MEASUREMENTS OF A LATENCY-BIASED EXPANDER TOPOLOGY IN THE TOR ANONYMITY SYSTEM

by

SUBHASISH DUTTA CHOWDHURI

Presented to the Faculty of the Graduate School of

The University of Texas at Arlington in Partial Fulfillment

of the Requirements

for the Degree of

MASTER OF SCIENCE IN COMPUTER SCIENCE

THE UNIVERSITY OF TEXAS AT ARLINGTON

August 2012

To my parents without whom I would not be where I am today and my brother Souvik for his unconditional support.

ACKNOWLEDGEMENTS

I would like to express my sincere gratitude to Dr. Matthew Wright for supervising me on this thesis and helping me in every step of my work. His patience, motivation and guidance helped me immensely during my research and the writing of this thesis. I am grateful to Dr. Donggang Liu and Dr. Manfred Huber for their invaluable advice and interest in my research and for taking time to serve on my thesis committee. I would like to thank all the members of the Information Security Lab for creating a positive and enjoyable work environment and being available at any time to discuss problems. I would also like to thank my friends Gaurav Hansda, Jaineel Mehta and Ankit Upadhyay for being there with me in my difficult times and supporting me throughout.

Finally, I thank my parents for their constant encouragement and unconditional support and my brother for his consistent motivation.

August 6, 2012

ABSTRACT

MEASUREMENTS OF A LATENCY-BIASED EXPANDER TOPOLOGY IN THE TOR ANONYMITY SYSTEM

Subhasish Dutta Chowdhuri, M.S.

The University of Texas at Arlington, 2012

Supervising Professor: Matthew Wright

Anonymous communication systems protect the privacy of their users by hiding who is communicating with whom. With the widespread use of the Internet, anonymity systems are all the more essential to support applications having strong privacy requirements such as intelligence gathering, military communications, or e-voting protocols. Anonymity systems must balance security and performance to remain popular with their users.

In this work, we perform measurements on anonymity systems to improve their performance. We use the Vivaldi network coordinate system to efficiently map out the relative delays between hosts. Using this data, we create an overlay expander network topology that is biased to use lower latency links instead of randomly selecting nodes. Our primary contribution is the design and execution of a set of experiments to evaluate the performance of this approach. These experiments are performed using a private deployment of Tor, a popular anonymity system, running on PlanetLab, a globally distributed testbed. Our testbed is comprised of 100 Tor relay nodes, five trusted directory servers and 10 geographically distributed clients, with each of the relays running a common implementation of Vivaldi to compute its virtual coordinates and reporting the same to a trusted directory server. The

directory server uses this information to construct an expander graph topology with a bias towards faster links. We show that when the network topology is created with a bias towards lower latency edges, there is a significant improvement in performance compared to using random links on our topology.

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

Figure Page

LIST OF TABLES

CHAPTER 1

INTRODUCTION

With the increased use of the internet in all aspects of daily life, the realization has dawned that privacy and confidentiality are essential and important requirements for the success and wide-spread use of many applications. Encryption alone does not provide the level of confidentiality required by users. It focuses primarily on protecting the confidentiality of the transmitted data, while the identities of the communicating parties remain unprotected. Traffic analysis can easily uncover information about the participants in a distributed application. User anonymity is the most important confidentiality criteria for many internet applications such as electronic voting, various forms of electronic commerce, peer-to-peer file sharing and anonymous web-browsing or email. Anonymity systems also benefit the government and law enforcement agencies enabling a medium of safe communication for whistle blowers and citizens willing to submit leads in criminal investigations as well as corporations seeking a means of doing competitive research and business deals.

Anonymity, however, has a few drawbacks. Abuse and illegal activity are the biggest drawbacks. Controlling illegal activity on the Internet is virtually impossible since anonymity ensures the identity of the perpetrator cannot be discovered or linked to specific actions. As a result, many organizations are dissuaded from fully embracing anonymity.

1.1 Anonymous Communication Systems

Anonymous communication systems were first proposed by Chaum [1] in which the message to be anonymized is relayed through a series of nodes called mix nodes. A mix can be thought of as a server which accepts incoming connections and forwards them in such a way that an eavesdropper cannot easily determine which outgoing connection corresponds to which incoming connection. Moreover, since any given mix can be compromised, traffic is usually

routed through a chain of mixes. Fig 1.1 shows an overview of an anonymous communication system.

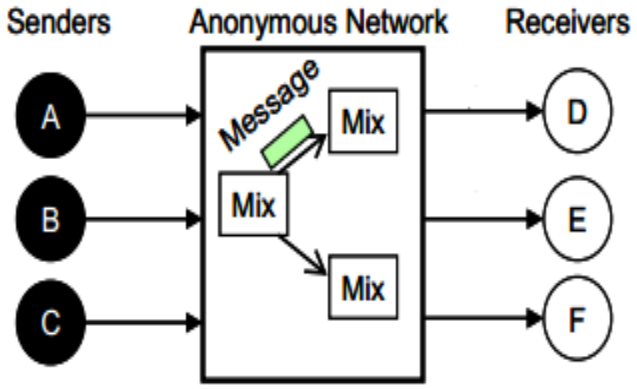


Figure 1.1 Anonymous Communication System

According to Pfitzmann and Waidner [14], there are three types of anonymities that can be provided by anonymous communication systems: sender anonymity, receiver anonymity, and unlinkability of sender and receiver. Sender anonymity means that the identity of the information sender is hidden, and receiver anonymity means that the identity of the information receiver is hidden. Unlinkability of sender and receiver refers to the property that the sender and receiver of a communication cannot be identified even if the sender and receiver are known to be of communicating with someone. Since anonymity is the state of lacking identity, anonymous communication can only be achieved by removing all the identifying characteristics from the network flows.

1.2 Types of Anonymous Communication Systems

Anonymous communication systems can be classified into two categories: systems catering to high-latency applications and systems for low-latency applications. High latency applications are those which do not require quick responses such as email systems. Low latency applications on the other hand are those that need real-time responses such as secure shell (SSH), web applications and instant messenger. Both types of systems are built on the idea proposed by Chaum [1] whereby unlinkability is provided by using a sequence of nodes

between the sender and its receiver and encryption is used to hide the message content. An intermediate node knows only its predecessor and its successor. An important difference between these two types of systems is that high-latency systems are message-based systems whereas low-latency ones are connection-based. Hence, high-latency systems have one message per path and new path is created for a new message whereas low-latency systems use a path for a period of time and send data as a stream of packets over the same path. High latency techniques attempt to hide timing information that could be used to perform traffic analysis. However, they are often impractical because of the fact that traffic must be delayed, reordered or manipulated in a manner to hide timing information. In contrast to high latency techniques, a number of approaches have been proposed to enable low latency anonymous communications that provide lower security guarantee but performance sufficient enough to support interactive web traffic.

A number of anonymous communication systems have been designed to provide anonymity to the communicating parties (e.g. Anonymizer.com [2], Web Mixes [3], Tor [4], Onion Routing [5], Crowds [6], Hordes [7]). We focus on low-latency systems whose main purpose is to protect the privacy of interactive internet communications such as web browsing. Tor is one of the most popular overlay networks for anonymizing TCP traffic. It is a low latency service that provides strong anonymity to its users.

