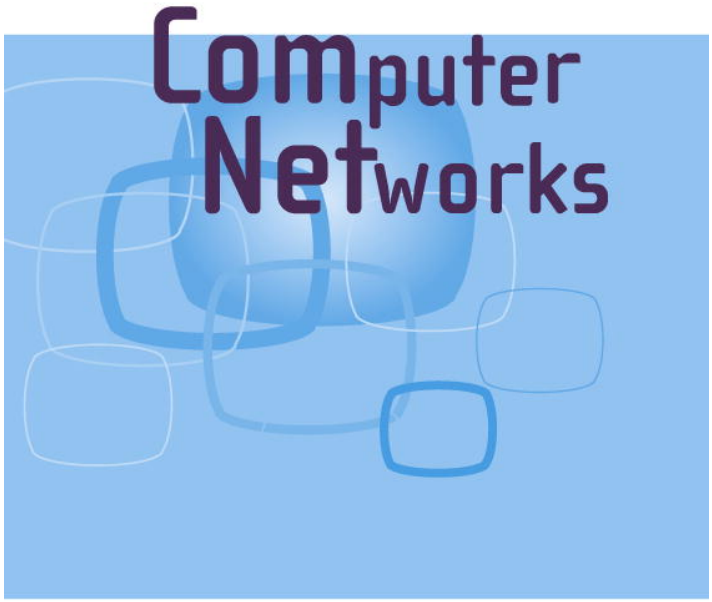
Volume 52 Issue 5 10 April 2008 ISSN 1389-1286



Available online at www.sciencedirect.com
ScienceDirect
Computer Networks 52 (2008) 915–934
Computer Networks
www.elsevier.com/locate/comnet

MuON: Epidemic based mutual anonymity in unstructured P2P networks ☆

Neelesh Bansod [1], Ashish Malgi [2], Byung K. Choi *, Jean Mayo

Department of Computer Science, Michigan Technological University, 1400 Townsend Drive, Houghton, MI 49931, United States

Received 19 June 2007; received in revised form 21 October 2007; accepted 14 November 2007
Available online 22 November 2007

Responsible Editor: M. van Steen

Abstract

A mutual anonymity system enables communication between a client and a service provider without revealing their identities. In general, the anonymity guarantees made by the protocol are enhanced when a large number of participants are recruited into the anonymity system. Peer-to-peer (P2P) systems are able to attract a large number of nodes and hence are highly suitable for anonymity systems. However, the churn (changes in system membership) within P2P networks, poses a significant challenge for low-bandwidth reliable anonymous communication in these networks.

This paper presents MuON, a protocol to achieve mutual anonymity in unstructured P2P networks. MuON leverages epidemic-style data dissemination to deal with churn. Simulation results and security analysis indicate that MuON provides mutual anonymity in networks with high churn, while maintaining predictable latencies, high reliability, and low communication overhead.

Keywords: Unstructured peer-to-peer (P2P) networks; Anonymity; Reliability; Epidemic protocol

1. Introduction

Reliable anonymous communication is required by certain online services operating over untrusted networks. Anonymity is needed to prevent third parties (or *adversaries*) from gathering information related to services and their clients. Examples of such applications include banking, e-voting, file sharing and searching, etc. Most online services have a common model of interactions. 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

☆ This material is based in part on work supported by the National Science Foundation under Grant No. CCR-9984682. Any opinions, findings, conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation. An earlier version of MuON was presented at IEEE ICNP 2005 in Boston MA.

* Corresponding author. Tel.: +1 906 487 3472; fax: +1 906 487 2283.

E-mail addresses: npbansod@mtu.edu (N. Bansod), asmalgi@mtu.edu (A. Malgi), bkchoi@mtu.edu (B.K. Choi), jmayo@mtu.edu (J. Mayo).

[1] Currently affiliated with Redback Networks, San Jose, CA 95134, United States.

[2] Currently affiliated with Microsoft Corporation, Redmond, WA 98052, United States.

1389-1286/$ - see front matter © 2007 Elsevier B.V. All rights reserved.
doi:10.1016/j.comnet.2007.11.011

the initiator. Based on this model of interactions, different types of anonymity [1–3] 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 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. All the terms related to anonymity research are well defined in [4].

Existing anonymity solutions use one of several basic approaches to achieve different forms of anonymity. In the simplest approach, a single proxy is used for communication between initiator and responder [5,6]. However, this approach fails if the proxy is compromised to reveal the identities of the communicating parties. Many anonymity protocols [7–10,2,11] overcome this single point of failure using indirection; messages from the sender (initiator/responder) are routed through intermediate relay nodes till they reach the final destination (responder/initiator). Finally, other protocols [12–14] provide anonymous communication by multicasting messages to a group of nodes. It is important to note that in these last two approaches, the anonymity in the system improves as the number of participant nodes increases.

This paper presents MuON, a protocol that shows a good potential to provide reliable, mutually anonymous communication with unlinkability. Inspired by the idea of multicasting, MuON uses epidemic-style data dissemination [15–17] to achieve reliable communication with the desired anonymity properties. As seen in prior work [15,17], epidemic protocols are much more efficient than reliable multicasting protocols. Intuitively, by broadcasting every communication message to all members reliably, this type of protocol should provide a very high potential to achieve anonymity. At the same time, however, broadcasting every message creates a concern on the inefficient use of limited network bandwidth. Thus, in general, limiting the scope of broadcast without (much) sacrificing the reliability will be an issue in this line of approach. This paper aims at extensively investigating a single scope of broadcasting using an epidemic protocol. To the best of our knowledge, this is the first study to use an epidemic protocol to provide anonymity. Multiple broadcast groups-based approaches will be the important future work.

MuON operates over unstructured P2P networks (such as the Gnutella file sharing system), which are known to be able to attract a large number of participants. Other anonymity protocols have been designed to operate over P2P networks. These protocols have used different kinds of P2P networks such as structured P2P systems [11], IP layer P2P systems [9] and hybrid P2P systems [3]. 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. However, unstructured P2P networks pose significant challenges. For example Gnutella is known to consume high bandwidth [18]. Other studies [19] have 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. Further, the nodes within the P2P network cannot be trusted. Peers may attempt to tamper with messages, masquerade as the responder, drop messages that they are supposed to forward, collude to violate the anonymity guarantees, or subvert the protocol by any other means.

Earlier work of MuON was presented previously in [20] and was shown to provide high reliability while maintaining low latencies and low overhead. Here we present an extended version of MuON with enhanced anonymity guarantees and additional performance metrics. The modifications ensure that the protocol terminates in dynamic networks without losing anonymity or reliability, thus bounding the consumed network resources. MuON has also been extended to operate without a trusted PKI (public key infrastructure). The extended version also includes modifications that enable MuON to withstand intersection attacks.

The paper is organized as follows: Section 2 summarizes the prior approaches for anonymity and introduces epidemic protocols. Section 3.1 discusses the goals of MuON. Section 3 describes MuON in detail, followed by the anonymity and performance evaluations in Section 4. The contributions and future work are summarized in Section 5.

2. Related work and motivation

This section reviews epidemic protocols and prior systems providing different kinds of anonymity over various network architectures.

2.1. Anonymity by mixes

Systems like **Mix-Net** [21], **Babel** [22], **MorphMix** [23] and **Mixminion** [24] are based on mix-networks. These systems hide communication by encrypting messages in layers of public key cryptography, and relaying messages through a group of dedicated message relays called 'mixes'. Each mix decrypts, delays and reorders messages before relaying them to another mix. While these systems achieve strong anonymity guarantees, the randomly introduced delays can result in high latencies unsuitable for interactive applications.

2.2. Anonymity by proxy

Systems like **Anonymizer** [5], **Lucent Personalized Web Assistant** [6], **PRA:Proxy for Responder Anonymity** [12] and **APFS Unicast** [12] use an intermediate proxy to provide anonymity. As these systems trust the proxy for performance and anonymity guarantees, they are vulnerable to failure if the proxy is compromised or if the proxy fails.

2.3. Anonymity by single-path forwarding

Many anonymity protocols forward 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 [10] provides anonymous communication using a dedicated set of message relays called 'onion routers'. The sender selects a path from the set of onion routers. It then wraps the data within encrypted layers to form an *onion*. The innermost layer of encryption in the onion uses the encryption key of the path's last hop, while the outermost layer uses the encryption key of the path's first hop. Onion routers co-operate and forward the onion from the sender to the destination.

Several systems use onion routing to create tunnels for anonymous communication. **Tor** [8] provides mutual anonymity by creating tunnels to rendezvous points, while **Tarzan** [9] provides initiator anonymity by building tunnels through an IP layer P2P system. Similarly, **MorphMix** [23] creates anonymous tunnels through P2P networks using nested encryption. **TAP** [11] uses replicated tunnels to provide reliable anonymous communication only within structured P2P networks.

Xiao et al. [3] propose protocols that use hybrid P2P networks to provide mutual anonymity. Trusted third parties coordinate anonymous message transfers between communicating entities. **Crowds** [2] provides initiator anonymity using random forwarding. Messages are sent to a randomly chosen peer called a *jondo*. Each jondo then randomly decides to either send the message to the responder or to another jondo.

In networks with high churn (nodes frequently join and leave the P2P network), approaches using single anonymous paths are likely to suffer from path losses. Consider an anonymous path through 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 network churn (increasing p), the probability that a given path remains 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 can be a high overhead operation and has not been extensively explored in the context of maintaining anonymity guarantees.

2.4. Anonymity by group communication

Many systems like **APFS Multicast** [12] and **Hordes** [14] multicast messages within a large group to achieve anonymous communication. Likewise **GAP** [7] uses controlled flooding to send a message throughout the network, while $\mathbf{P}^5$ [13] defines a logical hierarchy of broadcast groups, sending messages to groups of different sizes.

Protocols that depend on group communication primitives 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 [14] 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 [12,13]. GAP [7] takes a different approach, but achieves reliability by flooding, which may not scale well in large unstructured P2P networks.

