Making Peer-to-Peer Anonymous Routing Resilient to Failures

Yingwu Zhu
Department of CSSE, Seattle University
Seattle, WA 98122. Email: *zhuy@seattleu.edu*

Yiming Hu
Department of ECECS, University of Cincinnati
Cincinnati, OH 45221. Email: *yhu@ececs.uc.edu*

Abstract

One hurdle to using peer-to-peer networks as anonymizing networks is churn. Node churn makes anonymous paths fragile and short-lived: failures of a relay node disrupt the path, resulting in message loss and communication failures. To make anonymous routing resilient to node failures, we take two steps: (1) we use a simple yet powerful idea based on message redundancy by erasure coding and path redundancy to mask node failures; (2) we base mix choices of a path on node lifetime prediction and choose stable nodes as relay nodes, thereby prolonging single path durability. We present an allocation of erasure-coded message segments among multiple paths that provides a guideline on how to maximize routing resilience upon different node availabilities in real-world systems. Via detailed simulations, we compare routing resilience of our approach and existing anonymity protocols, showing that our approach greatly improves routing resilience while consuming modest bandwidth.

1. Introduction

Many Internet applications (e.g., anonymous web-browsing, anonymous e-mail services) need anonymizing networks to provide anonymity for participants such that their identities cannot be revealed by third-party observers. Initiator anonymity hides the identity of the initiator from all other nodes including the responder. Responder anonymity means that the identity of the responder is hidden to all other nodes including the initiator. Mutual anonymity provides both initiator anonymity and responder anonymity.

Most existing anonymity protocols can be categorized into two groups: *mix-based* and *multicast-based*. Mix-based protocols provide anonymity using redirection: messages from the initiator are routed through a set of relay nodes (called *mixes*) till they reach the destination. Multicast-based protocols, in contrast, provide anonymity using multicast groups: initiators and responders join multicast groups and messages are multicasted to all group members; cover traffics are employed to gain initiator or responder anonymity.

The recent success of peer-to-peer (P2P) systems has brought about much interest in using them as anonymizing networks, due to the following reasons. First, the open set of peer nodes offers a potentially large anonymity set for participants. Second, picking distributed peer nodes as mixes to relay traffics not only sidesteps political background and local jurisdiction issues, but also addresses the scalability issue facing current static anonymizing networks which operate a small number of fixed mixes. Finally, communication patterns and heterogeneity of peer node's locations render P2P networks an appealing environment for hiding anonymous traffics.

1.1. Motivation

However, membership churn is a hurdle to making the P2P network an attractive environment for anonymization, manifesting itself mainly in the following two aspects. First, churn complicates anonymous path construction in mix-based protocols: the initiator could get frustrated by frequent path construction failures (e.g., our experiments found that path construction success rate for current mix-based protocols is about 2.6% at the churn rate as those observed in deployed P2P networks) due to failures of relay nodes and that he/she has to choose another set of relay nodes for another attempt in path construction at the risk of another failure. Worst yet, path construction usually involves expensive asymmetric encryption and decryption.

Second, churn makes anonymous paths fragile and short-lived: the failures of relay nodes on a path disrupt the path, resulting in message loss (including requests and responses) and bad user experiences. Moreover, short-lived paths cannot support long-standing communication sessions and other applications such as anonymous email systems in which the reply email may fail to route back to the sender due to path failures.

A strawman solution to fragile paths in mix-based protocols is to use broadcasting/multicasting. While message multicasting is able to mask node/link failures and improve path resilience, it incurs costly bandwidth consumption due to massive messages and the cover traffics used to hide anonymous messages.

Current solutions including TAP [18] and Cashmere [19] use a group of nodes as a mix to mask node failures in anonymous routing. The group members have to share some secrecy such as public keys. However, the secrecy can be easily abused by the group members, thus endangering anonymity.

1.2. Our Approach

Our approach to resilient anonymous routing stems from a simple idea: resilience can be achieved by redundancy and wise choices of relay nodes. We use message redundancy by erasure coding [9] and path redundancy to mask node failures, improving routing resilience while minimizing bandwidth usage; we attempt to prolong path *lifetime* or *durability* by picking stable nodes as mixes.

Erasure coding breaks a message M of length $|M|$ into n coded segments, each of length $\frac{|M|}{m}$, so that m coded segments suffice for reconstructing the original message M. The key aspect is that when using erasure coding with a replication factor of $r = \frac{n}{m}$, only $\frac{1}{r}$ of the coded segments are required to reconstruct the message M.

Erasure coding is widely used in file and storage systems to mask disk and node failures for data availability. To achieve both anonymity and routing resilience, the initiator splits a message M into n coded message segments using erasure coding and evenly distributes coded segments over k disjoint paths using layered encryption (i.e., Onion Routing [17]) to the destination. By using erasure coding and path redundancy, our approach can tolerate $k(1 - \frac{1}{r})$ path failures while consuming modest bandwidth. We can strike a balance between routing resilience and bandwidth cost by adjusting parameters k and r. Moreover, we use node liveness prediction to make mix choices, resulting in higher durability for each single path.

While simple, our approach needs to explore a number of issues, which make our contributions:

(1) *We present an evaluation framework to assess routing resilience of anonymity protocols.*

(2) *We investigate routing resilience of current mixed-based protocols in terms of path construction and path durability.*

(3) *We study allocation of erasure-coded message segments among multiple anonymous paths for different node availabilities. Allocations of erasure-coded message segments with different replication factors allow a tradeoff between bandwidth cost and routing resilience.*

(4) *Mix choices affect path resilience significantly. We compare random mix choice and biased mix choice that is based on the node session time distribution observed in deployed P2P networks, and show that biased mix choice greatly improves path resilience.*

The remainder of the paper is structured as follows. Section 2 presents overview of related work. We discuss system goals and attack model in Section 3 and system design in Section 4. Section 5 provides anonymity analysis of our anonymity protocols. Section 6 presents experimental setup and results. We conclude the paper in Section 7.

