Using Randomized Routing to Counter Routing Table Insertion Attack on Freenet

Todd Baumeister, Yingfei Dong
University of Hawaii
Email: {baumeist, yingfei}@hawaii.edu

Guanyu Tian, Zhenhai Duan
Florida State University
Email: {tian, duan}@cs.fsu.edu

***Abstract*—As privacy becomes an increasingly important issue on today's Internet, attacks on anonymous tools gain more attention. In this paper, we investigate countermeasures to a routing table insertion (RTI) attack on Freenet. We analyzed the key factors supporting the RTI attack, and proposed a simple randomized routing method to mitigate the RTI or similar attacks. We implemented the proposed method on the Freenet Thynix simulator. Our results show that the proposed method works well on small world topologies and thus a feasible choice to deal with the RTI attack on Freenet. We are further investigating more effective countermeasures based on these results.**

***Index Terms*—anonymity, anonymous communication, p2p**

I. Introduction

As we become more connected with our mobile devices and PCs, service providers or governments gain more and more private information, and can exploit it for their purposes with or without user permissions. Therefore, the diminishing online privacy is an urgent issue on today's Internet. For example, augmented identity [1] is able to profile individuals using a combination of cloud computing, facial recognition, social networking, and augmented reality. More importantly, freedom of speech is increasingly under threat due to constant and multidimensional surveillance.

Luckily, anonymous P2P (ANP2P) systems can help users hide their identities such that an adversary cannot easily determine the origin (or destination) of a message. Two main types of anonymous communication systems are: (1) *anonymous access systems* such as TOR [2] or I2P [3]. They provide virtual tunnel services between two end points on top of common network services. While application data are stored on end systems, these tunnels help users exchange them anonymously. (2) *anonymous content sharing systems*, such as Freenet [4] and GNUnet [5]. They store user data in the systems, and provide common operations on the data such as insert or query. We focus on the second type of systems.

As the main idea of an anonymous system is to hide a user in a large crowd of other users, its strength is usually dependent on the total number of users in the system. Therefore, a key problem for ANP2P systems is to attract more users, typically by providing fast responses and easy accesses. However, such performance improvements often require sharing more information among peers and thus degrade anonymity. As a result, the tradeoff between performance and anonymity is usually a challenging issue in design and implementation of such systems. We are focusing on this issue and investigating quantitative methods to evaluate the tradeoff. We have conducted extensive analysis of Freenet [4], one of the most popular ANP2P systems, and identified several common performance improvement mechanisms used in its routing schemes. We have then exploited them and developed the RTI attack [6] to insert attack nodes into victim nodes' routing tables such that many other attacks can be performed, e.g., traceback attack [7]. In this paper, we focus on how to mitigate the RTI attack and provide the following contributions:

- We have carefully analyzed the basic requirements of RTI attack on ANP2P systems, and identified the key attack parameters.
- We have proposed a randomized routing method that makes the RTI attack more difficult to conduct.
- We have implemented the proposed method on the Freenet Thynix simulator and evaluated its effectiveness in both anonymity and performance.

The remainder of this paper is organized as follows. We discuss related work in Section II. We present background information on the RTI attack, identify the key requirements of RTI attack, and propose a randomized routing as a countermeasure in Section III. We evaluate the proposed countermeasure in Section IV, and conclude this paper and discuss our future work in Section V.

II. Related Work

Anonymous communications have always been an interesting research topic. Since Mixnet [8], many anonymous access systems have been proposed. Among them, the most famous one is Tor [2]. However, in this paper, we look into another type of systems: anonymous content sharing systems. Freenet supports anonymous data storage and retrieval via distributed peers organized as a distributed hash table (DHT). Freenet nodes form a small-world [9] topology, and use a greedy routing algorithm to insert and find contents. It has an Opennet mode and a Darknet mode. In the Opennet, a node takes any nodes as its peers for its communications. In the Darknet, a node only connects to friend nodes known via out-of-band channels a priori, and hopes that its (trusted) friends help it hide its identify in communications. We focus on the Opennet mode in this paper because it is the most common case.