2.5. Epidemic protocols

Epidemic (or gossip) protocols [25] are a well-studied class of protocols for low-cost reliable data dissemination within groups. They have been shown

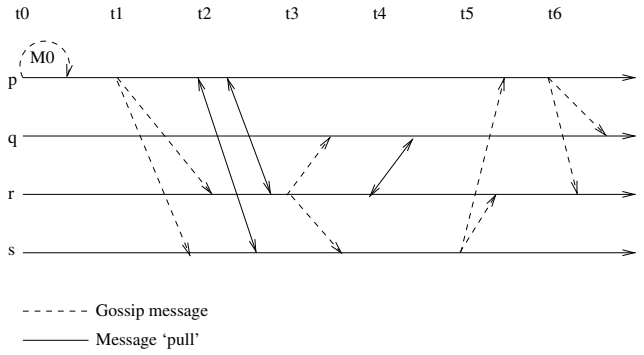


Fig. 1. Epidemic protocols.

to be much more efficient than flooding based approaches [26,27]. Epidemic protocols provide higher reliability and scalability while using lower bandwidth [16], when compared to other reliable multicast protocols. They provide a bimodal guarantee of reliability [15]; a 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 in the group. Due to these desirable characteristics, MuON uses an epidemic-style protocol for data dissemination.

A simplified gossip protocol is depicted in Fig. 1. Each node runs several rounds of the gossip protocol. In each round, a 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 (indicated 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 (indicated by solid lines between nodes).

Three important parameters impact gossip protocol performance. First is the number of gossip targets *FanOut* used in each round. (*FanOut* is two in the figure.) The second parameter is the time $D_{interval}$ between successive protocol rounds. (In the figure, as node p starts gossip rounds at time $t1$ and $t6$, the $D_{interval}$ is $t6 - t1$.) The final parameter is the number of rounds `GC` that a message is gossiped. These parameters determine the speed and efficiency of the protocol and have been rigorously studied by Birman et al. [15].

3. MuON

This section describes the details of MuON. We first give the goals, notation, system model and assumptions for deploying MuON, followed by a description of the basic data dissemination protocol. Finally, we describe the use of this data dissemination protocol to effect communication, between initiator and responder, with the desired properties.

3.1. Goals of MuON

MuON aims to balance between performance and anonymity in dynamic P2P networks. The main goals are described below:

1. **Mutual Anonymity**: Initiators and responders can communicate without knowing the actual identity of the other communicating party.
2. **Unlinkability**: Communication between initiators and responders cannot be correlated.
3. **Bounded Latency**: Latency is bounded.
4. **Reliability**: Reliable communication between initiator and responder(s).
5. **Communication Overhead**: Overhead incurred by a peer should be low.
6. **Scalability**: Metrics like reliability, anonymity, latency and overhead should scale well with the P2P network size and churn.
7. **Integrity and Confidentiality**: Peers cannot modify messages in-transit and cannot masquerade as responders; requests can be read only by the intended responder; responses can be read only by the appropriate initiator.

3.2. System model

MuON operates over an unstructured P2P network. Let N be the number of nodes within the overlay. We assume that nodes within the overlay know at least $\log(N)$ other peers in the overlay. This membership list for epidemic protocols can be maintained using services such as SCAMP [28], "Peer Sampling Service" [29], and DIMPLE [30]. MuON assumes that all initiators and responders are members of the P2P network. All protocol messages use low-cost unreliable transport (UDP) for communication.[3] Each peer is associated with a unique public key, which is used in lieu of actual node identities. The message sending protocol of MuON ensures that mutual anonymity and unlinkability are maintained, while the use of public and session keys maintains data integrity and confidentiality. The public keys

[3] Though messages between peers are sent unreliably, MuON reliably transfers messages between communicating entities.

are not tied to any specific algorithm; for example incomparable public keys [31] could be used.

To access a service in MuON, initiators require the public keys of peers that provide the corresponding service. MuON provides the ability to perform an anonymous network-wide search to map a *search query* (i.e. a desired service) to a set of public keys. Alternately a trusted PKI may be used to provide correct public keys corresponding to a search query. It is interesting to note that the communication between the PKI and the initiator must be anonymous. However, it is easy to conceive the PKI as a service within MuON, whose public key is well known and distributed out-of-band.

Peers are assumed to have approximately synchronized clocks; the difference between clock readings of any two peers at a single instant of time (called the *clock skew*) is within a known bound. Clocks may be synchronized in hardware, software, or some hybrid combination [32–34]. The increasing availability of global positioning system (GPS) based hardware allows physically dispersed systems to be synchronized [35] with a clock skew of a few microseconds[4] via their mutual synchronization to Coordinated Universal Time (UTC).[5]

Throughout this paper, the adversary is assumed to be passive and external. Depending on attacks, the adversary may be either local or global.

This paper considers logical network topologies induced by membership services, such as SCAMP [28] and DIMPLE [30], which are used for epidemic protocols. One common assumption in epidemic protocols is that each node can randomly select $\log(N)$ different nodes (fanout) in each cycle. As the network size grows, providing such information for each node is difficult for centralized service. Hence distributed algorithms and protocols have been sought in an effort to provide such membership information in a scalable manner. One common approach in that line of work is to locate a tiny subset (local view) of the entire membership at each node, and have the nodes exchange part of the local view with that of another randomly chosen node; this is known as *shuffling*. This seemingly simple operation in fact produces local views populated from the entire membership randomly and uniformly [36,37]. Thus a node knows at least as many nodes as the size of the local view. At any given instance, we can create a graph (called *knowledge graph*) reflecting the induced logical topology by adding an edge from P to Q if P knows Q. As reported in [37,30], such a knowledge graph is a random graph with the same number of nodes and edges. Assuming that MuON uses such a membership service, the network topology remains random independently of the physical networking.

Obviously, this claim does not directly involve shuffle message loss due to possible network congestion patterns. While congestion-embedded simulation would produce more realistic outcomes, applying flat message loss rates seems to be an alternative to focus on the effectiveness of the reliable broadcasting of MuON. Adopting this argument, this paper uses the simplified approach of using 10% packet loss rate throughout the simulation. Using network congestion patterns in simulation remains as future work.

An interesting question is whether this idea is directly applicable to wireless ad hoc mobile networks (MANET). The fundamental difference between MANETs and wired P2P systems is that while a peer can directly communicate with another peer using the underlying routing infrastructure, a mobile node is not able to without having other intermediate nodes forward a packet toward a destination in MANET. As a result, gossiping to fanout nodes would involve many other intermediate nodes in MANET. This, in turn, would make MuON-like gossiping impractical. Fundamental research needs to be done to support effective gossiping for MANET. The original question of whether MuON-like anonymous communication protocols can exist for MANET then could be answered after such fundamental work. This paper leaves this question open for future work.

3.3. Notation

The protocol uses the following notation.

Notation	Description
$k_{session}$	Symmetric session key
k_A^+, k_A^-	Public and private key of node A
r_1	Nonce
$\{data\}k_s$	*data* encrypted/signed using key k_s (k_s is public, private or session key)
$H(data)$	Cryptographic hash (e.g. SHA-1) computed over *data*

(*continued on next page*)

[4] Evaluation (Section 4) shows that MuON tolerates clock skews up to 18 × RTT (RTT = round trip time of UDP).

[5] UTC is independent of local timezone and daylight savings.

Notation	Description
self	IP-address of node
p_{inter}	Intermediate probability
α, β	Bounds on epidemic lifetimes
QUERY_HDR	Search query initiated by I. Format is $\{T_{deadline}, k_I^+, query\}$
MSG_HDR	Header associated with MSG. Format is $\{T_{deadline}, currOwner, hdr, H(hdr)\}$ Contents of *hdr* are defined in Section 3.5. If MSG_HDR is associated with a MSG, MSG_HDR. *currOwner* is non-null and indicates the node owning MSG
$T_{current}$	Current time on synchronized clock
T_{join}	$T_{current}$ recorded at start of session
$T_{deadline}$	Time at which message is invalidated
$skew_{max}$	Bounds on the clock skew within the system (accuracy is $\pm skew_{max}$)
$D_{interval}$	Interval between successive rounds
D_{avg}	Average life of peers in the system
FanOut	Number of gossip targets per round
GC	Number of rounds message is gossiped
MSG	Data (request/response) message

3.4. Data dissemination in MuON

MuON's data dissemination protocol 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. This protocol is then used for sending messages between initiators and responders and also for mapping services to the corresponding public keys (discussed in Section 3.5).

The data dissemination protocol operates via the messages MSG_HDR and MSG. MSG_HDR contains information such as identifiers and cryptographic keys, while MSG encapsulates the associated data. As MSG_HDR contains only protocol data, we assume[6] that the size of MSG_HDR is much less than MSG. The data dissemination protocol is depicted in Fig. 2, which shows node X sending MSG to node Y. MuON uses an epidemic protocol to disseminate

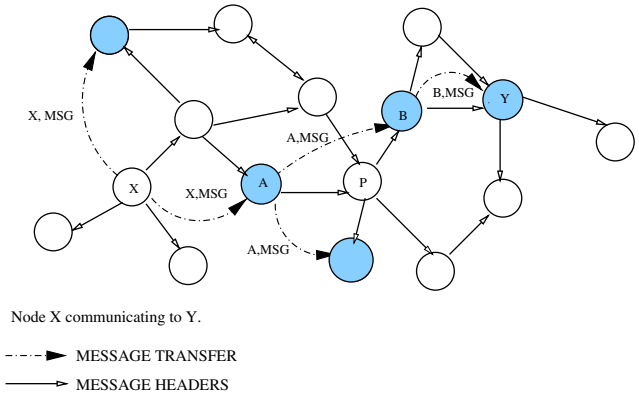


Fig. 2. Message sent from node X to Y.

