# MuON: Epidemic based Mutual Anonymity

Neelesh Bansod, Ashish Malgi, Byung Kyu Choi and Jean Mayo\*

{npbansod, asmalgi, bkchoi, jmayo}@mtu.edu
 Department of Computer Science
 Michigan Technological University
1400, Townsend Drive, Houghton, MI - 49931

*Abstract*—A mutually anonymous service hides the identity of a client from the service provider and vice-versa. Providing mutual anonymity usually requires a large number of participants. While peer-to-peer (P2P) networks are capable of recruiting a large number of participants, reliable anonymous communication in these architectures, with low bandwidth usage, still needs further investigation.

This paper presents MuON, a protocol to achieve mutual anonymity in unstructured P2P networks. MuON leverages epidemic-style data dissemination to deal with the high churn (changes in system membership) characteristic of unstructured P2P networks. The results from our security analysis and simulation show that MuON provides mutual anonymity over unstructured P2P networks while maintaining predictable latencies, high reliability, and low communication overhead.

#### I. INTRODUCTION

Many online applications such as banking, electronic voting, information sharing and searching etc, need anonymity to prevent third parties (or adversaries) from gathering information related to services and their clients. Most of these online services have a common model of interaction; a client (the *initiator*) sends a request to a node (the *responder*) that provides the service. The responder processes the request, and sends the corresponding response to the initiator. Based on this model of interactions, different types of anonymity [23, 26, 39] can be provided to applications: initiator anonymity, responder anonymity, mutual anonymity and unlinkability. Initiator anonymity hides the identity of the initiator from the responder and adversary. Responder anonymity hides the identity of the responder from the initiator and adversary. Mutual anonymity provides both initiator anonymity and responder anonymity. Unlinkability of initiator and responder means that the initiator and responder cannot be identified as communicating with each other, even though they can be identified as participating in some communication.

Different approaches have successfully achieved various forms of anonymity. In the simplest approach, a proxy is used for communication between initiator and responder [1, 14]. However, this system fails if the proxy itself reveals the identities of the communicating parties. To overcome this single point of failure, most anonymity protocols [3, 12, 13, 25, 26, 40] provide anonymous communication using indirection; messages from the sender (initiator/responder) are

\*The fourth author was supported in part by NSF CAREER grant CCR-9984682

routed through the intermediate relay nodes till they reach the final destination (responder/initiator). Some anonymity protocols [31, 33, 34] multicast messages to a large group of nodes to provide anonymous communication. It is important to note that in both approaches, the anonymity in the system improves as the number of participant nodes increases.

Experiences with P2P systems indicate the ability of these architectures to attract a large number of participants. There have been anonymity protocols that use different kinds of P2P systems such as structured P2P systems [40], IP layer P2P systems [13] and hybrid P2P systems [39] for providing anonymity. An unstructured P2P network does not impose a structure on its participant nodes and thus has several desirable characteristics such as administrative ease, ease of deployment and self-organization. Unstructured P2P networks however, pose significant challenges for anonymous communication protocols. An example of this kind of network is the Gnutella file sharing system, which is known to consume high bandwidth [28]. A study by Saroiu et al. [30] has shown that P2P systems exhibit high churn (changes in system membership); peers frequently leave/join the network and most peers are connected to the overlay for a short period of time. Similarly the nodes within the P2P network cannot be trusted by the anonymity protocol. These peers may attempt to tamper with messages, masquerade as the responder, drop messages that they are supposed to forward, or subvert the protocol by any other means. The peers within the network could also collude to violate the anonymity guarantees.

In this paper, we present MuON, a protocol for mutual anonymity and unlinkability over unstructured P2P (overlay) networks. The key contribution is its epidemic-style [6, 18, 36] message sending protocol. Our simulation results show that it achieves anonymous communication with high reliability while maintaining low latencies and low overhead.

The paper is organized as follows: Section II discusses the goals of MuON. Section III summarizes the prior approaches for anonymity and introduces epidemic protocols. Section IV describes MuON in detail, followed by the anonymity and performance evaluations in section V. Finally, section VI summarizes the contributions of MuON and future work.

## II. GOALS OF MUON

The main motivation of MuON is to strike a balance between performance and anonymity in a dynamic unstructured P2P network. The main goals are described below:

**Mutual Anonymity:** An initiator sends a request for a service without knowing which node actually provides the service. Likewise, the responder sends responses without knowing the identity of the initiator.

**Unlinkability:** The identities of the communicating parties (initiators and responders) are hidden from the adversaries.

Bounded Latencies: Communication latency is bounded.

**High Reliability:** The messages sent by the initiator to the responder and vice-versa are delivered reliably.

**Communication Overhead:** The communication overhead incurred by each peer within the P2P network should be low.

**Scalability:** Metrics like reliability, anonymity, communication latency and overhead should scale well with the size of the P2P network and the churn within the network.

Message Integrity and Confidentiality: Messages cannot be changed in transit; requests sent by an initiator can be read only by the corresponding responder; responses can be read only by the initiator, which sent the corresponding request; and intermediate nodes cannot masquerade as responders.

## III. RELATED WORK AND MOTIVATION

This section reviews previous approaches that provide different kinds of anonymity over various network architectures.

#### A. Anonymity by Mixes

One of the earliest proposed approaches for anonymous communication is Chaum's **Mix-Net** [7], which is the basis of subsequent systems like **Babel** [16] and **Mixminion** [10]. These approaches hide the communication between sender and receiver by encrypting messages in layers of public key cryptography. The messages are relayed through a group of dedicated message relays called "mixes". Each mix decrypts the messages, delays, and reorders the message before relaying it to another mix.

Approaches based on mix networks achieve strong anonymity guarantees at the cost of latency. While they provide anonymity against powerful global adversaries, the random delaying within the mix network results in unbounded and high latencies unsuitable for interactive applications.

