Abstract

For many internet applications, it is essential and even critical to protect the identity of participants. This paper presents a protocol for anonymous communication over internet. Our protocol provides sender-, receiver-, and sender-receiver anonymity. Our System is designed to provide anonymity under a rather strong adversarial model with low latency of data delivery and the link utilization. We present a description of the protocol, an analysis of its anonymity and communication efficiency, and evaluate its performance using message-level simulations.

1 Introduction

With the rapid growth and public acceptance of the Internet as a means of communication, anonymity has become an essential requirement for many Internet applications [7]. Anonymity protects the identity of one or both endpoints of a communication, i.e. no one, including other parties in the communication should be able to identify the anonymous parties [1]. Sender anonymity protects the identity of the original sender from the receiver(s) of the message and other third parties. IP multicast can be considered as an example of practical sender anonymous communication, since a sender of packets is hard to be recognized. Receiver anonymity protects the identity of the receiver from the original sender and third parties. An ordinary mess media such as newspaper is an example of receiver anonymous communication where indented recipient is impossible to be identified. Sender-Receiver anonymity protects the identity of both the sender and the receiver from each other and other third parties. Internet forum, blog can be considered the examples of sender-receiver anonymity where no one knows who is the sender and who is the receiver expect the sender or the receiver themselves.

We present a system provides all three kind of anonymity described above, specifically, our system supports bi-way anonymous communications. Our system is built on top of peer-to-peer networks which is completely distributed, self-organized, and scalable with thousands of active participants may communicate simultaneously. Our system can provide high level of anonymity while reducing the latency of data delivery and the link utilization. We achieve the anonymity and reductions by making use of optimized application layer multicast communication. The number of peers joined to a multicast tree is dynamic and unknown to peers, and these properties that make multicast useful for anonymity. Also, our system is designed to against passive and active attacks from outsider nodes and insider peers.

The basic approach is that we first initialize the system by building an efficient multicast tree in a decentralized manner, and then senders send messages by broadcasting it through the multicast tree with itself as the root. Any peer sets up its own filter to get messages it interested in. In that way, both sender and receiver anonymity is achieved.

In Section 2, we overview related work. We describe our protocol in detail in section 3. In section 4, we present various attacks and analysis the anonymity provided by our system. In section 5, we analysis the performance of the protocol. We show and discuss the results from our message-level simulator in Section 6. We offer concluding remarks in Section 7.

2 Related Works

2.1 Dining Cryptographers (DC-net)

DC-net [3] assumes a public key infrastructure, and users send encrypted broadcasts to the entire group thus achieving receiver anonymity. All members of the group are made aware of when a message is sent, only one user can send at a time, so it takes additional bandwidth to handle collisions and contention [13]. A DC-net participant fixes its anonymity vs bandwidth trade off when joining the system; there are no provisions to resale that trade off when others join the system.

2.2 Xor-Trees

Xor-Trees [5] provides sender-, receiver-, and sender-receiver anonymity. In Xor-trees, only a single uer may send at any one time in an Xor-Tree, thus, the performance degrades due to collisions as the number of users increase [13]. Also Xor-tree requires each participating node has complete knowledge of the network, which could not scale well in large size networks.

2.3 Onion Ring

Onion ring [11] is composed of a fixed set of routers with complete knowledge of all other onion routers. The connection is made to an Onion Routing Proxy, and proxy makes a path through the rerouting network [13]. The last rerouter connects to a destination proxy that connects to the destination host. When the anonymous socket is set up, an end-to-end layered asymmetric cryptography is used for distributing symmetric keys [1]. After the setup, the symmetric keys are used between the rerouters on the path to transport data. Onion Ring provides sender anonymity.

2.4 Crowds

Crowds [12] [7] protocol is similar in operation to Onion Routing; the difference is that the path through cooperating proxies is chosen randomly, on a hop-by-hop basis [7]. Once a path out of the crowd is chosen, it is used for all anonymous communication from the initiator to any responder within a 24-hour period [7]. Subsequent packets between the initiator and responder always follow the same path. Reply messages in crowds follow the same path until a periodic path reformation occurs, usually hourly. This design is prone to trace pack attacks.

