SHAPING AND MAINTENANCE OF LOW-LATENCY ANONYMITY SYSTEMS

by

HARSHA DORESWAMY

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

December 2012

To my parents, family and friends who have supported and encouraged me.

ACKNOWLEDGEMENTS

I would like to express my profound appreciation to my supervising professor Dr. Matthew Wright for supervising me on this thesis and guiding me in every step of my work. And I would like to thank Dr. Mohan Kumar and Dr. Gergely Zaruba for their invaluable advice and interest in my research and for taking time to serve on my thesis committee. I would also 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.

Finally, I would like to thank my parents, family and friends. Were it not for their support throughout my academic career, I wouldn't have made it as far as I have. I will never forget it.

November 19, 2012

ABSTRACT

SHAPING AND MAINTENANCE OF LOW-LATENCY ANONYMITY SYSTEMS

Harsha Doreswamy, M.S.

The University of Texas at Arlington, 2012

Supervising Professor: Matthew Wright

Surveillance of users on the Internet is growing. This makes it increasingly important to protect users' identity when they communicate online, especially for users like journalists, whistleblowers, and military officials. Anonymity systems can conceal users' identities and provide anonymity for their online activities. Low–latency anonymity systems like Tor are designed to provide support for applications like Web browsing, video streaming, and online chat. To make the commutation between source and destination unlinkable, Tor routes all the traffic through three Tor relays, which are spread across the globe. These three relays are usually chosen without considering the delay to get from one to the next. If the relays selected are located too far from each other, it reduces the performance of the system. On the other hand, if we pick the three relays closest to the sender, then it would make it easier for the attacker to identify the sender. In other words, we would lose anonymity. One of biggest challenges in low–latency anonymity systems is maintaining anonymity while improving performance.

In this thesis, we try to improve the performance of Tor–like low–latency anonymity systems without sacrificing anonymity. The current version of Tor tries to improve the performance of the system in the relay selection process by biasing clients to select nodes with higher bandwidth. However, the performance gain from bandwidth–based biasing alone is not sufficient for many users. To further increase performance while ensuring strong anonymity properties, we propose to arrange the nodes and links of the network into restricted network topologies, through which anonymized traffic is routed. In building these restricted topologies, we bias the construction process to select lower latency edges to improve performance, while also ensuring that the network's bandwidth capacity is well utilized. We examine two restricted topologies: an expander graph topology and a novel clustering model topology. In the expander graph topology, the system constructs an expander graph with Hamiltonian cycles in which the graph edges are biased towards latency. In the clustering model topology, we cluster the relays based on their bandwidth and construct the graph in two steps. First, the system calculates the number of edges for given node that connects to each cluster based on the ratio of the total number of edges of the cluster to the total number of all edges in the system. Then nodes are selected from these clusters with a bias towards low–latency edges.

A key challenge in the deployment of a topology is maintaining the structure and properties of the topology, in spite of churn— nodes joining and leaving the network. We have shown that our topologies, of size 750 nodes, have maintained the structure and properties of the topology for up to 25,000 churn operations without significant degradation to anonymity and performance when compared with the initial topology. We also compare the anonymity and performance of the expander graph versus the clustering model topology on various parameters, such as latency bias, number of hops, and number of nodes joining and leaving.

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

With the ever growing use of the Internet in all aspects of our daily lives, privacy and confidentiality are becoming increasingly important to the success and wide–spread use of many online applications. Hiding user's identity is Anonymity. In particular providing anonymity to users on internet is a growing concern. Encryption alone does not provide the level of privacy required by many users, since it only protects the transmitted data instead of the identities of the communicating parties. User anonymity is important in Internet applications such as electronic voting, e–commerce, peer–to–peer file sharing, and anonymous Web–browsing or email. Anonymity can also benefit government and law enforcement agencies, as it creates a medium of safe communication for whistleblowers and citizens willing to submit leads in criminal investigations. It can also help corporations seeking for a means of doing competitive research and business deals. For journalists and more specifically for citizen journalists, anonymity systems could be a very important tool.

Tor is free software and an open network that provides online anonymity [1]. Tor is one of the most widely used anonymity systems with around 500,000 users [2]. Tor provides anonymity in terms of unlinkability of sender and receiver, i.e., the sender and receiver of a communication cannot be identified even if the sender and receiver are known to be of communicating through Tor. The Tor client software routes all the traffic though a sequence of three Tor relays, called a circuit, to ensure that no single router can link the sender and receiver. Tor relays are operated by volunteers and there are around 3,000 volunteer–operated Tor relays spread across the globe [3].