#### B. Anonymity by Proxy

Several systems use a proxy to provide anonymity. Examples include **Anonymizer** [1] and **Lucent Personalized Web Assistant** [14] which use an intermediate proxy to provide anonymity to users. Likewise **PRA: Proxy for Responder Anonymity** [31] uses a proxy to provide responder anonymity, while **APFS Unicast** [31] uses an intermediate proxy and onion routing to provide mutual anonymity. Proxy based systems place a great deal of trust on the proxy. Thus they are vulnerable to failure if the proxy is compromised and reveals the identity of the communicating parties.

### C. Publisher Anonymity

Freenet [9] and FreeHaven [12] provide anonymous information storage and retrieval. The target applications of these protocols differ from those of MuON; FreeNet and FreeHaven are used for publishing and accessing documents anonymously, while MuON is used for anonymous communication in applications with online communicating parties.

## D. Anonymity by Single-Path Forwarding

Many protocols provide anonymous communication by forwarding messages along a single anonymous path, formed through the group of nodes within the infrastructure. This anonymous path can be specifically created, or is formed by random forwarding.

**Onion Routing** [25] provides anonymous communication using a dedicated set of message relays called "onion routers". The sender first selects a path from the set of onion routers. The data is then wrapped within encrypted layers, forming a unit called the *onion*. In the onion, the innermost layer of encryption is performed by the encryption key of the path's last hop, while the outermost layer uses the encryption key of the path's first hop. The onion routers co-operate and forward the onion from the sender to the destination.

**Tor** [12] is the second generation Onion Routing [25] protocol. It uses a modified form of Onion Routing to provide initiator anonymity, and uses rendezvous points to provide responder anonymity.

Xiao et al. [39] propose two protocols for mutual anonymity in hybrid P2P networks. These protocols, named **Center-Directing** and **Label-Switching**, use trusted third parties to provide anonymity. They also proposed **Shortcut-Responding** [39], which combines onion routing and broadcasting to provide mutual anonymity within unstructured P2P networks.

**Tarzan** [13] is an anonymous IP layer P2P system that provides initiator anonymity. Initiators create tunnels through the overlay by distributing session keys. The data is then passed through this tunnel using layered encryption/decryption (analogous to onion routing).

**TAP** [40] provides initiator anonymity in dynamic structured P2P networks by building replicated tunnels. The replicated tunnels enable the protocol to combat network churn.

**Crowds** [26] provides initiator anonymity using random forwarding. The initiator sends the (suitably encrypted) message to a randomly chosen node in the network called *jondo*. Each jondo randomly decides to either send the data to the responder or to forward it to another jondo.

In networks with high churn (nodes frequently join and leave the P2P network), approaches that utilize single anonymous paths are bound to suffer from path losses. Consider an anonymous path of length n nodes. If p is the probability of a node leaving the overlay, then a given path is valid with a probability of  $(1-p)^n$ . With increasing path lengths (increasing n) and increasing churn within the network (increasing p), the probability that a given path is valid diminishes. Hence approaches using a single-path will incur with greater probability, the additional overhead of detecting and rebuilding failed paths. Providing this kind of fault tolerance will likely be a high overhead operation and has not been extensively explored in the context of maintaining anonymity guarantees.



Fig. 1. Epidemic protocols

#### E. Anonymity by Group Communication

Many systems use group communication primitives like multicasting and flooding to achieve anonymity.

 $P^5$ , Peer-to-Peer Personal Privacy Protocol [33] proposes a novel approach for mutual anonymity using broadcast channels. It defines a logical hierarchy of broadcast groups, and the nodes within the P2P network join one or more of these groups when entering the system.

**GAP** [3] (part of GNUnet) uses controlled flooding to achieve initiator and responder anonymity in a P2P network.

**APFS** (Anonymous Peer-to-Peer File Sharing) [31] includes **APFS Multicast**, a protocol that uses multicasting to provide mutual anonymity within P2P file sharing applications.

**Hordes** [34] provides initiator anonymity using multicasting. A multicast group is formed by all the initiator nodes. Initiators send requests to responders using Crowds or Onion routing, while the responder multicasts the response to the group of initiators.

Protocols that depend on group communication primitives like multicasting are ideally suited for networks with high churn, because the departure of a few nodes does not substantially impact the communication between the sender and receiver. Previous work [34] also indicates that the use of multicasting helps reduce communication latencies. However, the lack of widespread deployment of IP multicast infrastructure inhibits deployment of protocols based on this type of multicast [31, 33]. GAP [3] takes a higher level approach, but achieves reliability by flooding, which may not scale well in large unstructured P2P networks. Hordes [34] uses singlepath forwarding to send requests, and potentially incurs the additional overhead of detecting and rebuilding failed paths.

## F. Epidemic Protocols

Epidemic (or gossip) protocols [11] are a well-studied class of protocols for low-cost reliable data dissemination within groups. They have been shown to be much more efficient than flooding based approaches [24, 35]. Epidemic protocols provide higher reliability and scalability while using lower bandwidth [18], when compared to other reliable multicast protocols. They provide a bimodal guarantee of reliability [6]; the message reaches all members of the group with a high probability, and the probability that it will reach to just a few members of the group is very low. Studies have shown that the time required to disseminate data to the entire group is log(N), where N is the number of nodes within the group. Due to these desirable characteristics, MuON uses an epidemic-style protocol for data dissemination.