2.5 Hordes

Hordes [7] is an application level anonymity system similar to crowds which adds support for anonymous multicast receivers. Hordes relies on the deployment of IP multicast, a technology that has yet to receive wide scale adoption for a variety of reasons[10].

2.6 Tarzan

Tarzan [6] is a system designed to provide anonymous communication at the IP-level [1]. All participants are potential rerouters and there are no centralized components [1]. When a participant wants to communicate with another host, a variable length path is build by choosing a path randomly and session keys are distributed to rerouters on the path using asymmetric cryptography [1]. They system uses dummy traffic to make traffic analysis harder. The path is chosen at random at the initiating endpoint [1].

Requesters in tarzan must predetermine message paths, which requires them to have knowledge of a significant portion of the network [10]. To accomplish this, peer discovery in tarzan is implemented using a gossip-based protocol with the aim of producing a fully connected network of peers. Such an architecture limit tarzan's scalability [10].

2.7 P5

P5 [13] allows secure anonymous connections between a hierarchy of progressively smaller broadcast groups and allows individual users to trade off anonymity for communication efficiency. Because the P5 logical broadcast hierarchy is a binary tree which is constructed using the public keys instead of basing on physical connection and network resource usage, the resulting tree may be very inefficient with respect to end-to-end delay and link usage. Further, to construct of the tree, peers need to consult an "oracle" which maintains an up to date list of channel memberships, which could not scale well in large size network if the "oracle" is achieved in distributed manner or could be prone to single point of failure if the "oracle" is a centralized server.

3 Our Solution

3.1 Structure

3.1.1 Topology

In our system, we organize all the participating peers into an application layer multicast tree, and each leaf peer connecting a number of randomly selected leaf peers. That is for intermediate peers, each peer connects to a parent, a number of children, and for leaf peers, each peer connects to one parent, and a number of friends. As shown in Figure 3-1, all the peers form a multicast tree, and each leaf peer connected to at least two other leaf peers.

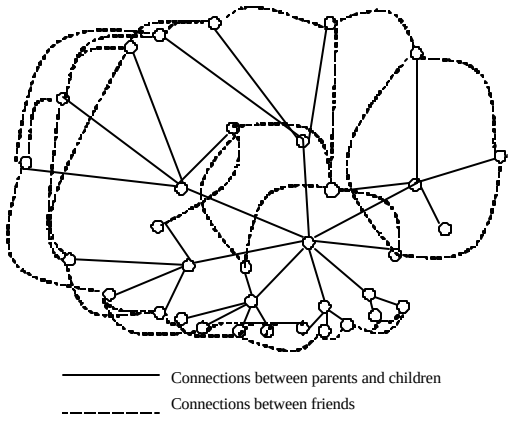

Figure 3-1. System Topology

In order to improve the network efficiency and reduce system communication cost, we adapt [9]'s ideas on building a tree based on the network condition, including latency, and bandwidth. And the goal of tree building is to mimic the underlining physical network connections,

so that the link usage and the latency could be minimized. The reason, for which we add friends, is to keep each leaf peer connect to at least n friends to provide a desired level of anonymity (Details discussed in section 3.2.2). The more number each peer connected to, the higher level of anonymity.

3.1.2 Initialization

In our solution, we assume that each peer knows a list of other peers from bootstrap, and the peers in the list are called neighbors of the peer. If a peer does not know neighbors, it could advertise itself through other ways, like URL, email, and record the peers who response to its advertisement, and then set up its own neighbor list.

We achieve the tree building in the following three steps. First, the source sends out tree-creating-message to all of its neighbors. Second, for each peer that receives the tree-creating-message will broadcast the tree-create-message with TTL(Time To Live) equals to two, and if it is interested in the service, it will send ion message to its parent (it locate its parent through the information given in the tree-creating-message). Third, for each peer receives joining request will follow a tree-joining algorithm. This algorithm is used to deal with the received joining request and adjust the fan-out according to the capability of the peer. If the peer has the capability to accommodate one more children, then accepts it, otherwise, it will compare its worst child with the requesting one, and accept the better one, reject the worse one. Peers can always join/rejoin the tree at any time from any peer in the tree.