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 Fig. 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 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 *headerBuffer* to store the message headers and the other *messageBuffer* to store the corresponding messages. Every node tracks the number of protocol rounds *gossipCount* each header has been gossiped. Algorithm 1 describes the details for handling these buffers. Algorithm 2 explains the protocol executed by each node after every $D_{interval}$ 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[7] 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

[6] This assumption holds true in applications with large responses (e.g. file-transfer and web-browsing). In these applications, MuON achieves substantial bandwidth savings compared to other group communication based anonymity protocols. We anticipate that MuON will also 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.

[7] If the decrypted message contains an expected value like a known identifier or public key, the node can conclude "successful" decryption.

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 indicates itself as the message owner. With this property, MuON achieves its anonymity guarantees as there are many equiprobable owners of the same 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 in MuON behaves exactly the same as any other node in the network (with the exception that it always pulls the message).

```
/*Adding a MSG_HDR or QUERY_HDR*/
addheader(HDR):
begin
    slot = free slot in the headerBuffer
    headerBuffer[slot].HDR = HDR
    headerBuffer[slot].gossipCount = 0
end
/*Add MSG and the corresponding MSG_HDR*/
addmessage(MSG, MSG_HDR):
begin
    addheader(MSG_HDR)
    Add MSG to the messageBuffer
    Associate MSG with MSG_HDR.H(hdr)
end
/*Sending message MSG with header hdr*/
sendMessage(hdr, MSG):
begin
    Let lifetime = random number between α and β
    T_deadline = T_current + lifetime
    currOwner = NULL
    if MSG is not NULL then
        currOwner = self
    MSG_HDR={T_deadline, currOwner, hdr, H(hdr)}
    addmessage(MSG, MSG_HDR)
end
/*Sending a search query message*/
sendQuery(search query):
begin
    Let lifetime = random number between α and β
    T_deadline = T_current + lifetime
    QUERY_HDR={T_deadline, k+_self, search query}
    addheader(QUERY_HDR)
end
```

Algorithm 1: Common procedures

```
/*Runs every D_interval units of time*/
gossipRound:
begin
    gossipMesssage = all HDR ∈ headerBuffer
    for i=0 to FanOut do
        Randomly select a peer n_i from the overlay
        Send gossipMessage to n_i
    for every used slot in headerBuffer do
        headerBuffer[slot].gossipCount++
        if headerBuffer[slot].gossipCount > GC then
            Free headerBuffer[slot] and remove
            corresponding MSG from messageBuffer
end
```

Algorithm 2: Gossip round

```
/*When node receives gossipMessage*/
onRecvGossipMessage:
foreach HDR ∈ gossipMessage do
    if HDR.T_deadline < T_current then next header
    if HDR ∈ headerBuffer then next header
    if HDR= {T_deadline, k+_X, search query} then
        deliver HDR
        addheader(HDR)message
    if HDR.currOwner = NULL then
        if HDR.hdr decrypted then deliver HDR
        else addheader(HDR)message
    p_current = currentProbability()
    if HDR.hdr can be deciphered then
        /*MSG identified by H(hdr)*/
        HDR.currOwner for MSG
        when MSG arrives then
            deliver MSG
            if true with probability p_current then
                /*Setting self as currOwner*/
                .currOwner = self
                addmessage(MSG, HDR)
            else
                addheader(HDR)
        end
    else
        if true with probability p_current then
            /*MSG identified by
            H(hdr)*/HDR.currOwner for MSG
            when MSG arrives then
                /*Setting self as currOwner*/
                .currOwner = self
                addmessage(MSG, HDR)
            end
        else
            addheader(HDR)
currentProbability:
begin
    if D_avg = ∞ then return p_inter
    else return min(1, p_inter * (T_current − T_join)/D_avg)
end
```

Algorithm 3: Receiving a gossip message

3.5. Communication in MuON

This subsection first describes how a node within MuON can anonymously obtain the public key of a responder. We then describe how the data dissemination protocol is used for anonymous communication between initiator and responder.

Obtaining public keys of responders: To obtain public keys of responders providing a service, an initiator I multicasts a *search-query* describing the service by invoking *sendQuery*(search-query). *sendQuery* encapsulates the search-query and key k_I^+ into a `QUERY_HDR` and disseminates it throughout the network using a push-epidemic protocol (The relevant portions of Algorithms 2 and 3.). Thus the query is delivered to each peer within the network. On receiving a search-query, a peer R that provides a matching service anonymously sends its public key k_R^+ to node I (This is equivalent to invoking *sendMessage*($\{k_R^+\}k_I^+$, *NULL*)). On receiving k_R^+, node I can now anonymously access the service.[8]

Initiator and responder communication: While the data dissemination protocol achieves anonymity, cryptographic measures are needed to ensure message integrity and confidentiality. The following describes how the data dissemination protocol is used by initiators and responders for secure anonymous communication.

Sending a request: An initiator I uses the following steps to initiate communication with node R by sending the request *data*.

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. The `MSG` is generated as $\{r_1, data\}k_{session}$.
4. I creates a header *hdr*, corresponding to `MSG` as $hdr = \{r_1, k_{session}, k_I^+, \{H(D)\}k_I^-\}k_R^+$ where $D = \{r_1, k_{session}, k_I^+, \texttt{MSG}\}$ and $H()$ is a cryptographic hash function.
5. I invokes *sendMessage*(hdr, `MSG`).

Responding to a request: Algorithm 3 eventually delivers `MSG_HDR` and `MSG` to the responder R, which proceeds with the following steps.

1. R decrypts *hdr* using k_R^-, 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 obtain *data*.
3. Let *response* be the corresponding reply, which R needs to send to I. R creates $\texttt{MSG} = \{r_1, reponse\}k_{session}$ using $k_{session}$ and r_1 as recovered in step 1.
4. R creates a header *hdr* corresponding to the response as $hdr = \{r_1, \{H(D)\}k_R^-\}k_I^+$, where $D = \{r_1, \texttt{MSG}\}$ and $H()$ is a cryptographic hash function.
5. R invokes *sendMessage*(hdr,`MSG`).

3.6. Miscellaneous protocol properties

Epidemic Termination: Epidemic protocols achieve reliability by gossiping messages until all protocol participants receive (with very high probability) the message. However, in dynamic P2P networks, the set of protocol participants changes unpredictably over time. Reaching all protocol participants would then result in a perpetually surviving epidemic. Hence, MuON requires some other mechanism for termination of the epidemic without compromising its anonymity properties.

A fixed value of GC is not adequate to terminate the epidemic. Each time a node joins the network and receives a given header for the first time, it will disseminate the header GC additional times. Since the network membership changes continually, epidemics may survive perpetually. Another simple approach is to choose some maximum integer count of rounds for which a message is propagated. The initiator chooses this count value and appends it to the message; each subsequent propagation of the message by any node decrements the count by one. When the count reaches zero, the message is no longer propagated. However, implementing this approach without sacrificing anonymity is difficult. Suppose each initiator randomly selects the round count from within some range of values. Then any node that chooses the maximum value of the range can be identified as initiator.

MuON applies approximately synchronized clocks toward epidemic termination. A MuON header originating at time T is associated with a deadline $T_{deadline}$ that specifies the wall-clock time at which the header is to be invalidated. This ensures that even in dynamic networks, the associated epidemic terminates at approximately $T_{deadline}$ and the epidemic has a *lifetime* of approximately $T_{deadline} - T$.

[8] A query may result in multiple responses. A reputation scheme [38,39] may be used to select one trusted response.

If all messages have the same lifetime, a peer directly receiving a header from the initiator can guess the identity of the initiator. Hence the message lifetime is chosen between α and β, where $\beta > \alpha$. MuON uses a value of α equal to the maximum communication latency for epidemic protocols in static networks ($\approx log_2$(overlay size) [15]). This ensures that with high probability all nodes that are in the network when the dissemination begins, that remain in the network over the *lifetime* interval, and that remain reachable, will receive the message. Though all nodes choose a lifetime value between α and β, $T_{deadline}$ is computed by adding the selected lifetime value to the local synchronized clock value. Since local clocks are only approximately synchronized, the identity of the initiator is obscured. Using $T_{deadline}$ an adversary receiving a header directly from the initiator can guess the message's time of origin (and detect the initiator) with a probability $\frac{1}{(\beta-\alpha)\times(1+2\times skew_{max})}$.[9] Thus clock skew (and clock synchronization) is essential for terminating MuON epidemics anonymously in this way.

Aligning with long-lived nodes: In dynamic P2P networks, peers alternately join and leave the network. As discussed in section 15, this continual change in system membership aids an adversarial attack called an *intersection attack*. To enhance anonymity, peers in MuON pull data messages with a probability p proportional to the time that has elapsed since joining the system (termed *age*). Thus in MuON long-lived nodes pull data messages with a higher probability.[10] For simplicity, we consider p increasing linearly with the peer's age (as computed by method *currentProbability* in Algorithm 1), though other relations may be suitable depending on the network model.

4. Evaluation

In this section, we describe MuON's simulation model and evaluate the performance, anonymity and other security guarantees.

4.1. Simulation model

Measurement studies of unstructured P2P networks [40,19,41] 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* (or *age*) 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 [42,43]. Prior experiences [44,42,43] indicate that network churn impacts the performance of protocols over P2P networks. Hence we evaluate MuON over P2P networks of varying sizes and varying churn.

We model network churn using an approach similar to that described by Liben-Nowell et al. [45]. This model has also been used for evaluating distributed hash tables over P2P networks [42,43]. 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 work [46] has 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.

Simulating network churn has been relatively well addressed in the literature. One issue is whether a good mathematical distribution exists to model network churn. Some work [47] suggest the use of a distribution function backed up by real measurements, while some others [45] advocate the same method adopted in this paper. A second issue in simulating network churn is whether the network size should remain constant. A common approach used in the literature is to maintain a fixed network size in individual simulation runs. This helps in decoupling the impact of network size from that of network churn. Furthermore, to simulate networks with varying sizes, one requires an underlying management service which can provide the network size dynamically, since epidemic protocol parameters depend on the network size. Estimating network size can be found in separate work such as [48]. Including dynamically changing network sizes and different distribution models for network size, would produce more realistic outcomes. However, since this paper focuses on anonymity these improvements are left for future work.

