Securing Anonymity in P2P Network

Byung Ryong Kim[(1)] & Ki Chang Kim[(2)] & Yoo Sung Kim[(2)]

[(1)] School of Computer Science & engineering, Inha university, 253 Yonghyun-dong, Nam-gu, Incheon 402-751, Korea
doolyn@super.inha.ac.kr

[(2)] School of Information & Communication engineering, Inha university, 253 Yonghyun-dong, Nam-gu, Incheon 402-751, Korea
kichang@inha.ac.kr, yskim@inha.ac.kr

Abstract

Basically flooding-based P2P systems provide anonymity and thus it is not possible to find the initial sender of packet and the designated receiver of that packet. However it does not provide anonymity where the IP addresses of nodes uploading and downloading contents are revealed. So in order to maintain anonymity we propose and test our techniques that the receiver node of response packets on retrieval query is agent node and the agent node provides contents service between server and client. Through the proposed techniques it was found that the identity of node is secured without using encryption techniques which have been deployed in the former anonymity protection techniques and control data communications through all the nodes located between server and client. The application of this concept of which result was evaluated may be extended to many other current P2P systems under operation.

1. Introduction

The recent P2P retrieval systems can largely be divided into flooding-based and distributed hash table-based models. FreeNet[2] and Gnutella[1,15] belong to Flooding model and Tapestry[3,5], CAN[4] and Chord[6] to distributed hash table model.

P2P systems are too free and irresponsible to secure reliability as achieved in server-client environment. However it cannot manage the nodes by nature and each node should maintain itself with independent authority that each node must have anonymity. Hence we propose packet-preemptive proxy service techniques that maintain both high speed and anonymity in flooding-based P2P file share systems requiring anonymity.

In a flooding-based model broadcasted retrieval query and ping packet basically provide anonymity but dynamic routing[16] is not used so that they do not support anonymity when uploading and downloading. Therefore this may result in unintentional exposure of node information in P2P network in which non-specific majority is participating. Thus attacks, such as denial-of-service and storage overflow, may occur to the exposed node, whether it is malicious or not.[7,8]

The existing techniques to secure anonymity include MUTE[10], Onion Routing[12,13], Crowds[11] and Mantis[9]. In order for MUTE to protect anonymity, file share is made through other clients such as jondo. However this slows speed in a large file because it transfers file through many jondos. In Mantis, model supplementing this consideration, UDP channel is used without passing through jondos but this requires additional control data communication for UDP channel. Onion Routing uses data encryption to secure anonymity. Client should connect to proxy performing encryption to firstly connect to Onion Routing. Crowds was developed for user privacy while browsing web site and which is similar to MUTE. Client participating in Crowds requests contents not to server but to other client participating in Crowds to get desired contents. Anonymity is guaranteed by this technique.

Our goal is to preserve anonymity and provide retrieval and file share service in a quick and easy way without using additional control data communication for UDP communication and file sending method passing through number of jondos.

2. Summary

The core of this paper to protect anonymity is proxy service by agent node receiving QueryHit between server and client. We discovered the techniques to easily preserve anonymity in flooding-based model using the properties of Query, QueryHit and PUSH packets used in Gnutella protocol in an effort to find simple way to secure anonymity without using existing complicated techniques.

Passing through lots of intermediate nodes(jondos), QueryHit is sent to client on the basis of dynamic routing. When one of these nodes initially receives QueryHit the node is preempted by relay node. After replacing IP address and Port number of QueryHit components by relay node's own ip and port, the relay node sends it to client through dynamic routing. And this information is stored into table of which key is server's session UUID value. Therefore the changed information can be known later on using session UUID value. QueryHit, replaced through dynamic routing, is sent to client. Client is unable to know the server's address because of replacement of QueryHit. Download is requested to relay node because of the replacement. Using server's session UUID value PUSH packet is created by relay node to which download service is requested. Request to transfer file is made by sending the created PUSH packet to server. Then receiving PUSH packet the server communicates with relay node according to Gnutella protocol. When it starts receiving file from the server the file is sent to client without delay. Since agent node is dynamically preempted when one transaction is finished, the relay role is terminated although it is the same server-client pair. In the next transaction dynamically other node is preempted.