GNUnet [5] is a somewhat improved version of Freenet, and it also organizes peers into a DHT. The two systems use

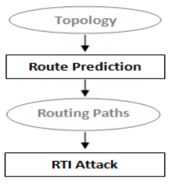

Fig. 1. Overview of RTI attack model.

different semi-structured topologies and routing algorithms. Freenet uses a small-world [9] topology, and GNUnet uses a Kademlia-like [10] topology. Freenet uses a greedy routing algorithm to insert and query contents. GNUnet [11] routes a message in two phases based on its hop-to-live (HTL) counter. It first randomly routes the message, and then routes it greedily towards its destination once the HTL reaches a threshold.

Most anonymous networks select randomized paths on a random or semi-structured topology to defeat traffic analysis, e.g., TOR, Crowds [12], or Tarzan [13]. However, random paths may seriously damage query performance, e.g., TOR limits its default path length to three hops. In this paper, we examine how to take advantage of randomized routing on Freenet to mitigate RTI or similar attacks.

III. RTI Attack and Randomized Routing as Countermeasure

A. RTI Attack

The RTI attack allows an adversary to use collaborating attack nodes to force a victim node to accept an attack node into its routing table, by exploiting peer replacement policies designed to introduce randomness into the network. Fig.1 shows the basic flow of the RTI attack model. First, we gather the network topology and peer relationships, and pass them to the route prediction procedure. It will then predict routing paths between a pair of nodes. We mostly only need to predict the routing paths for the subset of attack nodes in a network. Furthermore, we pass the predicted paths to the RTI attack module controlled by an adversary. Last, the RTI attack module inserts attack nodes into a target node's routing table. We will review the details of the RTI attack in the following. More details about RTI and the route prediction procedure are presented in [6].

Three basic components needed in the RTI attack are insertion node, query node, and intersection node. An insertion node is used to insert keys into the intersection node (along the insertion path). The query node is used to request the keys inserted in the intersection node (along the query path). Using these three components, an attacker can identify target (victim) nodes that are vulnerable to the RTI attack, and then insert an attack node into their routing tables. An attacker controls the insertion and query nodes, and uses the route prediction procedure to find the intersection and target nodes.

In Fig.2, A_i is the insertion node, A_q is the query node, I is the intersection node, and T is the target node. The basic attack steps are: (1) the attack node A_i inserts keys into node I via the insertion path without target node T on the path. (2) Another attack node A_q fetches the inserted keys with a query path from A_q to I with target node T on the path. Note that

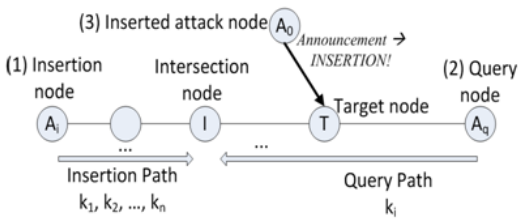

Fig. 2. Illustration of RTI attack.

the insertion path from A_i to node I is not overlapped with the query path from A_q to I, except at node I. The reason behind this requirement is that Freenet performs extensive caching to improve both performance and anonymity. When the insertion path and the query path are disjointed, no cached copies of the newly-inserted keys are on the query path. (Note that only attackers know the IDs of these keys.) As a result, caching does not affect this attack.

Furthermore, to insert an attack node A_0 (we found in practice that letting $A_0 = A_q$ is the easiest approach) into the routing table of target node T, we choose multiple files (e.g., 30) with routing keys that are mapped to intersection node I and insert them via insertion node A_i. Right after these insertions, we have newly-inserted files that are located at node I. We then request these files from query node A_q. After we have successfully retrieved the files for a number of times more than the given threshold (the current default is 10 in Freenet), the peer replacement policy is triggered at the target node T. Then we let attack node A_0 announce itself to the network with an identifier that is close to node T. Based on the peer replacement policy, some existing peers of T may be dropped and attack node A_0 is likely to be accepted as a peer by T. If this fails the first time, we simply repeat this attack until A_0 is accepted by the target node as a peer.

Moreover, we can maintain the attack node A_0 in T's routing table by frequently querying known files at A_0 from other attack nodes via node T. Due to the LRU peer replacement policy, A_0 is never the least recently used one to be replaced in T's routing table. In addition, once we have A_0 in T's routing table, it becomes much easier to insert other attack nodes into the table. We can then completely surround T with our attack nodes. Consequently, we can completely control messages in and out of target node T, determine if a request originated from T, or determine which file T currently holds, and break its anonymity.