[9] If initiator with clock skew *skew* sends a message at wall-clock time T, then $T_{deadline} = T + lifetime + skew$. For a given $T_{deadline}$, an adversary can guess *lifetime* with a probability $\frac{1}{\beta-\alpha}$ and *skew* with a probability $\frac{1}{1+2\times skew_{max}}$.

[10] P2P applications using MuON for communication, may use this property to align with the network topology, though details are not discussed within this paper.

MuON is simulated using PeerSim [49], 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 unreliable (UDP) message transmission with a loss rate of 10%.[11] This UDP loss rate of 10% is a rough model of congestion. All time in the simulations is measured in terms of cycles, which is the smallest unit of time within the simulator. In the simulation model, a network with churn 0 is a static network, which does not change during the simulation. At churn 0, the average session time was chosen as 100 cycles (a factor of 10 over the maximum time for one run of the protocol in a system of 1000 nodes), 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, another immediately joins, thus keeping the overlay size constant. This helps us to understand the impact of overlay size and churn independently. We also model networks with varying clock skew. If the clock skew in a simulated network is *skew*, then the clock skew of each node is selected uniformly between $-skew$ and $+skew$. The node's clock skew remains constant throughout its lifetime.

In the simulations, `FanOut` and `GC` are maintained at $\log_2$(overlay size) and $D_{interval}$ is maintained at one cycle. These parameters are common to all epidemic protocols and their impact on performance is similar to that determined by previous studies [15]. We maintain $\alpha = log_2$(overlay size) and $\beta = 2 \times \alpha$ to bound epidemic lifetimes such that message delivery may be attempted but the epidemic is terminated eventually. The parameter D_{avg} controls the rate of increase of $p_{current}$. We simulate two different approaches for determining the value of D_{avg}. In the first approach (denoted by $\gamma = 0$), each node computes D_{avg} using the session-time of its previous sessions.[12] In the second approach, the membership service[13] is used to provide D_{avg}. We use the following values for D_{avg}; $0.5 \times D_{life}, D_{life}$ and $2 \times D_{life}$ (denoted by $\gamma = 1, 2$ and 3, respectively) where D_{life} is the average session time within the network. Thus $\gamma = 1, 2$ and 3 represents increasing D_{avg}.

4.2. Performance evaluation

This section discusses MuON's performance in networks with varying sizes and churn. Since MuON cannot provide reliable communication at churn 1, we measure performance metrics (other than reliability) in networks with churn between 0 and 0.8. Performance metrics are averages over multiple messages exchanged between initiator and responder for a given set of parameters such as skew, γ and p_{inter}.

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. When delivery ratio is one, it indicates that all requests that were sent were eventually delivered at their destination, thus indicating reliable communication. Fig. 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 Communication Latency: One of MuON's goal 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 message losses. Since MuON operates over an overlay network, we measure the latency in terms of number of protocol cycles required for the message to be

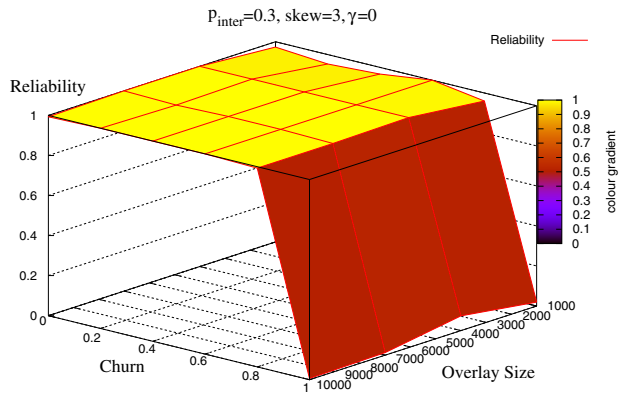


Fig. 3. Reliability.

[11] In general, the time required for unidirectional UDP message transmission is 0.5 × round trip time. Based on reported average round trip times [50], one cycle is approximately 50–100 ms.

[12] $D_{avg} = \infty$ during the first session.

[13] Membership services [28] using unsubscriptions or leases can estimate the average session-time of participant peers.

delivered at its destination. (The duration of a protocol cycle corresponds roughly to the time required for delivery and processing of a single UDP message.). Fig. 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 similar at each network size irrespective of network churn, indicating that the latencies in MuON are predictable and bounded. The latency does increase sublinearly with increasing network size, since the message lifetime is proportional to log(network size). When a peer pulls a data message, it does not gossip the header till the data message arrives. Hence as seen in Figs. 5 and 6, when data messages are pulled by a greater number of peers (caused by increasing p_{inter} and decreasing D_{avg}), the latency increases marginally.

Epidemic Termination: As discussed in Section 3.6, epidemic protocols tend to survive perpetually in dynamic networks. MuON terminates message epidemics by associating a deadline with each header in a system with approximately synchronized clocks.

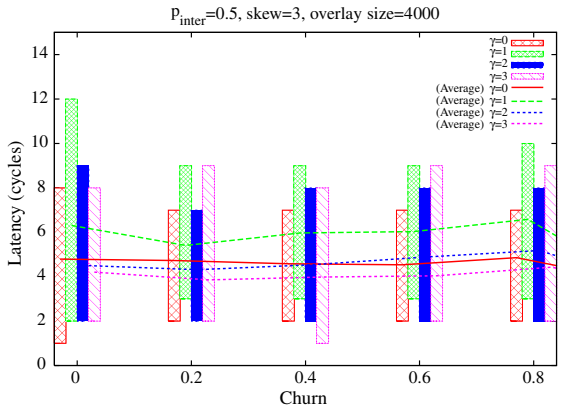


Fig. 6. Impact of D_{avg} on latency.

We study the effectiveness of this approach by measuring the epidemic lifetime in the networks of varying sizes and churn. A message's epidemic lifetime is measured as the duration of time (measured in cycles) during which at least one peer actively gossips the corresponding header. Figs. 7–10 show that the epidemic lifetime is not affected by overlay size,

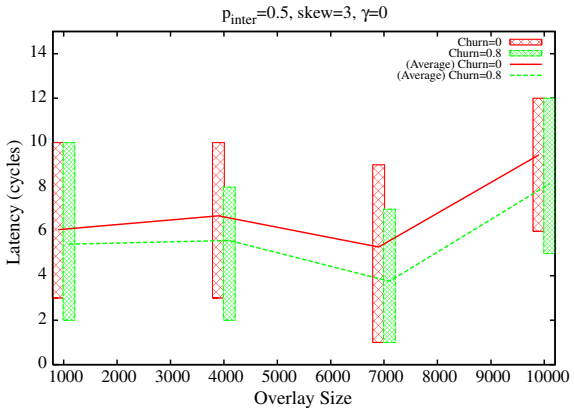


Fig. 4. Latency of message delivery.

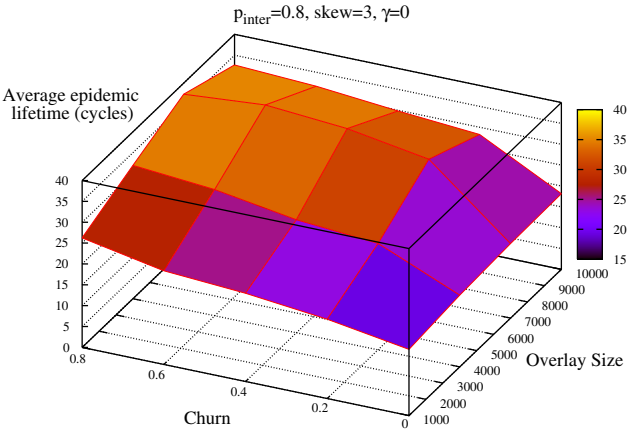


Fig. 7. Epidemic lifetime.

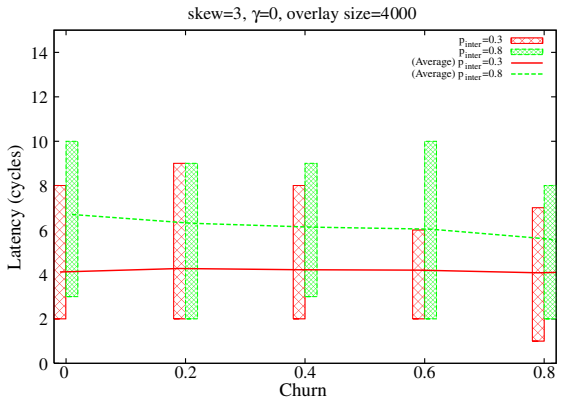


Fig. 5. Impact of p_{inter} on latency.

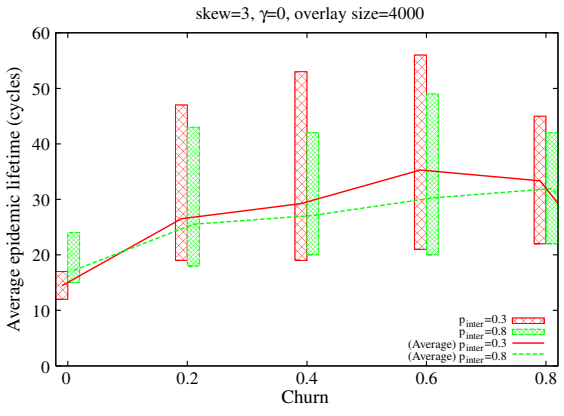


Fig. 8. Impact of p_{inter} on epidemic lifetime.