2. Related Work

Most anonymous routing protocols basically can be categorized into two groups: mix-based and multicast-based. Mix-based systems [1, 3] provide anonymity using the principle of Onion Routing [17]. Tor [5] is the second generation of Onion Routing. In contrast, multicast-based systems achieve anonymity using multicast groups. Many such systems exist including P^5 [15], Horders [16] and APFS [14].

As P2P networks are becoming an appealing platform for anonymizing networks, a number of P2P-based anonymous systems have been proposed. Such systems including Tarzan [6] and MorphMix [11] are essentially based on layered encryption and multi-hop routing. In contrast, Crowds [10] achieves anonymity using probabilistic random forwarding: upon receiving a message, a relay node forwards the message to the responder with probability p and sends it to another randomly-chosen relay with probability $1 - p$.

However, churn in P2P networks makes anonymous paths fragile and short-lived. To make path resilience to node failures, TAP [18] and Cashmere [19] decouple paths from fixed relay nodes and use a group of nodes in structured P2P overlays [12] to mask single relay node failures. The main limitation is that the group members share some secrecy such as public keys. In contrast, MuON [2] is a mutual anonymity system that uses epidemic-style data dissemination to handle network dynamics in unstructured P2P networks.

Most existing anonymous systems assume a public key infrastructure (PKI) such that each participating node has public-private key pairs. Recently, Katti et al. [8] proposed to use a combination of information slicing and source routing to achieve anonymous routing without a PKI.

3. Goals and Attack Model

Goals. Our approach uses an Internet-wide pool of nodes, numbered in thousands, to relay each other's traffics to gain anonymity. We aim to improve anonymous routing resilience while minimizing bandwidth consumption. In particular, the goals are to meet the following requirements:

Initiator anonymity: the identity of an initiator is hidden to all other nodes including the responder.

Routing resilience: anonymous routing are fault-tolerant to node/link failures and messages are delivered reliably.

Low latency: anonymous messages are delivered at low latency.

Low bandwidth cost: routing resilience does not incur costly bandwidth consumption.

In this paper, we mainly focus on the routing resilience issue for initiator anonymity. However, responder anonymity and mutual anonymity can be easily achieved by extending our design, i.e., using an additional level of redirection.

Attack model. We assume the attacker controls a fraction of nodes. These compromised nodes collude and share each other's information, attempting to break other legitimate users' anonymity. The attacker can observe some fraction of network traffics and there is zero latency for messages sent between compromised nodes.

4. System Design

The system relies on a PKI and assumes each node learns other nodes' public keys through some mechanism (e.g., out-of-band or piggybacking in messages). The system uses gossip protocols to manage node memberships by which each node learns information about other nodes and uses a subset of the learned nodes to construct anonymous routing paths. The system uses layered encryption and mutihop routing through a set of relay nodes to achieve initiator anonymity. To make anonymous routing resilient to node failures, we take two steps: (1) we use a simple yet powerful idea based on erasure coding and path redundancy to provide routing redundancy and mask path failures; (2) we bias mix choices on nodes that tend to stay longer in the system, thereby increasing path success probability.

When an initiator wishes to send a message M to a responder, the initiator uses erasure coding to split M into n coded segments, each of length $\frac{|M|}{m}$. The initiator then distributes the n coded segments over k disjoint paths each of which consists of a set of L relay nodes. Section 4.7 will discuss in details allocation of coded message segments among k disjoint paths to maximize the probability that at least m segments are successfully delivered to the responder. Note that the responder can reconstruct M upon m received segments, and some time later he/she may send back the coded response segments over the k paths. Replication can be thought of as a special case of erasure coding where $m = 1$. The major advantage of erasure coding over replication is bandwidth cost. We provide detailed comparisons between replication and erasure coding in Section 6.

4.1. Path Construction

We decouple forwarding paths from message segment payload. Let P_i $(1 \leq i \leq L)$ be a relay node of a forwarding path with length L and P_{L+1} be the responder D. The initiator I generates a forwarding path by

$$Path_i = \begin{cases} \perp(termination) & i = L+1 \\ < P_{i+1}, R_i, Path_{i+1} >_{PubKey_{P_i}} & 1 \leq i \leq L \end{cases}$$

To perform path construction, the initiator I generates a random stream ID sid, caches the tuple $[sid, P_1]$, and sends the tuple $[Path_1, sid]$ to the first relay node P_1. In general, upon receiving the message $[Path_i, sid_{i-1}]$ from the upstream node, the i-th relay node strips off the outer layer of $Path_i$ using its public key $Pubkey_{P_i}$, revealing the next hop P_{i+1} and symmetric key R_i. Provided that $Path_{i+1}$ is not $\perp$ then the i-th relay node generates a random stream ID sid_i, caches the tuple $[P_{i-1}, sid_{i-1}, P_{i+1}, sid_i, R_i]$, and sends $[Path_{i+1}, sid_i]$ to P_{i+1}. Otherwise, it caches $[P_{i-1}, sid_{i-1}, \perp]$, implying end of the forwarding path.