A simplified gossip protocol is depicted in Figure 1. Each node runs several rounds of the gossip protocol. In each round, the node selects a random node as its gossip target. The node sends the gossip target(s) a gossip message containing a list of message identifiers that it has heard of (represented by dotted lines between nodes). If the list contains a message identifier which the gossip target has not received, the gossip target will request the node to send it (represented by solid lines between nodes). Three important parameters that impact gossip protocols are FanOut,  $T_{interval}$  and GC. FanOut is the number of gossip targets used in each round (FanOut is two in the figure).  $T_{interval}$  is the time between successive protocol rounds. (In the figure, node p is seen to start gossip rounds at time t1 and t6, resulting in  $T_{interval} = t6 - t1$ ). GC (Gossip Count) determines the number of rounds a message is gossiped by a node. These parameters determine the speed and efficiency of message sending and have been rigorously studied by Birman et al [6].

## IV. MUON

In this section, we describe the system model and assumptions for deploying MuON. We then describe the data dissemination (message sending) protocol along with the notation used. The message sending protocol is used in both directions, from initiator to responder and vice-versa. Finally, we describe how the message sending protocol is used for communication between the initiator and responder.

#### A. System Model

MuON operates over an unstructured P2P network. Let N be the number of nodes within the overlay (referred to as the *overlay size*). We assume that nodes within the overlay know at least g = log(N) other members of the overlay. The membership list for epidemic style protocols can be maintained by means of services such as SCAMP [15] and "Peer Sampling Service" [19]. MuON assumes that all initiators and responders are members of the P2P network. All protocol messages use low-cost unreliable transport (UDP) for communication.

Services are identified by a *service identifier*. To send a request for a particular service, the initiator obtains the public key corresponding to the service. This public key is used for initiating the communication between the initiator and responder. Thus the identity of the responder node that provides the service, is not revealed to the initiator. The message sending protocol of MuON ensures that initiator and responder anonymity and unlinkability are maintained. The use of public and session keys ensure that data integrity and confidentiality are maintained. The public keys of MuON are not tied to any specific algorithm; for example incomparable public keys [38] could be used.

The protocol assumes that there exists some mechanism



Fig. 2. Data Dissemination in MuON

that provides public keys corresponding to the service identifier. The system places some trust on this mechanism; the mechanism provides correct public keys only and it does not reveal the identity of the node corresponding to the public key. For convenience, the protocol description assumes the presence of a trusted PKI (Public Key Infrastructure), though an initiator could obtain the public keys out-of-band. It is interesting to note that the communication between the PKI and the initiator itself must be done anonymously. However, it is easy to conceive the PKI as a service within MuON itself, whose public key is well known and distributed out-of-band.

#### B. Notation

Before looking into the details of the MuON, we first look at the notation used within the protocol.

$k_{session}$	Symmetric session key
$k_{A}^{+}, k_{A}^{-}$	Public and private keys of node A respectively
$\{\hat{d}ata\}k_s$	$data$ encrypted/signed using key $k_s$
	$(k_s \text{ is public, private or session key})$
$r_1$	Nonce
H(data)	Cryptographic hash computed over data
	(e.g. SHA-1)
self	Identity of the node executing the protocol
	(e.g. IP-address)
$p_{inter}$	Intermediate probability, parameter controlling
	anonymity and performance of MuON
$T_{interval}$	Time interval between successive protocol rounds
FanOut	Number of gossip targets per protocol round
GC	Number of protocol rounds a message is gossiped
MSG	Data message (a request or a response message)
MSG_HDR	The header corresponding to MSG. Format is
	{currOwner, $hdr$ , $H(hdr)$ }, where currOwner is
	the node that has the corresponding MSG and $hdr$
	depends on type of MSG (refer to section IV-D).

#### C. Message Sending in MuON

The message sending protocol of MuON is unidirectional; it is used to send requests from an initiator to a responder and then again to send a response from the responder to the initiator. Let MSG denote the encapsulated data to be sent (thus MSG may be a request or a response). MuON generates

```
/* Adding a header (MSG_HDR) to the headerBuffer */
addheader(MSG_HDR):
begin
   slot = free slot in the headerBuffer
   headerBuffer[slot].MSG_HDR = MSG_HDR
   headerBuffer[slot].gossipCount = 0
end
/* Add message (MSG) and its header (MSG_HDR) */
addmessage(MSG, MSG_HDR):
begin
   addheader(MSG_HDR)
   Add MSG to the messageBuffer
   Associate MSG with H(hdr) contained in MSG_HDR
end
/* Sending message MSG with header hdr. The node
* sending MSG computes the message's hdr (described
* in later sections). self indicates the identity of the node
* that executes this method */
sendMessage (hdr, MSG):
begin
   MSG_HDR = \{self, hdr, H(hdr)\}
   addmessage(MSG, MSG_HDR)
end
```

Algorithm 1: Common procedures used by algorithms





a header, denoted MSG\_HDR, that corresponds to MSG. We assume<sup>1</sup> that the size of MSG\_HDR is much less than MSG, since MSG\_HDR contains only the required identifiers and cryptographic keys (details are in section IV-D).

The basic operation of the protocol is depicted in Figure 2, which shows node X sending MSG to node Y. MuON uses an epidemic protocol to disseminate MSG\_HDR to all nodes within the P2P network, while the larger MSG is disseminated to only a few nodes within the network (shaded within Figure 2). As explained in detail later, the number of nodes which receive MSG depends on the value of  $p_{inter}$ . The protocol ensures that the responder always gets MSG. As the larger MSG is not sent to the entire network, MuON substantially reduces the bandwidth usage. Also, since multiple nodes within the

<sup>1</sup>This assumption holds true in applications with large responses (e.g. filetransfer and web-browsing). In these applications, MuON achieves substantial bandwidth savings compared to other group communication based anonymity protocols. We anticipate that MuON will provide a bandwidth reduction for applications with small data messages (e.g. e-voting) that require reliable delivery, though these applications are not evaluated in this paper.



Algorithm 3: Receiving a gossip message

network receive MSG (all the shaded nodes), multiple nodes are potential receivers and senders of MSG, giving MuON its anonymity guarantees. MuON derives its properties of reliability and bounded latencies from its epidemic nature.