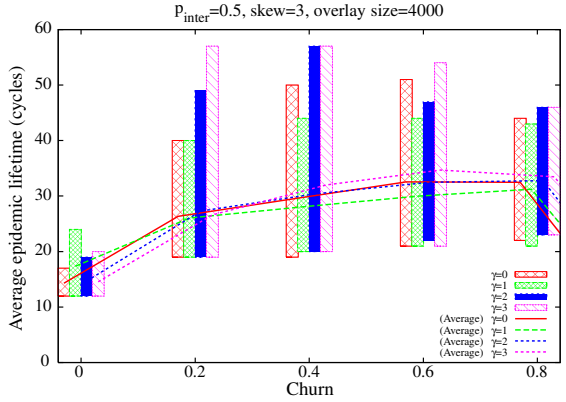


Fig. 9. Impact of D_{avg} on epidemic lifetime.

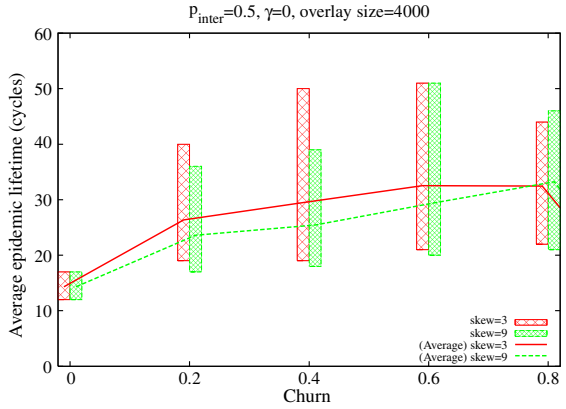


Fig. 10. Impact of skew on epidemic lifetime.

D_{avg}, p_{inter} and skew. Fig. 7 shows that MuON successfully terminates epidemics despite of high churn, though the epidemic lifetime increases gradually.

Node Overhead: When a peer joins MuON, it contributes its resources to forward messages from other peers. These resources are needed to send, encrypt, decrypt and store messages and are proportional to message size. Similar to prior work in epidemic protocols [32], we study the node overhead in MuON by measuring the header and data throughput. The throughput is measured as the number of header (or data) messages processed per cycle per node for each anonymous message sent in the network. Fig. 11 shows that the throughput is essentially independent of the overlay size and increases with churn. It can be seen that the throughput of larger data messages is much lower than the header throughput. Header messages are small in size and thus the processing overhead for each header, storage and bandwidth is low. MuON uses private/public key encryption for small headers and faster symmetric cryptography for large data messages,

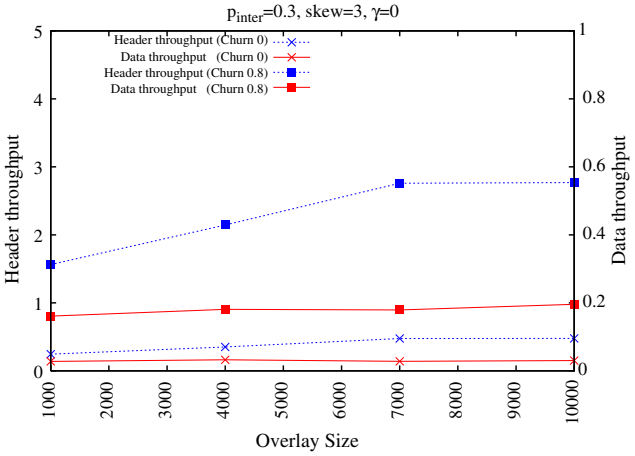


Fig. 11. Node overhead.

to reduce the encryption overhead. Fig. 12 indicates that D_{avg} does not significantly impact the node overhead. Fig. 13 shows that the data throughput increases with p_{inter}, while not impacting the header throughput.

Network Resources: MuON's message sending protocol is designed to use low network resources as compared to previous multicast-based anonymity protocols. The network resources can be measured by the network bandwidth consumed, when one data message is sent anonymously.

Let HDR_{size} and $DATA_{size}$ be the size of header and data messages, respectively, and N be overlay size. When a message is sent anonymously, a multicast-based anonymity protocol multicasts this message to N nodes. Hence the consumed bandwidth will be at least $N \times DATA_{size}$. 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

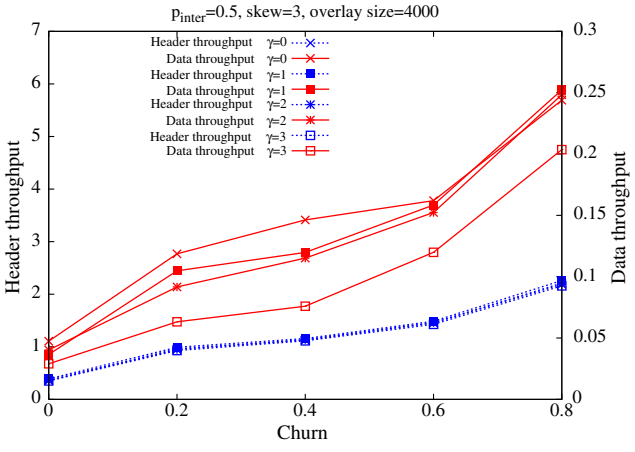


Fig. 12. Impact of D_{avg} on node overhead.

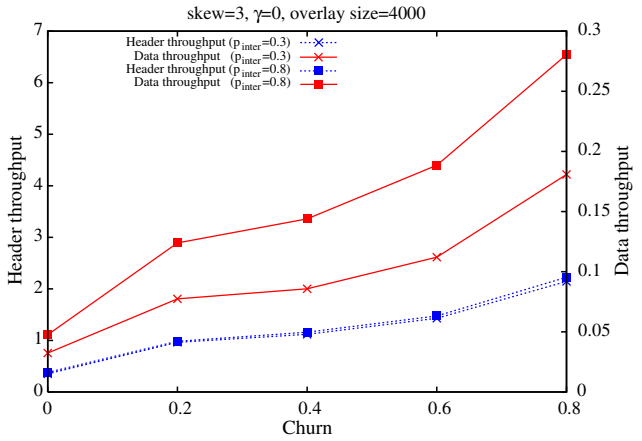


Fig. 13. Impact of p_{inter} on node overhead.

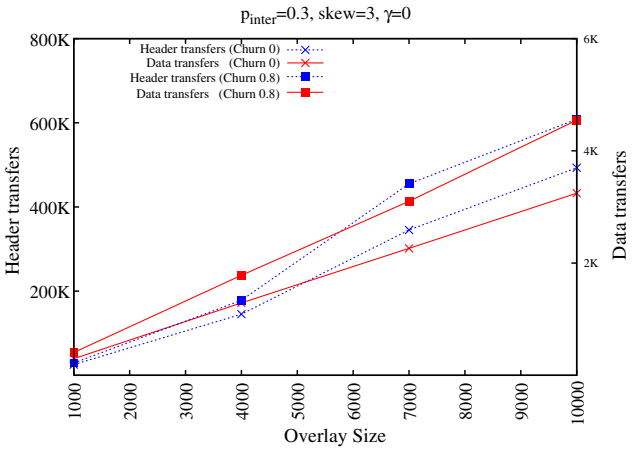


Fig. 14. Network overhead.

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. The bandwidth consumed in MuON is $\theta \times HDR_{size} + \phi \times DATA_{size}$, where θ and ϕ is the number of header and data transfers in the network. Considering the values of θ and ϕ from Fig. 14, it can be seen that MuON provides reliable anonymous communication at a cost lower than $N \times DATA_{size}$.[14] Fig. 17 depicts the bandwidth consumed in MuON as compared to a multicast based approach, while Figs. 15 and 16 show that the bandwidth consumed by data messages increases with increasing p_{inter} and decreasing D_{avg}, while the resources consumed by header messages are unaffected.

Scalability: Scalability is an important characteristic in systems designed for P2P networks. It is evident from the presented results that MuON is scalable. The protocol exhibits desirable performance irrespective of the overlay size and churn.

4.3. Anonymity guarantees

This section describes how MuON achieves mutual anonymity, followed by an evaluation of anonymity in overlays of varying sizes and churn. We then describe the protocol behavior under various adversarial attacks. The anonymity metrics are averages over multiple messages exchanged between multiple initiator and responder pairs for a given set of parameters such as skew, γ and p_{inter}.

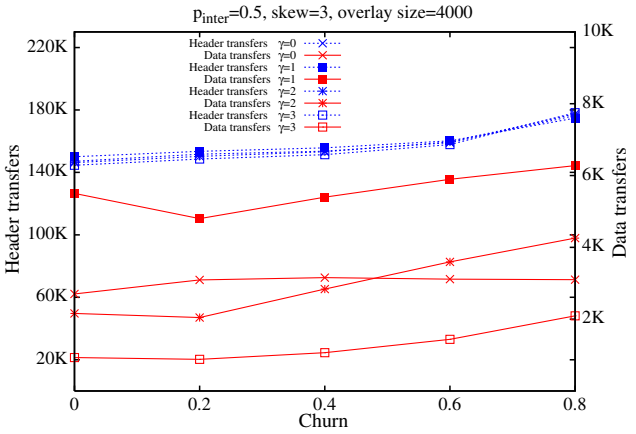


Fig. 15. Impact of D_{avg} on network overhead.

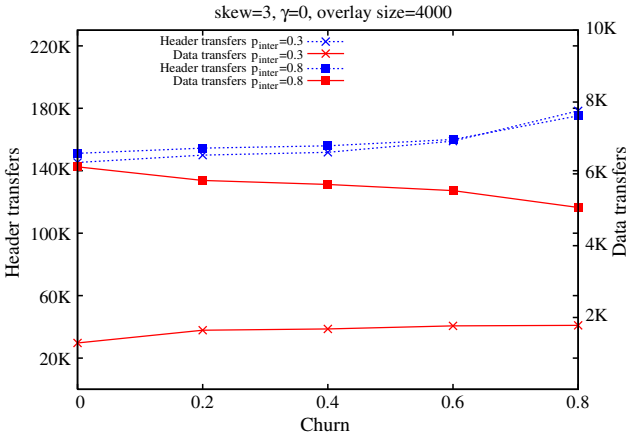


Fig. 16. Impact of p_{inter} on network overhead.