4.2. Message Sending and Receiving

Consider that the initiator I wishes to send a message M to the responder D after path construction. The initiator splits M into n segments M_i $(1 \leq i \leq n)$, each of length $\frac{|M|}{m}$, and sends these segments over k disjoint paths to D $(P_{L+1} = D)$. For ease of exposition, we denote the coded segments a single path is responsible for by M_p. The initiator generates the payload for each path:

$$PayLoad_i = \begin{cases} < MID, M_p >_{R_i}, < R_i >_{PubKey_{P_i}} & i = L+1 \\ < PayLoad_{i+1} >_{R_i} & 1 \leq i \leq L \end{cases}$$

The initiator finds the cached sid corresponding to the cached tuple $[sid, P_1]$ and sends the tuple $[sid, PayLoad_1]$ to the first relay node P_1. In general, upon receiving $[sid_{i-1}, PayLoad_i]$, the i-th relay node looks up its cache for the corresponding symmetric key R_i, stream ID sid_i and next hop P_{i+1}. Then, it strips off the outer layer of $PayLoad_i$ using R_i. Provided that P_{i+1} is not $\perp$ then the i-th relay node forwards $[sid_i, PayLoad_{i+1}]$ to P_{i+1}. Otherwise, the responder D waits for the coded segments transmitted over the other $k-1$ paths. Note that MID is a unique message ID generated by the initiator, allowing the responder to correlate coded segments received to the same message. Once it receives m segments , the responder reconstructs the message M. Some time later, the responder splits the response message, encrypts each coded segment using the received symmetric key, and sends the message segments back over the k paths. On each reverse path, the payload is encrypted by the cached symmetric key at each

m as a parameter can be included in the payload.

hop until reaching the initiator who strips the onion and reconstruct the response message. Note that we eliminate the need to perform asymmetric encryption/decryption on payload due to the symmetric keys.

We can perform path construction and message sending in the same time such that the message contains both path information and payload: each node on the path caches the path state and forwards the stripped payload to the next hop. This allows the initiator to form paths *on-demand* in case some paths in use have failed or will be failing soon, without message delays.

4.3. Path States

Caching path states consumes resources on the relay nodes. If the initiator wishes to tear down a path, it can ask each relay node to release the cached path state. However, node churn complicates path release operations: the failure of an upstream node makes it difficult for the initiator to remove states on the relay nodes which are downstream nodes of the failed node. The orphaned states on the relay nodes raise an issue of *resource depletion*. We therefore associate a TTL (time-to-live) with each cached path state during path construction. The path refreshing messages (the payload messages can serve the purpose of refreshing messages) keep the path alive.

4.4. Path Reuse

Caching path states on the relay nodes allow for path reuse. The initiator can reuse a path by multiplexing different streams intended for different responders. Given a path consists of relay nodes $P_1, \cdots, P_L$, the initiator generates a payload:

$$PayLoad_i = \begin{cases} < MID, M_p >_{R_i}, < R_i >_{PubKey_{P_i}} & i = L + 1 \\ < PayLoad_{i+1}, D >_{R_i} & i = L \\ < PayLoad_{i+1} >_{R_i} & 1 \leq i < L \end{cases}$$

Upon receiving the message $[sid_{L-1}, PayLoad_L]$, the last relay node P_L decrypts the payload using the cached R_i (corresponding to the cache entry $[P_{L-1}, sid_{L-1}, P_{L+1}, sid_L, R_L]$), revealing the next hop D which *overrides* the cached P_{L+1}. P_L generates a new stream id sid'_L, caches $[P_{L-1}, sid_{L-1}, D, sid'_L, R_L]$, and sends the tuple $[PayLoad_{L+1}, sid'_L]$ to D. D decrypts the message, caches the tuple $[P_L, sid'_L, \perp, R_{L+1}]$, and waits for the other coded segments from other paths. On the subsequent communications, P_L looks up its cache using combination of P_{L-1}, sid_{L-1} and D on the forwarding path and using combination of D and sid'_L on the reverse path.

4.5. Path Failure Detection, Prediction and Reconstruction

A path fails if one of its relay nodes fails. To detect failures, the initiator relies on end-to-end acknowledgments. The initiator can detect the point of failure by using timeout and retry mechanisms [6]. Moreover, our system can predict path/node failures by using node lifetime predictor that is presented in Section 4.9. Periodically, the initiator checks each relay node's lifetime predictor: if the calculated predictor for a node is less than certain threshold, we treat the node as a potential point of failure. After detecting or predicting a failure, we use the same mechanism discussed in Section 4.1 to construct a new path replacing the one that has failed or is likely to fail.

4.6. Cover Traffic

An initiator seeking anonymity and routing resilience needs to send erasure-coded message segments over k paths. However, sending messages over k paths allows a passive observer to mount a statistical attack and trace a communication. On the one hand, we argue that communication patterns and heterogeneity of peer node's locations in the P2P network make it hard for the attacker to mount such an attack. On the other hand, we resort to cover traffics: each node, at all times, generates *cover messages* and sends them over k paths to a randomly chosen destination. The k paths used for cover traffics consists of random nodes. Note that k is unnecessary system-wide parameter and each node may pick a value corresponding to its bandwidth constraints. With cover traffics, there is no discernible difference between *real* messages and cover messages. In general, only the source and destination of a communication can distinguish real messages and cover messages. They are treated with equal disdain at all other nodes.

4.7. Allocation of Erasure-coded Message Segments

We now formulate the problem of erasure-coded message segment allocation. Currently, we only consider a simple erasure coding allocation (**SimEra**): the initiator divides n coded message segments of length $\frac{|M|}{m}$ *equally* among k disjoint paths (for simplicity we assume that k is a multiple of replication factor $r = \frac{n}{m}$). The goal of SimEra is to maximize the probability that at least m coded segments can be successfully delivered in the presence of path failures. In other words, SimEra aims to maximize the probability that at least $\frac{k}{r}$ out of the k paths succeed in delivering the message segments. We denote this probability by $P(k)$.