To add friends, peers are required to give a list of peers they know to their children, so that their children could include the list into their list and then randomly pick peer from lists to pass down until the leaf peers. A leaf peer searches for friends by sending out be-friend request. Peers, who receive a be-friend-request, could either accept it by adding the requester to its list, or reject it and send back a list of peers they know, from which the requester could choose to become friend. The requester either adds a friend upon receiving the acceptance or selects and send s request to another peer until the lower bound achieved.

3.1.3 Maintenance

To maintain the system topology, we use two heartbeat timers at each peer in the tree. One timer is used to send out heartbeat message periodically and the other is for detecting connections to its neighbors (parent, children and friends). If it does not receive its parent's heartbeat on time, then it will start selecting new parent. If it does not receive its children or friends' heartbeat, delete the child/friend from its children/friend list.

While in peer-to-peer network environment, the peer-to-peer overlay networks are dynamic, heterogeneous and unpredictable. The peers who were offline or new to the service can join the system topology from the root of the tree. In our system topology, when the peer is unavailable, it will only have impact on its neighbors (parent, children and friends). *For its parent:* since it will query its children at each heartbeat, once it detects the child is not available, it will just delete the child from its children list. *For its children:* who lost its connection to its parent will rejoin the tree in the following way: since it stores a candidate parent list during tree building process, and it knows a set of peers who is in the tree, like its previous ancestors, its previous children, it could ask these peers or their decanters to be its new parent. *For its friends:* since it will query its friends at each heartbeat, once it detects the friend is not available, it will just delete the friend from its friend list. If the number of friends drops under the lower bound, it will select and add new friends.

3.1.4 Optimization

In the process of optimization, the overlay topology mimics the real physical network connections so that the difference between them is minimized. The peers execute the optimization algorithm periodically to adjust their positions in the Tree dynamically to achieve better system performance and scalability. Peers in the tree receive heartbeat message from their parent. The heartbeat messages contain the children list of the parent, the ancestor list of the parent, etc. The peer follows the grid optimization algorithm to find a better parent upon receiving the heartbeat message from its parent.

The basic idea of the optimization algorithm is that: first, if there is no penalty for the peer to get the message from its sibling, that's the physical path from the parent to the peer goes through the sibling, then we use the sibling as the peer's new parent, then we can reduce the link stress (the number of duplicated packets go through the physical link) between the parent to the sibling, that means we saved network resources. Second, if it is possible for the peer to move closer to the source in the tree, that is if the capacity of its ancestor is not full, it's more efficient to add the peer under its ancestor than under its parent. A threshold is set to adjust the computing time of the peer to forward a message. Peers receive join requests from siblings or decanters will follow tree-joining algorithm.

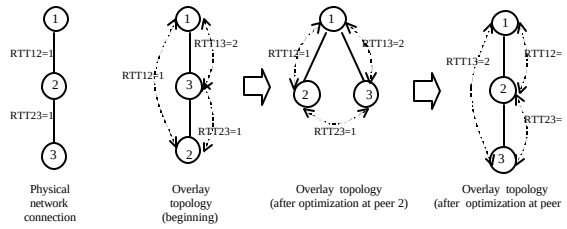

Figure 3-1 Example of Overlay Topology Optimization

An example is shown in figure 3-1. At the beginning, the overlay topology is different from the physical network connection, where peer 3 is in the middle in the former one, and peer 2 is in the middle in the latter one. The messages from peer 1 to peer 2 always go through peer 3 in the overlay topology, but according to the underlying network connection, the message should first pass peer 2 and then peer 3 and then back to peer 2, which shows the overlay topology is not efficient. In the first step of the optimization, peer 2 will compare the RTT 13 plus RTT23 with RTT12, and since RTT 12 is shorter, peer 2 will move directly under peer1. Then in the optimization of peer 3, it compares RTT12 plus RTT23 with RTT13, and since there is no difference between the two paths, peer 3 will move down to be child of peer 2 reduce the like stress between peer 1 and peer 2.