1.3 Tor: The second generation onion routing design

Tor is the second generation of the onion routing design which provides low latency anonymity for TCP-based applications [4]. It is the most widely deployed anonymous communication system with an estimated 250000 users. The primary design goal of Tor is to ensure low enough latency to facilitate the use of interactive applications such as instant messaging and web browsing. Tor's system architecture consists of Tor routers, which are volunteer-operated servers, a set of trusted directory servers that advertise information about the Tor routers such as their IP addresses, public keys, exit policies, self-reported bandwidth

capacities etc. and Tor proxies (or clients). Tor clients query one of the authoritative directory servers to obtain a signed list of the available Tor routers and then establish paths, or virtual circuits, through the Tor network by choosing precisely three Tor routers and establishing a shared symmetric key with each, using authenticated Diffie-Hellman and a telescoping key agreement procedure. The client encrypts their data in fixed 512 byte cells in a layered fashion with each key and sends the encrypted cell to the first router on the circuit, called the entry guard. The entry guard removes one layer of encryption using the symmetric key shared with the client, revealing the IP address of the middle router. The cell is forwarded in this manner, removing one layer of encryption at each router until the final router in the circuit, called the exit router, removes the last layer of encryption, revealing the cell's destination. The exit router finally forwards the message to the destination. The entry guard only knows the client's identity and only the exit router knows the destination's identity. For efficiency, the Tor software uses the same circuit for connections that happen within the same ten minutes or so. Later requests are given a new circuit, to keep people from linking your earlier actions to the new ones. More details about Tor can be found in its design document [4].

Entry guards are Tor routers that are used as the first node in a client's circuit. To mitigate the threat from adversaries setting up Tor routers and profiling a large number of clients over time, only those routers are chosen as entry guards which have high uptime and high bandwidth. Clients choose a fixed number of entry guards (three by default) to use on their circuits. A router is marked as a Guard node by the authoritative directory servers only if its mean time between failures is above the median of all "familiar" routers (A router is "familiar" if one-eighth of all active routers have appeared more recently) than it and its bandwidth is greater than or equal to 250KB/s [17]. By default, clients choose precisely three entry guards to use for their circuits. To ensure that there is sufficient guard bandwidth available, guard node selection is weighted by $(G - (T/3))/G$, where G is the amount of available guard bandwidth and T is the total bandwidth available. If $G < T/3$, then guard nodes are not considered for non-guard

positions. Exit routers are the Tor routers that allow connections to leave the Tor network. Anonymous communication systems are sometimes used for abusive or malicious purposes [9]. Hence Tor allows router operators to exercise control over the types of traffic they wish to exit. Routers can be configured to exit to specific ports or they can be configured to connect only to other Tor routers (in which case the router may be only used as an entry guard or a middle router). The router's exit policy specifies the ports to which an exit router may connect to. In order to ensure that there is sufficient exit bandwidth available, the bandwidth of Exit routers is weighted differently depending on the fraction of bandwidth that is available from non-Exit routers. Suppose that the total exit bandwidth is E and the total bandwidth available is T. If $E < T/3$, then Exit routers are not considered for non-exit positions. Otherwise, their bandwidth is weighted by $(E - (T/3))/E$ [10].

Each router's entry guard status and exit policy are advertised by the trusted directory servers. Tor's router selection algorithm [10] chooses routers with the following constraints:

- A router may only be used once per circuit.
- Only one router per /16 network and two routers per IP address may be used on a circuit. This prevents an attacker who controls a single network from deploying a large number of routers in an attempt to attract traffic.
- The first router on the circuit must be marked as an entry guard by the authoritative directory servers. Clients select precisely three entry guards to use on their circuits, and choose new guards periodically.
- The exit router must allow connections to the client's chosen destination host and port.

Tor clients query one of the authoritative directory servers to obtain a signed list of the available Tor routers, their public keys, bandwidth advertisements, exit policies, uptime, and other flags indicating their entry guard status and other information. Routers for each position of the circuit are chosen in proportion to their self-advertised bandwidth. Fig. 1.2 provides an illustration of Tor's system architecture.

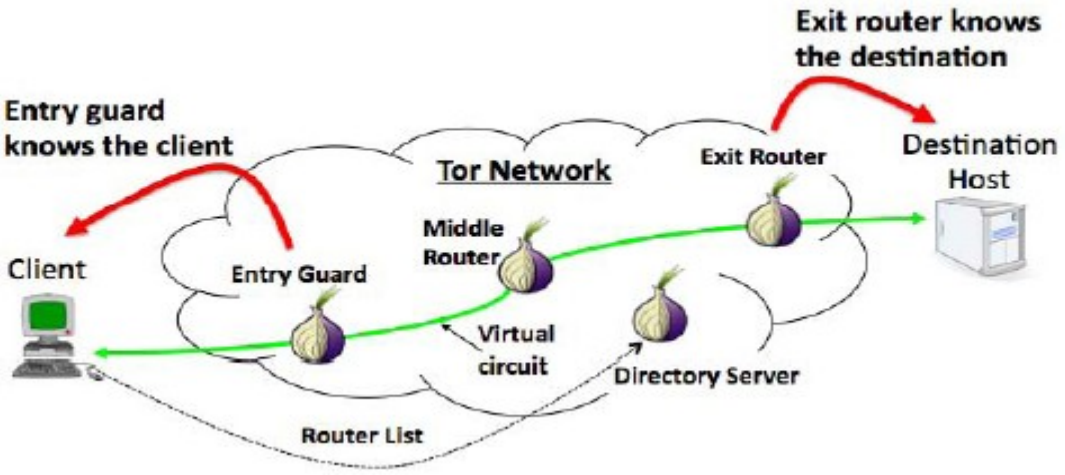


Figure 1.2 Tor's system architecture [8]

Persistent applications such as FTP and SSH that establish long-lived sessions require more stable circuits than applications with short-lived sessions like HTTP. For such long-lived applications, Tor builds circuits solely with routers that are marked as stable by the trusted directory servers. A router is stable if it has been observed by the directory servers for 30 days or if it is above the median of all routers in terms of mean time between failures [11]. Routers for each position of the circuit are chosen in proportion to their self-advertised bandwidth. Since Tor routers self-advertise their bandwidth capabilities, an adversary can falsely report high bandwidth values and increase the probability of attracting traffic and controlling the end points of circuits [12], [13]. To mitigate the effectiveness of this attack, all bandwidth advertisements are capped at 10 MB/s. More details about Tor's path selection algorithm may be found in the Tor path specification [10].

CHAPTER 2

BACKGROUND

Tor is the second generation of the onion routing design which provides low latency anonymity for TCP-based applications [4]. It is the most widely deployed anonymous communication system with an estimated 250000 users. The primary design goal of Tor is to ensure low enough latency to facilitate the use of interactive applications such as instant messaging and web browsing. Tor's system architecture consists of onion routers, which are volunteer-operated nodes, a set of trusted directory servers that advertise information about the Tor routers such as their IP addresses, public keys, exit policies, self-reported bandwidth capacities etc. and Tor proxies (or clients). Tor clients query one of the authoritative directory servers to obtain a signed list of the available Tor routers and then establish paths, or virtual circuits, through the Tor network by choosing precisely three onion routers and establishing a shared symmetric key with each. More details about Tor can be found in its design document [4].

Low latency anonymity systems strive to achieve a balance between security and performance. One of the frequent problems faced by Tor users is that it significantly slows down Web browsing speed. For an average user, the performance penalty introduced by Tor is very high for daily use. Efforts to improve the performance of Tor may decrease user's anonymity. For example, Tor does not add intentional delays or use cover traffic to ensure performance that is sufficient to support interactive applications such as Web browsing or instant messaging. However, this increases Tor's vulnerability to end-to-end traffic correlation. Another important design decision in Tor is the length of the circuits. Tor uses exactly three routers in a circuit to mitigate any single router's ability to link a source and destination. However, three-hop paths incur a performance cost over shorter circuits. The current Tor design tries to find a compromise

that satisfies both those users who desire strong anonymity and those for whom performance is more of a priority.

Another reason for the slow performance of Tor can be attributed to the volunteer-operated nature of the onion routers-many of them are on slow connections or shared with other activity [30]. Therefore, when selecting the nodes in the circuit, it is important to make the best use of the limited capacity available. The onion routers on each path are selected by the initiator to prevent an attacker from manipulating path selection. For best performance, the path selection algorithm must fairly distribute connections based on the capacities of the onion routers.