The volunteer–operated nature of Tor relays creates two problems. First, the volunteers provide a wide range of bandwidths for their Tor relays. This variation in bandwidth not only makes system performance unreliable, it can also lead to a loss of anonymity due to attacks like throughput fingerprinting [4]. In this attack, an attacker observes the throughput of a Tor flow, which can be used as a fingerprint of the bottleneck relay— the relay with the least bandwidth in the circuit. Through these observations, it is possible to identify when two or more circuits share the same bottleneck relay, which can then be used as a stepping stone for de–anonymizing the user. Second, the variations in latency— round trip time (RTT) between Tor relays. With Tor relays spread across the globe unevenly, the latency between the relays can hugely vary. If circuits are built with high–latency links between the Tor relays, then it would reduce the performance of the system. On the other hand, always building circuits with low–latency links between the relays, would make it easier for the attackers to identify the relays used and can also lead to de–anonymizing the user.

1.1 Contributions

Our goal is to improve the performance of low–latency anonymity systems like Tor without reducing users' anonymity. To this end, we propose to organize Tor relays into *restricted network topologies*, through which Tor traffic is routed. The restricted topologies provide more cover traffic for users, which improves the anonymity. Keeping in mind the influence of latency on Tor's performance and anonymity, we try to include this property in the construction of our topologies to increase the performance of Tor. Tor studies appear to indicate that bandwidth is a limited resource. We treat bandwidth as a shared resource and design a topology with both high utilization of the available bandwidth and fair access to the bandwidth.

We examine two network topologies: an expander graph topology and a novel clustering model topology. In the expander graph topology, which closely follows the work of Mallesh et al [5], the system arranges nodes into an expander graph with the graph edges selected using a bias towards low–latency edges. We construct our expander graph with Hamiltonian cycles, which helps in maintaining the graph's structure and properties even after churn operations. In the clustering model topology, we cluster the relays based on their bandwidth using the K–means algorithm and construct the graph in two steps. First, the system assigns each node a number of edges to each cluster based on the ratio of the bandwidth of that cluster to the total bandwidth in the system. In the second step, for each of the assigned edges to a given cluster, nodes from that cluster are selected using a bias towards low–latency edges.

Our evaluation process consists of constructing both topologies in simulation and calculating latency and entropy of the topologies. In real–world systems, nodes join and leave the network all the time. To take into consideration of this churn—nodes joining and leaving, and the effects of churn on the topologies, we simulate churn operations varying from 0 to 25,000 and then checking for changes in latency and entropy of the topologies . The graph size of each of these topologies is around 750 nodes. We used MIT's King dataset [6] for assigning latency values and sample Tor bandwidth values for assigning bandwidth values. We find that the performance of the clustering model topology can be significantly increased, without affecting the anonymity of the system, by increasing the biasing towards low–latency links. We also found that the maintenance process of the expander graph should be carried out without latency biasing, whereas the maintenance process of the clustering model topology should be carried with latency biasing. A comparison between the expander graph and the clustering model shows that the expander graph offers better anonymity while the clustering model offers better performance.

In Chapter 2, we provide the background context for our work, including a brief description of Tor and discussions of related work on improving the performance of Tor and topology maintenance in distributed systems. We also discuss the similarities and differences of our work with other topology maintenance techniques. Chapter 3 describes the construction and maintenance process of the expander graph and clustering model topologies. Chapter 4 describes the experimental setup and simulation results. Chapter 5 concludes our work by considering the outcomes of our results and discussing ideas for possible future work.

CHAPTER 2

BACKGROUND

In this chapter, we will discuss the background of my thesis. We start with a brief description of working of Tor. Then ways of improving performance of Tor and various parameters of Tor used in this process. We also discuss one of a kind anonymity system called LAP. A brief discussion of topology maintenance in distributed systems. Then get into a prior work on a restricted network topology—expander graph.

2.1 Tor

Tor (The Onion Router) is free software and an open network which provides online anonymity. Tor conceals its users' identities and their network activity from surveillance and traffic analysis. The Tor client software routes all the traffic though a sequence of three Tor relays, called a circuit, to ensure that no single router can link the sender and receiver. Tor is built on a second–generation onion routing design that provides low–latency anonymity for TCP–based applications [1]. The term onion routing refers to layered encryption, where the original message is encrypted multiple times and then sent through a series of Tor relays. Each of these relays decrypts one layer of encryption, allowing it to learn of the next hop in the path, and passes it on to the next relay until it reaches the destination.