3.2 Anonymous Communications

The system provides three types of anonymous communications, sender anonymous communication, receiver anonymous communication and sender-receiver anonymous communications. These anonymous communications are achieved through the cooperation of all participant peers. Each peer provide four basic functions, based on these basic functions, the system provides four system functions, and then any peer can use the system for anonymous communications as they desire.

3.2.1 Basic functions

To support the system level functions, and the anonymous communications between peers, each peer performs the following four functions

- **Send**

Any peer can be the originator of a message, and function send is used when a peer wants to send messages. For each message to be send, the message should first be encrypted using the private key of the first hop peer, and also the message should be padded to the system pre-define size. If the message's length exceed the pre-define size, a section random number and a message sequence number should be included in the messages' content, so the receivers are able to re-construct the whole message. The send function's first hop receiver is different with respect to the peers' location in the tree. For the intermediate peers, they send their messages by broadcasting it to its parent and all of its children. For the leaf peers, they send their message by randomly choosing one of its friends, and pass the message to that friend.

- **Receive**

In our system, all the messages are sent to all of the peers. Each peer set a filter for all the messages come in, only interested messages could be accepted and read, so no one knows which peers read which messages except the peers themselves.

- **Broadcast**

All peers in the system are responsible for message relay. For any message come in, the peers should broadcast it to all of its neighbors. Any peer broadcast messages by sending the message to its parent and all of its children except the one from whom received the message.

- **Pass**

Pass is the function preformed by leaf peers only, and the function is used to pass one message from one leaf peer to another or send out. Any leave peers who get messages from their friends, with probability of m% to send the message to a randomly choose friend or broadcast it by send the message to its parent. The selection of m will affect the latency of the message delivery and the level of the anonymity.

3.2.2 System functions

To support anonymous communication, the peers in the system cooperate to perform the following four system functions.

- **Message Broadcasting**

Message broadcasting is used to broadcast a message to all of the peers in the system. Any peer can start message broadcasting. The sender sends the message to all of its children and its parent, who ever received the message passes it to its parent and all of its children except the one who sends it. Eventually, all the nodes in the tree will receive the message.

- **Secure channel for leaves**

The secure channel is used for the leaf peers to broadcast message without expose their identity. Since there is no children for leaf peers, if leaf peers broadcast message as stated above, their parents will know that which the leaf peers are the actual senders. To overcome this, leaf peers sends their messages though a secure channel to

another leave peer, and that leave peer will broadcast the message. To build the secure channel, the leaf peer passes the message to other leave peers, until a leaf peer decide to broadcast the message.

- **Cover traffic**

To provide anonymity against a global eavesdropper, we use cover traffic to maintain peers' traffic patterns, i.e. the traffic pattern should be statistically independent of it originating data traffic. Each peer send messages to their parent and children and friends at a system pre-defined rate, the messages are padded to the same system pre-defined size. When a peer wants to send a message, it exchanges the random dummy message with the real meaningful message, and there is a fixed bit indicate whether it's a dummy message or not. The messages are encrypted with the public key of the next hop, so it's not possible for the outsiders to know which messages are dummy messages or not. For each message to pass on, the node decrypt and check its integrity first, then re-pad it and re-encrypt it, last reorder the message with other messages and send them one by one.

- **Tree Adjustment**

The multicast tree is build to mimic the underlining physical connections; the tree structure will remain relative stable once the structure is settled down. The attackers could figure out the neighbors of a targeted peer, and compromising all the peer's neighbors will result in the expose of message sending from the targeted peer. Besides the instability of the peers which goes on and off constantly, we design a protocol to let the tree adjust itself periodically, so that the tree structure is not predictable at the degree that the attacker can figure out and compromise all neighbors of a peer within that adjustment period. Each peer with children will periodically randomly choose one of its children to switch with it, i.e. the selected child will become the parent and the original parent and all other children of that parent will become the children of the selected child, the children of the selected child will become the children of the original parent. With the length of the adjustment period and the optimization cycle, the multicast tree will be always changing and it is much harder for the attackers to break in.