2.1 Improving the performance of Tor

When Tor was first released for public use, it was composed of only a few high-bandwidth routers and had few users. Hence, selecting routers uniformly at random did not have any impact on performance. However, as the number of Tor users keeps growing [31], random selection of routers led to the degradation in performance for its users.

There are a lot of factors that influence the performance of Tor. Tor's design decision to build paths with precisely three routers strikes a correct balance between security and performance [16]. However, compared to two-hop paths, three-hop paths incur a performance penalty. Bauer et al. [16] experimentally evaluate several key benefits and drawbacks of two-hop and three-hop paths. Though two-hop paths may improve performance, a disadvantage of a two-hop design from the security perspective is that exit routers can discover client's entry guards since they communicate directly. Two-hop paths are vulnerable to adaptive surveillance and introduce potential liabilities for exit node operators. Though shorter paths result in an improvement in performance, there is no strong argument to reduce Tor's path length.

The Tor users are heterogeneous in their requirements. Some users require high anonymity whereas other users may be less privacy-sensitive. The Tor router selection algorithm is a compromise between performance and anonymity. The bandwidth values used in

the Tor load-balancing algorithm are self-reported by each node and are not verified in any way. This can lead to attacks where malicious nodes can report a higher-than-actual bandwidth so that a larger fraction of tunnels are routed through them. To mitigate the effectiveness of this attack, all bandwidth advertisements are capped at 10 MB/s [10]. Despite the enforced upper bound on the reported bandwidth, the attack can be quite successful: Bauer et al. [13] report that a small fraction of attacker nodes can attain the first and last node positions on nearly half the tunnels. Even when the nodes are honest, the reported values can be a poor indicator of the available bandwidth at a node due to changing network conditions and other factors. This makes the performance of Tor highly variable. The current Tor load-balancing algorithm is a compromise between performance and anonymity. Snader and Borisov propose that end-users should have the ability to decide whether to weigh router selection towards higher bandwidth routers or to select routers uniformly at random, eliminating bias in router selection [15]. They propose to replace the self-reported bandwidth mechanism in Tor with a scheme for opportunistic bandwidth measurement. The topology of the Tor network allows each router to interact with most other routers and observe their performance over time. This mechanism accurately predicts the performance of the routers and is less susceptible to low-resource attacks in which low-resource nodes are perceived to be high-resource ones by reporting false resource claims to centralized directory authorities. A user-tunable mechanism for selecting routers based on their bandwidth capabilities is proposed which lets users select between anonymity and performance and make router selections accordingly.

While the current Tor path selection algorithm picks nodes with a probability proportional to their contribution to the total network bandwidth, the Snader Borisov (S-B) tunable variant only uses advertised node capacity to produce a rank ordering of nodes. The probability that a particular node will be selected depends solely on its position within this ordering. Let the family of functions f_s be defined as follows:

$$f_s(x) = \frac{1-2^{sx}}{1-2^{s}} \quad (\text{for } s \neq 0) \tag{1}$$

$$f_0(x) = x \tag{2}$$

To select each node, the *n* nodes are sorted based on the descending order of their bandwidth and a number *x* is selected uniformly at random from the interval [0, 1). The selected node is at index $\lfloor n \times f_s(x) \rfloor$. The value of *s* is selected by the client according to their preference for the type of performance they need. For *s* = 0, nodes are selected uniformly (for users requiring high anonymity) but as *s* increases, faster nodes will be preferred (for users willing to compromise anonymity for better performance). Experiments show that users can achieve great improvements in performance without much compromise in anonymity or significantly increase anonymity protection without any loss in performance. Moreover, this approach is dynamic: if a router's available bandwidth fluctuates over time, it will be noticed by its peers and used accordingly [15,28]. We use this tuning method in our proposal which is explained in detail in Chapter 3.

In [32], the authors describe a link-based path selection strategy that chooses in favor of higher performing links. Link-based relay selection supports more flexible routing, enabling anonymous paths with low latency, jitter, and loss, in addition to high bandwidth. In comparison to node-based techniques in which relay selection is biased by the relay node characteristics (i.e., bandwidth), link-based selection enables the generation of high performance paths across multiple metrics: latency, jitter, loss, and bandwidth. Another area of investigation is the improper application of TCP's congestion control mechanisms which degrade the performance of Tor. Traffic between any pair of routers is multiplexed over a single TCP connection. This results in interference across circuits during congestion control resulting in packet dropping or packet reordering.

In [33], the authors propose to use a TCP-over-DTLS (Datagram Transport Layer Security) between routers. Each stream of data has its own TCP connection with the TCP

headers being protected with DTLS which would otherwise give stream identification information to an attacker. Through experiments, they demonstrate that their proposal resolves the cross-circuit interference. In [34], the authors propose a fair resource allocation among circuits. Tor's circuit scheduling algorithm allows busy circuits (those with continuous traffic) to crowd out bursty circuits (those with short bursts of traffic). The authors in [34] propose to implement an advanced scheduling algorithm that treats circuits differently based on their recent activity. As a result, bursty circuits such as those used for web browsing can gain higher priority over busy ones such as used for bulk transfers. This improves the performance of Tor with minimal added overhead. In [35], the author's seek to improve Tor's performance by reducing unnecessary delays due to poor flow control and excessive queuing at intermediate routers. To improve flow control while reducing congestion, they implement N23, an ATM-style per-link algorithm that allows Tor routers to explicitly cap their queue lengths and signal congestion via back-pressure. Experimental results show that N23 offers better congestion and flow control, resulting in improved web page response times and faster page loads compared to Tor's current design and the other window-based approaches.

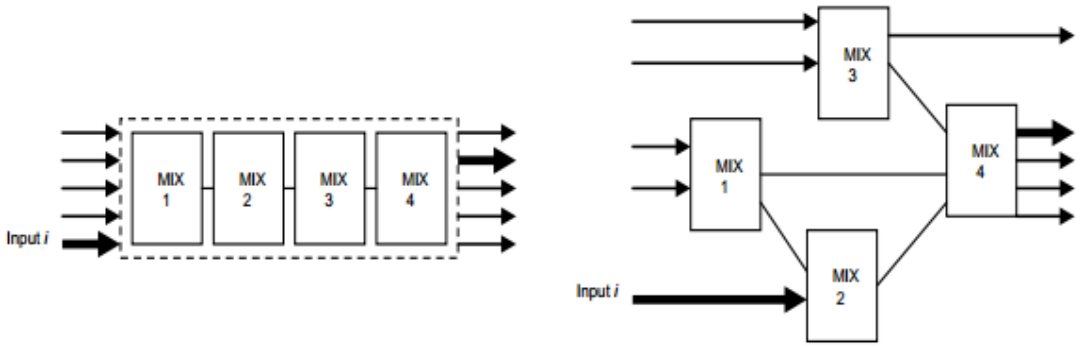
In addition to the above factors that have an impact on the performance of Tor, another important factor is the network topology. Basically, Tor forms an overlay network through which clients build circuits and forward traffic. Our works deals with the impact of the network topology on the performance and anonymity of low-latency anonymity systems.

2.2 Impact of Network Topology on Anonymity Networks

Mix networks were introduced by Chaum [1] as a technique to provide anonymous communication where messages are relayed through a sequence of intermediate nodes called mixes. A mix hides the relation between incoming and outgoing messages. This is done by collecting a number of messages and reordering them before sending them on their way. The topology of a mix network plays an important role in its efficiency and traffic analysis resistance properties.