Every node running MuON maintains two buffers; one to store the message headers (called *headerBuffer*) and the other to store the corresponding messages (called *messageBuffer*). Every node tracks the number of protocol rounds each header has been gossiped (called the *gossipCount*). The details for handling these buffers are described in Algorithm 1.

Algorithm 2 explains the protocol executed by each node after every  $T_{interval}$  units of time. This algorithm describes an epidemic protocol for disseminating the headers. Each node selects FanOut random nodes from the group as gossip targets and sends them a list with each message header MSG\_HDR currently within *headerBuffer*. As given in algorithm 3, whenever A gets the message, A tries to decrypt<sup>2</sup> the message. If A can decrypt the message, it indicates that the message was intended for A and thus A contacts currOwner and pulls the message. In this case A also gossips with its neighbors that it has the message to send. Thus the responder

<sup>2</sup>If the decrypted message contains an expected value such as a known identifier or public key, the node can conclude "successful" decryption

in MuON behaves exactly the same as any other node in the network (with the exception that it always pulls the message). If A cannot decrypt the message then A performs one of two actions: it may just add the header to its header buffer or with some probability  $p_{inter}$  it may go back and get the corresponding MSG from B. In the first case, A gossips with its neighbors that B currently has the message. In the second case, when A gets the MSG from B, it changes the currOwner field of MSG\_HDR to A. Thus when A gossips the header, it will indicate itself as the owner of the message. With this property, MuON achieves its anonymity guarantees as there are potentially many owners of the same message.

#### D. Initiator and Responder Communication

While the message sending protocol of MuON helps achieve anonymity, cryptographic measures are required to ensure message integrity and confidentiality. This subsection describes how the dissemination protocol is used by initiators and responders for secure anonymous communication.

Sending a request: The following steps are performed, when an initiator I sends a request for service S. Let *data* represent information contained in the request message and *id* be an application specific message identifier.

- 1) I generates a symmetric session key  $k_{session}$ , which is used to encrypt all data messages.
- 2) I generates a nonce  $r_1$ , which is used to correlate responses with this request.
- 3) Using the PKI, I obtains the public key  $k_s^+$  associated with the service S.
- 4) The MSG is generated as  $\{r_1, id, data\}k_{session}$ .
- 5) *I* creates a header *hdr*, corresponding to MSG as *hdr* =  $\{r_1, k_{session}, k_I^+, \{H(D)\}k_I^-\}k_s^+$  where  $D = \{r_1, k_{session}, k_I^+, MSG\}.$
- 6) I now invokes sendMessage(hdr,MSG) (Algorithm 1).

**Responding to a request:** Algorithm 3 ultimately delivers  $MSG\_HDR$  and MSG to the peer providing the service S. Let some node R provide the service S. Suppose R receives hdr and its corresponding MSG. R proceeds with the following steps.

- 1) R decrypts hdr using  $k_s^-$ , to obtain  $k_{session}$ ,  $r_1$  and the initiator's public key  $k_I^+$ . R now runs integrity checks with the cryptographic hash.
- 2) Using  $k_{session}$ , R decrypts MSG to recover the request.
- 3) Let *response* be the corresponding reply, which the responder R needs to send to I. R creates  $MSG = \{r_1, id, reponse\}k_{session}$ . Here  $k_{session}$  and  $r_1$  are values sent by the initiator and recovered by R in step 1.
- 4) R creates a header hdr corresponding to the response as  $hdr = \{r_1, \{H(D)\}k_S^-\}k_I^+$ , where  $D = \{r_1, MSG\}$ .
- 5) R now invokes sendMessage(hdr,MSG) (Algorithm 1).

## V. EVALUATION

In this section, we evaluate the performance, anonymity and other security guarantees of MuON. We first study the impact of overlay size and churn on the various performance metrics of MuON. We then evaluate the anonymity and other security guarantees of the protocol.

#### A. Performance Evaluation

Measurement studies of unstructured P2P networks [5, 30, 32] indicate that these systems exhibit dynamic membership, because peers alternately join and leave the network. Peers participate in the protocol only during the time between joining and leaving the network. This time is called the *session time* and the resultant dynamism is called the *network churn*. The network churn is related to the average session time of the peers within the network. As the average session time decreases, the membership of the P2P network changes at a faster rate and is said to exhibit a higher churn [21, 27].

Prior experiences [8, 21, 27] indicate that network churn impacts the performance of protocols over P2P networks. Hence we evaluate MuON by simulating the protocol over dynamic unstructured P2P networks of varying sizes and varying churn. We model network churn using an approach similar to that described by Liben-Nowell et al. [22]. This model has also been used for evaluating distributed hash tables over P2P networks [21, 27]. The peers within the network are assigned exponentially distributed session times. When a peer reaches the end of its session time, it leaves the network. Prior studies [17] have shown that the average session time (amount of churn) within a network depends on the application. Since MuON is not specific to any application, we simulate networks with varying churn (session time).

We simulate MuON using PeerSim [2, 20], a P2P simulator designed specifically for epidemic protocols. The simulator executes the protocol in a series of cycles, where the time interval between each cycle is assumed to be sufficient for unidirectional message transmission. The loss rate in a direct UDP communication between any pair of nodes is 5%.

In our simulation model, a network with churn 0 is a static network, which does not change during the run of the simulation. At churn 0, the average session time was chosen as 150 cycles (a factor of 10 over the maximum time for one run of the protocol), to enable simulation of several rounds of MuON simultaneously. An increase of 0.1 in network churn decreases the average session time of the nodes by a factor of  $\frac{1}{10}$ . When a node leaves the network, it is replaced by a new node, thus keeping the overlay size constant. This helps us to understand the impact of overlay size and churn independently.