3.2.3 Anonymous Communications

Based on the basic functions provided by each participating peers and the system functions provided by the cooperation of all the peers, the system supports all three types of anonymous communications as we stated in introduction.

- **Sender-receiver anonymous communication**

Sender-receiver anonymous communication is achieved by performing message broadcasting if the sender is an intermediate peer and passing through secure channel and then broadcast if the sender is a leaf peer. The receiver can reply this kind of message by simply sending the reply messages as a new message.

If the sender is a leaf node, it sends the message through a secure channel, since the forming peers of the channel are randomly chosen at each step; any peer in it except the sender could not know who the sender is. Once the message is broadcasted, the fist intermediate node could not know who the send is either, it could be any leave node in the tree, and the anonymous set for other intermediate node is even larger. Hence, the communication is sender anonymous. Since every node receives every message, and they filter out everything they want by themselves, it is receiver anonymous.

- **Sender anonymous communication**

The sender encrypt the message using the receiver's public key, and then broadcast the message, only the receiver can decrypt and read it. Also, the sender could include a symmetric key in the message, and then the receiver can reply the message by encrypting it with the symmetric key, and only the original sender can decrypt and read it. Since the message is sent to all the peers, only the real receiver can decrypt it, so no one knows who the targeted receiver is except the real receiver.

- **Receiver anonymous communication**

The sender broadcast the message, and only interested receiver will read the message and there is no way for the sender to know who received it. The receiver could reply to the send in the sender anonymous way described above.

4 Attacks

In this section, we analysis the possible attacks on the system, and we show that our system can provide anonymous communications in spite of these attacks.

4.1 Attacks from Outsiders

Anyone outside the peer-to-peer system, it can monitor all the traffic going through the network.

4.1.1 Passive Attacks

- Eavesdropper

Since all messages are encrypted at each hop, the hacker can not know the content of the messages.

- Traffic analysis

Since the traffic through the system maintains a stable traffic pattern, each peer maintains a fix incoming/outgoing rate and all the messages are encrypted, there is no way to figure out who is the sender or the receiver even the hacker can monitor all the traffic.

- Correlate messages

Since all the messages are padded to the same size and the padding is changed at each hop, and the messages are re-ordered before passing on, the hackers can not correlate the incoming and outgoing messages.

4.1.2 Active Attacks

- Alter messages

Integrity checking is conducted at each hop, altered messages will be dropped.

- Replay

A replay attack will re-send an incoming packet and watch for an outgoing packet, a duplicate that will correlate the incoming and outgoing packet [1]. In our system, messages are sent to all the peers in the network through a stable topology, replay can not expose the receiver, or the sender. Also, each peer passing messages at a fixed rate, the DoS attack won't work.

- Delete messages

The sender will resent messages if there is no reply from any receivers.

4.2 Attacks from the internal adversary

The internal adversary can monitor all the communication between peers and in addition is also trying to compromise the internal peers of the network. An adversary agent at such a compromised peer can gather information about messages that traverse the peer [7]. If the compromised peer is involved in the message rerouting, it can discover and report the immoderate predecessor and successor node for each message traversing the compromised peer [7].

4.2.1 Passive Attacks

- Collaborate

In case the sender is an intermediate node, the only way to know which the sender is that the attacker compromise all of its parent and children, and so can know that its children didn't send it the message, yet its parent received the message, then figure out the intermediate node is the sender. For k compromised nodes over n total nodes, the probability of these k nodes is round some nodes is very low. Also the tree is dynamically changing due to the instability of the peers which go on and off constantly, it makes it harder to figure out who are the neighbors of the targeted peer and it leave limited time to compromise all the neighbors of the target peer.

In case the sender is a leaf node, the only way to know which the sender is that the attacker compromises all the friends of that node, when there is an outgoing message and no income messages from the friends, the leaf node is the sender. Again, probability that attackers happened to compromised all the siblings of one leaf node is very low and the peer picks new friends constantly. It's harder to compromise all the friends of the targeted peer.

4.2.2 Active Attacks

- Drop

The sender can crosscheck the message it sends out with its neighbors, if it finds somebody did not receive it, it will re-send the message.