Tor is the most widely deployed anonymous communication system with around 3000 volunteer–operated nodes [3] and an estimated 500,000 users [2]. Tor's primary goal is to ensure anonymity with low enough latency to facilitate the use of interactive applications such as instant messaging and Web browsing. The architecture of the Tor system consists of Tor proxies (or clients), Tor relays, a set of trusted directory servers that advertise information about the Tor relays such as their IP addresses, public keys, exit policies, and self–reported bandwidth capacities. To initiate anonymous communication, a Tor client first queries one of the directory servers to obtain a signed list of the available Tor relays. It then establishes paths, called *circuits*, through the Tor network by choosing three Tor relays and performing a key establishment protocol with each relay in turn. The number of Tor relays chosen initially can be more than three relays, if the user desires greater anonymity, but Tor requires a minimum of three relays to maintain sender–receiver unlinkable anonymity.

One of the most common problems faced by Tor users is that it significantly slows down Web browsing speed. Addressing this performance issue is not easy and might lead to a decrease in the user's anonymity. For example, Tor does not add intentional delays or use cover traffic, as these measures could hurt performance for interactive applications such as Web browsing or instant messaging [1]. However, this also increases Tor's vulnerability to end–to–end traffic correlation. One more important design decision in Tor is the length of the circuits. The default Tor uses three relays in a circuit [7] to mitigate any single relay's ability to link a source and destination. Reducing the number of relays in the circuit could improve performance, but it leaves the system vulnerable to attacks. For example, consider a path length of two and an attack model where the attacker controls one out of two relays on the circuit. Then the attacker can carry out traffic surveillance on the other relay and correlate the traffic from both relays, which can lead to de–anonymizing the whole communication. The current Tor design tries to find a compromise that satisfies both those users who desire strong anonymity and also those for whom performance is more of a priority.

One more reason for the low performance of Tor can be attributed to the volunteer–operated nature of the Tor relays, as many of them are on slow connections or share bandwidth with other activity [8]. Hence, when selecting nodes in the circuit, it is important to make the best use of all the limited available capacity. The Tor relays on each path are selected by the client 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 Tor relays.

2.2 Improving the performance of Tor by path selection

We can assume that Tor users have different requirements and expectations. Some users require high anonymity, whereas other users require higher performance. Hence, there should be a provision for compromise between performance and anonymity in Tor. This provision can be addressed during the selection of relays for constructing Tor circuits. The default relay selection process in Tor is biased towards high bandwidth relays to improve performance and load balancing. The bandwidth values used in this process are reported by the owners of the relays and are not verified. This can lead to attacks where an attacker can insert malicious nodes into the Tor network and report a higher–than–actual bandwidth so that the bandwidth–biased relay selection algorithm would pick these malicious nodes in most of their circuits. To mitigate the effectiveness of this attack, all bandwidth advertisements are capped to 10 MB/s [9]. In spite of this upper bound, the attack can be quite successful. Bauer et al. report that a small fraction of attacker controlled relays can attain the first and last node positions on nearly half of the Tor circuits built using this attack, which could lead to de–anonymizing the whole communication [10]. Even when the nodes are honest, the reported values may not truly indicate the available bandwidth at a node due to changing network conditions and other factors. This makes the performance of Tor highly variable.

To address these issues, Snader and Borisov propose to replace the self–reported bandwidth mechanism in Tor with a scheme for opportunistic bandwidth measurement [11]. The current topology of the Tor network allows each relay to interact with most other relays and observe their performance over time. These observations can be reported by each of the Tor relays to the centralized directory servers. Then the centralized directory servers can consolidate the observed data of other relays' performance and can more accurately predict the actual performance of the relays. This can mitigate the issue of false bandwidth self–reporting.

Snader and Borisov also propose that end–users should have the ability to decide whether to bias the relay selection towards higher bandwidth relays or to select relays uniformly at random [11]. In the current Tor path selection algorithm, the probability of a node being selected is proportional to their contribution to the total network bandwidth. On the other hand, in the Snader–Borisov (S–B) tunable path selection approach, the probability of selecting a node depends on the bandwidth–based rank ordering of nodes. In the S–B approach, all the n nodes in the network are sorted in descending order of their measured bandwidths and then ranked. The node with highest bandwidth will have rank 1 and node with least bandwidth will have rank n. To select a node from this rank order, we apply the following equation:

$$f_s(x) = \frac{1-2^{sx}}{1-2^{s}} \quad (for\ s\ \neq 0) \tag{2.1}$$

$$f_0(x) = x$$

where x is selected uniformly at random from the interval [0, 1) and the value of s is selected by the user according to their preference for the type of performance they need. Once $f_s(x)$ is calculated, then node is selected from the rank order by selecting the node at index $\lfloor n * f_s(x) \rfloor$. In the above Equation(2.1) if $s = 0$, then the nodes are selected uniformly at random. This is for users who require high anonymity. As s value increases, better ranked nodes will be preferred, for users who are willing to compromise anonymity for better performance. In their experiments, Snader and Borisov show that users can use this approach to attain reasonable performance and anonymity across a range of trade–off points. We adapt the S–B approach to our proposed designs.