The simulations are used to study the impact of increasing overlay size and churn on the various protocol metrics. In the simulation FanOut and GC are maintained at  $log_2$ (overlay size) and  $T_{interval}$  is maintained at 1 cycle. These parameters are common to all epidemic protocols and do not impact anonymity guarantees. Their impact on performance is expected to be similar to that determined by previous studies [6] and is currently under evaluation. The impact of intermediate probability  $p_{inter}$  is studied in section V-B. Unless specified, the value of  $p_{inter}$  is assumed to be 0.5.



Fig. 3. Reliability of MuON



Fig. 4. Number of cycles for message delivery

**Reliability** The reliability of MuON is measured by the delivery ratio achieved in networks of varying sizes and churn. The delivery ratio is the fraction of the sent requests that were ultimately delivered at the final destination. Thus delivaryratio =  $\frac{\text{number of requests delivered}}{\text{number of requests sent}}$ . When delivery ratio is one, it indicates that all requests that were sent were eventually delivered at their destination, thus indicating reliable communication.

Figure 3 shows the delivery ratio for networks with varying sizes and churn. It can be seen that MuON maintains a high delivery ratio of almost one, independent of the overlay size and churn. This high reliability indicates its suitability for highly dynamic P2P networks.

**Bounded Latency** One of the goals of MuON is to achieve communication within a predictably bounded time interval. This characteristic is important from the application's point of view; shorter latencies are important for application interactivity while bounded latencies are needed by applications to set timeouts and detect messages losses.



Fig. 5. Average number of messages processed on each node

Since MuON operates over an overlay network, we measure the latency in terms of number of protocol cycles required for the message to be delivered at its destination.

Figure 4 shows the average number of cycles required for messages to be delivered; the bars indicate the variation in the delivery latency. It can be seen that the delivery latency is almost constant irrespective of the overlay size and churn, indicating that the latencies provided by MuON are predictable and bounded.

**Resource Consumption** When a peer joins MuON, it has to contribute some of its resources to help forward the messages from other peers. Hence it is important to study the amount of resources a peer has to expend, to help other peers achieve anonymity.

In MuON, a peer needs resources to send, encrypt, decrypt and store messages. In general, the resources consumed by a node are directly proportional to the number of messages (header as well as data) it has to process. Figure 5 shows the average number of headers and data messages that are processed on each peer, whenever some peer sends an anonymous message. It can be seen that irrespective of the overlay size and churn, the number of header messages processed is bounded and relatively low. The graph also indicates that each peer has to process very few data messages. The header messages are small in size and thus the processing overhead for each header, storage and bandwidth is low. The protocol uses private/public key encryption for small headers and faster symmetric cryptography for large data messages, to reduce the overhead due to encryption [29].

**Comparative Bandwidth Use** MuON's message sending protocol has been designed to use low bandwidth as compared to previous multicast-based anonymity protocols. Let  $HDR_{size}$  and  $DATA_{size}$  be the size of header and data messages respectively and N be the number of nodes within the overlay. Consider the bandwidth consumed when one data



Fig. 6. Comparative Bandwidth Use (Churn 0)

message is sent anonymously. A multicast-based anonymity protocol will multicast this message to N nodes. Hence the bandwidth consumed will be at least  $N * DATA_{size}$ . Note that this is a conservative estimate, since it ignores the bandwidth consumed by control messages and data message re-transmissions required in the presence of network churn. On the other hand, MuON disseminates the data message to only a subset of nodes within the overlay, ensuring that the final destination is a member of this subset. Let  $\beta$  be the size of the subset of nodes that receive the message. The bandwidth consumed in MuON is  $k*N*HDR_{size} + \beta*DATA_{size}$ , where k is the number of headers processed by each node. Since Figure 5 indicates that the value of k is low and  $HDR_{size}$  is small<sup>3</sup>, we approximate the bandwidth consumed as  $\beta * DATA_{size}$ . The value of  $\beta$  depends on the intermediate probability  $p_{inter}$ . We generated figure 6 by simulation, which indicates the value of  $\beta$  with values of  $p_{inter}$  of 0.3, 0.5 and 0.8. The value of N is also shown for clarity. It can be seen that the value of  $\beta$  is always lower than N, indicating that MuON would use lower bandwidth compared to other multicast-based approaches. It is also interesting to note that as the value of  $p_{inter}$  decreases, the bandwidth consumption decreases.

**Scalability** An important characteristic that is evident from the results presented above, is the scalability of MuON. The protocol's reliability, latency bounds and resource consumption are almost constant irrespective of the overlay size and churn.

#### B. Anonymity Guarantees

In this section, we first discuss how MuON achieves mutual anonymity and the parameters that impact it. We then evaluate the anonymity guarantees by describing the protocol behavior under various attacks from the adversary.

<sup>&</sup>lt;sup>3</sup>Considering 128 bit cryptographic hash, 128 bit cryptographic keys, 32 bit IP addresses and 32 bit nonce, the maximum size of hdr is 416 bits and  $HDR_{size} = 576$  bits (72 bytes). If the data being transferred is a 1 MB media file then  $DATA_{size} = 1048576$  bytes. For an overlay of size 10,000 at churn 0, k = 12 from figure 5 and  $\beta = 3000$  from figure 6. Hence the volume of MuON header messages is 8437.5KB and the volume of MuON data messages is 3072000KB.

Mutual Anonymity in MuON Similar to anonymity protocols that use multicasting [31] or broadcasting [33], MuON achieves mutual anonymity on the virtue that several intermediate peers receive the messages. When an intermediate node receives a MSG, it gossips the corresponding MSG\_HDR with itself as the owner. From an observer's perspective, any node claiming to be the current owner could be the actual sender of the message. Similarly, when an intermediate node receives MSG\_HDR, it pulls the corresponding MSG with a probability of  $p_{inter}$ . Thus from the observer's perspective, any intermediate node that eventually receives the MSG could potentially be the receiver. Thus in the protocol, an observer (initiator, responder or intermediate nodes) cannot differentiate the initiator and responder from the other peers. The use of public keys also enables the initiator and responder to communicate without knowing the identity of each other. Thus mutual anonymity and unlinkability is achieved.