- Update

Both the sender or the receive can check the integrity of the message through crosschecking, once an integrity broken, the sender will resent, and the receiver will discharge the altered message.

5 Performance Analysis

Since in our approach, all of the messages will be sent to all the participating peers, performance should be a big concern for this kind of system. However, in order to beat against various attacks, including traffic analysis, message correlation and collaboration, more and more anonymous communication solutions use cover traffic, which means each participating nodes sends out noise messages to cover the real messages. Comparing to the systems using cover traffic at the same transition rate as our system's, our system does not show any performance degrading, because in both kind systems, each nodes sends out messages to its n neighbors, and for the same level of security, the number n in both systems should be close. Comparing to other peer-to-peer solutions, our topology is built to mimic the underlining physical network connections, so the link usage and the latency in our system will be better than those systems which use the application layer topology directly. From the security point of view, at the same transition rate, we send messages to all the nodes to enhance the anonymity, which could show better anonymity than the systems that only send messages to dedicated receivers, the rate of useful message over noise message will be higher in our system.

More current solutions only support sender anonymous communications. There are two solutions support sender-receiver anonymous, one is DC net, since it requires a bus for all the participating nodes, it is not

realistic, the other is Xor-tree, in this system, nodes should know all the other nodes, and only one message is allow at one time. Our system adapts their ideas of broadcasting messages to all the participants to achieve receiver anonymous. However, we use application multicast tree which mimic the underling physical connections for broadcasting, which provide high level anonymity in a cheap, scalable and efficient way.

The mix-net systems, like remailer system, onion routing, the anonymity is provided through a small, fixed core set of relays [6]. If a corrupt reply received traffic from a non-core node, the relay can identify that node as the ultimate origin of the traffic [6]. Colluding entry and exit relays can use timing analysis to determine both source and destination [6]. In our system, all peers are potential senders, receivers, and relays. Such a scalable design lessens the significance of targeted attacks and inhibits network-edge analysis [6]. Further, since the mix-net systems provide anonymity through a small, fixed core set of replays, which are prone to single point of failure or service degrading due to the attacks on a number of nodes. Since our system is built on top of peer-to-peer system, which is highly distributed, self-organized, the single point of failure will not happen, also is much harder for the attacker to attack most of the peers. Also, we take advantage of peer-to-peer system's ability to harness a massive amount of resources available on the Internet, the cost of building such system is much cheaper than systems using dedicated servers.

6 Simulation

In this section we present results from our message level simulator. Our simulator is written in C#, and can simulate the entire peer-to-peer service system with thousands of nodes. Through our simulation experiments, we used Georgia Tech [16] random graph generators to generate topologies of the peer-to-peer network.

We designed and implemented three basic experiments:

- Measure system performance as the network size increases.
- Measure the effect of different number of friends' lower bound
- Measure the effect of different buffer queue limit.

In our simulation, we generated random network topology which models the different latencies between pairs of peers. We keep the traffic rate fixed by using a timer; at every time out, the peers either send out noise messages or signal messages in the output queue. We assume an unbounded input queue length and a bounded output queue. Output queue limit is enforced, if the output queue is larger than its maximum specified size, messages will be discarded. To simplify the problem, we simulated networks in which peers share one common interest, i.e. peers in the network are either interested in the topic or not, so messages are all about one interest. We instrumented the simulator to record the end-to-end latency, drop rates, total number of noise messages, and percentage of number of signal messages over total number of messages.

6.1 Network Size

We measured the system performance as the network size increases. For each network size, we chose five different sets of interested peers who may generate or receiver signal messages. The size of the sets is 50% of the group size (number of peers interested in the topic). Each bar group in the figure is corresponding to a sending rate. In the "1/2S" case, each peer generates a message at each time stop with probability 1/(2*S), where S is the size of the network.

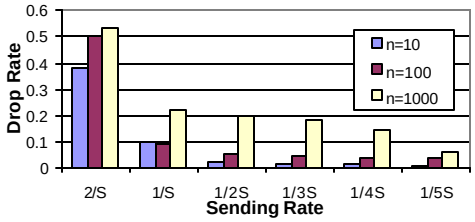

Figure 6-1. Drop Rate vs Sending Rate in different sized networks.