The proposal by Chaum [1] assumes a fully connected graph, whereby all nodes can communicate with all other nodes, therefore forming a fully connected network. Clients relay their messages by choosing at random a sequence of nodes from the set of all existing mix nodes. The fully connected nature of the mix networks would seem to improve the anonymity provided by the system. However, they are vulnerable to intersection attacks, especially if many of the mixes in the network are compromised [23]. If an adversary controls all the mix nodes except one, an attacker can use intersection attacks to reduce the anonymity set of the sender and the anonymity of the messages going through the mix will most likely be compromised. Moreover, if two or more messages follow the same route, attacks become easier.

As a solution, a cascade of mixes was proposed [23]. Each cascade is a sequence of mixes with an equal number of mixes in every cascade. Users of the network cannot choose the route to take and must route their messages through a predefined sequence of mixes. Using a cascade of mixes allows only a fixed routing position for each mix for all the messages it processes. This prevents the partitioning of the input message batches and so prevents the intersection attack. However, mix cascades have some disadvantages. Cascades do not scale well to handle heavy loads and are vulnerable to denial of service attacks, since disabling one node in the cascade will stop the functioning of the whole system. Another disadvantage of a mix cascade is that the cascade consists of default mixes which have to be used. A user cannot express his trust in certain mixes by using them or his distrust by not using them. But a user may choose that cascade he wants to use and trust in.



(a) Cascades (b) Free-route

Figure 2.1 Mix Topologies [22]

Danezis [24] proposed a middle path between free-route mix networks and mix cascades that uses a restricted-route mix network based on expander graphs-graphs with low degree and short paths between any pair of nodes. An expander graph $G_{N,D}$ is a random graph that has a homogeneous topology, having N nodes each with a degree D. Compared to a complete graph topology, this topology has fewer edges and fewer possible paths. Thus, traffic is concentrated through fewer mixes instead of dispersing traffic through many mixes as in a complete graph topology. To choose a path in the network, the user does a random walk on the topology graph of the network. Additionally, restricted topologies offer better anonymity properties. Danezis [24] showed that a restricted network topology such as an expander graph improves security against intersection attacks. Intuitively, more cover traffic and messages flow over the same links, allowing them to provide cover for each other.

Our work also leverages expander topologies, but in the context of low-latency anonymity systems.

CHAPTER 3

MEASUREMENTS OF A LATENCY-BIASED EXPANDER TOPOLOGY

We propose the creation of an underlying network topology using network latency based on expander graphs. The topology is created with a bias towards low-latency edges. The latency between the nodes is measured using a decentralized network coordinate system that allows an efficient estimation of the latency between any pair of nodes.

3.1 Vivaldi: A Decentralized Network Coordinate System

Network coordinate systems assign virtual coordinates to every node in the network which allows the efficient estimation of latency between any pair of nodes. The communication costs are greatly reduced compared to directly measuring the $O(n^2)$ pairwise latencies because each of the nodes in the network computes its coordinates based on the round-trip time to a few other nodes. Coordinates are assigned to hosts such that the distance between their coordinates predicts the RTT between those hosts.

Vivaldi is a decentralized network coordinate system which has a low convergence time, low error and an accurate mapping of the virtual coordinate network. It calculates the coordinates as a solution to the spring relaxation problem. Its behavior is analogous to a physical model made of springs and balls, in which each ball represents a network node. The spring connecting any two balls is longer when the latency between those nodes is larger. Over a period of time, such a model reaches a stable state. The resting position of the spring equals the network latency between the pair of nodes. At the beginning, a Vivaldi node selects an arbitrary set of peers, and sets its initial coordinate to the origin. It then begins an iterative algorithm that pulls it closer to peers with lower latencies, and pushes it away from peers with higher latencies. After many iterations, the coordinate system reaches an equilibrium, and subsequent changes are due only to the changing latency between the nodes. On each

iteration, a Vivaldi node sends a probe packet to each of its peers. It receives a response to each probe packet containing the peer's current coordinate and self-reported error estimate, and learns its latency to that peer from the round trip time of the transaction. It then computes a new position that is closer to the peer if the estimated latency is too large, and farther from the peer if the estimated latency is too small. Vivaldi uses Euclidean coordinates of *d* dimensions with an additional height value: $x = x_1, \ldots\ldots, x_d, x_h$. To calculate the distance between two nodes *x* and *y*, the distance of their Euclidean coordinates is calculated first and then the heights of both nodes are added.

$$\textit{Vivaldi distance}(x,y) = \sqrt{\sum_{i=1}^{d} \left(x_i - y_i\right)^2} + x_h + y_h \qquad (1)$$

3.2 Expander Graph topologies and construction

Expander graphs are graphs which are sparse, show high connectivity and are well known to have excellent expansion properties. Intuitively, they are graphs for which any "small" set of vertices has a relatively "large" neighborhood. They are used to model social networks and relationships between species and organizations.

A graph G = (V,E) in which every vertex has exactly D neighbors is called D-regular graph. A D-regular graph is an (A,K)-expander if for every subset S⊆V of vertices in G, such that | S |≤ K, then | N(S) |> A | S |. Here | S | is the number of vertices in S and | N(S) | is the number of vertices in S′ that share an edge with any vertex in S. Although expander graphs contain a relatively small number of edges per vertex compared to a complete graph with the same number of vertices, they possess excellent connectivity properties enabling them to be extremely fast mixing. Mixing time of a graph is the time that it takes a random walk on the graph to approach the stationary distribution of that graph. In fast mixing graphs, a random walk will converge to a stationary distribution in a fewer number of hops. The property of having fewer numbers of edges per vertex in combination with the fast mixing property makes

expander graphs an attractive alternative to build the underlying topology of anonymity networks which could be leveraged to provide better security and performance.

The expander graphs are constructed using the distributed method described in [26]. The method begins with an initial set of three nodes that are connected to each other with D/2 hamiltonian cycles, where D is the vertex degree of the expander graph. The graph grows incrementally from the initial three nodes when new nodes join the existing nodes. When a new node joins the network, a random existing node is selected as the new node' neighbor. The process of random neighbor selection continues until the new node is connected to D other nodes in the network. An expander graph constructed in such a manner is a randomly-constructed expander.

We propose an expander topology that is biased toward low-latency edges which would simultaneously achieve both the security benefits of an expander topology and improved performance. If the topology is based on the latency between nodes, the resulting expander will have a greater number of high performance links as compared to a random expander. A path or circuit chosen through this shaped expander network has a higher probability of experiencing better performance than a path chosen through a random expander or a fully connected topology. The topology which we get is a shaped expander topology.

3.3 Shaping and Routing Bias

Snader and Borisov [15] propose a method to choose nodes to use in a circuit in a way so as to bias selection towards nodes that provide higher performance or higher security. This tuning method is used in our work to bias neighbor selection during the construction of the expander graph. We use two parameters SBIAS and RBIAS to bias the selection of neighbors during the creation of the expander topology. The parameter SBIAS is used to vary the shaping bias of the topology. As the value of SBIAS increases, the resulting graph that is generated is more highly shaped and contains a greater number of high performance links. The tuning function in [15] is also used to pick nodes in the anonymity network thereby introducing a

routing bias. RBIAS is the parameter for routing bias and is used to vary the amount of routing bias in the expander topology. We measure the performance of a shaped expander topology on a private deployment of Tor for varying values of SBIAS and RBIAS.

One area of concern is that the topology generated by biased edge selection is not theoretically an expander and may lack the expansion properties of a random expander. However, previous work has shown that biasing the expander construction does not lead to a topology that requires significantly more hops to reach maximum entropy [27]. Therefore, shaped expander graphs can be used to improve link performance without substantial loss of anonymity.