Sherr et al [12] describe a link–based relay selection process for flexibly tuning the performance and anonymity properties of anonymous paths. In the link–based relay selection strategy, the relays are chosen in such a way that the links connecting the relays provide high performance gain. In comparison to node–based techniques, in which relay selection process is biased only by the node characteristics (i.e., bandwidth), link–based selection enables the choice of high performance paths across multiple metrics: latency, jitter, and loss, as well as bandwidth. In this approach, all three relays for the candidate paths are chosen uniformly at random, then the end–to–end cost (i.e., weight) of each generated candidate paths is calculated based on the desired link characteristics. These candidate paths are then sorted based on weights and the S–B [12] biasing technique (described above) is used to select the relays to build the circuits.

Akhoondi et al. [13] design and implement a new Tor client called LASTor. The authors show that the performance of Tor can be improved by only client–side modification. They show that LASTor can deliver significant gain in performance over the default Tor client by using the inferred locations of Tor relays while choosing paths. LASTor's path selection process focuses on two issues: 1. relays selected from the same AS (Autonomous systems) and 2. geo–location services, to improve the performance, while protecting anonymity. The authors state that if two relays are selected from the same AS, then the AS can link the two relays using statistical analysis. Hence path selection algorithm makes sure that its relays are selected from different ASes. LASTor uses geo–location services to map the relay's geographical locations and use this data in path selection to improve performance. Since a 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 trade–off between performance and anonymity by specifying a value between 0 (highest performance) and 1 (highest anonymity) for a single parameter. They also propose an efficient and accurate algorithm to identify paths on which an anonymity system can correlate traffic between the entry and exit segments. This algorithm enables LASTor to identify and avoid such paths and improve the user's anonymity.

2.3 Other ways of improving Tor performance

In this section we discuss ways of improving the performance of Tor using approaches which require significant changes to the Tor system.

Another area of investigation is the improper application of TCP's congestion control mechanisms which degrade the performance of Tor. In Tor, the traffic between any pair of relays is multiplexed over a single TCP connection. This will result in interference across circuits during congestion control resulting in packet dropping or packet reordering. Reardon et al. [**?**] propose to use a TCP–over–DTLS (Datagram Transport Layer Security) between relays. Each stream of data has its own TCP connection with 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 issues and improves performance.

Streaming videos online is greatly affected by the low performance of Tor. AlSabah et al. [14] propose Conflux, a multipath circuit construction and stream–splitting approach that increases performance, especially for clients using low–bandwidth bridges, unadvertised proxies that provide Tor access to users whose ISPs or countries block Tor. In this design, Tor clients use Conflux to build a number of circuits (two or more) that intersect at a common Tor exit relay. The packets are split at client end and are reconstructed at the exit relay. This approach decreases latency and increases throughput, particularly for low–bandwidth bridge users and streaming videos.

Tor depends on volunteers to donate their resources. Hence Ngan et al. [8] explore the idea of giving incentives to Tor users to route Tor traffic. The idea is that if users contribute resources to Tor overlay, they should receive faster service in return. They propose a solution in which the central Tor directory authorities measure the performance of each relay and construct a list of reliable *gold star* relays. Relays obtain this list from the central directory authorities during normal updates. Traffic from these gold star relays are marked as high priority by other relays and they receive a better treatment along the whole circuit.

2.4 LAP

There is a new one of a kind system called LAP: Lightweight Anonymity and Privacy [15] for future Internet. LAP is an efficient and practical network–based solution featuring lightweight path establishment and efficient communication. LAP attempts to enhance anonymity by obscuring an end–host's topological location, based on two building blocks: packet–carried forwarding state and forwarding–state encryption.

Unlike Tor, in which all the relays in the circuit are predetermined, in LAP, each packet carries its own forwarding state. The encryption schema used in LAP allows each Autonomous Domains (AD) to use a secret key to encrypt and decrypt forwarding information in packet headers. As a result, an AD's forwarding information can be hidden from all the other entities while a LAP packet remains the same at each hop.

