dynamic trust computations. About 9% of all transactions had around 5% of the transactors online for trust computations. Only 1% of the transactions had 50% successful operations, and less than 0.70% had 100% successful operations for computing their PeerTrust PSM values.

The two experiments clearly indicate the vulnerability of the PeerTrust algorithm to a lack of trust data. On the other hand, our decoupled approach lends itself reasonably well to networks experiencing churn. At the end of each transaction, a requester, x, provides feedback updates to all peers that previously transacted with provider y. We define *buffer time* as the amount of time a requester will hold feedback update values if the target is not online. We vary the buffer time from one to two hours. A requester periodically probes targets for online availability every 60 seconds. Here, the percentage of successful operations is given by the ratio of total number of successful feedback updates over the total number of feedback updates that need to be communicated by our decoupled algorithm. As seen in Figure 12, our decoupled approach is equally vulnerable to network churn as PeerTrust. However, we overcome the problem of churn by employing a buffer time window which enables requesters to communicate their feedback update values to a greater number of target peers. As expected, the larger the buffer time window, the greater the number of successful feedback update operations.

7. DISCUSSION

Reputations are not a guaranteed solution to the problem of maliciousness in peer-to-peer networks. They only serve as a risk-management technique, reducing the chances of a peer deceiving another in an online transaction. A reputation system assumes that a peer's past behavior is indicative of its future behavior. This assumption, however, proves to be incorrect when a peer is compromised. By ensuring that a good reputation is difficult to gain and easy to lose, our decoupled approach is robust to common attack strategies. We safeguard our reputation system from malicious, colluding, and oscillating peers.

We note several implications of using a decoupled reputation system. First, the use of dual service and feedback reputations is likely to impact the way users choose with whom they transact. In a traditional reputation system, a peer requesting a service checks the service reputations of available peers to select a trustworthy peer with which to transact. Because feedback reputations are also available in our decoupled system, service providers now have an incentive to choose from whom they accept requests. A provider might avoid peers that have poor feedback reputations, since those peers might inaccurately rate the service provider's performance. In an actual system, this behavior will likely lead to the isolation of both nodes who perform bad service and nodes who give bad ratings.

The dual reputation approach imposes additional storage and management overhead for the feedback reputation. However, we note that the feedback reputation can be stored and managed using the same mechanisms as service reputations. The additional overhead is clearly justified given the increase in reputation accuracy and higher transaction success rates. Finally, while users generate service ratings, feedback ratings are generated automatically without user involvement.

8. CONCLUSIONS

Reputation systems establish peer trustworthiness in P2P networks. A number of feedback attacks, however, compromise the reliability of reputations generated by a reputation

system. In this paper, we discuss these attacks and the different reputation-based trust approaches that have been proposed to counter these attacks. After a detailed theoretical and experimental analysis of existing reputation mechanisms, we recommend that decoupled trust approaches be employed as they result in more robust reputations.

In this paper, we propose our own scalable globally decoupled trust model and demonstrate, using simulation-based and trace-based experiments, reputation improvement by removing the assumption of correlation between service quality and feedback quality. We demonstrate the effectiveness and scalability of our decoupled approach as compared to PeerTrust, an alternative mechanism proposed for decoupled reputations. Our decoupled approach incorporates global reputations of both the service provider and the requester in the computation of trust values and, in this way, makes our model more robust to peer maliciousness.

ACKNOWLEDGMENTS

The authors would like to thank the anonymous reviewers for their helpful feedback. This work is supported in part by the National Science Foundation under Career Award no. CNS-0546216 and by DARPA under the Control Plane program.

REFERENCES

[1] Gnutella, "The gnutella protocol specification v0.4," 2001.

[2] Symantec, "Vbs.gnutella worm," 2000, http://securityresponse.symantec.com/avcenter/venc/data/vbs.gnutella.html.

[3] E. Damiani, S. Vimercati, S. Paraboschi, P. Samarati, and F. Violante, "A reputation-based approach for choosing reliable resources in peer-to-peer networks," in *Proceedings of the 9th ACM Conference on Computer and Communications Security (CCS '02)*, pp. 207–216, Washington, DC, USA, November 2002.

[4] K. Walsh and E. G. Sirer, "Fighting peer-to-peer SPAM and decoys with object reputation," in *Proceeding of the 3rd Workshop on Economics of Peer-to-Peer Systems (P2PECON '05)*, pp. 138–143, Philadelphia, Pa, USA, August 2005.

[5] K. Aberer and Z. Despotovic, "Managing trust in a peer-2-peer information system," in *Proceedings of the 10th International Conference on Information and Knowledge Management (CIKM '01)*, pp. 310–317, Atlanta, Ga, USA, November 2001.