In Figure 6-1, we show the drop rate as the network size increases. The results show that the drop rate decreases when the sending rate decreases, and the larger the network, the higher the drop rate. The reason is that, with the sending rate increases, the output queues are easily full, and system is closer to saturate, so the drop rate increases.

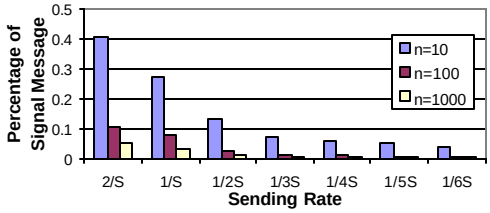

Figure 6-1. Percentage of Signal Message vs Sending Rate in different sized networks

In Figure 6-2, we show the percentage of signal messages over all messages including cover traffic as the network size increase. The results show that the percentage of signal messages decreases as the sending rate decreases, and the large the network, the lower the percentage of signal messages. The reason is that the with the sending rate decreases, the number of signal messages generated will decrease, then the number of noise messages increase, since the overall traffic rate is fixed. Also, since number of friends bound is higher in larger networks and only a low percentage of friends are used to send signal messages each time, a lot more cover traffic generated between friends. So the large the network is, the lower percentage of the signal messages is.

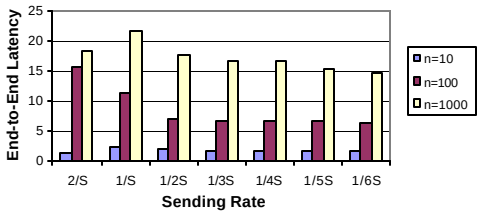

Figure 6-2. End-to-end Latency vs Sending Rate

In Figure 6-3, we show that the end-to-end latency increases as the network size increase and the end-to-end latency decrease as the sending rate decrease. There are three way to affect the latency, one is the time cost during the message transition, two is the hops between the two ends and three is the time cost in waiting in output queues. As network size increase, the number of hops between two peers increases, so the end-to-end latency increases. As the sending rate increases, the messages take longer time to wait in the queues, so the end-to-end latency increases. Note, the reason that the end-to-end latency is lower at sending rate equals 2/S is that the drop rate is high at this sending rate, messages are more likely to reach the peers close to the messages' sources, and then got dropped in the way to the farther peers. The end-to-end latency is not counted for the messages not reaching the peers on longer paths, this is why in the case of 2/S sending rate, the end-to-end latency could be lower.

6.2 Number of Friends

We measure the effect of different number of friends' lower bound. In this experiment, we used n=100 network with sending rate=1/S and number of friends' lower bound set to 10, 20, 30, 40 50 respectively. Figure 6-4 shows that with the increasing of the friends' number limit, the number of noise message increases

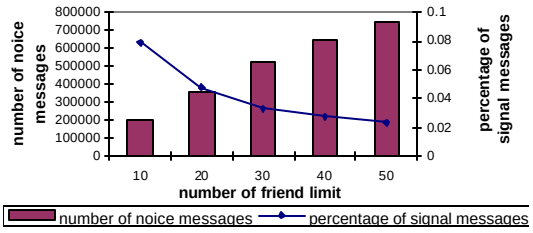

Figure 6-4. Number of noise messages vs percentage of signal messages with different number of friends limit

linearly, and the percentage of signal message decreases. Since only one out of the set number of friends will be chosen to pass signal messages each time, the traffic to other friends is all noise. So more friends peers have, the more noise message sent and then the less percentage of signal messages is.

6.3 Output Queue Size

We study the effect of different output queue limit. In this experiment, we used n=100 network with sending rate set to 2/S, 1/S and 1/2S and the output queue size set to 1, 2, 3, 4, 5 respectively.