3. Existing Techniques to Secure Anonymity

Techniques to secure anonymity of server and client include MUTE, Onion Routing, Crowds, and Mantis as shown on Fig. 1. MUTE does not provide file share service because server is directly connected to client in order to secure anonymity. Passing through many intermediate nodes such as jondo it sends data(information/contents). So sending high capacity file such as .avi or .divx will incur sizable waste of bandwidth because it passes through many nodes.

In order to secure anonymous connection Onion Routing uses data encryption to conceal routing header and make statistical computation hard to detect routing path. To make the first

connection to Onion Routing, client must make routing path and connect to proxy that encrypts data. Then client gets to destination following the routing path through Onion routers. Each router removes one layer of cryption. By repeating this process client knows the next onion router and finally reaches to the final destination. Inversely when getting back it reaches to the final destination by adding encrypted data layer one by one. Although it secures anonymity, it needs proxy between network infrastructure and application and costs a lot for the encryption.

Crowds is developed to protect user's privacy during web browsing. To get contents client participating in Crowds does not directly request contents to server having known the server's address but requests contents to other jondos participating in Crowds and anonymity is maintained this way.

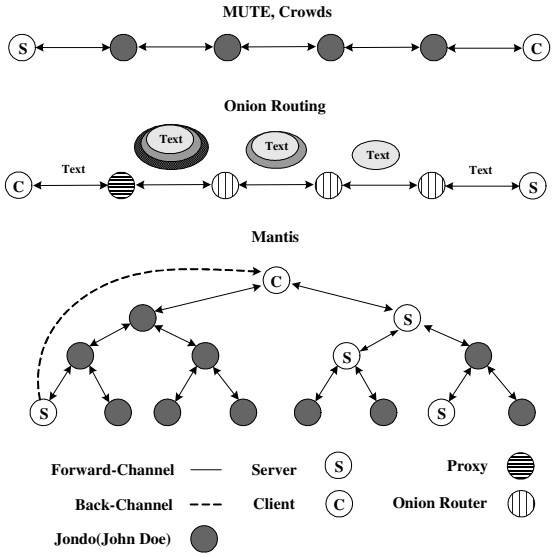

Figure 1. Existing Techniques to Secure Anonymity

Mantis is similar to Crowds but it instantly sends answer without passing through jondo as shown on Fig. 1. Concealing its own IP address, server sends file using UDP but control data communication is necessary to control packet loss aroused in the course of UDP communication between server and client and to control retransmitting. Control data communication is performed by passing through many jondos between server and client.

4. Service Model

The packet-preemptive proxy service model that we propose is based on Gnutella protocol. Packets used here are Query, QueryHit, and Push. In this section the basic definition of the proposed model and communication protocol will be covered.

4.1. Concept

Our service model is based on dynamic routing. Using dynamic routing enables to select agent node that well understands the relation between server and client and enables the selected agent node to perform proxy role of server and client. In the course of carrying out proxy service, we used Query, QueryHit and PUSH packets which have been used in general flooding based model. The format of each packet is shown on Fig 2. Bold border indicates packet header.

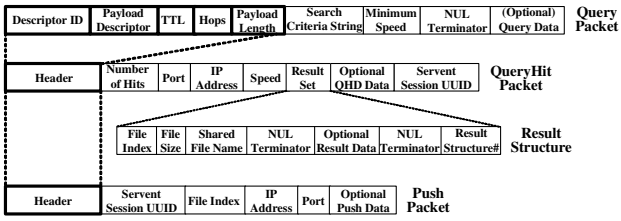

Figure 2. The Format of Query, QueryHit and PUSH Packet

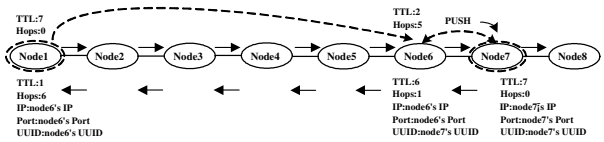

Figure 3. Contents Download by Switched QueryHit

Assuming that TTL is 7 and overlap receiving is not allowed in Fig 3, if node1 broadcasts Query then it is sent to node8 by dynamic routing. QueryHit is sent at node7 and node6, the first recipient of this packet, replaces IP and Port of this packet by its own IP and Port. Then node6 remembers session UUID value of node7 to create PUSH packet. The replaced QueryHit is sent to node1 by dynamic routing. Since the information of node having retrieved contents is replaced from node7 to node6, node1 requests download to node6.