**Impact of Intermediate Probability** The degree of anonymity provided by multicast based anonymity protocols depends on the number of nodes that have an equiprobable chance of playing a certain role (initiator/responder). Let S (called the *anonymity set*) denote the set of nodes that have an equiprobable chance of being the initiator/responder in the anonymity system. Shields et al. [34] show that the degree of anonymity provided by the protocol is  $1 - \frac{1}{|S|}$ .

In MuON, for a given communicating pair of initiator and responder, any node that receives MSG has an equiprobable chance of being the initiator or responder. Hence the anonymity set is the set containing all nodes that receive MSG. Using simulations, we measured the size of the anonymity set for a given MSG, when it is delivered at its destination. Figure 7 shows the average size of the anonymity set expressed as percentage of nodes within the overlay that received a given MSG. It can be seen that the anonymity set increases with increasing values of  $p_{inter}$ . However, for a given value of  $p_{inter}$ , the anonymity set remains fairly constant independent of the churn within the network. This indicates that the degree of anonymity provided by MuON is independent of the churn of the network.

The anonymity set (and hence the degree of anonymity provided by MuON) and the bandwidth consumed (Figure 6) increase as the value of  $p_{inter}$  increases. Thus when  $p_{inter}$  is 1, the protocol would provide the maximum degree of anonymity. However, the protocol will perform a multicast and thus the bandwidth consumption would be maximized. On the other hand, when  $p_{inter} = 0$  the protocol performs a unicast, resulting in minimum anonymity and the bandwidth consumed would be minimized. Thus  $p_{inter}$  represents a tradeoff <sup>4</sup> within MuON between performance and anonymity.



Fig. 7. Impact of Intermediate Probability

Attacks by Adversary Anonymity protocols are susceptible to several possible attacks. However, the successful completion of these attacks may require the adversary to utilize varying amounts of resources. To evaluate the anonymity guarantees of MuON, we describe various attacks and describe MuON's behavior under attack. It can be seen that though some attacks are possible against MuON, the resources required by the adversary to successfully complete the attack are substantial.

Local eavesdropper: A local eavesdropper is an adversary that is able to monitor all communications sent to or received from one particular protocol participant. This adversary tries to detect the identity of communicating parties by recording and comparing all incoming and outgoing messages of a particular node. In MuON a local eavesdropper on an intermediate node, cannot confirm the identities of the communicating parties, even if the message and its header are received by the

<sup>&</sup>lt;sup>4</sup>The degree of anonymity decreases gradually with  $p_{inter}$ . Consider a network of 10,000 nodes with churn 0. If  $p_{inter} = 1$  then |S| = 10,000 and degree of anonymity is 0.9999. From figure 7, if  $p_{inter} = 0.8$  then |S| = 0.50 \* 10,000 and degree of anonymity is 0.9998.

intermediate node. This is because the header and message do not contain any form of identification of initiator and responder.

*Collusion Attack:* In a collusion attack, nodes within the system collaborate to identify the communicating entities. It has been seen that the degree of anonymity in MuON is  $1 - \frac{1}{|S|}$  where S is the anonymity set. This implies that as long as a single intermediate node within the anonymity set does not collaborate, it is hard for colluding nodes to differentiate with certainty, the initiator, responder and the honest intermediate nodes from one another. However, if all |S| nodes within the anonymity set collaborate, the degree of anonymity provided by MuON reduces to 0 and the identities of the communicating parties can be revealed. However, since the anonymity set changes for every MSG exchanged between the initiator and responder, all nodes within the network must collaborate to launch this attack successfully.

*Timing attack:* In a timing attack, the adversary (behaving as an initiator) attempts to identify the responder by analyzing the round trip time (RTT) of a request, since short RTT indicates that the responder is nearby. In MuON since the messages are transferred over the overlay network, RTT measurements do not reflect actual network locations. Thus launching a timing attack is difficult.

An adversary can launch a variant of the timing attack against MuON, by identifying the initiator as the first node to gossip a particular MSG. To launch this attack, the adversary would have to trace outgoing messages of every node within the network, to identify a particular node as the first node to gossip a message. However in this attack, the adversary cannot identify the responder, since the responder behaves like an intermediate node and continues to gossip the MSG.

*Traceback attacks:* There are two kinds of traceback attacks: *passive traceback* and *active traceback*. In a passive traceback attack, the adversary examines the stored routing state of the peers within the network to identify the path(s) between initiator and responder. To launch a passive traceback against MuON, the adversary needs to look at the application level message buffers at every node within the network. However, since the messages are periodically removed from the buffers, to perform a successful traceback the adversary must collect the information before it is removed.

In an active traceback attack, the adversary has control of the network infrastructure and is able to follow an active and continuing stream of packets back through the network to their point of origin. In MuON, such an adversary can identify the sender of the message (as it is the starting point of the message paths). However, the recipient is not revealed (since the path does not terminate at the recipient).

*Predecessor attacks:* These attacks occur if the same path is used by the initiator while communicating to the responder. If a compromised node records its predecessor, then most of the time the initiator will be the predecessor. However in MuON as every node randomly picks up the gossip target, different messages follow different paths. Hence this type of attack is not possible in MuON. *Message volume attack:* An adversary can differentiate responders from other nodes by observing the volume of data transmitted, since initiators generate less volume of data as compared to responders. This attack is possible against MuON, if the adversary is a global adversary and can observe the volume of data from all nodes within the network.