CHAPTER 4

EXPERIMENTS AND RESULTS

To demonstrate the efficacy of our proposed improvements, we design and execute a set of experiments. These experiments are performed using a private deployment of Tor running on PlanetLab [18]. PlanetLab is a geographically distributed platform for deploying, evaluating and accessing planetary-scale services and serves as a testbed for computer networking and distributed systems. As of June 2010, it was composed of 1090 nodes at 507 sites worldwide.

Our testbed is comprised of 100 Tor relay nodes, five trusted directory servers, and 10 geographically distributed clients, with each of the relays running Pyxida [19], a common implementation of Vivaldi, to compute its virtual network coordinates and reports the same to a trusted directory server. We use Pyxida in our experiments since it uses the Vivaldi algorithm to compute the coordinates of the nodes in the network, is a stable deployment of the network coordinate system and has been used in large scale deployments. The directory server uses the coordinate information to construct an expander graph topology with a bias towards faster links. Using this topology, Tor builds circuits for routing data from the source to the destination. We measure the performance of a normal deployment of Tor compared with one using a biased underlying network topology for different values of SBIAS and RBIAS.

4.1 Experimental Parameters and Evaluation Metrics

Our testbed is comprised of 100 Tor relay nodes, five trusted directory servers, and 10 geographically distributed clients. Since PlanetLab is a globally distributed testbed, we have chosen the Tor relays to be as geographically diverse as possible so as to obtain a realistic model of the actual Tor network. We deployed a standalone Pyxida service on the relay nodes with each of the nodes reporting its virtual coordinates to the directory server. We wait until the

network stabilizes before constructing the expander graph topology. The shaped expander graph constructed has a degree of 20. After the graph is constructed, the Tor client randomly chooses a guard node from its downloaded network consensus and builds its circuit in accordance with the expander topology.

We pick 10 PlanetLab nodes to use as Tor clients, in keeping with the distribution across countries of Tor clients [20]. The top 20 countries having the highest number of daily Tor users (direct and bridge) for June 2012 are shown in Table 4.1. We use this data to select 10 clients which would be used in all the experiments.

Table 4.1 Top-20 countries by average daily Tor users (direct and bridge) for June 2012 [20]

Country	Average daily Tor users (direct and bridge)	Percentage of users
USA	61193	13.2%
Iran	45835	9.9%
Italy	43791	9.4%
Germany	41362	8.9%
France	30885	6.6%
Spain	30286	6.5%
Syria	17976	3.8%
Brazil	14696	3.1%
Great Britain	11182	2.4%
Russia	11165	2.4%
Saudi Arabia	9998	2.1%
Poland	7128	1.5%
Israel	6988	1.5%
Canada	6952	1.5%
South Korea	6532	1.4%
Netherlands	6487	1.4%
Japan	5087	1.1%
Argentina	4059	0.87%
Australia	3966	0.85%
India	3963	0.85%

Since Tor is primarily used for low-latency communication such as web browsing, the evaluations that we do focus on the performance metrics that is particularly important to the

quality of service of an end-user from that perspective. We measure the time-to-first-byte (TTFB) which is the time a user must wait from the time they issue a request for data until they receive the first byte (duration from the web client making a HTTP request to the first byte of the page being received by the browser). The time-to-first-byte is two end-to-end circuit RTTs: one RTT to connect to the destination web server, and a second RTT to issue a request for data (e.g., HTTP GET) and receive the first byte of data in response. We also measure the download time which is how long a user must wait for a web page to load.

To evaluate the time-to-first-byte, we measure the latency between the 10 geographically distributed clients and the top 100 websites as reported by Alexa [21]. We also resolve the URLs of these websites in advance to exclude the DNS lookup time. We measured the latency between every client node and every website as the median latency of 5 requests. We first measured the latencies when Normal Tor was running on the PlanetLab nodes. Next, we repeat the same with communication happening over a Tor network using a shaped expander topology for different values of SBIAS and RBIAS.

Next, we evaluate the download time where a 1 MB file was fetched over HTTP via the Tor network. According to HTTP Archive [29], the average size of a web page as on July 2012 is 1097 KB. The list of URLs used by HTTP Archive to come to get this figure is based on the Alexa [21] top 1,000,000 sites. We measure the download time of 1 MB data which roughly corresponds to the size of an average web page from 10 geographically distributed clients. Each experiment concludes after the web client completes 100 downloads. The clients use the wget web browser. We first perform the experiments using a normal deployment of Tor running on the PlanetLab nodes. Next, we repeat the same with communication happening over a Tor network using an expander topology for different values of SBIAS and RBIAS.

4.2 Results

We first evaluate the time-to-first byte for accessing the top 100 websites as reported by Alexa [21] from 10 clients using a shaped expander topology. We compared the performance of

normal Tor with one using a shaped expander topology for SBIAS and RBIAS values of 0, 1, 8 and 15. Fig 4.1 shows the cumulative distribution of the time-to-first-byte for normal Tor and Tor with a shaped expander topology using RBIAS = 0, 1, 8 and 15. Fig 4.2 shows the cumulative distribution of the time-to-first byte for normal Tor and Tor with a shaped expander topology using SBIAS = 0, 1, 8 and 15. The time-to-first byte for normal Tor is 0.748 seconds at the median. As shown by Fig 4.1 and Fig 4.2, as the values of the parameters SBIAS and RBIAS are increased, the performance of the shaped expander topology improved compared to normal Tor.

Next, we evaluate the download time of 1 MB of data from 10 clients using a shaped expander topology. We compared the performance of normal Tor with one using a shaped expander topology for SBIAS and RBIAS values of 0, 1 and 8. The download time for normal Tor is 4.3 seconds at the median. Table 4.3 and Fig 4.4 show the download times for different values of SBIAS and RBIAS. We observe that as the values of SBIAS and RBIAS increase, their in a significant improvement in the download times.