LAP also supports different privacy levels such that an end–host can trade the anonymity for improved performance. LAP's generic design can work with a wide range of routing protocols, which includes the inter–domain routing protocols. LAP considers a relaxed threat model, where the attacker can compromise any AD except the first–hop AD, where the victim end–host resides. Because of this weaker attacker model, LAP is only suitable for users who trust their local ISPs but want protection from tracking by ISPs that are further away and from being tracked by Websites.

2.5 Topology maintenance in distributed systems

One of the biggest problem with organizing nodes in an overlay into a topology is churn, the continuous process of node arrival and departure. Rhea et al. [16] address the churn problem in Distributed Hash Tables (DHTs), with their new design called *Bamboo*. In Bamboo, they look at three important factors for handling churn: 1. Reactive versus periodic recovery from failures, 2. Calculation of message timeouts during look–ups, and 3. Choice of nearby over distant neighbors. In reactive recovery, as soon as a node fails it is replaced with another node immediately, whereas in periodic recovery, all the failed nodes are recovered periodically irrespective of the churn rate. When the churn rates are high, reactive recovery can lead to a positive feedback cycle in which the whole DHT is continuously dealing with topology changes on reaction to the churn, and these changes harm system performance. For medium and high churn rates, the authors found that periodic recovery is a better solution. Also, they compared the different ways of choosing nodes that are used to replace the failed nodes, and they found that choosing neighbor nodes with lower latency improves performance over random selection of nodes. Our findings surprisingly indicate that this is not always the case.

Godfrey et al. [17] show various selection strategies to minimizing churn in distributed systems. These strategies can be categorized into predictive replacement strategies that are based on the information collected about nodes and agnostic replacement strategies that do not consider any information. In the predictive replacement strategy, there are three types: 1. Max expectation: select the node with the greatest expected remaining uptime, 2. Longest uptime: select the node with the longest current uptime, and 3. Optimal: select the node with longest time until next failure. In the agnostic replacement strategy, there are also three types: 1. Random replacement: the failed node is replaced with an available node selected uniformly at random, 2. Passive preference list: the nodes are ranked and the failed node is replaced with the most preferable node, and 3. Active preference list: similar to passive preference list, the nodes are ranked and the failed node is replaced with the most preferable node, and when a new node becomes available, then all the nodes are ranked again and the node is replaced with the most preferable node again. Based on their experiments [17], they find that, in several node failure cases, the random replacement strategy would be easier to implement and may offer a better trade–off between churn and system complexity.

Law et al. [18] describe two ways of dealing with node failures in network topologies in order to maintain the properties of the topology: 1. Youngest node, and 2. Shell recycling. In the youngest node approach, whenever a node leaves/fails, the failed node is replaced with the youngest node in the graph. This way the impact on the properties of the network topology is minimal. To replace the failed node by the youngest node, the challenge is to maintain the youngest node. The authors propose a global server to manage the current youngest node at all times, and thus all the nodes can identify the youngest node when a node fails. In the shell recycling approach, the failed node's *shell*— simulation of the node, is adopted by one of its neighbors, and this neighbor node processes the requests of the failed node and along with its own requests. Each node has a maximum limit of adopting failed nodes' shells. When all the nodes reach their maximum limits, then the whole topology is reconstructed. In our work, we take into consideration of node's bandwidth and latency to construct and maintain the topology. Similar to Rhea's Bamboo [16], we could also use periodic recovery if the churn is high. Godfrey et al. [17] propose considering additional properties to better reduce the effects of churn in distributed systems. They consider node properties like node availability, uptime, and expected time to failure, and they maintain a preference list of nodes to replace the failed nodes. These additional parameters can also be included in our approach during biasing. In this thesis, however, we focus more on increasing performance of the system without affecting its anonymity, rather than considering all possible attributes of the node during node failure.

2.6 Expander graph

Expander graphs are sparse graphs which are highly connected. To disconnect an expander graph, very large portion of edges have to be deleted. They are called expander graphs because they have good expansion properties [19].

Mallesh et al. propose a system that uses expander graphs as restricted network topology to improve the anonymity and performance of low–latency anonymity systems like Tor [5]. They show that expander graph possess excellent connectivity properties enabling them to be extremely fast mixing. Along with that, fewer numbers of edges per vertex would provide cover traffic (i.e. more traffic on the same set of edges, making it difficult to distinguish between the traffic coming from different nodes) for the users and there by increase their anonymity. All these properties make expander graphs a good choice to build the underlying topology of anonymity networks.