We think of path failures as Bernoulli distribution: when a path fails due to node failures, all the messages sent on it are lost; when a path succeeds, all the messages sent on it are successfully delivered. Assume node availability in the system is p_a and node failures are independent. The number of relay nodes in a path is L. Then, the probability of path success $p = {p_a}^L$ (we assume the responder node is available for communication). Thus, for SimEra, the probability that at least $\frac{k}{r}$ out of k paths succeed is

$$P(k) = \sum_{i=\frac{k}{r}}^{k} p^i (1-p)^{k-i} \binom{k}{i}$$

We have three observations regarding the behavior of $P(k)$ as a function of k for different values of r and p (proof omitted):

OBSERVATION 1: *when $pr > \frac{4}{3}$, we have, $\forall k$, $P(k+1) > P(k)$. In this case, it is beneficial to split coded segments among as many paths as possible.*

OBSERVATION 2: *when $1 < pr \leq \frac{4}{3}$, we have, $\exists k_0$, such that $\forall k > k_0$, $P(k+1) > P(k)$. In this case, it is beneficial to split coded segments only if k is sufficiently large.*

OBSERVATION 3: *when $0 < pr \leq 1$, we have, $\forall k$, $P(k+1) < P(k)$. In this case, it is never beneficial to split coded segments among more than r paths (since we assume k is a multiple of r).*

In SimEra, k measures the degree of splitting. We should carefully choose k upon different node availabilities and replication factors, in order to maximize the probability of successful message delivery. Section 6 validates the observations using simulations. We also consider simple replication allocation (**SimRep**). SimRep sends one copy of the original message M over each of k paths. Section 6 provides comparison studies between SimEra and SimRep.

4.8. Node Membership Management

Each node seeking anonymity needs to maintain a node cache which keeps track of nodes such that he/she can pick cached nodes as mixes to construct anonymous paths. We use epidemic/gossip protocols [4] to exchange and update node membership changes in the system. Epidemic protocols provide low-cost, reliable and scalable data dissemination: each node periodically selects random nodes as *gossip target* and sends the gossip target a gossip message that contains node membership changes (e.g., node joining and leaving) that it has heard of. The gossip target will merge all received gossip messages and update the node cache. Studies have shown that time required to disseminate data to the entire system is $\log N$ where N is the network size. However, how to optimize epidemic protocols is out of the scope of the paper.

4.9. Mix Choices

Existing mix-based anonymity protocols do not consider node lifetime when making mix choices in path construction. Most protocols *randomly* select a set of nodes as mixes to construct a path. If nodes on a path are prone to fail or leave, the resulting path is fragile and short-lived. Intuitively, choosing nodes that stay longer in the system as mixes can improve path durability. We thus make efforts to prolong path lifetime by wisely choosing relay nodes. However, key to wise mix selection is prediction of node lifetime. In this section, we start by presenting node lifetime distribution. We then discuss a node liveness predictor and a biased mix selection algorithm that improves path durability.

Node Lifetime Distribution

A measurement study [13] of deployed P2P systems has shown that the distribution of node lifetimes is heavy-tailed: nodes that have been alive for a long time tend to stay an even longer time. Figure 1 compares the Gnutella node lifetime distribution measured by Saroiu et al. with a synthetic heavy-tailed Pareto distribution with shape parameter $\alpha = 0.83$ and scale parameter $\beta = 1560$ sec. The CDF of Gnutella node life distribution closely matches the Pareto distribution. We thus assume node lifetimes follow Pareto distribution in this paper.

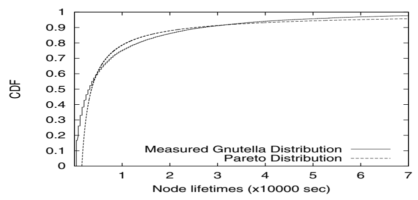

Figure 1. Cumulative distribution of the measured Gnutella node lifetime distribution compared with a Pareto distribution with α= 0.83 and β= 1560 sec.

Node Liveness Predictor

Given the Pareto distribution of node lifetimes, the probability of a node dying before time t is

$$P_r(lifetime < t) = 1 - (\frac{\beta}{t})^{\alpha}$$

where α and β are the shape parameter and scale parameter respectively.

Let Δt_{alive} be the time for which a node has been staying alive, measured from the node's last join time to the time when it was last heard. Let Δt_{since} be the time between the time when the node was last heard and now (current local time). The conditional probability of the node being alive, given that it has already been alive for Δt_{alive}, is

$$p = P_r(lifetime > \Delta t_{alive} + \Delta t_{since} | \\ lifetime > \Delta t_{alive}) \\ = \frac{(\frac{\beta}{\Delta t_{alive}+\Delta t_{since}})^{\alpha}}{(\frac{\beta}{\Delta t_{alive}})^{\alpha}} = (\frac{\Delta t_{alive}}{\Delta t_{alive} + \Delta t_{since}})^{\alpha} \quad (1)$$

As stated in Equation 1, calculating p requires three values: Δt_{alive} and Δt_{since} for a given node, and the shape parameter α of the Pareto distribution. By introducing a node liveness predictor q as

$$q = \frac{\Delta t_{alive}}{\Delta t_{alive} + \Delta t_{since}} \quad (2)$$

we have $p = q^{\alpha}$. p is a monotonically increasing function of q: big values of q mean high probability of being alive. Thus, biased mix choice will be based on q rather than p.