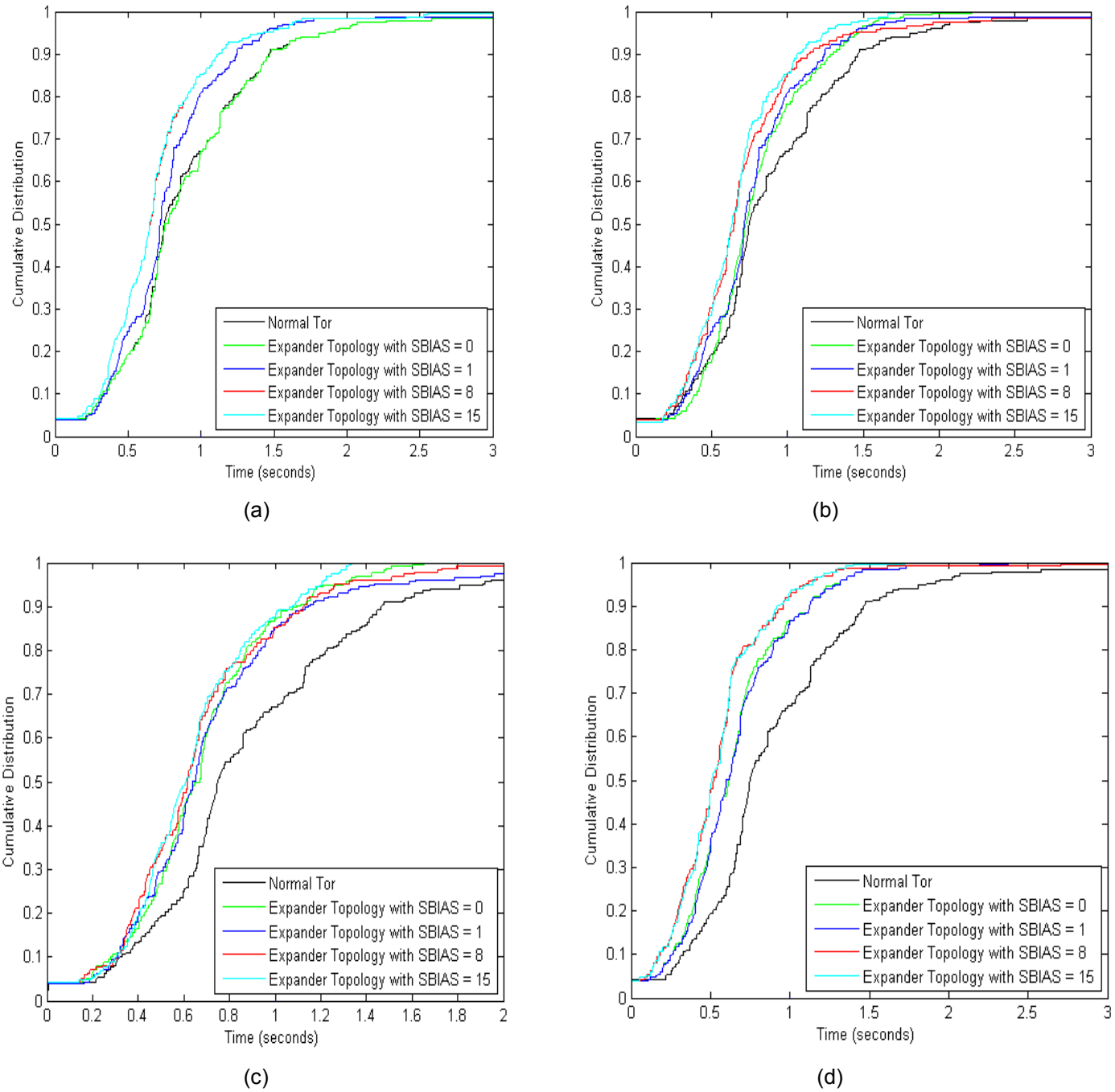
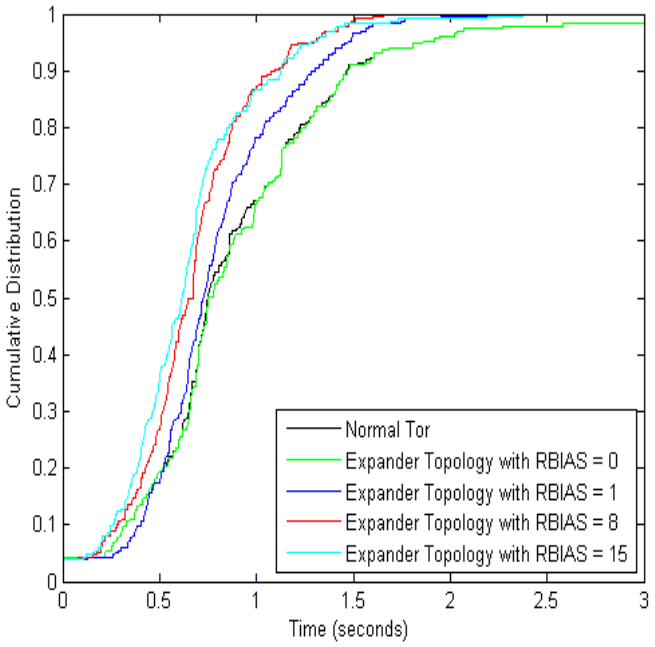
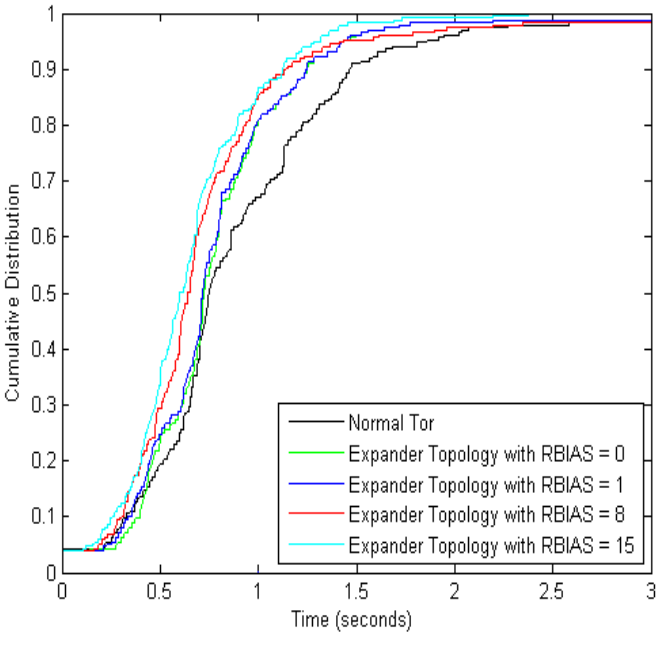


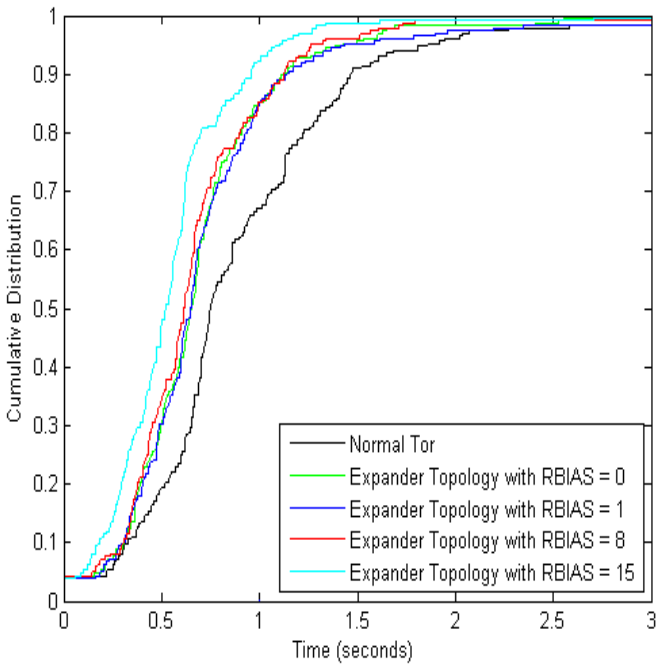
Figure 4.1 Tor client's time-to-first byte for different values of RBIAS (a) RBIAS=0 (b) RBIAS=1 (c) RBIAS=8 (d) RBIAS=15.



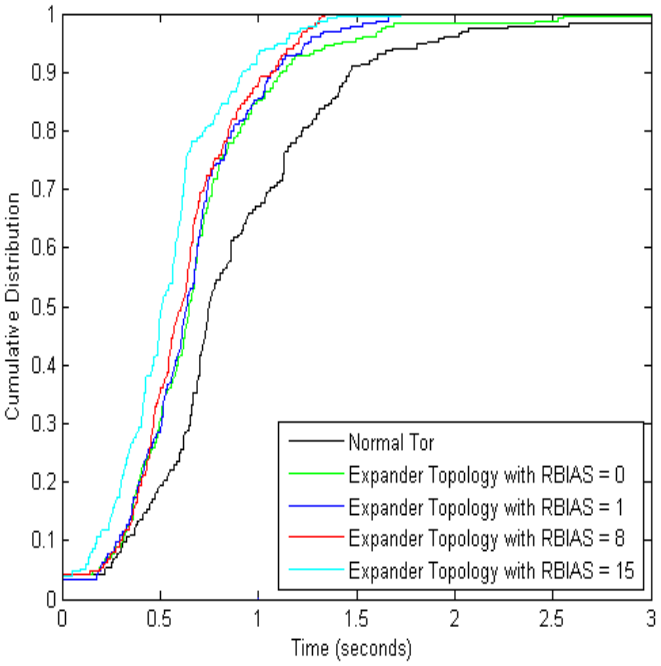
(a)



(b)



(c)



(d)

Figure 4.2 Tor client's time-to-first byte for different values of SBIAS (a) SBIAS=0 (b) SBIAS=1 (c) SBIAS=8 (d) SBIAS=15.