To perform the RTI attack, an attacker must first identify the three key components. The insertion node and query node are in a set of nodes controlled by the attacker. The attacker can try different combinations of attack nodes as the insertion and query roles. Then it can use the set of attack nodes and their predicted routing paths to identify potential intersection nodes. With these three components, the attacker can then identify the potential target nodes.

We have verified that the RTI attack is effective using a test bed that we constructed with Freenet code. Details on the effectiveness of the RTI attack can be found in our previous work [6]. In addition, the RTI attack uses legitimate network behavior to manipulate a victim's routing table. This makes detecting the RTI attack difficult, since it is hard to separate it from normal inserts and queries.

The RTI attack can be used to enable many other attacks. For example, we have developed a traceback attack (see [7] for details) based on the RTI attack. The RTI attack allows attack nodes to be placed in key locations in the network. These nodes can then be used for surveillance, setting up for other attacks, or surrounding a victim node. Once an attacker surrounds a victim node with attack nodes, it has complete control over the messages in and out of the victim node and breaks its anonymity.

Network churns due to nodes joining, leaving, or swapping locations in the network affects the effectiveness of the RTI attack, because these events may change the topology and cause routing paths to change. Less accurate routing path knowledge may limit the success rate of the RTI attack. If a subset of the topology used in the attack is changed, we have to reexamine the predicted routing paths to see if the network change affected the attack insertion and query paths. An attacker can reduce the effect of churns affecting their RTI attacks by reducing the number of hops between the three key components: insertion, query, and intersection nodes. When the distances are smaller, the effect of churns are lesser. Reducing the distances also increases the accuracy of the RTI attack and increases its success rate.

Key RTI Attack Assumptions. Although the RTI attack is developed on Freenet, the basic idea can be applied to other P2P systems having the following characteristics.

- *Decentralized.* Each node operates distributedly. This is the common case for almost all P2P systems.
- *Routing Table Management.* On an ANP2P, a node usually has a limited number of peers, and follows a given policy to replace them. This is a common requirement for anonymous networks to avoid ”super” nodes that connect with many peers and easily collect too much information about peers. In the meantime, to make the network more dynamic, a node is often forced to replace some of its peers based on a trigger mechanism, e.g., periodically or based on certain events. In the case of Freenet, a least-used peer is marked as replaceable after a node sees 10 successful queries.
- *Node Announcement Protocol.* A node has to announce itself to other nodes in order to establish direct connections with them. The announcement protocol is usually deterministic or follows a predictable path to make sure that a node can find a sufficient set of peers for connectivity and performance. Normally, an announcement is first sent through a directory service or a seed node in a node's bootstrap process. Once a node is up and running, it may also announce itself to other nodes to improve connectivity and routing efficiency based on certain performance metrics. In this second phase, a node usually directly contacts other peers without the help of seed nodes. Many P2P systems use such announcement procedures.
- *Recursive Routing.* For anonymous P2P routing, recursive routing must be used to avoid a pair of source and destination being easily identified. (TOR is not a pure P2P system because it uses source routing based on a central directory service that is not available in common P2P systems.) In addition, the routing algorithm usually uses a simple set of rules for efficiency to deal with generic cases. As a result, it is highly predictable when the topology and peer relationships are known. Most ANP2P systems use greedy forwarding or similar schemes, because it is arguably the most practical solution on large-scale distributed systems without central control. Such semi-deterministic predictable routes are exploited by the RTI attack. Unpredictable routes on a large-scale P2P system generally mean poor routing efficiency, which usually causes unacceptable performance.
- *Predicted Routing Path Knowledge.* For the RTI attack, an attacker must be able to predict possible routing paths in the network. This often requires a method for gathering topology information. However, an attacker does not require the entire topology of a network. They can still perform the RTI attack with only a subset of the topology, assuming the subset is fairly accurate.