[14] 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 transferred data is a 1 MB media file then $DATA_{size} = 1,048,576$ bytes. From Fig. 14, if the overlay has 10,000 nodes with churn 0.8, then $\theta = 600,000$ and $\phi = 4500$. Hence the volume of MuON header messages is 41.2 MB and the volume of MuON data messages is 4500 MB. In this system with 10,000 nodes, $N \times DATA_{size} = 10,000$ MB.

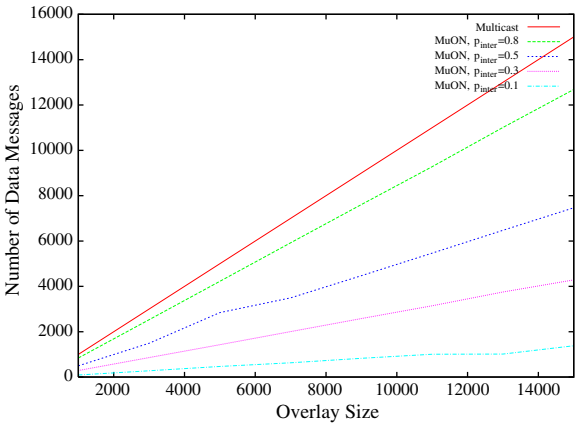


Fig. 17. Comparative bandwidth use.

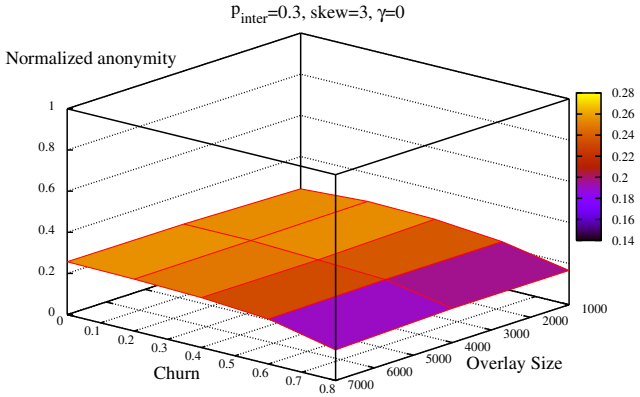


Fig. 18. Anonymity.

4.3.1. Mutual anonymity in MuON

Since MuON does not make use of cover traffic and padding, and does not hide information such as message lengths, $T_{deadline}$ and *currOwner*, epidemic traffic can be observed at each individual node. Consequently as discussed in Section 4.3, MuON is vulnerable to a sustained attack by a global adversary. However as discussed in the sections below, the anonymity guarantees of end-to-end traffic between initiators and responders are maintained in the presence of adversaries that can observe only a limited portion of the network for a given period of time.

Similar to anonymity protocols that use multicasting [12] or broadcasting [13], MuON achieves mutual anonymity on the virtue that several intermediate peers receive a message. 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 probabilistically pulls the corresponding `MSG`. Hence from the local observer's perspective, any peer that eventually pulls `MSG` could potentially be the receiver. Thus in the protocol, an observer (initiator, responder or intermediate node) cannot differentiate the initiator and responder from the other peers. The use of public keys also enables the initiator and responder to communicate without the identity of each other. Thus mutual anonymity and unlinkability is achieved.

The degree of anonymity provided by an anonymity system 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. Shields et al. [14] show that the degree of anonymity in the system is $1 - \frac{1}{|S|}$. In MuON, for a given communicating pair of initiator and responder, any node that receives `MSG` has some probability of being the initiator or responder. In this paper, all such nodes are collectively considered the anonymity set. This is because a local adversary cannot tell if the probability of being the sender or receiver is equal for all such nodes without a large scale collusion. Intersection attacks by a global adversary would make this anonymity set smaller. Formal analysis of this anonymity set using an information theoretic technique such as introduced by [51] is left as an important future work. Therefore, we measure the anonymity of MuON as the fraction of peers within the overlay that received a given `MSG` (called the *normalized anonymity*). Fig. 18 shows that the anonymity is almost constant in networks of varying sizes and churn. Figs. 19 and 20 indicate that the anonymity is controlled by p_{inter} and D_{avg}.

Attacks by Adversary: Anonymity systems are susceptible to several possible attacks. However, the adversary must utilize varying amounts of resources to complete these attacks successfully. To withstand powerful adversaries, several systems [10,9,3,11,8] use techniques like cover traffic generation, message buffering and per-hop message encryption. This paper evaluates MuON without using these high-cost[15] approaches, by describing the protocol behavior under various known attacks.

[15] Cover traffic increases network bandwidth consumption, message delays introduce communication latencies, and per-hop message encryption necessitates negotiation of shared keys between each pair of nodes in the dynamic network.

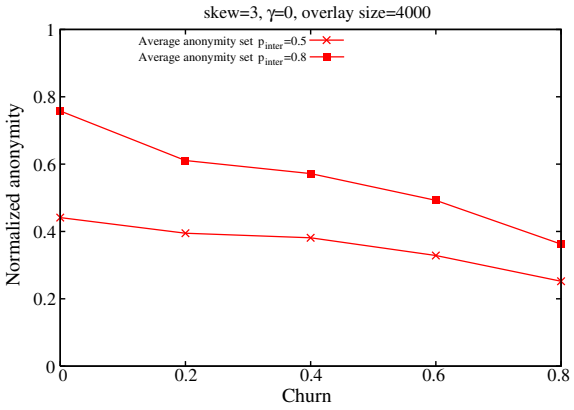


Fig. 19. Impact of p_{inter} on anonymity.

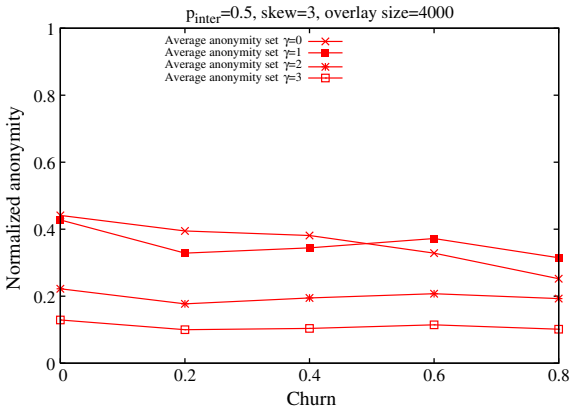


Fig. 20. Impact of D_{avg} on anonymity.

In addition, this paper assumes that a good pseudo ID management is in place. An intermediate node in MuON can collect pseudo IDs, such as public key as a pseudo ID, by participating the gossiping behavior. In order to prevent such a problem, a mechanism needs to be in place which changes the pseudo ID randomly at every interval. The general problem of managing pseudo IDs is left as future work. Follow up work will incorporate a mechanism into MuON.

Local eavesdropper: A local eavesdropper is an adversary that can monitor all communications sent to or received from one particular peer. 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 peer, cannot confirm the identities of the communicating parties, even if the message and its header are received by the peer. This is because the peer cannot determine the identity of either sender or receiver with certainty based on the information of the header. Likewise a local eavesdropper an initiator (or responder) cannot deduce the identity of the responder (or initiator).

Collusion Attack: In a collusion attack, peers collaborate to identify communicating entities. These colluding peers could be physically different nodes or a single node with multiple identities. 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 pair of peers within the anonymity set do not collaborate, it is hard for colluding nodes to differentiate the initiator, responder and the honest peer(s) from one another. If all $|S|$ nodes within the anonymity set collaborate, MuON's degree of anonymity becomes 0 and the identities of the communicating parties can be revealed. However, since the anonymity set changes for every `MSG` in the system, a large fraction of peers must collaborate to successfully launch this attack.

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, 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 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 a message (it is the starting point of the message paths). However the recipient is not revealed (since paths do 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 unlikely in MuON.

Traffic volume attack: An adversary can differentiate responders from other nodes by observing the volume of data transmitted, since initiators generate less data as compared to responders. This attack is possible against MuON, if the adversary can observe the traffic generated by each node in the network.

Intersection Attack: An adversary can record the members within the anonymity set of a message by launching a traffic volume or traceback attack. To perform an intersection attack, the adversary first records the anonymity sets of messages for an extended period of time. The adversary then computes the *effective anonymity set*, which is the intersection of all observed anonymity sets. Since the initiator and responder are present in each anonymity set,[16] they are always included in the effective anonymity set. Thus if the network is observed for a sufficient period, the effective anonymity set can be reduced substantially. We measure the progress of an intersection attack by measuring the size of the effective anonymity set at regular time intervals, relative to the size of the first observed anonymity set. For simplicity, we assume the adversary observes the entire network.

In dynamic networks, an intersection attack is assisted[17] due to changes in system membership. To offset this effect, MuON peers pull data messages with a probability proportional to their age in the system. Fig. 21 shows the fraction of peers within different age-groups that pull a given message. Hence the anonymity sets of successive messages include the same subset of long-lived nodes, resulting in larger effective anonymity sets. Fig. 22 shows the progress of an intersection attack in MuON

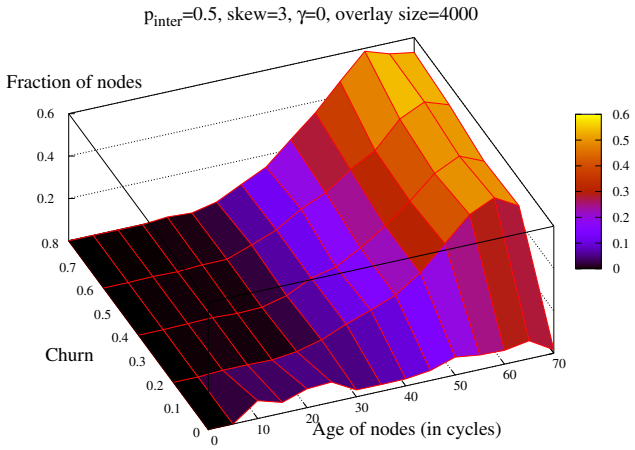


Fig. 21. Message distribution.