Node6 that has received HTTP GET request, knows that contents are in node7, comparing session UUID value which it remembers with server session UUID value requested by HTTP GET. node6 creates PUSH packet and sends it to node7. node7 has received PUSH packet so that it begins to send file according to Gnutella protocol. node6 receives file and simultaneously sends the file to node1. node1 and node7 are secured with respect to anonymity for node6 takes a proxy role between node1 and node7. If download request is concentrated in a particular relay node, because relay role can be assigned to the neighboring node that it can prevents relay load from concentrating into specific a node.

4.2. A Case that server node is in firewall/NAT environment

If the IP of QueryHit header is private IP, the concept described in 4.1 is to be applied, as it is, to the first node receiving QueryHit from server node located in firewall/NAT environment. This is because to receive QueryHit packet means that the server and relay node is connected. Therefore it is solved if the relay node sends PUSH packet to server node. Originally PUSH packet was made to share file with Firewalled Servent

4.3. Case that relay node(agent) is in firewall/ NAT environment

If the first node receiving QueryHit is under firewall/NAT environment, client cannot request service directly to relay node. In this case without replacing QueryHit relay node(node6) sends it to the neighboring node through dynamic routing. So the relay role is to be assigned to the neighboring node(node5). The delegated node conforms to the packet-preemptive proxy service techniques we proposed in 4.1.

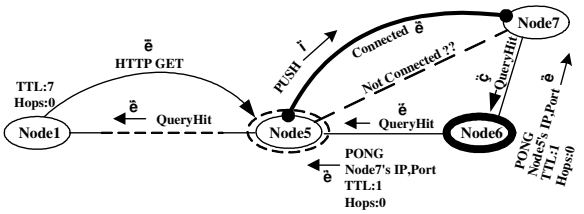

Figure 4. The Process flow in case that relay node is in firewall/NAT environment

Although the delegation is made as shown on Fig 5 if there is no direct connection between the new relay node(node5) and server(node7), how will the PUSH packet be sent to server? To solve this problem the following procedure is to be performed to firewalled relay node (node6).

1. Firewalled relay node pretends to receive PING from server and new relay node(node5) and send PONG as response value.
2. Then not replacing QueryHit it sends QueryHit to new relay node.
3. New relay node receiving PONG and server try to connect to each other. New relay node replaces packet and sends it to client.
4. Client sends HTTP GET to new relay node.

In the long run, Firewalled relay node delegates relay role by not replacing QueryHit but sending it to the neighboring node

5. Simulation and result

Simulation was based on minism simulator[14]. Minism generates a random topology with a particular node degree, then runs a few simulations with a given ttl to see how many nodes are reachable and how many packets are wasted. We applied our service algorisms to it and executed test by developing new simulator.

The number of nodes participating in P2P network was maximum 10^5, the number of connection maximum 7 and TTL value 5. Network was randomly composed and client broadcasted Query. It was assumed that every node having contents among the nodes receiving Query sends QueryHit and client downloads only one choosing from many QueryHits. Four aspects were evaluated; first, if there is a case that relay node is not selected or not, second, if double relay node is selected or not, third, if download requests are concentrated to a certain relay node, and lastly, we evaluated jondo overhead which was chosen as relay node.

Fig. 5 shows the average rate of download that did not use relay node among the whole 10^3 contents downloads. The number of nodes sharing file was set as 10% to 50% of the total nodes and each was evaluated. According to the test result, download without relay node can be made since our service uses TTL and Hops but when the number of node was more than 500 almost every download service was made via relay node. And double selection was not found.

To evaluate whether downloads are concentrated on a specific node it was assumed that retrievable contents are only in specific nodes and simulation was made to find whether there is a case that preemption is excessively made to a specific node among the neighboring nodes. When it was assumed that the most popular contents are only in a specific server, the relay rate of neighboring nodes was evaluated. The test result suggested when the number of nodes was very small the difference was big but in general it displayed uniform distribution. Also it suggested that download relay role was uniformly performed by server's neighboring nodes which may be selected as relay node.