2.6.1 Construction

Mallesh et al. employ the distributed construction method of Law et al. [18]. To construct expander graphs with D degree, we use $\frac{D}{2}$ *Hamiltonian cycles*, which are closed loops through a graph that visits each node exactly once. The method begins with an initial set of three nodes that are connected to each other in all $\frac{D}{2}$ Hamiltonian cycles. Then the graph grows incrementally, with new nodes joining the existing nodes on each of the Hamiltonian cycles. When a new node joins the network, it chooses a node at random from the graph. Then it randomly picks one more node from the chosen node's neighbors. This chosen node and its neighbor will then become the neighbors of the new node, and the connection between the chosen node and its neighbor is disconnected, adding the new node the graph. This construction method leads to a random expander graph.

Mallesh et al. propose an expander topology that is biased towards low–latency edges, which would simultaneously achieve both the security benefits of an expander topology and improved performance [5]. An expander graph generated with a bias towards low–latency edges, called a *shaped expander*, results in higher performance than a random expander graph and also higher anonymity than a fully connected topology. The low–latency bias method to generate a shaped expander graph was implemented by using the S–B approach (Equation 2.1), to bias the node selection process towards low–latency links. Due to this biasing, the shaped expander graph may lack the expansion properties of a random expander. However, Mallesh et al.

show that biasing the expander graph does not lead to a topology that requires significantly more hops to reach maximum entropy [5]. Hence, shaped expander graphs can be used to improve link performance without substantial loss of anonymity.

CHAPTER 3

SYSTEM DESIGN

In this chapter, we explain the construction and maintenance of our two restricted network topologies, the expander graph topology, and the clustering model topology.

3.1 Expander graph topology

3.1.1 Construction

We construct our expander graphs much similar to Mallesh et al. [5] but with modest changes in the implementation, to help with topology maintenance. Mallesh et al. propose that when a new node joins the network, it chooses its neighbor nodes at random, with latency bias in case of the shaped expander, from all the nodes in the network. We instead keep track of the $\frac{D}{2}$ Hamiltonian cycles. When a new node joins the network, for each Hamiltonian cycle, a random connected pair of nodes are chosen as the new node's left and right neighbor nodes in the cycle. In the case of a shaped expander graph, latencies between the new node and all the connected pairs of nodes of that Hamiltonian cycle are calculated and ranked. Then by using the S–B (Equation 2.1), a connected pair of nodes are chosen as the new node's left and right neighbor nodes. This selection process is repeated for all the $\frac{D}{2}$ Hamiltonian cycles.

3.1.2 Maintenance

Maintaining the expander graph topology means handling churn operations without modifying the basic properties of the graph. When a node leaves the network, for each of $\frac{D}{2}$ Hamiltonian cycles, the left neighbor of the leaving node will be directly connected to its right neighbor. In other words, the left and right neighbors will be directly linked, filling the void created by the leaving node. This way, the cycle is maintained and so is the expander graph's properties. A node joining the network follows the same process as explained above in the construction of the expander graph. Node joining can also be made biased towards low–latency, using the same method as used to construct the initial graph.

3.2 Clustering model topology

While the shaped expander graph uses latency information to improve the performance of the system, it does not consider the bandwidth of the nodes. In the expander graph, since all the nodes have same number of edges, the bandwidth used by all the nodes would also be more or less the same, even if the nodes have variable bandwidths. Tor, for example, has a large range of bandwidths among its relays. Tor relays with high bandwidth can handle more circuits at a time as compared with low–bandwidth nodes. Hence, the high bandwidth nodes should have more edges compared to the low–bandwidth nodes. To develop a topology that takes both bandwidth and latency into consideration, we propose the *clustering model topology*. In this topology, we cluster the nodes in the network based on bandwidth and then select nodes from each of these clusters with a bias towards low–latency edges. This makes sure that high bandwidth nodes will be highly connected, while also biasing the topology towards faster links.

Figure 3.1 depicts the clustering model using data sampled from our simulations described in Chapter 4. There are 5 clusters with 750 nodes. Each cluster shows the number of nodes the cluster contains, the average number of edges of all nodes, the