Learning Node Liveness Information

To calculate node lifetime predictor q, we need to know Δt_{alive} and Δt_{since}. Each node in the system keeps track of its Δt_{alive} based on its last join and includes Δt_{alive} in every packet it sends. As mentioned earlier, we use epidemic protocols to update and propagate node membership changes. By piggybacking node liveness information (i.e., Δt_{alive} and Δt_{since}) onto the gossip messages, a node learns other nodes' liveness information as follows:

- When the node hears from node A *directly*, it records the current local timestamp as t_{last} in the cache for A, updates the associated Δt_{alive} with the newly-received Δt_{alive}, and resets the associated Δt_{since} to 0.
- When the node hears information about node B *indirectly* from node A, it needs to check if B exists in its cache. If not, the node places B into its caches, recording the supplied Δt_{alive} and Δt_{since} and setting the associated t_{last} to the current local timestamp. Otherwise, the node compares the received Δt_{since} value with the currently stored value associated with B. A smaller received Δt_{since} indicates fresher information about B, and so the node records the received Δt_{alive} and Δt_{since} for B in the cache and sets the associated t_{last} to the current local timestamp.

Whenever the node needs to calculate node liveness predictor q for a node in its cache, q can be computed as

$$q = \frac{\Delta t_{alive}}{\Delta t_{alive} + \Delta t_{since} + (t_{now} - t_{last})} \quad (3)$$

where t_{now} is the current local timestamp. Whenever a node needs to piggyback node C's liveness information onto a gossip message, it calculates a current value for Δt_{since} by adding the saved Δt_{since} value and $(t_{now} - t_{last})$.

Biased Mix Choice

As discussed above, each node maintains a cache that reflects node membership changes and stores liveness information for each node in the cache. When an initiator wants to construct a path, it chooses a set of nodes with highest liveness predictor values from the cache. This mix choice is based on node liveness predictor and called *biased*. We compare the performance of biased and random mix choices in Section 6.

4.10. Summary

We stab the routing resilience issue of anonymity protocols from two angles. First, we use erasure coding and path redundancy to mask node failures while minimizing bandwidth cost incurred by redundancy. Our scheme using erasure coding can tolerate up to $k(1 - \frac{1}{r})$ path failures. Parameters k and r allow users to make a tradeoff between bandwidth cost and routing resilience. Second, we base our mix choices on node liveness prediction and pick nodes that tend to stay longer as mixes, resulting in more stable paths. We use layered encryption and multi-hop routing to gain initiator anonymity, similar to Onion Routing. We decouple forwarding paths from payload, thereby allowing path reuse. Cover traffic is utilized to protect anonymity. In addition, the three observations in SimEra give a guideline on how to maximize routing resilience upon different node availabilities by carefully choosing k and r, without incurring much bandwidth cost.

5. Security Analysis

Our approach to initiator anonymity essentially uses Onion Routing [17]. In spite of path and message redundancy, there is no information included in messages making it possible for the attacker (except the responder) to correlate coded message segments sent over different paths. Communication patterns and heterogeneity of peer node's locations in the P2P network, together with cover traffics, complicates the statistical attack the attacker could mount. We argue that our approach provides similar degree of initiator anonymity as in Onion Routing. Message confidentiality is achieved by symmetric encryption.

We analyze initiator anonymity using three parameters: N (number of nodes in the system), f (fraction of malicious nodes that collaborates each other), and L (number of relay nodes in a path). To simplify the analysis, we assume that

Freshness of cached information depends on intervals of epidemic protocols.

L is constant and known to the attacker. This assumption degrades anonymity, making the results lower bounds. Otherwise, variable Ls make it harder for the attacker to guess the initiators.

Consider a path that is initiated by a non-malicious node and on which the attacker occupies one or more positions. The goal of the attacker is to determine the node that initiates the path. Since messages are encrypted, the attacker has no reason to suspect any node other than the one immediately preceding it. All other non-malicious nodes are each *equally* likely to be the initiator, but are also obviously less likely to be the initiator than the attacker's immediate predecessor. We now analyze the probability that the immediate predecessor is in fact the initiator. We distinguish two cases:

Case 1: the first relay node is malicious. In this case, the malicious node can guess its immediate predecessor is the initiator with probability of 1. The probability of Case 1 occurring is, $P(Case1) = \sum_{i=1}^{L} \frac{i}{L} f^i (1-f)^{L-i}$.

Case 2: the first relay node is non-malicious. In this case, the attacker guesses the initiator with probability of $\frac{1}{N(1-f)}$. The probability of Case 2 occurring is, $P(Case2) = 1 - P(Case1) = 1 - \frac{1}{L}\sum_{i=1}^{L} i f^i (1-f)^{L-i}$.

Therefore, the probability a node x is the initiator I:

$$P(x = I) = \frac{1}{L}\sum_{i=1}^{L} i f^i (1-f)^{L-i} + \frac{1}{N(1-f)}(1 - \frac{1}{L})\sum_{i=1}^{L} i f^i (1-f)^{L-i} \quad (4)$$

6. Evaluation

6.1. Experimental Setup

Simulator. Our evaluation is based on p2psim. P2psim includes OneHop [7] which provides schemes to disseminate membership changes quickly and efficiently so that nodes maintain accurate and complete membership information in the presence of churn. The protocol to manage memberships in OneHop can be thought of as a *hierarchical gossip protocol* (among slice leaders, unit leaders and unit members). We augment OneHop by piggybacking node liveness information onto the gossip messages. We implement three anonymity protocols on top of the augmented OneHop: current mix-based protocols (**CurMix**), mix-based protocols using simple replication (**SimRep**), and mix-based protocols using erasure coding (**SimEra**). Mix choices in each protocol have two options: *random* and *biased* (on node liveness predictor). The number of relay nodes in a path is $L = 3$ by default, unless otherwise noted.