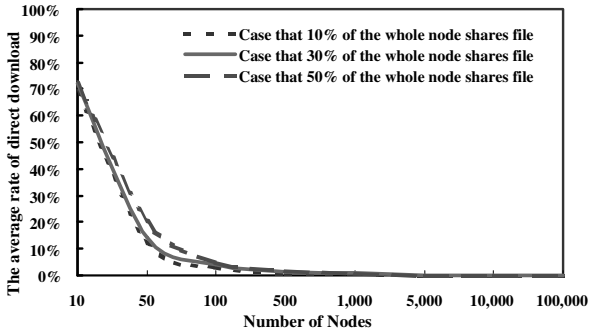

Figure 5. The average rate of download that did not use relay node

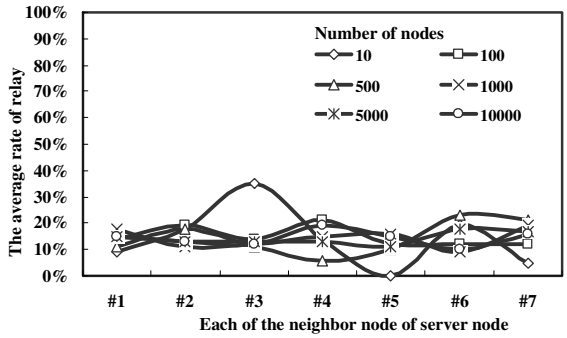

Figure 6. Rate of relay by 7 neighboring nodes among 103 downloads on specific server

Fig. 7 compares the amount of packets relayed by jondos in packet preemptive proxy service vs. MUTE.Our tests were performed in the network simulator Simnet[16], with the client and server transferring 1000 packets through a several relay jondo.Packet amount was produced by calculating the number of packets relayed by relay jondo.

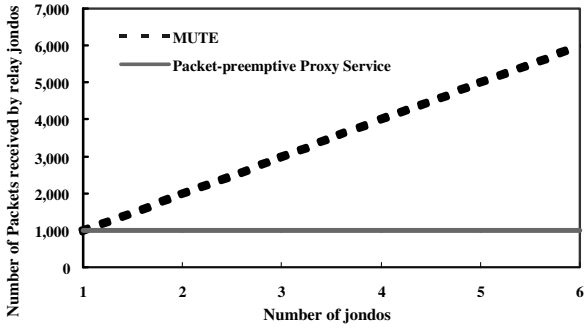

Figure 7. Packets forwarded by relay Jondos in packet-preemptive proxy service vs. MUTE

Basically bulk-data sent from crowds and MUTE is relayed hop-by-hop in P2P systems. But lots of jondos are not used in packet preemptive proxy service techniques. The relay role is performed by a jondo uniquely preempted between the server and client. Bulk-data is maximum 32kb including 16Kb of the actual data and header in Mute. When sending the maximum 32Kb packet if bulk multimedia file is sent through each jondo it can be expected that increasing jondo number will cause the increase in relayed packets as well. But The number of relayed packets was constant in packet preemptive proxy

service since it preempts only the unique jondo as relay node as shown on Fig. 3.

6. Discussion

Our aim is to preserve anonymity between server and client without such methods as encryption, control data transfer passing through many jondos for UDP communication and passing through many jondos connected between server and client. The mentioned methods require high cost for encryption or additional control data communication using many jondos for transfer control. Moreover downloading large data aroused heavy network overhead because it passes through many jondos. And download speed will be that of the slowest jondo on the link

In order to solve these problems, we proposed and tested our service techniques. Anonymity of server and client was maintained by preempting the first jondo receiving retrieval query response packet as relay jondo. Therefore in the techniques proposed in this paper when sending file it passes through not many jondos but just only one relay jondo. The test results found that complete anonymity was not preserved because occasionally it does not pass through relay jondo in case of very small P2P. Except that, anonymity was fully secured. Therefore if network size is not so small, we evaluate with the proposed techniques anonymity can preserved without encryption technique, UDP communication and file share service technique via many jondos.