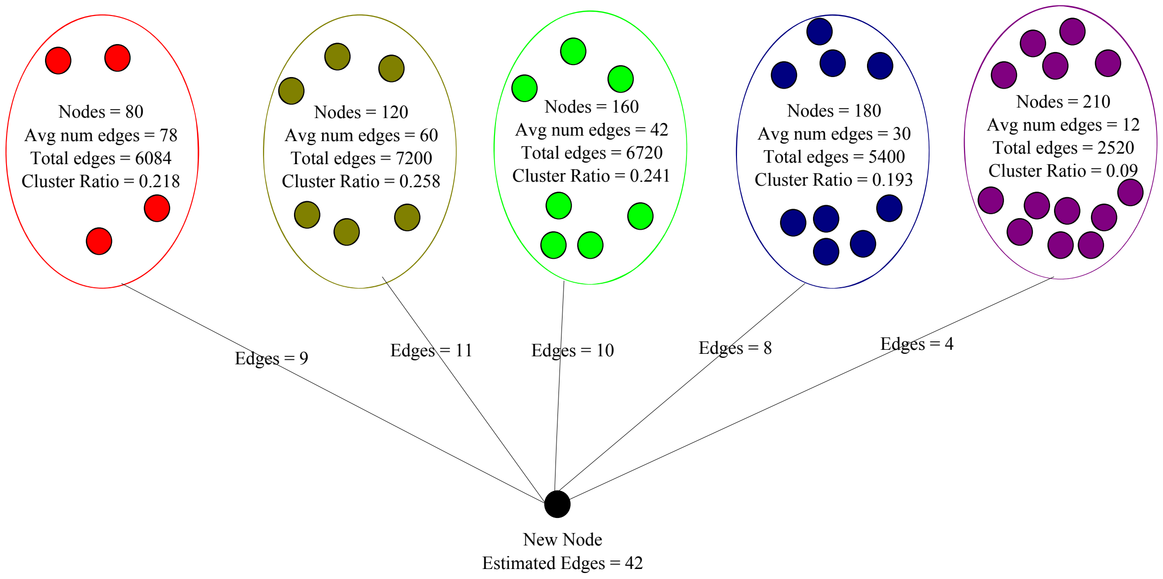


Figure 3.1. Clustering model with five clusters and sample values from our simulations.

total sum of all the edges, and cluster ratio. These clusters are formed based on bandwidth of the node. High bandwidth nodes form one cluster and similarly low–bandwidth nodes form another. When a new node joins, based on its estimated number of edges and the cluster ratio, we can calculate the number of edges of the new node connecting to a particular cluster.

3.2.1 Construction

We apply the K–means clustering algorithm [20] to cluster nodes by bandwidth. K–means clusters similar data into a predefined number of clusters K. Algorithm 1 shows the steps in clustering model.

In the clustering model algorithm, we begin with a list of the nodes and their bandwidths, and a predefined K value for K–means clustering. For each node, we estimate the ideal number of edges the node should have based on its bandwidth. This

Algorithm 1 Clustering model Algorithm

```
Require: $nodeList$, $K$ value for clustering
1:  for each $node \leftarrow nodeList$ do
2:      $estimatedEdgesList \leftarrow calculateEstimatedEdgesFromBandwidth(node)$
3:  end for
4:  $clusterList \leftarrow kMeansOnEstimatedEdges(estimatedEdgesList, K)$
5:  for $i = 1 \rightarrow K$ do
6:      $clusterRatio_i = \left(\frac{sumOfEdgesInCluster_i}{sumOfAllEdges}\right)$
7:  end for
8:  for each $node \leftarrow nodeList$ do
9:      for each $cluster \leftarrow clusterList$ do
10:         $numEdges \leftarrow calculateNumEdgesInCluster(node, cluster)$
11:         for $i = 1 \rightarrow numEdges$ do
12:             $neighborNode \leftarrow getLatencyBiasedNode()$
13:             ADD an edge from new node to this selected $neighborNode$
14:         end for
15:     end for
16: end for
```

is done by mapping a range of bandwidth values to the number of edge values proportionately. We map bandwidth to edges so that it helps us to simply the topology to just nodes and weighted edges by converting node weight to number of edges. In our bandwidth–to–estimated–edges mapping schema, we assign minimum of 10 edges to nodes with bandwidth of 50 KBps or less, and a maximum of 100 edges to nodes with bandwidth of 950 KBps or more. For every increase of 10 KBps of bandwidth, we increase the estimated edges by one. We have also provided a parameter which can increase or decrease the number of estimated edges to the bandwidth ratio in the mapping. After the estimated ideal number of edges is calculated for each node, we use this number as a property of the node for applying one–dimensional K–means clustering to produce K clusters. For each of these clusters, we calculate the *cluster ratio*. The cluster ratio is the ratio of the sum of all the edges in that cluster to the sum of all the edges in the whole network. In Line 10 of Algorithm 1, we calculate the number of edges of a node that should be connected to a cluster by multiplying the cluster ratio by the node's estimated edges. At this point, we know how many edges a node should have and what portion of those edges go to each of the other clusters. The next task (Line 12) is to identify the nodes in those clusters to which these edges will be connected. To identify these nodes, we consider both node availability and latency. We rank all the candidate nodes based on availability and latency using the ranking algorithm described below.