Table 4.2 Time-to-first-byte in seconds for different values of SBIAS and RBIAS.

	RBIAS=0	RBIAS=1	RBIAS=8	RBIAS=15
SBIAS=0	0.747	0.728	0.671	0.616
SBIAS=1	0.723	0.712	0.640	0.588
SBIAS=8	0.653	0.645	0.615	0.521
SBIAS=15	0.649	0.636	0.611	0.511

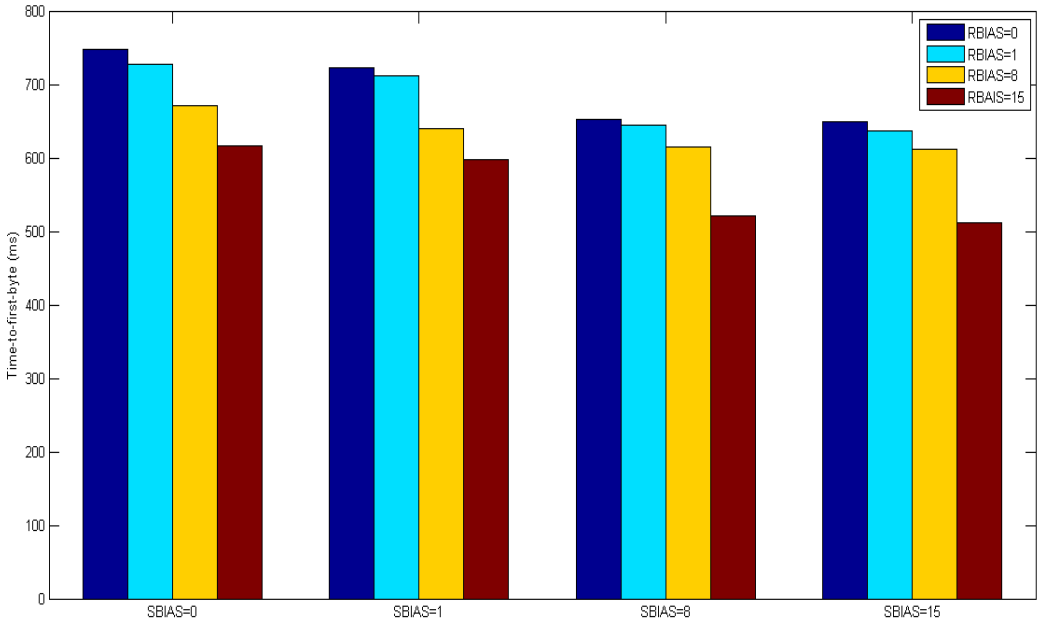


Figure 4.3 Tor client's time-to-first-byte for different values of SBIAS and RBIAS.

Table 4.3 Download time in seconds for different values of SBIAS and RBIAS.

	RBIAS=0	RBIAS=1	RBIAS=8
SBIAS=0	4.2	3.8	3.4
SBIAS=1	3.9	3.7	3.3
SBIAS=8	3.4	3.3	3.0

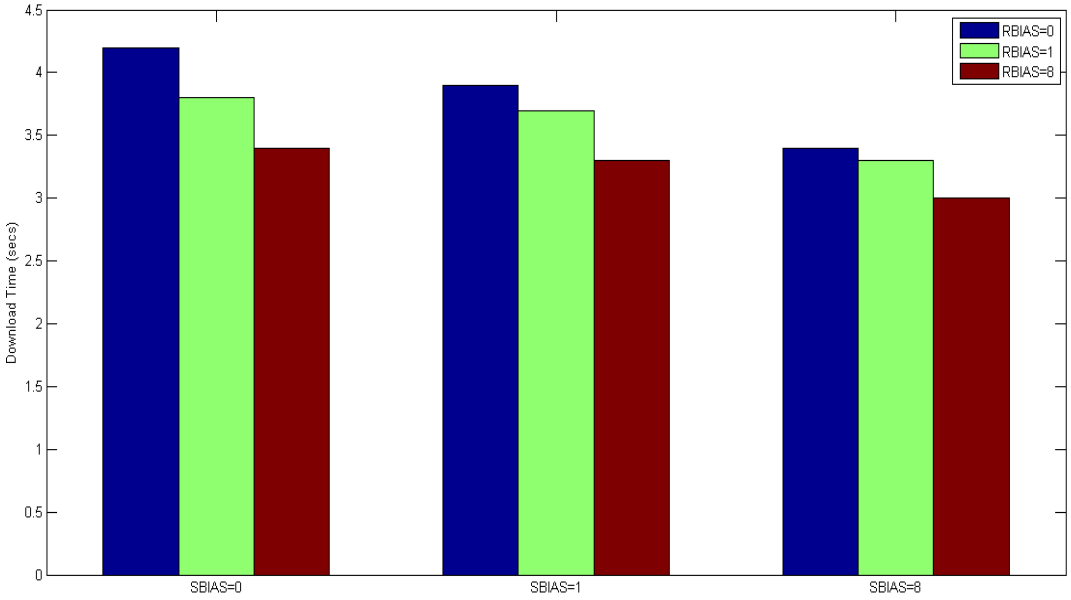


Figure 4.4 Tor client's download time for different values of SBIAS and RBIAS.

From Table 4.2, we observe that as the values of SBIAS and RBIAS increase, the performance of Tor improves. We find that the time-to-first byte reduces by roughly 13% to 22% compared to an unbiased topology. If we observe the shaping bias parameter, we see that there is significant improvement in the performance up to a value of SBIAS=8 for a fixed value of RBIAS after which we do not see much improvement. However, with a simultaneous increase in both SBIAS and RBIAS, we observe a significant improvement in the performance of Tor.

From Table 4.3, we observe that as the values of SBIAS and RBIAS increase, the download time reduces significantly. We find that the download time reduces by roughly 7% to 28% compared to an unbiased topology. The best performance is obtained when both the SBIAS and RBIAS parameters are 8.

From the above results, we can conclude that using a shaped expander topology biased towards low-latency links as the underlying network topology for Tor leads to a significant improvement in the performance of Tor.

CHAPTER 5

DISCUSSION

In our work, we propose to use a latency-biased network topology in Tor. This requires a significant revamp of the Tor architecture. In this section, we compare our work with [36] which shows that the performance of Tor can be improved with only client-side modifications. The author's design and implement a new Tor client, LASTor. They show that LASTor can deliver significant latency gains over the default Tor client by simply accounting for the inferred locations of Tor relays while choosing paths. Since the preference for low latency paths reduces the entropy of path selection, LASTor's path selection algorithm is designed to be tunable. A user can choose an appropriate tradeoff between latency and anonymity by specifying a value between 0 (lowest latency) and 1 (highest anonymity) for a single parameter. An efficient and accurate algorithm is proposed to identify paths on which an anonymity system can correlate traffic between the entry and exit segments. This algorithm enables LASTor to avoid such paths and improve a user's anonymity, while the low runtime of the algorithm ensures that the impact on end-to-end latency of communication is low.

By applying techniques similar to those used in our work for their experiments and evaluations, the authors in [36] show that in comparison to the default Tor client, LASTor reduces median latencies by 25% which is roughly the same as we have achieved in our experiments. However, LASTor is better compared to our proposal because it does not require any change in the Tor architecture and can be implemented by client-side modifications only.

CHAPTER 6

CONCLUSION AND FUTURE WORK

This thesis seeks to improve the performance of Tor, the most widely used privacy enhancing technology, using an underlying shaped expander network topology comprised of low-latency links. To that end, we offer a detailed analysis of Tor with the aim of improving its performance problems.