Previous studies were complicated but the advantages of the proposed techniques are its simplicity and easiness applicable to existing P2P network. We believe that our service is very efficient in terms of low cost and minimizing the loss of anonymity when providing P2P file share service.

7. Conclusion

Anonymity is important issue in P2P file sharing systems. We have studied to find simple way to preserve anonymity without data transfer passing through many jondos and additional control data transfer for UDP communications through many jondos.

Consequently we proposed and implemented our service systems to secure anonymity in both server and client side. The core of this service is that the random jondo, located in transfer path of QueryHit sent by server, is selected as agent node and proxy service for data transfer is provided between server and client by the selected jondo.

Implementation showed that file share is performed with anonymity secured between the server and client since relay node provides proxy service between the two parties. The test result showed that the preempted relay node was never overlapped and the chance for the server's neighboring nodes to be selected as relay node displayed uniform distribution. Therefore it is expected that this can be effectively used in not only the current Gnutella but in many other P2P file share systems.

We will try to preserve complete anonymity of server and client in network. With this proposed techniques, anonymity of server and client can be preserved only when there is at least one node in transfer path of QueryHit. Therefore we will study how to choose other relay node outside transfer path of QueryHit.

Acknowledgements

This work has been supported by INHA UNIVERSITY Research Grant.

References

[1] *The Annotated Gnutella Protocol Specification v0.4 (1).*, http://rfc-gnutella.source forge.net/developer/stable/index. Html

[2] *The Freenet Project.*, http://freenet.sourceforge.net/

[3] Kirsten Hildrum, John Kubiatowicz, Satish Rao and Ben Y. Zhao. (2004) *Distributed Object Location in a Dynamic Network*, Theory of Computing Systems.

[4] Sylvia Ratnasamy, Paul Francis, Mark Handley, Richard Karp, Scott Schenker, *A scalable content-addressable network*, Proceedings of the 2001 conference on Applications, technologies, architectures, and protocols for computer communications table of contents.

[5] Ben Y. Zhao, Ling Huang, Jeremy Stribling, Sean C. Rhea, Anthony D. Joseph, and John Kubiatowicz. (2004) *Tapestry: A Resilient Global-scale Overlay for Service Deployment*, IEEE Journal on Selected Areas in Communications.

[6] Ion Stoica, Robert Morris, David Liben-Nowell, David R. Karger, M. Frans Kaashoek, Frank Dabek, Hari Balakrishnan. (2003) *Chord: a scalable peer-to-peer lookup protocol for internet applications*, IEEE/ACM Transactions on Networking.

[7] Neil Daswani, Hector Garcia-Molina. (2002) *Query-flood DoS attacks in gnutella*, Proceedings of the 9th ACM conference on Computer and communications security table of contents.

[8] P. Krishna Gummadi, Stefan Saroiu, Steven D. Gribble. (2002) *A measurement study of Napster and Gnutella as examples of peer-to-peer file sharing systems*, ACM SIGCOMM Computer Communication Review.

[9] Stephen C. Bono, Christopher A. Soghoian, Fabian Monrose. (2004) *Mantis: A Lightweight, Server-Anonymity Preserving, Searchable P2P*, Information Security Institute of The Johns Hopkins University, Technical Report TR-2004-01-B-ISI-JHU.

[10] *MUTE: Simple, Anonymous File Sharing*., http://mute-net.sourceforge.net/

[11] Michael K. Reiter, Aviel D. Rubin. (1998) *Crowds: anonymity for Web transactions*, ACM Transactions on Information and System Security (TISSEC).

[12] Roger Dingledine, Nick Mathewson, Paul Syverson. (2004) *Tor: The Second-Generation Onion Router*, Proceedings of the 13th USENIX Security Symposium.

[13] Michael G. Reed and Paul F. Syverson. (1999) *Onion Routing*, Proceeding of AIPA '99.

[14] *Gnutella Developer Forum*., http://groups.yahoo.com/ group/the_gdf/

[15] The Gnutella Protocol Specification v0.41 Document Revision 1.2.

[16] A. Oram, Peer-to-Peer, p.106-107, O'Reilly, Mar, 2001.