B. *Randomized Routing to Counter RTI Attack*

To mitigate the RTI attack, we need to break one or more of its basic assumptions listed in the above. After carefully analyzing these requirements, we found that some of them are basic P2P properties and have very little room to explore, because information sharing among peers is usually minimized for anonymity. For example, we must limit the number of peers for a node to prevent a broad exposure to a single attack node; we must replace peers to deal with churns, and besides LRU, there are only a few choices to pick a peer to replace (e.g., FIFO or random replacements may hurt the performance seriously); we must use recursive routing for anonymity. Therefore, reducing an attacker's ability to predict routing paths is arguably the most feasible way to counter the RTI attack, as long as the routing performance is not serious damaged. In the following, we will investigate the impacts of adding randomness to the greedy routing algorithm, and evaluate both the performance and anonymity effects of this approach.

Randomized routing reduces an attacker's ability to predict routing paths. If the attacker cannot accurately predict the routing paths, they will not be able to perform the RTI attack as illustrated in Fig.2. There are different ways to add randomness into a routing algorithm, e.g., GNUnet's R^5N routing algorithm adds randomness to the initial portion of a path. It splits a path of a message in two phases based on the message's HTL counter (that is used to limit the life of the message in the network). In the first phase, a message is randomly routed for a fixed number of hops (relative to the estimated network diameter). In the second phase, a greedy routing is used. Since an attacker can simply bypass the first phase of R^5N in the RTI attack, we have to consider a more generic case: adding randomness at each node independently, instead of based on HTL counters. Each node randomly routes a message with a given probability that is not affected by the behavior of other nodes in the network, i.e., a message may be randomly routed at any node on its path. To illustrate the basic performance of this method, we assume that all nodes in the network have the same probability in our simulations.

Because it is extremely difficult to define a generic anonymity metric on P2P systems, we propose to use the expected number of randomized routing occurred on a message forwarding path to quantify the anonymity strength of the randomized routing. We can estimate this number based on the length l of forwarding path of a message and the probability p that a node may randomly route the message, as function $R(p, l) = p * l$. We use this simple estimation for comparison the basic effects, and more elaborated R can be defined based on the knowledge about the message path and the network churns.

IV. Simulation Evaluation

We present the performance and anonymity evaluation of the randomized routing in this section.

A. Our Simulator and Experiment Design

We used the Thynix [14] simulator to run our experiments. Thynix is a tool for simulating the behavior of probe routing and path folding in Freenet, provided by the Freenet Project. We revised it for our purpose and branched off the Thynix source to allow us to simulate randomized routing in Freenet.

When a file is inserted into Freenet, there is no guarantee it will be stored in the optimal node (that has the closest address to the file's routing key CHK, content hash key). Actually, Freenet inserts several copies of the file to increase the chance that a requester finds it near the optimal node. Thynix simplifies this content storage process and focuses on testing reachability between a pair of nodes in the network.

We designed our experiments to take several key network characteristics into account.

- *Node Count:* We set the total number of nodes in a given network as 1000, 3000, 5000, 8000, or 10000. Consider that the current Freenet has about 3000 to 4000 nodes simultaneously online on average (from Freenet statistics).
- *Node Degree:* Freenet chooses the average number of peers (i.e., node degree) as 8, 11, 15, 19, 26, 34, or 40, proportional to the node's available bandwidth.
- *Topology Type:* We consider two types of network topologies as in many P2P systems: random topology and small-world topology. In a random topology, nodes are randomly connected. In a small-world topology, a node has a higher probability connecting to the nodes close to it. As a result, a small-world topology has a large clustering coefficient. Freenet attempts to build a small-world topology, but it is not guaranteed that it can achieve this goal. It is possible that it may end up with a random topology or some variation between the two.
- *Randomized Routing:* To test the performance impacts of randomized routing algorithm, we let each node have an independent probability to randomly pick the next hop, instead of using the greedy choice. We varied such a probability as 0%, 5%, 10%, 15%, or 20%. Here, 0% means a node uses the greedy routing; 5% means that each node has a 5% probability to randomly route a message; and so on.
- *Look-ahead:* We also examine the performance impacts of using various look-ahead values with randomized routing. Freenet currently uses a look-ahead of two hops, i.e., every node knows about the addresses of all peers within two hops. So, we evaluate the performance with one-hop and two-hop look-ahead.
- *Degree Distribution:* We use two degree distributions: fixed and Poisson. In the fixed degree case, all nodes have the same degree (or as close as possible); in the Poisson degree case, the node degrees follow a Poisson distribution. In fact, we have found that the node degree distribution has very little effect on our experiments. For this reason, we do not report their effects in the following.