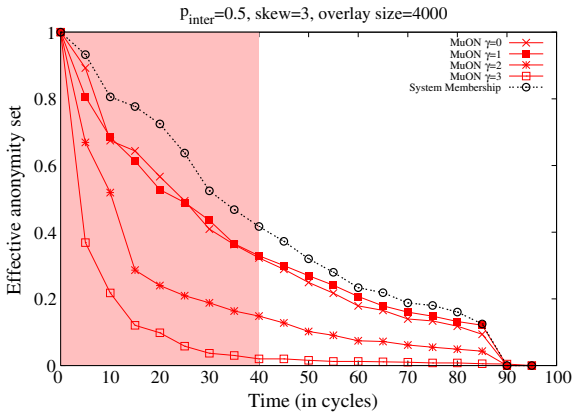


Fig. 22. Progress of an intersection attack (churn=0.6).

(depicted by solid lines) along with the effective anonymity set solely due to network churn (depicted by the dashed line). It can be seen that for low values of D_{avg}, an intersection attack proceeds primarily due to network churn. According to the simulation model, since the network in Fig. 22 has churn 0.6, the session time of peers is exponentially distributed with an average of 40 cycles (shaded region). Thus by the time the communicating entities complete their communication, the effective anonymity set is still greater than 0.3. Fig. 23 shows the progress of an intersection attack in networks with varying churn. Figs. 22 and 24 indicate that defense against intersection attacks can be improved by decreasing D_{avg} and increasing p_{inter}.

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 [52].

[16] This assumes that the initiator and responder are within the portion of the network observed by the adversary.

[17] This motivates some systems [13,14] to restrict system membership changes, and perform periodic system resets.

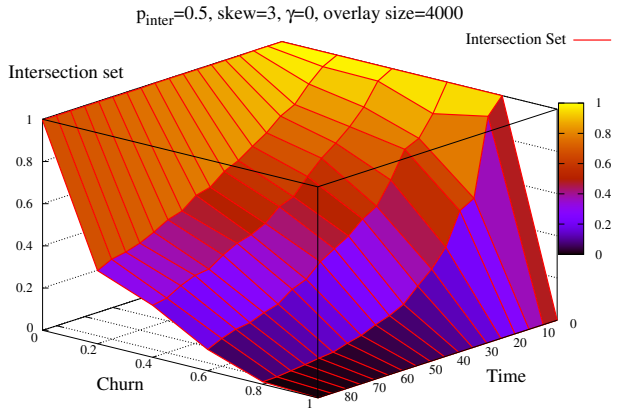


Fig. 23. Effective anonymity set.

4.4. Security guarantees

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

When the initiator sends the request, it generates a nonce r_1 and a session key $k_{session}$. The initiator then generates a header containing the nonce, session key and the initiator's public key. The header is then encrypted using the responder's public key. Similarly, the responder 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. If the responder keeps track of nonce values of the past requests, it can detect the replay of requests.

It is important to note that `MSG_HDR` contains an unsigned value for $T_{deadline}$, since an intermediate node cannot verify a signature without knowing the public key of the originator. Hence a misbehaving intermediate node may increase or decrease the value of $T_{deadline}$ to either cause the epidemic to die slowly or quickly; a slow dying epidemic increases the resources consumed by the protocol, while a quick dying epidemic would prevent the message from reaching the destination. However due to the inherent redundancy within the epidemic, a significant percentage of peers need to be misbehaving to consistently cause delivery failures. In either scenario, the anonymity and security guarantees of the protocol remain intact. In general, promoting

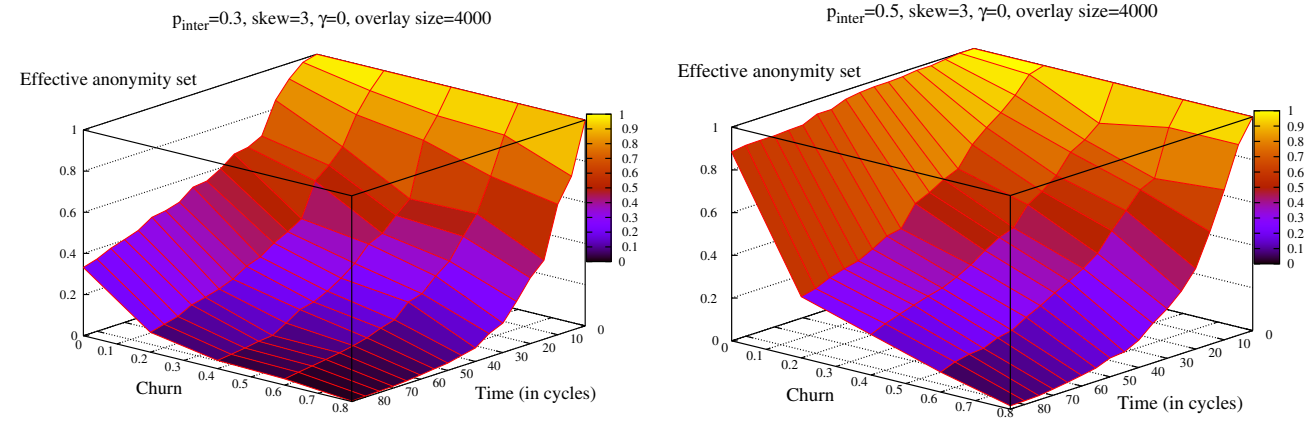


Fig. 24. Impact of p_{inter} on effective anonymity set.

compliance within P2P anonymity systems [53] is an important research problem, which we plan to investigate in future work.

5. Conclusion and future work

We have presented MuON, a protocol for providing mutual anonymity in dynamic P2P networks. There are two key contributions of MuON; the protocol provides reliable mutually anonymous communication over dynamic P2P networks, while maintaining low bandwidth and processing overhead compared to existing reliable multicasting approaches, and it exhibits application friendly characteristics such as bounded communication latency and message integrity and confidentiality. A unique feature of MuON is the ability to maintain anonymity against sustained adversarial attack for a reasonable amount of time. In order to help resolve the inefficient bandwidth use problem due to the broadcasting nature, a methodology to create multiple broadcast groups with high inter-group connectivity for reliable message delivery is currently under investigation. In the future, we plan to investigate the use of MuON for creating censorship resistant services and 'Denial-of-Service' DoS tolerant anonymous services.

References

[1] Pfitzmann Andreas, Waidner Michael, Networks without user observability, Computers and Security 6 (2) (1987) 158–166.

[2] Michael K. Reiter, Aviel D. Rubin, Crowds: anonymity for web transactions, ACM Transactions on Information and System Security 1 (1) (1998) 66–92.

[3] Xiao Li, Xu Zhichen, Zhang Xiaodong, Low-cost and reliable mutual anonymity protocols in peer-to-peer networks, IEEE Transactions on Parallel and Distributed Systems 14 (9) (2003) 829–840.

[4] Andreas Pfitzmann, Marit Hansen, <http://freehaven.net/anonbib/cache/terminology.pdf>.

[5] AT&T, The anonymizer, <http://anonymizer.com/>.

[6] Eran Gabber, Phillip B. Gibbons, David M. Kristol, Yossi Matias, Alain Mayer, Consistent, yet anonymous, web access with LPWA, Communications of ACM 42 (2) (1999) 42–47.

[7] Krista Bennett, Christian Grothoff, GAP-practical anonymous networking, in: PET'03: Privacy Enhancing Technologies Workshop, Dresden, Germany, March 2003, pp. 141–160.

[8] Roger Dingledine, Nick Mathewson, Paul Syverson, Tor: The second-generation onion router, in: Security'04: Proceedings of the 13th USENIX Security Symposium, San Diego, CA, August 2004, pp. 303–320.

[9] M. Freedman, R. Morris, Tarzan: A Peer-to-Peer Anonymizing Network Layer, in: Proceedings of the 9th ACM Conference on Computer and Communications Security, Washington, D.C., November 2002, pp. 193–206.

[10] Michael G. Reed, Paul F. Syverson, David M. Goldschlag, Anonymous connections and onion routing, IEEE Journal on Selected Areas in Communications 16 (4) (1998) 482–493.

[11] Yingwu Zhu, Yiming Hu, TAP: A novel tunneling approach for anonymity in structured P2P systems, in: ICPP'04: International Conference on Parallel Processing, Montreal, Canada, August 2004, pp. 21–28.

[12] Vincent Scarlata, Brian Neil Levine, Clay Shields, Responder Anonymity and Anonymous Peer-to-Peer File Sharing, in: ICNP'01: The 9th International Conference on Network Protocols, Riverside, CA, Nov. 2001, pp. 272–280.

[13] Rob Sherwood, Bobby Bhattacharjee, Aravind Srinivasan, P^5: A Protocol for Scalable Anonymous Communication, in: Proceedings of the 2002 IEEE Symposium on Security and Privacy, Berkeley, CA, May 2002, pp. 58–70.

[14] Clay Shields, Brian Neil Levine, A protocol for anonymous communication over the internet, in: Proceedings of the 7th ACM Conference on Computer and Communications security, Athens, Greece, 2000, pp. 33–42.

[15] Kenneth Birman, Mark Hayden, Oznur Ozkasap, Zhen Xiao, Mihai Budiu, Yaron Minsky, Bimodal multicast, ACM Transactions Computer Systems 17 (2) (1999) 41–88.

[16] I. Gupta, K. Birman, R.V. Renesse, Fighting fire with fire: using randomized gossip to combat stochastic scalability limits, Journal on Quality and Reliability Engineering International 29 (8) (2002) 165–184.

[17] Werner Vogels, Robbert van Renesse, Ken Birman, The power of epidemics: Robust communication for large-scale distributed systems, ACM SIGCOMM Computer Communication Review 33 (1) (2003) 131–135.