[6] K. Burton, "Design of the openprivacy distributed reputation system," May 2002, http://www.peerfear.org/papers/openprivacy-reputation.pdf.

[7] P. Dewan and P. Dasgupta, "Pride: peer-to-peer reputation infrastructure for decentralized environments," in *Proceedings of the 13th International Conference on World Wide Web (WWW '04)*, pp. 1212–1213, New York, NY, USA, May 2004.

[8] S. D. Kamvar, M. T. Schlosser, and H. Garcia-Molina, "The eigentrust algorithm for reputation management in P2P networks," in *Proceedings of the 12th International Conference on World Wide Web (WWW '03)*, pp. 640–651, Budapest, Hungary, May 2003.

[9] B. C. Ooi, C. Y. Liau, and K.-L. Tan, "Managing trust in peer-to-peer systems using reputation-based techniques," in *Proceedings of the 4th International Conference on Advances in Web-Age Information Management (WAIM '03)*, pp. 2–12, Chengdu, China, August 2003.

[10] L. Xiong and L. Liu, "PeerTrust: supporting reputation-based trust for peer-to-peer electronic communities," *IEEE Transactions on Knowledge and Data Engineering*, vol. 16, no. 7, pp. 843–857, 2004.

[11] S. Buchegger and J. Le Boudec, "A robust reputation system for P2P and mobile ad-hoc networks," in *Proceedings of the 2nd Workshop on the Economics of Peer-to-Peer Systems (P2PECON '04)*, Cambridge, Mass, USA, June 2004.

[12] R. Sherwood, S. Lee, and B. Bhattacharjee, "Cooperative peer groups in NICE," *Computer Networks*, vol. 50, no. 4, pp. 523–544, 2006.

[13] Z. Zhang, S. Chen, and M. Yoon, "MARCH: a distributed incentive scheme for peer-to-peer networks," in *Proceedings of the 26th IEEE International Conference on Computer Communications (INFOCOM '07)*, pp. 1091–1099, Anchorage, Alaska, USA, May 2007.

[14] S. Marti and H. Garcia-Molina, "Identity crisis: anonymity vs. reputation in P2P systems," in *Proceedings of the 3rd International Conference on Peer-to-Peer Computing (P2P '03)*, pp. 134–141, Linkvping, Sweden, September 2003.

[15] eBay, "ebay," 2005, http://www.ebay.com/.

[16] M. Feldman, K. Lai, I. Stoica, and J. Chuang, "Robust incentive techniques for peer-to-peer networks," in *Proceedings of the 5th ACM Conference on Electronic Commerce (EC '04)*, vol. 5, pp. 102–111, New York, NY, USA, May 2004.

[17] M. Srivatsa, L. Xiong, and L. Liu, "TrustGuard: countering vulnerabilities in reputation management for decentralized overlay networks," in *Proceedings of the 14th International Conference on World Wide Web (WWW '05)*, pp. 422–431, Chiba, Japan, May 2005.

[18] K. Walsh and E. G. Sirer, "Experience with an object reputation system for peer-to-peer filesharing," in *Proceedings of the 3rd Symposium on Networked System Design and Implementation (NSDI '06)*, San Jose, Calif, USA, May 2006.

[19] C. Dellarocas, "Immunizing online reputation reporting systems against unfair ratings and discriminatory behavior," in *Proceedings of the 2nd ACM Conference on Electronic Commerce (EC '00)*, pp. 150–157, Minneapolis, Minn, USA, October 2000.

[20] J. Douceur, "The sybil attack," in *Proceedings of the 1st International Workshop on Peer-to-Peer Systems (IPTPS '02)*, pp. 251–260, Cambridge, Mass, USA, March 2002.

[21] A. Cheng and E. Friedman, "Sybilproof reputation mechanisms," in *Proceedings of the 3rd Workshop on Economics of Peer-to-Peer Systems (P2PECON '05)*, pp. 128–132, Philadelphia, Pa, USA, August 2005.

[22] D. E. Knuth, *The Stanford GraphBase: A Platform for Combinatorial Computing*, ACM Press, New York, NY, USA, 1993.

[23] K. L. Calvert, M. B. Doar, and E. W. Zegura, "Modeling internet topology," *IEEE Communications Magazine*, vol. 35, no. 6, pp. 160–163, 1997.

[24] M. Yang, H. Chen, B. Y. Zhao, Y. Dai, and Z. Zhang, "Deployment of a large-scale peer-to-peer social network," in *Proceedings of the 1st Workshop on Real Large Distributed Systems (WORLDS '04)*, San Francisco, Calif, USA, December 2004.