We randomly generate several topologies for each combination of the above network variables (node count, node degree, degree distribution, topology type). Results on these topologies help us filter out most outlier characteristics that a particular topology may have. For ease of comparison, we keep the topology unchanged through simulations. This helps us eliminate the impacts of network churns and gives us a basic picture of the effect of randomized routing. In our follow-up work, we will examine the effect of network churns as they always exist in P2P systems.

To evaluate routing schemes in the simulator, we perform the following steps: 1) Select two random nodes in the topology. 2) Route from the source to the destination. 3) Record the routing path length in hops. Repeat steps 1 through 3, until a sufficient number of samples have been taken. When the simulation is finished, we take the average length of all paths and its standard deviation. The shorter the average path length means the better routing performance.

B. Experiment Analysis

On Random Topologies. Fig. 3 and 4 show the performance of randomized routing on random topologies. Each series represents a given randomized routing probability. The X-axis is the average network path length between nodes on a random topology. It is a graph property that is not affected by routing. We use it as the X-axis because it allows us to combine the node count and node degree variables into a single value. This reduces the dimensions of our output data and shows the combined effect. A large average network path length means that a topology is more spread with a large diameter. The Y-axis is the average path length obtained by the randomized routing in each simulation.

We will first discuss the performance impacts of the look-ahead on routing performance. Fig. 3 shows the performance of randomized routing when one look-ahead is used, and Fig. 4

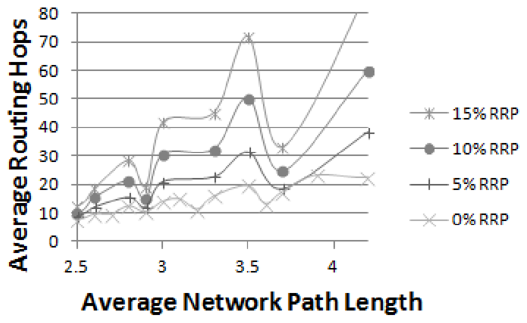

Fig. 3. Routing performance in random topologies with one look-ahead and various randomized routing probabilities (RRP).

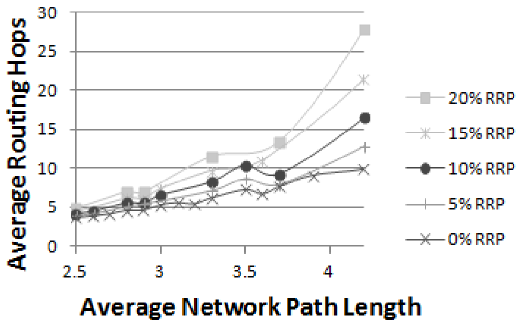

Fig. 4. Routing performance in random topologies with two look-ahead and various randomized routing probabilities (RRP).

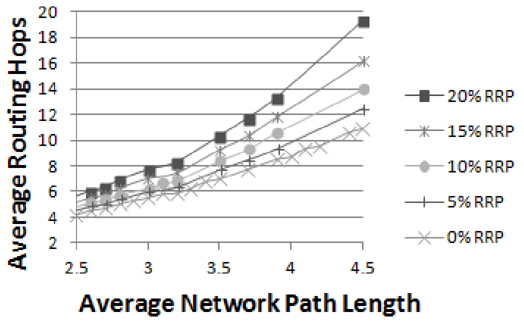

Fig. 5. Routing performance in small-world topologies with one look-ahead and various randomized routing probabilities (RRP).

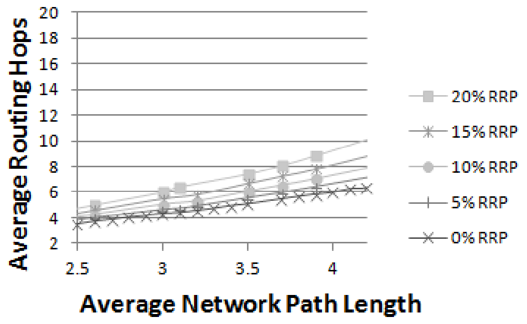

Fig. 6. Routing performance in small-world topologies with two look-ahead and various randomized routing probabilities (RRP).

shows that of two look-ahead. By default, Freenet is configured with two look-ahead; however, it is possible for nodes to be configured with only one look-ahead. As expected, two look-ahead greatly increases the routing performance of the base greedy routing case with a randomized probability 0% (note the Y-axis are not on the same scale). In addition, using two look-ahead reduces the performance degradation due to the randomized routing. With two look-ahead, the differences between the average path lengths of randomized routing and that of the base case are much smaller.