6.1 Conclusion

We propose to improve the performance of Tor by using an underlying shaped expander graph network topology that is biased towards low-latency edges rather than using random links. We evaluate our proposal on a private deployment of Tor on PlanetLab. We measure the time-to-first-byte of popular websites and the download time of 1 MB of data which corresponds to the size of an average web page. We observe that compared to a normal deployment of Tor, using a biased expander topology leads to a performance improvement of roughly 7% to 28% in Tor.

6.2 Future Work

Our proposal creates an underlying network topology that favors the nodes and links in the network which have high bandwidth and low latency. As a result, different parts of the network may have skewed traffic volumes. High performing nodes will carry majority of the traffic and the nodes having low bandwidth will have low utilization. The bandwidth distribution of all the nodes should be taken into consideration when constructing a shaped expander.

Another important area of interest is the churn in the network. Considering the fact that nodes may join or leave the network, an interesting area of study would be to evaluate how long an expander topology remains stable and accurate in such a scenario. Future work could

involve evaluating how long an expander topology sustains itself in a dynamic network environment after which it would need to rebuild itself.

REFERENCES

[1] D. Chaum, “Untraceable Electronic Mail, Return Addresses, and Digital Pseudonyms,” in Communications of the ACM, 4(2), Feb.1981.

[2] Anonymizer Inc. http://anonymizer.com.

[3] O. Berthold, H. Federrath, and M. Kohntopp, “Project anonymity and unobservability in the internet,” in Proceedings of Computers Freedom and Privacy, April 2000.

[4] R. Dingledine, N. Mathewson, and P. Syverson, “Tor: The second generation onion router,” in Proceedings of the 13th USENIX Security Symposium, August 2004.

[5] M. Reed, P. Syverson, and D. Goldschlag, “Anonymous connections and onion routing,” in IEEE JSAC Copyright and Privacy Protection, 1998.

[6] M. K. Reiter and A. D. Rubin, “Crowds: Anonymity for web transactions,” in ACM Transactions on Information and System Security, 1(1):66–92, November 1998.

[7] C. Shields and B. N. Levine, “A protocol for anonymous communication over the internet,” in Proceedings of the 7th ACM Conference on Computer and Communications Security (CCS 2000), pages 33–42, 2000.

[8] K. Bauer, D. Grunwald and D. Sicker, “Predicting Tor Path Compromise by Exit Port,” in Proceedings of 2nd IEEE International Workshop on Information and Data Assurance , Phoenix, AZ, December, 2009.

[9] D. McCoy, K. Bauer, D. Grunwald, T. Kohno, and D. Sicker, “Shining light in dark places: Understanding the Tor network,” in Proceedings of the 8th Privacy Enhancing Technologies Symposium, July 2008.

[10] R. Dingledine and N. Mathewson, “Tor path specification,” https://git. torproject.org/checkout/tor/master/doc/spec/path-spec.txt.

[11] N. Matthewson, “Base “stable” flag on mean time between failures,” https://svn.torproject.org/svn/tor/branches/hidserv-perf/doc/spec/proposals/108-mtbf-based-stability.txt.

[12] L. Øverlier and P. Syverson, “Locating hidden servers,” in Proceedings of the IEEE Symposium on Security and Privacy, May 2006.

[13] K. Bauer, D. McCoy, D. Grunwald, T. Kohno, and D. Sicker, “Low resource routing attacks against Tor,” in Proceedings of the ACM Workshop on Privacy in the Electronic Society, October 2007.

[14] A. Pfitzmann and M. Waidner, “Networks without user observability - design options,” Computers and Security, 6(2):158–166, 1987.

[15] R. Snader and N. Borisov, “A Tune-up for Tor: Improving Security and Performance in the Tor Network,” in Proceedings of the Network and Distributed Security Symposium - NDSS '08, February 2008.

[16] K. Bauer, J. Juen, N. Borisov, D. Grunwald, D. Sicker and D. McCoy, “On the Optimal Path Length for Tor,” 3rd Hot Topics in Privacy Enhancing Technologies, July 2010.

[17] R. Dingledine and N. Mathewson, “Tor directory protocol, version 3,” https://gitweb.torproject.org/torspec.git/blob/HEAD:/dir-spec.txt.

[18] PlanetLab: https://www.planet-lab.org.

[19] Pyxida: http://pyxida.sourceforge.net.

[20] “Tor metrics portal: Users,” https://metrics.torproject.org/users.html.

[21] “Alexa top websites,” http://www.alexa.com/topsites.

[22] K. Sampigethaya and R. Poovendran, “A Survey on Mix Networks and Their Secure Applications,” IEEE, vol. 94, no. 12, pp. 2142-2181, 2006.

[23] O. Berthold, A. Pfitzmann, and R. Standtke, “The disadvantages of free MIX routes and how to overcome them,” in Designing Privacy Enhancing Technologies, LNCS Vol. 2009, pages 30-45. Springer-Verlag, 2000.

[24] G. Danezis, “Mix-networks with restricted routes,” in Proceedings of Privacy Enhancing Technologies workshop (PET 2003). Springer-Verlag, LNCS 2760, March 2003.

[25] F. Dabek, R. Cox, F. Kaashoek, and R. Morris, “Vivaldi: A Decentralized Network Coordinate System,” in the Proceedings of ACM SIGCOMM (2004).

[26] C. Law and K.Y. Siu, “Distributed construction of random expander networks,” in IEEE Infocom (2003) 2133–2143.

[27] N. Mallesh and M. Wright, “Shaping Network Topology for Privacy and Performance,” in Proceedings of ACM Conference on Computer and Communications Security (CCS '11), [Poster Session], Oct 2011.

[28] S.J. Murdoch and R.N.M. Watson, “Metrics for Security and Performance in Low-Latency Anonymity Networks,” in the Proceedings of the Eighth International Symposium on Privacy Enhancing Technologies (PETS 2008), Leuven, Belgium, July 2008, pages 115-132.

[29] HTTP Archive: http://httparchive.org.

[30] T. Ngan, R. Dingledine and D.S. Wallach, “Building Incentives in Tor,” in Proceedings of the 14th international conference on Financial Cryptography and Data Security(FC’10), January 2010, pages 238-256.

[31] K. Loesing, “Evaluation of client requests to the directories to determine total numbers and countries of users,” Technical report, The Tor Project, June 2009.

[32] M. Sherr, M. Blaze, and B. T. Loo, “Scalable Link-Based Relay Selection for Anonymous Routing," in 9th Privacy Enhancing Technologies Symposium (PETS '09), August 2009.

[33] J. Reardon and I. Goldberg, “Improving Tor Using a TCP-over-DTLS Tunnel,” in the 18th USENIX Security Symposium, August 2009.

[34] C. Tang and I. Goldberg, “An Improved Algorithm for Tor Circuit Scheduling ,” in the Proceedings of the 17th ACM Conference on Computer and Communications Security (CCS '10), October 2010.

[35] M. AlSabah, K. Bauer, I. Goldberg, D. Grunwald, D. McCoy, S. Savage and G. M. Voelker, "DefenestraTor: Throwing out Windows in Tor," in the 11th Privacy Enhancing Technologies Symposium, July 2011.

[36] M. Akhoondi, C. Yu, and H. V. Madhyastha, "LASTor: A Low-Latency AS-Aware Tor Client," In the Proceedings of the 2012 IEEE Symposium on Security and Privacy, May 2012.

BIOGRAPHICAL INFORMATION

Subhasish Dutta Chowdhuri was born in Calcutta, India in 1983. He received his B.E. (Computer Science and Engineering) degree from the University of Burdwan, India in 2005. From 2005 to 2010, he was with IBM in Calcutta, India as a UNIX Systems Administrator. He has been a part of ISec, the Information Security Lab, from 2010. From August 2012, he will join LinkedIn Corporation as an Operations Engineer.