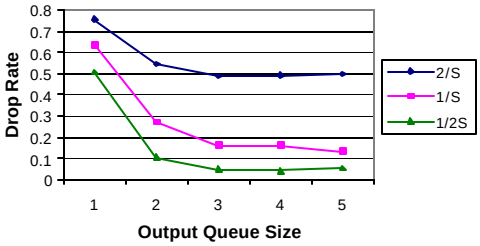

Figure 6-5. Output Queue Size vs Drop Rate

In Figure 6-5, we show that the drop rate decreases as the output queue size increase with various sending rate. The reason is that the longer the queue, the larger the buffer the peers can get to store the unsent messages due to heavy traffic going on, and get better chance to send these messages during off-peak period. So the drop rate is lower if the queue size is larger.

7 Conclusion

In this paper, we present a protocol which provides sender-, receiver-, and sender-receiver anonymity. Our system is built on top of peer-to-peer networks which is completely distributed, self-organized, and scalable with thousands of active participants may communicate simultaneously.

From our analysis on anonymity and performance, and various simulations using our message-level simulator, we show that our system provides high level of anonymity while reducing the latency of data delivery and the link utilization. It is not only resilient to various passive and active attacks from outsider nodes and insider peers, but also achieve high level anonymity in a cheap, scalable and efficient way.

Reference

[1].C. Boesgaard, Anonymous Communication in Practice, Department of Computer Science, University of Copenhagen, Denmark, 2003

[2]. D. Chaum, Untraceable Electronic Mail, Return Addresses, and Digital Pseudonyms, CACM, v.24, n.2, pp. 84-88, 1981.

[3]. D. Chaum. The Dining Cryptographers Problem: Unconditional Sender and Recipient Intractability. Journal of Cryptography, 65-75, 1989

[4] George Danezis, Roger Dingledine, and Nick Mathewson. Mixminion: Design of a type iii anonymous remailer protocol, 2003.

[5] Sholmi Dolev and Rafail Ostrovsky. Xor-Trees for efficient Anonymous Multicast Receiption. Advances in Cryptography 1997

[6]. M. J. Freedman, E. Sit, J. Cates, and R. Morris. Tarzan: a peer-to-peer anonymizing network layer. In Proceedings of the ACM Conference on Computer and Communications Security, Washington, D.C., Nov. 2002.

[7]. Yong Guan, Xinwen Fu, Riccardo Bettati, and Wei zhao. An optimal strategy for anonymous communication protocols. Texaz A&M University.

[8]. Yong Guan, Xinwen Fu, Riccardo Bettati, and Wei zhao, A Quantitative Analysis of Anonymous Communications, IEEE Transactions on Reliability, Vol. 53, No. 1, March 2004

[9]. J. Jannotti, D.K. Gifford, K. L. Jognson, M. F. Kaashoek and J. w. O'Toole Jr, Overcast: Reliable Multicasting with an Overlay Network, Operating System Design and Implementation (OSDI), Oct. 2000

[10].A. Mislove, G. Oberoi, A.Post, C. Reis, P. Druschel, AP3: Cooperative, decentralized anonymous communication, In Proc. SIGOPS-EW, Leuven, Belgium, Sept. 2004

[11] M.G. Reed, P.F. Syverson, and D. M. Goldschlag. Anonymous Connections and Onion Routing. IEEE Journal on Selected Areas in Communication, 1998.

[12]. Michael K. Reiter and Aviel D. Rubin. Crowds: Anonymity for Web Transactions. ACM Transactions on Information and System Security, 66-92, 1998

[13].Rob Sherwood, Bobby Bhattacharjee, and Aravind Srinivasan. P5: a protocol for scalable anonymous communication. University of Maryland

[14]. Clay Shields and Brian Neil Levine. A protocol for anonymous communication over the Internet. In Proceedings of the 7 th ACM Conference on Computer and Communications Security, pages 33-42, N.R. November 12 2000

[15].M. Waidner and B. Pfitzmann. The Dining Cryptographers in the Disco: Unconditional Sedner and Recipient Untraceability with Computationally Secure Serviceability,. In Eurocrypt, 1989.

[16]. Modeling Topology of Large Internetworks , http://www.cc.gatech.edu/fac/Ellen.Zegura/graphs.html