Simulated network. The simulated network consists of 1024 nodes with inter-node latencies derived from measuring the pairwise latencies of 1024 DNS servers on the Internet using King method. The average round-trip time for the simulated network is 152ms. Unless otherwise specified, our experimental results presented in the paper are based on this simulated network. To simulate network churn, each node alternately leaves and rejoins the network. The interval between successive events for each node follows a Pareto distribution with median time of 1 hour (i.e., $\alpha = 1$ and $\beta = 1800$ sec), unless otherwise specified. The interval is called node's lifetime or session time.

Evaluation framework. We use four metrics to evaluate performance and cost of anonymity protocols: (1) *Latency*: it measures successful routing latency. A successful routing means that the responder can either receive a copy of the original message (e.g., CurMix, SimRep) or reconstruct the original message (e.g., SimEra) sent by the initiator. (2) *Bandwidth cost*: it measures the total bandwidth cost of successful routing by which the responder successfully receives a message anonymously sent by the responder. The message size $|M|$ by default is 1KB, unless otherwise noted. (3) *Path setup success rate*: it measures the success rate of path construction. CurMix measures only single path while SimRep and SimEra measures k disjoint paths as messages or coded message segments will be sent over the k paths. For SimRep, path construction success comes when at least 1 out of k paths is successfully formed. For SimEra, path construction success also depends on replication factor r, and it comes when $\frac{k}{r}$ out of k paths are successfully formed. (4) *Path durability*: it measures path lifetime and implies routing resilience degree. For CurMix, path lifetime is terminated if any node on the path has failed. For SimRep, path lifetime is terminated if all k paths have failed. For SimEra, path lifetime is terminated if more than $k(1 - \frac{1}{r})$ paths have failed.

6.2. Experimental Results

Validation of Three Observations in SimEra

In the first set of experiments, we validated the three observations of SimEra presented in Section 4.7 using simulations upon various node availabilities of 0.70, 0.86 and 0.95. Figure 2 plots the results for three node availabilities where $r = 2$ and $L = 3$. The x-axis represents the number of paths used while the y-axis denotes the probability of successfully reconstructing the original message by the responder. The simulation results confirm the three observations. For example, the curve for Observation 2 has an initial dip but it increases when $k \geq 4$. Moreover, higher node availability has higher probability of success.

Figure 3 shows results for node availability $p_a = 0.70$ with different replication factors of 2, 3, and 4. We observed similar behaviors on other node availabilities. Note that a relatively bigger replication factor r dramatically in-

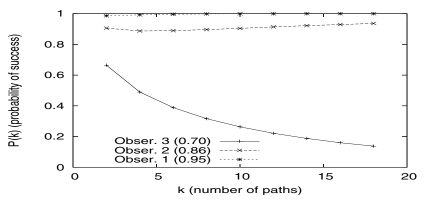

Figure 2. Validation of three observations. $r = 2$ and $L = 3$.

creases the probability of success. The side-effect of a bigger r is increased bandwidth cost. Figure 4 shows the total bandwidth cost when the initiator sends a message of 1KB over k paths and the responder successfully reconstructs the message.

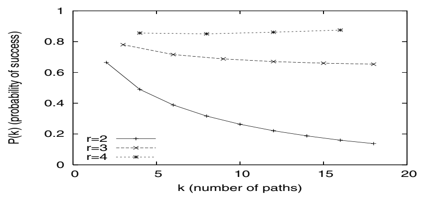

Figure 3. Results for varying replication factor r, where $p_a = 0.70$ and $L = 3$.

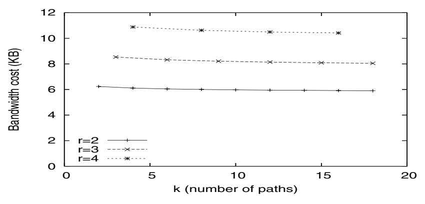

Figure 4. Bandwidth cost for varying replication factor, where $p_a = 0.70$ and $L = 3$.

Path Construction

In this set of experiments, we simulated node churn using the Pareto distribution with median time of 1 hour, as described in Section 6.1. For each experiment, the total simulation time was 2 hours. After the first hour of simulation, each node scheduled path construction events using exponential distribution with an average inter-arrival time of 116 seconds. The total path construction events for each anonymity protocol were about $16,000$. Table 1 shows path setup success rates for three anonymity protocols. The replication factor for SimRep and SimEra is 2. The first row represents the results for random mix choice while the second represents the results for biased mix choice. Two main observations can be made from the table: (1) using redundancy by replication or erasure coding improves path setup success rate, by about 1.9 times. (2) Biased mix choice on node liveness prediction significantly increases path setup success rate.

Figure 5 shows path setup success rates for varying k and r in SimEra. With either random or biased mix choice, a higher replication factor results in higher path construction success rate. For random mix choice, as shown in Figure 5(a), the number of paths k impacts the path setup success rate significantly: as k increases, the path setup success rate decreases. For biased mix choice, as shown in Figure 5(b), surprisingly, k does not have much impact on the success rate. This is mainly because biased mix choice makes the top $\frac{k}{r}$ paths very stable by selecting relay nodes that are likely to stay long in the system.