*Intersection Attack:* This attack can be launched by a global adversary, who can observe all the various communication paths within the network. The initiator and responder will always be on the communication path, and the intersection of these paths would reveal the identity of the initiator and responder. Like several other anonymity protocols, MuON is vulnerable to intersection attacks.

In summary, we see that MuON can resist most kinds of attacks in the absence of a global adversary. We believe that such an adversary is impractical for large and dynamic P2P systems, though many of these attacks can be thwarted by means of cover traffic [4].

#### C. Security Guarantees

In MuON, message confidentiality and integrity are achieved using cryptographic techniques such as cryptographic hash, public/private keys and session keys. Hence these guarantees are constrained by the strengths of the cryptographic algorithms actually used.

When the initiator sends the request, it generates a nonce  $r_1$  and a session key  $k_{session}$ . The initiator then generates a header, which contains the nonce, session key and the initiator's public key. The header is then encrypted using the responder's public key. Similarly, the responder also includes the nonce in the header for the response and encrypts this header with the initiator's public key. Thus the nonce and session key always remain confidential. MSG always encrypts the data and nonce, using the session key. Since the session keys are not reused, encrypting the data with session keys helps thwart dictionary attacks. Thus confidentiality is maintained.

The header, hdr always contains a cryptographic hash signed by the private key of the sender (initiator in case of requests and responder in case of responses). The cryptographic hash is computed over MSG and the required fields of hdr and is signed by the sender's private key. This signed cryptographic hash has several uses. It allows the receiver to verify the correspondence between a given MSG and its MSG\_HDR. The signed cryptographic hash helps the receiver detect if an adversary changed the contents of the message or the nonce. Similarly, when an initiator receives a response, the initiator can verify that the response originated from the responder, because the cryptographic hash is signed by the responder's private key. Thus an adversary cannot masquerade as the responder. Likewise, the nonce contained within each header and data message can be used by the initiator to detect a replay of a response. Likewise, if the responder keeps track of nonce values of the past requests, it can detect the replay of requests.

#### VI. CONCLUSION AND FUTURE WORK

We have presented MuON, a protocol for providing mutual anonymity in dynamic P2P networks. The contributions of MuON are twofold; the protocol provides reliable mutually anonymous communication over dynamic P2P networks, while maintaining low bandwidth and processing overhead; and it exhibits application friendly characteristics such as bounded communication latency and message integrity and confidentiality. Since network outages can be modeled as churn, we believe that MuON provides resilient communication between initiator and responder.

In the future, we plan to incorporate a cover traffic scheme in order to enhance MuON's anonymity guarantees. We also plan to investigate the use of MuON for creating censorship resistant services [37] and 'Denial-of-Service' (DoS) resistant services.

#### REFERENCES

- [1] The anonymizer, http://anonymizer.com/.
- [2] Peersim peer-to-peer simulator, http://peersim.sourceforge.net/.
- [3] K. Bennett and C. Grothoff. GAP-practical anonymous networking. In Privacy Enhancing Technologies Workshop (PET'03), Mar. 2003.
- [4] O. Berthold and H. Langos. Dummy traffic against long term intersection attacks. In Proc. of Privacy Enhancing Technologies workshop (PET 2002). Springer-Verlag, LNCS 2482.
- [5] R. Bhagwan, S. Savage, and G. Voelker. Understanding availability. In Proc. of the 2nd International Workshop on P2P Systems (IPTPS '03).
- [6] K. Birman, M. Hayden, O. Ozkasap, Z. Xiao, M. Budiu, and Y. Minsky. Bimodal multicast. ACM Trans. Comput. Syst., 17(2):41–88, 1999.
- [7] D. Chaum. Untraceable electronic mail, return addresses, and digital pseudonyms. *Communications of the ACM*, 24(2):84–90, 1981.
- [8] Y. Chawathe, S. Ratnasamy, L. Breslau, N. Lanham, and S. Shenker. Making gnutella-like p2p systems scalable. In SIGCOMM '03: Proc. of the 2003 conference on Applications, technologies, architectures, and protocols for computer communications, pages 407–418, 2003.
- [9] I. Clarke, O. Sandberg, B. Wiley, and T. Hong. Freenet: A Distributed Anonymous Information Storage and Retrieval System. In *Designing Privacy Enhancing Technologies: International Workshop on Design Issues in Anonymity and Unobservability*, pages 46–66, New York, 2001.
- [10] G. Danezis, R. Dingledine, and N. Mathewson. Mixminion: Design of a type iii anonymous remailer protocol. In SP '03: Proc. of the 2003 IEEE Symposium on Security and Privacy, page 2, 2003.
- [11] A. Demers, D. Greene, C. Hauser, W. Irish, J. Larson, S. Shenker, H. Sturgis, D. Swinehart, and D. Terry. Epidemic algorithms for replicated database maintenance. In PODC '87: Proc. of the 6<sup>th</sup> annual ACM Symp. on Principles of distributed computing, pages 1–12, 1987.
- [12] R. Dingledine, N. Mathewson, and P. Syverson. Tor: The secondgeneration onion router. In *Proc. of the 13th USENIX Security Symposium*, San Diego, CA, Aug. 2004.
- [13] M. J. Freedman and R. Morris. Tarzan: A Peer-to-Peer Anonymizing Network Layer. In Proc. of the 9th ACM conference on Computer and communications security, pages 193–206, Washington, DC, 2002.
- [14] E. Gabber, P. B. Gibbons, D. M. Kristol, Y. Matias, and A. Mayer. Consistent, yet anonymous, web access with lpwa. *Commun. ACM*, 42(2):42–47, 1999.
- [15] A. J. Ganesh, A.-M. Kermarrec, and L. Massoulié. Peer-to-peer membership management for gossip-based protocols. *IEEE Trans. Computers*, 52(2):139–149, 2003.
- [16] C. Gülcü and G. Tsudik. Mixing E-mail with Babel. In Proc. of the Network and Distributed Security Symposium - NDSS '96, pages 2–16, Feb. 1996.
- [17] K. P. Gummadi, R. J. Dunn, S. Saroiu, S. D. Gribble, H. M. Levy, and J. Zahorjan. Measurement, modeling, and analysis of a peer-to-peer filesharing workload. In SOSP '03: Proc. of the nineteenth ACM symposium on Operating systems principles, pages 314–329, 2003.