As the average network path length increases in Fig. 4, the average routing path length also increases as expected. Increases in the average network path length mean larger network diameters, which means that on average message routes are longer. This behavior is common across all simulations, and can be seen in all figures that compare the average path length of randomized routing over the average network path length.

As shown in Fig. 4, as a node's randomized routing probability increases, the average path length f randomized routing also increases. Randomized routes cause a message's forwarding to regress. When we look at the topologies that have an average network path length of 3.7 hops, the average path length of randomized routing is 7.6 hops with the greedy routing. However, when with the 20% randomized routing, the average path length almost doubles to 13.51 hops. Consider that the greedy routing is the base line. We calculate the expected times that randomized routing may occur using function R from the previous section, $R(.2, 13.51) = 2.702$ This means that with the 20% randomized routing, the path grows an additional 5.91 hops (¿2.7 in the above). This suggests that every time random routing occurs, it also adds several hops to the routing path.

Freenet currently uses a maximum HTL value of 18, and there is a probability that the HTL value can be extended a few hops more than 18 during routing. In Fig. 4, most of average path lengths of randomized routing are under the 18-hop ceiling (except for the cases of 15% and 20% randomized routing on topologies with an average network path length over 4). However, we need more than just the average path length to be less than 18 hops. We also want the range of path lengths within two standard deviation to be under the 18-hop ceiling. Two standard deviations account for 95.45% of the average path lengths of randomized routing. The standard deviations of the average path length with various randomized routing on random topologies and small-world topologies are shown in Fig. 7. When we include two standard deviations from the average path length of randomized routing, we find that a portion of the 10% randomized routing now is above the 18-hop ceiling.

On Small-world Topologies. Fig. 5 and 6 show the average routing performance of randomized routing on small-world topologies. We notice that the trends are similar as on random topologies as shown in Fig. 4. As the average network path length increases, so does the average path length of randomized routing. Adding more randomness to the routing algorithm also increases the average path length. However, we see much shorter average path lengths on small-world topologies.

When we compare the base line case between random and small-world topologies (Fig. 3 vs. Fig. 5 and Fig. 4 vs. Fig. 6), we see better performance on small-world topologies. The

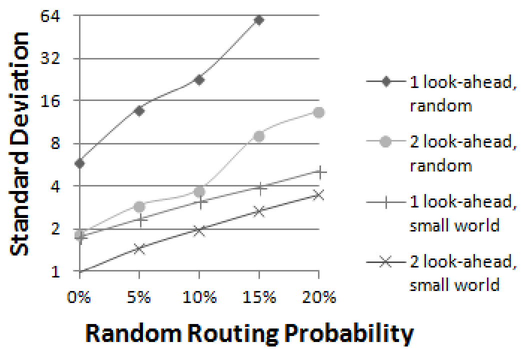

Fig. 7. Standard deviation in hops of routing path lengths.

number of extra hops added to paths due to randomized routing is decreased, compared with the average paths lengths on random topologies. When using two look-ahead on small-world topologies, the average path lengths in all cases (including their standard deviations) fall under the 18-hop ceiling.

Standard Deviation. Fig. 7 shows the standard deviations of the randomized routing experiments. The X-axis is the various randomized routing probabilities that we used. The Y-axis is the standard deviation in hop counts. As the randomized routing probability increases, the average routing length increase (see Fig. 3, Fig. 4, Fig. 5, Fig. 6), and their standard deviations also increases. We see not only better performance on small-world topologies than on random topologies but also smaller standard deviations.

Anonymity. We estimate the number of times randomized routing is triggered using function R, the randomized routing probability, and the average path lengths. Higher return values from function R indicate a more random routing in a path and, in turn, a less deterministic route. The concern is that there is no guarantee random routing will occur, or where it will occur along the path.