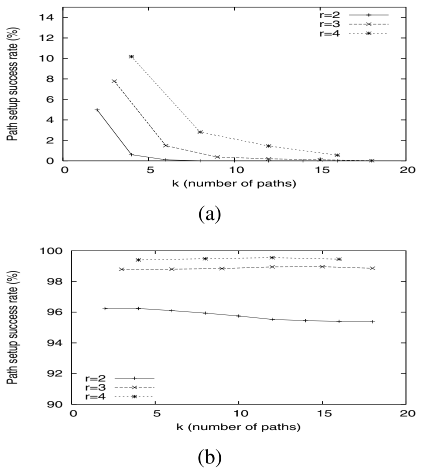

Figure 5. Path setup success rates for SimEra with varying k and r. (a) Random mix choice. (b) Biased mix choice.

Performance Comparison

In this set of experiments, we simulated node churn using the Pareto distribution with mean time of 1 hour. Two nodes kept staying in the system without leaving throughout the simulation: one node acts as an initiator and the other node acts as a responder. The simulation time was 2 hours. After the first hour of simulation, the initiator attempted to construct paths and then periodically sent a message of 1KB to the responder through the paths at an interval of 10 seconds. The continuing successful delivery of messages measures path durability and indicates routing resilience to node failures. The path durability is capped by 1 hour (3600 seconds). For each protocol (e.g., CurMix, SimRep, SimEra), we ran simulations 10 times with different seeds. Table 2 shows the results averaged over the 10 simulations. For each pair of values, the first value represents the result for a protocol with random mix choice while the second represents the result with biased mix choice. We omit the results for SimEra($k = 2$,$r = 2$) since its results are same as SimRep($r = 2$). Several important observations can be made from this table: (1) message and path redundancy greatly improves path durability and routing resilience, though at the cost of increased bandwidth. (2) Bi-

Table 1. Path setup success rates for three anonymity protocols.

Mix choice	CurMix	SimRep($r = 2$)	SimEra($k = 2$,$r = 2$)
random	2.64%	4.98%	4.98%
biased	80.62%	96.26%	96.24%

Table 2. Performance comparison among three anonymity protocols.

Protocols	Durability(sec)	Path construction attempts	Latency(ms)	Bandwidth(KB)
CurMix	$[700, 1153]$	$[8.4, 1]$	$[374, 266]$	$[4, 4]$
SimRep($r = 2$)	$[1140, 1167]$	$[2.8, 1]$	$[270, 257]$	$[6.2, 6.8]$
SimEra($k = 4$,$r = 4$)	$[1377, 2472]$	$[2.4, 1]$	$[406, 231]$	$[8.8, 10.4]$

ased mix choice has higher routing resilience than random mix choice. (3) For all protocols, biased mix choice significantly reduces number of attempts in path construction. As mentioned earlier, path construction generally involves expensive asymmetric encryption/decryption. Thus, it is beneficial to make biased mix choice in the three protocols.

Effect of Churn
One question remaining is how SimEra behaves under different levels of churn. Table 3 shows the results for SimEra($k = 4$,$r = 4$) with varying median node lifetime. Lower median node lifetimes correspond to higher churn. We ran simulations with median node lifetime ranging from 1200 seconds to 7200 seconds. Again, the first value in the pair represents the result for random mix choice while the second represents the result for biased mix choice. Several observations can be made from this table: (1) lower churn rates increases path durability and routing resilience. (2) Lower churn rates reduce number of attempts in path construction, especially when SimEra uses random mix choice. (3) Biased mix choice enhances routing latencies, because biased mix choice increases path stability, making more paths able to successfully deliver the message segments. In this case, one single path routing success allows the responder to reconstruct the message; if we have two paths or more able to deliver the message segments successfully, the responder can reconstruct the message upon the fastest path delivery, thereby improving the latency. However, biased mix choice consumes more bandwidth since more paths deliver the message segments. (4) Biased mix choice reduces number of attempts in path construction.

Impact of Node Lifetime Distribution
SimEra assumes a Pareto distribution of node lifetimes when making biased mix choice. In such a distribution, nodes that have been alive a long time tend to remain alive. Biased mix choice exploits this property by preferring to pick long-lived nodes as relay nodes in paths. However, if the distribution of node lifetimes is not what SimEra expects, biased mix choice may make mistakes about what nodes to select as relay nodes, thereby hurting performance. Table 4 shows the results for three different node lifetime distributions: a Pareto distribution (as discussed earlier with median lifetime of 1 hour), exponential distribution and uniform distribution. In the exponential distribution, node lifetimes are exponentially distributed with a mean of 1 hour. The probability of a node being alive does not depend on its join time. In the uniform distribution, node lifetimes are chosen uniformly at random between 6 minutes and nearly two hours, with an average of 1 hour; nodes that have been part of the network longer are more likely to fail soon. As shown in Table 4, path durability is higher with Pareto distribution than both uniform and exponential distributions. Surprisingly, in spite of the uniform or exponential distribution, biased mix choice still has higher performance than random mix choice in path durability, latency and path construction, at the cost of modest increased bandwidth cost. This means biased mix choice is viable in lifetime distribution other than Pareto distribution.

7. Conclusions and Future Work

To make P2P anonymous routing resilient, we exploit message redundancy by erasure coding and path redundancy to mask mix failures, and we base mix choice on node lifetime prediction to prolong single path durability. Via detailed simulations, we compare performance of our approach and existing mix-based protocols. We also fully investigate performance of our approach SimEra with random and biased mix choices. The results show that SimEra greatly improves routing resilience while consuming modest bandwidth. Moreover, we make three observations in SimEra upon different node availabilities and replication factors, and the three observations provide users a guideline on how to make a tradeoff between routing resilience and bandwidth cost in real-world systems.

Currently, SimEra assumes *even* allocation of coded message segments among multiple paths. In our next step, we plan to explore a *weighted* allocation scheme: more segments are allocated to the paths that are more likely to be stable. In biased mix choice, nodes that have been alive a long time are more likely to be chosen as relay nodes. So, the attacker may attempt to stay longer in the system with the hope of being relay nodes of many paths and breaking

Table 3. Performance of SimEra($k = 4, r = 4$) with varying median node lifetime.