- [18] I. Gupta, K. Birman, and R. Renesse. Fighting Fire With Fire: Using Randomized Gossip To Combat Stochastic Scalability Limits. *Journal* on Quality and Reliability Engineering International: Secure, Reliable Computer and Network Systems, 29(8):165–184, May 2002.
- [19] M. Jelasity, R. Guerraoui, A.-M. Kermarrec, and M. van Steen. The peer sampling service: experimental evaluation of unstructured gossip-based implementations. In Proc. of the 5th ACM/IFIP/USENIX international conference on Middleware (Middleware 2004), pages 79–98, 2004.
- [20] M. Jelasity, A. Montresor, and O. Babaoglu. A modular paradigm for building self-organizing peer-to-peer applications. In *Engineering Self-Organising Systems*, pages 265–282, July 2003.
- [21] J. Li, J. Stribling, R. Morris, M. F. Kaashoek, and T. M. Gil. A performance vs. cost framework for evaluating dht design tradeoffs under churn. In *Proc. of the 24th Infocom*, Miami, FL, March 2005.
- [22] D. Liben-Nowell, H. Balakrishnan, and D. Karger. Analysis of the evolution of peer-to-peer systems. In PODC '02: Proc. of the twentyfirst annual symposium on Principles of distributed computing, pages 233–242, 2002.
- [23] A. Pfitzmann and M. Waidner. Networks without user observability. Computers and Security, 6(2):158–166, 1987.
- [24] M. Portmann and A. Seneviratne. Cost-effective broadcast for fully decentralized peer-to-peer networks. *Computer Communications*, 26(11):1159–1167, 2003.
- [25] M. G. Reed, P. F. Syverson, and D. M. GoldSchlag. Anonymous Connections and Onion Routing. *IEEE Journal on Selected Areas in Communications*, 16(4):482–493, May 1998.
- [26] M. K. Reiter and A. D. Rubin. Crowds: Anonymity for Web Transactions. ACM Transactions on Information and System Security,, 1(1):66– 92, Nov. 1998.
- [27] S. Rhea, D. Geels, T. Roscoe, and J. Kubiatowicz. Handling churn in a DHT. In Proc. of the 2004 USENIX Annual Technical Conference (USENIX '04), Boston, Mass., Jun. 2004.
- [28] M. Ripeanu, I. T. Foster, and A. Iamnitchi. Mapping the gnutella network: Properties of large-scale peer-to-peer systems and implications for system design. *CoRR*, cs.DC/0209028, 2002.
- [29] M. Roe. Performance of block ciphers and hash functions one year later. In *Fast Software Encryption*, pages 359–362, 1994.
- [30] S. Saroiu, P. K. Gummadi, and S. D. Gribble. A measurement study of peer-to-peer file sharing systems. In *Proc. of Multimedia Computing* and *Networking 2002 (MMCN '02)*, San Jose, CA, Jan. 2002.
- [31] V. Scarlata, B. N. Levine, and C. Shields. Responder Anonymity and Anonymous Peer-to-Peer File Sharing. In *The Ninth International Conference on Network Protocols (ICNP'01)*, pages 272–280, Mission Inn, Riverside, CA, Nov. 2001.
- [32] S. Sen and J. Wang. Analyzing peer-to-peer traffic across large networks. *IEEE/ACM Trans. Netw.*, 12(2):219–232, 2004.
- [33] R. Sherwood, B. Bhattacharjee, and A. Srinivasan. P<sup>5</sup>: A protocol for scalable anonymous communication. In *Proc. of the 2002 IEEE Symposium on Security and Privacy*, pages 58–70, Berkeley, May 2002.
- [34] C. Shields and B. N. Levine. A protocol for anonymous communication over the internet. In Proc. of the 7th ACM conference on Computer and communications security, pages 33–42, Athens, Greece, 2000.
- [35] S. Tanaraksiritavorn and S. Mishra. Evaluation of gossip to build scalable and reliable multicast protocols. *Perform. Eval.*, 58(2+3):189–214, 2004.
- [36] W. Vogels, R. van Renesse, and K. Birman. The power of epidemics: robust communication for large-scale distributed systems. *SIGCOMM Comput. Commun. Rev.*, 33(1):131–135, 2003.
- [37] M. Waldman, A. Rubin, and L. Cranor. Publius: A robust, tamperevident, censorship-resistant and source-anonymous web publishing system. In *Proc. of 9<sup>th</sup> USENIX Security Symp.*, pages 59–72, Aug. 2000.
- [38] B. R. Waters, E. W. Felten, and A. Sahai. Receiver Anonymity via Incomparable Public Keys. In Proc. of the 10th ACM conference on Computer and Comm. Security, pages 112–121, Washington, DC, 2003.
- [39] L. Xiao, Z. Xu, and X. Zhang. Low-Cost and Reliable Mutual Anonymity Protocols in Peer-to-Peer Networks. *IEEE Transactions on Parallel and Distributed Systems*, 14(9):829 – 840, Sept. 2003.
- [40] Y. Zhu and Y. Hu. TAP: A Novel Tunneling Approach for Anonymity in Structured P2P Systems. In *International Conference on Parallel Processing (ICPP'04)*, pages 21–28, Montreal, Quebec, Canada, Aug. 2004.