[18] Matei Ripeanu, Ian T. Foster, Mapping the gnutella network: macroscopic properties of large-scale peer-to-peer systems, in: IPTPS'01: Proceedings of the 1st International Workshop on Peer-to-Peer Systems, Cambridge, MA, March 2002, pp. 85–93.

[19] Stefan Saroiu, P. Krishna Gummadi, Steven D. Gribble, A measurement study of peer-to-peer file sharing systems, in: MMCN'02: Proceedings of the Multimedia Computing and Networking 2002, San Jose, CA, January 2002.

[20] Neelesh Bansod, Ashish Malgi, Byung Kyu Choi, Jean Mayo, MuON: Epidemic based Mutual Anonymity, in: ICNP'05: Proceedings of the 13TH IEEE International Conference on Network Protocols (ICNP'05), Boston, MA, 2005, pp. 99–109.

[21] David Chaum, Untraceable electronic mail, return addresses, and digital pseudonyms, Communications of the ACM 24 (2) (1981) 84–90.

[22] Ceki Gülcü, Gene Tsudik, Mixing E-mail with Babel, in: NDSS'96: Proceedings of the Network and Distributed System Security Symposium, San Diego, CA, February 1996, pp. 2–16.

[23] M. Rennhard, B. Plattner, Introducing morphmix: peer-to-peer based anonymous internet usage with collusion detection, in: WPES'02: Proceedings of the 2002 ACM workshop on Privacy in the Electronic Society, New York, NY, USA, 2002, pp. 91–102.

[24] George Danezis, Roger Dingledine, Nick Mathewson, Mixminion: Design of a Type III Anonymous Remailer Proto-

col, in: SP'03: Proceedings of the 2003 IEEE Symposium on Security and Privacy, Berkeley, CA, 2003, pp. 2–15.
[25] Alan Demers, Dan Greene, Carl Hauser, Wes Irish, John Larson, Scott Shenker, Howard Sturgis, Dan Swinehart, Doug Terry, Epidemic algorithms for replicated database maintenance, in: PODC'87: Proceedings of the 6th annual ACM Symp. on Principles of Distributed Computing, Vancouver, British Columbia, Canada, August 1987, pp. 1–12.
[26] Marius Portmann, Aruna Seneviratne, Cost-effective broadcast for fully decentralized peer-to-peer networks, Journal of Computer Communications 26 (11) (2003) 1159–1167.
[27] Soontaree Tanaraksiritavorn, Shivakant Mishra, Evaluation of gossip to build scalable and reliable multicast protocols, Performance Evaluation 58 (2+3) (2004) 189–214.
[28] Ayalvadi Ganesh, Anne-Marie Kermarrec, Laurent Massoulié, Peer-to-peer membership management for gossip-based protocols, IEEE Trans. on Computers 52 (2) (2003) 139–149.
[29] Márk Jelasity, Rachid Guerraoui, Anne-Marie Kermarrec, Maarten van Steen, The peer sampling service: experimental evaluation of unstructured gossip-based implementations, in: Middleware 2004: Proceedings of the 5th ACM/IFIP/USENIX International Conference on Middleware, Oct. 2004, pp. 79–98.
[30] J. Sun, P. Weber, B. Choi, R. Kieckhafer, DIMPLE: dynamic membership protocol for epidemic protocol, Broadnets 2007, Raleigh, NC, 2007.
[31] Brent R. Waters, Edward W. Felten, Amit Sahai, Receiver Anonymity via Incomparable Public Keys, in: Proceedings of the 10th ACM Conference on Computer and Communication Security, Washington, D.C., 2003, pp. 112–121.
[32] Kenneth Birman, Building secure and reliable network applications, Manning Publications Co., 1996.
[33] Parameswaran Ramanathan, Kang G. Shin, Ricky W. Butler, Fault-tolerant clock synchronization in distributed systems, Computer 23 (10) (1990) 33–42.
[34] B. Simons, J.L. Welch, N. Lynch, An overview of clock synchronization, 1990, pp. 84–96.
[35] Paulo Veríssimo, Luís Rodrigues, Antonio Casimiro, Cesiumspray: a precise and accurate global time service for large-scale systems, Real-Time Systems 12 (3) (1997) 243–294.
[36] A. Allavena, A. Demers, J.E. Hopcroft, Correctness of a gossip based membership protocol, in: Proceedings of the 24th ACM Symposium on Principles of Distributed Computing (PODC'05), Las Vegas, NV, July 2005, pp. 292–301.
[37] S. Voulgaris, D. Gavidia, M. van Steen, Cyclon: inexpensive membership management for unstructured p2p overlays, Journal of Network and Systems Management 13 (2) (2005) 197–217.
[38] Sepandar D. Kamvar, Mario T. Schlosser, Hector Garcia-Molina, The eigentrust algorithm for reputation management in p2p networks, in: WWW'03: Proceedings of the 12th international conference on World Wide Web, New York, NY, USA, 2003, pp. 640–651.
[39] Sergio Marti, Hector Garcia-Molina, Identity crisis: Anonymity vs. reputation in p2p systems, in: P2P'03: Proceedings of the 3rd International Conference on Peer-to-Peer Computing, Washington, D.C., USA, 2003, pp. 134.
[40] Ranjita Bhagwan, Stefan Savage, Geoffrey M. Voelker, Understanding availability, in: IPTPS'03: Proceedings of the 2nd International Workshop on P2P Systems, February 2003, pp. 256–267.
[41] Subhabrata Sen, Jia Wang, Analyzing peer-to-peer traffic across large networks, IEEE/ACM Transactions Networking 12 (2) (2004) 219–232.
[42] Jinyang Li, Jeremy Stribling, Robert Morris, M. Frans Kaashoek, Thomer M. Gil, A performance vs. cost framework for evaluating DHT design tradeoffs under churn, in: Proceedings of the 24th Infocom, Miami, FL, March 2005.
[43] Sean Rhea, Dennis Geels, Timothy Roscoe, John Kubiatowicz, Handling churn in a DHT, in: USENIX'04: Proceedings of the 2004 USENIX Annual Technical Conference, Boston, Mass., June 2004, pp. 127–140.
[44] Yatin Chawathe, Sylvia Ratnasamy, Lee Breslau, Nick Lanham, Scott Shenker, Making gnutella-like P2P systems scalable, in: SIGCOMM'03: Proceedings of the 2003 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications, August 2003, pp. 407–418.
[45] David Liben-Nowell, Hari Balakrishnan, David Karger, Analysis of the evolution of peer-to-peer systems, in: PODC'02: Proceedings of the Twenty-first Annual Symposium on Principles of Distributed Computing, July 2002, pp. 233–242.
[46] Krishna P. Gummadi, Richard J. Dunn, Stefan Saroiu, Steven D. Gribble, Henry M. Levy, John Zahorjan, Measurement, modeling, and analysis of a peer-to-peer file-sharing workload, in: SOSP'03: Proceedings of the 19th ACM Symposium on Operating Systems Principles, Oct. 2003, pp. 314–329.
[47] D. Stutzbach, R. Rejaie, Towards a better understanding of churn in peer-to-peer networks, November, 2004.
[48] L. Massoulie, E.L. Merrer, A.-M. Kermarrec, A. Ganesh, Peer counting and sampling in overlay networks: random walk methods, in: International Conference on Principles Of Distributed Computing (PODC'06), Denver, CO, July 2006, pp. 123–132.
[49] Márk Jelasity, Alberto Montresor, Ozalp Babaoglu, A modular paradigm for building self-organizing peer-to-peer applications, in: Engineering Self-Organising Systems, July 2003, pp. 265–282.
[50] B. Huffaker, M. Fomenkov, K. Claffy, D. Moore, Macroscopic analyses of the infrastructure: measurement and visualization of internet connectivity and performance, in: PAM2001: Proceedings of the 2001 Workshop on Passive and Active Measurements, April 2001.
[51] Claudia Díaz, Stefaan Seys, Joris Claessens, Bart Preneel, Towards measuring anonymity, in: Roger Dingledine, Paul Syverson, (Eds.), Proceedings of the Privacy Enhancing Technologies Workshop (PET 2002), LNCS 2482, Springer-Verlag, April 2002.
[52] Oliver Berthold, Heinrich Langos, Dummy Traffic against Long Term Intersection Attacks, in: PET'02: Proceedings of the Privacy Enhancing Technologies Workshop, Apr. 2002, pp. 110–128.
[53] Daniel Figueiredo, Jonathan K. Shapiro, Donald F. Towsley, Incentives to promote availability in peer-to-peer anonymity systems, in: ICNP, 2005, pp. 110–121.

Neelesh Bansod received an MS in Computer Science from Michigan Technological University in 2005. His thesis addresses an epidemic protocol-based anonymity system. He is currently affiliated with Redback Networks in San Jose, CA.

Ashish Malgi received an MS in Computer Science from Michigan Technological University in 2005. His thesis addresses anonymity system-based DDoS (Distributed Denial of Service) avoidance. He is currently affiliated with Microsoft in Redmond, WA.

Byung Choi received PhD in Computer Science from Texas A&M University in 2002. He has been an Assistant Professor in the Department of Computer Science at Michigan Technological University since 2002. Recently he has investigated peer-to-peer systems and network security issues centered around anonymity. His research has been supported in part by the State of Michigan and Korea Electronics Technology Institute (KETI). Prior to his graduate studies, he was a team leader for LG Information and Communications (LGIC) R&D Center in Korea, where he worked on various switching system designs which have been widely used in Korea.

Jean Mayo is an Associate Professor at Michigan Technological University (MTU). She joined MTU upon receiving the PhD in Computer Science from the College of William and Mary in Virginia. Her research interests are in distributed systems and security. Her research programs have been supported by several NSF grants including a NSF-CAREER. Prior to her graduate studies, she was a mechanical engineer specializing in control system design and real time simulations.