Lifetime(minutes)	20	30	60	80	120
Durability(sec)	$[987, 1263]$	$[1101, 1889]$	$[1377, 2472]$	$[2448, 3014]$	$[2549, 3304]$
Path construction attempts	$[27.4, 1]$	$[10, 1]$	$[2.4, 1]$	$[1.4, 1]$	$[1, 1]$
Latency(ms)	$[270, 262]$	$[371, 182]$	$[406, 231]$	$[365, 274]$	$[288, 225]$
Bandwidth(KB)	$[7.4, 11]$	$[8.2, 12]$	$[8.8, 12.4]$	$[9.2, 12.6]$	$[10.4, 12.8]$

Table 3. Performance of SimEra($k = 4, r = 4$) with different node lifetime distributions.

Distribution	Pareto	Uniform	Exponential
Durability(sec)	$[1377, 2472]$	$[284, 1467]$	$[1271, 2256]$
Path construction attempts	$[2.4, 1]$	$[2.2, 1]$	$[3.4, 1]$
Latency(ms)	$[406, 231]$	$[370, 219]$	$[415, 256]$
Bandwidth(KB)	$[8.8, 12.4]$	$[8.4, 11.6]$	$[7.8, 11]$

other's anonymity. For example, if a malicious node receives a message from a node which just joined the system, the malicious node may guess that the newly joined node is a real initiator. However, this is not a big issue since we use cover traffic. Biased mix choice, instead, may provide an incentive for nodes seeking anonymity to stay longer in the system: a node that has been alive a long time reduces the risk of being identified as an initiator by the attacker.

Acknowledgment. This research was supported in part by the NSF under grant CCF-0541103.

References

[1] Anonymous remailer. http://www.lcs.mit.edu/research/.

[2] N. Bansod, A. Malgi, B. K. Choi, and J. Mayo. MuON: Epidemic based mutual anonymity. In *Proceedings of 13th International Conference on Network Protocols (ICNP)*, Boston,MA.

[3] D. L. Chaum. Untraceable electronic mail, return addresses, and digital pseudonyms. *Communications of the ACM*, 24(2):422–426, Feb. 1981.

[4] 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 *Proceedings of the sixth annual ACM Symposium on Principles of distributed computing*, pages 1–12, New York, NY, USA, 1987. ACM Press.

[5] R. Dingledine, N. Mathewson, and P. Syverson. Tor: The second-generation onion router. In *Proceedings of the 13th USENIX Security Symposium*, San Diego, CA, Aug. 2004.

[6] M. J. Freedman and R. Morris. Tarzan: A peer-to-peer anonymizing network layer. In *Proceedings of the 9th ACM CCS*, Washington, D.C., November 2002.

[7] A. Gupta, B. Liskov, and R. Rodrigues. Efficient routing for peer-to-peer overlays. In *Proceedings of first Symposium on Networked Systems Design and Implementation (NSDI)*, Mar. 2004.

[8] S. Katti, D. Katabi, and K. Puchala. Slicing the onion: Anonymous routing without PKI. In *Proceedings of the 4th ACM Workshop on Hot Topics in Networks (HotNets)*, College Park, MD, November 2005.

[9] M. O. Rabin. Efficient dispersal of information for security, load balancing, and fault tolerance. *Journal of the ACM*, 36(2):335–348, Apr. 1989.

[10] 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.

[11] M. Rennhard and B. Plattner. Introducing morphmix: Peer-to-peer based anonymous internet usage with collusion detection. In *Proceedings of the Workshop on Privacy in the Electronic Society*, Washington, DC, November 2002.

[12] A. Rowstron and P. Druschel. Pastry: Scalable, decentralized object location, and routing for large-scale peer-to-peer systems. In *Proceedings of the 18th IFIP/ACM International Conference on Distributed System Platforms (Middleware)*, pages 329–350, Heidelberg, Germany, Nov. 2001.

[13] S. Saroiu, K. P. Gummadi, and S. D. Gribble. Measuring and analyzing the characteristics of Napster and Gnutella hosts. *Multimedia Systems Journal*, 9(2):170–184, Aug. 2003.

[14] V. Scarlata, B. N. Levine, and C. Shields. Responder anonymity and anonymous peer-to-peer file sharing. In *Proceedings of IEEE International Conference on Network Protocols (ICNP 2001)*, Nov. 2001.

[15] R. Sherwood, B. Bhattacharjee, and A. Srinivasan. P5: A protocol for scalable anonymous communication. In *Proceedings of 2002 IEEE Symposium on Security and Privacy*, pages 58–70, May 2002.

[16] C. Shields and B. N. Levine. A protocol for anonymous communication over the internet. In *ACM Conference on Computer and Communications Security*, pages 33–42, Nov. 2000.

[17] P. F. Syverson, D. M. Goldschlag, and M. G. Reed. Anonymous connections and onion routing. In *IEEE Symposium on Security and Privacy*, pages 44–54, Oakland, California, May 1997.

[18] Y. Zhu and Y. Hu. TAP: A novel tunneling approach for anonymity in structured p2p systems. In *Proceedings of International Conference on Parallel Processing (ICPP)*, pages 21–28, Aug. 2004.

[19] L. Zhuang, F. Zhou, B. Y. Zhao, and A. Rowstron. Cashmere: Resilient anonymous routing. In *Proceedings of the 2nd Symposium on Networked Systems Design and Implementation (NSDI)*, Boston, MA, 2005.