We found that small-world topologies can handle randomized routing much better than random topologies. A 20% randomized routing on a small-world topology with two look-ahead will still achieve an average path length within the 18 hops (including its standard deviation). For this case, we expect that a random routing occurs every five hops. On the random topologies with two look-ahead, the only case can achieve an average path length completely under the 18-hop ceiling was the 5% randomized routing. However, in this case, we expect a random routing occurs every 20 hops.

Randomized routing may help us break deterministic routing paths, but it comes with a heavy routing cost. If not carefully, adding small amounts of randomness may greatly decrease the routing performance. However, our simulation results shows that applying randomized routing on small-world topologies will not result in heavy performance degradation. We have examined one type of random routing in this paper. We will further investigate more advanced randomized schemes to explore basic ANP2P properties in order to maintain anonymity without heavy performance costs.

V. Conclusions and Current Work

We have identified the basic requirements of RTI-like attacks on P2P systems, and used randomized routing to mitigate RTI or similar attacks. Although such randomized routing greatly decreases the performance on random topologies, our simulation results show that small world topologies can handle this type of randomized routing much better. As a result, the proposed method is a feasible choice to mitigate RTI attacks on Freenet. We will further investigate more elaborated schemes to limit RTI or similar attacks on P2P systems, e.g., based on more information about a particular path or network churns.

Current Work. We are working on generalizing our findings. First, we limited the network churn in the current simulations to isolate the impact of randomized routing. Although this approach allows us clearly understand the impact of randomized routing, we will combine network churn with the randomized routing to show more practical effect. Second, our current results are based on simulations. We are working on revising our testbed to run actual Freenet code to further evaluate the proposed solutions, especially with content storage. Moreover, we are working on other randomized routing methods to limit performance impact while improving anonymity, e.g., instead of picking the greedy or a random choice at each hop, we can use the second or third choices with certain probability to semi-randomized paths while having minor performance lost.

References

[1] E. Jonietz, "Augmented Identity," *Technology Review*, 2010.
[2] R. Dingledine, N. Mathewson, and P. Syverson, "Tor: The second-generation onion router," DTIC Document, Tech. Rep., 2004.
[3] "I2P Anonymous Network." [Online]. Available: http://www.i2p2.de/
[4] I. Clarke, S. G. Miller, T. W. Hong, O. Sandberg, and B. Wiley, "Protecting free expression online with Freenet," *Internet Computing, IEEE*, vol. 6, no. 1, pp. 40–49, 2002.
[5] D. Kügler, "An analysis of gnunet and the implications for anonymous, censorship-resistant networks," in *Privacy Enhancing Technologies*. Springer, 2003, pp. 161–176.
[6] T. Baumeister, Y. Dong, Z. Duan, and G. Tian, "Routing Table Insertion (RTI) Attacks on Freenet," *ASE/IEEE International Conference on Cyber Security*, 2012.
[7] G. Tian, Z. Duan, T. Baumeister, and Y. Dong, "A Traceback Attack on Freenet," *IEEE INFOCOM*, 2013.
[8] D. L. Chaum, "Untraceable electronic mail, return addresses, and digital pseudonyms," *Communications of the ACM*, vol. 24, no. 2, pp. 84–90, 1981.
[9] O. Sandberg, "Distributed routing in small-world networks," in *2006 Proceedings of the 9th Workshop on (ALENEX)*, 2005, pp. 144–155.
[10] P. Maymounkov and D. Mazieres, "Kademlia: A peer-to-peer information system based on the xor metric," *Peer-to-Peer Systems*, pp. 53–65, 2002.
[11] N. S. Evans, *Methods for secure decentralized routing in open networks*. Network Architectures and Services, Dept. of Computer Science, 2011.
[12] M. K. Reiter and A. D. Rubin, "Crowds: Anonymity for web transactions," *ACM Transactions on Information and System Security (TISSEC)*, vol. 1, no. 1, pp. 66–92, 1998.
[13] M. J. Freedman and R. Morris, "Tarzan: A peer-to-peer anonymizing network layer," in *Proceedings of the 9th ACM conference on Computer and communications security*. ACM, 2002, pp. 193–206.
[14] Thynix, "Simulator," 2012. [Online]. Available: https://wiki.freenetproject.org/Simulator