The ranking algorithm works as follows. Let e_i be the estimated ideal number of edges for a node i,and let c_i be the current number of edges that i has. Then we define a *difference ratio* $d_i = \left(\frac{e_i - c_i}{e_i}\right)$, the *attractiveness* a of the node i as:

$$a_i = \begin{cases} (d_i^2) \quad if \quad d_i > 0 \\ -(d_i^2) \quad if \quad d_i \leq 0 \end{cases}$$

and the *total weight* m_i of node i as: $m_i = a_i - \omega \times \left(\frac{\ell_i}{\ell_{ave}}\right)$, where ℓ_i is the latency of node i to the new node, ℓ_{ave} is the average of latencies of node i to all the candidate nodes, which puts ℓ_i into perspective, and ω is a weighting parameter, used to tune the relative importance of latency. Based on our experiments, we set the value of ω to 0.1, so that it improves the latency without affecting the anonymity. The difference ratio of a node indicates the difference between estimated ideal number of edges and current number of edges with respect to estimated ideal number of edges. A positive difference ratio indicates that more edges can be added to the node and negative difference ratio indicates that the node has more number of edges than the estimated ideal edges. The attractiveness of a node captures the difference ratio and increase or decreases its importance (attractiveness) based on the value of difference ratio.

The weight m_i that is calculated for each candidate node i is then sorted in descending order and ranked, and used in node selection process. In node selection process, we select the neighbor node by using the S–B (Equation 2.1).

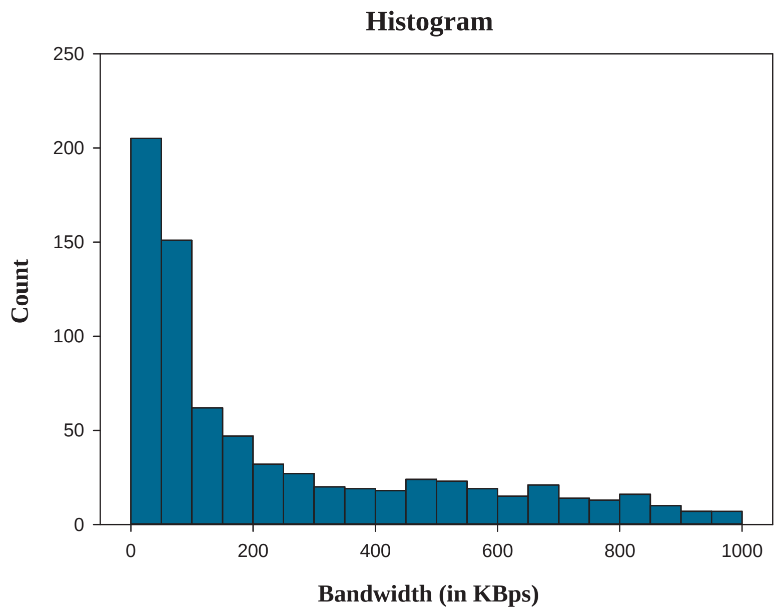


Figure 3.2. Tor Bandwidth distribution.

The reason we choose to cluster the bandwidths is two fold. First, the bandwidth distribution in Tor is not uniform. There are large number of low–bandwidth nodes compared to a very small fraction of high bandwidth. Figure 3.2 shows the distribution of bandwidth of Tor nodes, extracted from a Tor descriptor file . We extract the Tor bandwidths from the Tor descriptor file, a file downloaded by each Tor client from the directory server, which contains all the information about the Tor relays. This figure shows a large population of low–bandwidth relays and a long tail, hinting at a heavy–tailed distribution, though the bandwidth is capped. By clustering the nodes and then selecting the neighbors from each of the cluster, we make sure that low–bandwidth nodes are connected to atleast one high bandwidth node. We show in Chapter 4 that this approach provides high connectivity even for fairly low degree.

3.2.2 Maintenance

Maintaining the clustering model graph means handling the churn operations in such a way that the graph properties are maintained. When a node leaves the network, we delete all the links that are connected to that leaving node and also remove that node from its cluster. We then update the cluster ratios, which helps in maintaining the graph. By keeping track of the cluster ratios, we can determine whether or not the graph is disconnected. Over all of our experiments, we did not find any disconnected graphs, even with 25,000 churn operations. Thus, no additional measures were taken in managing the leaving nodes. If needed, then we could assign a lower limit on the number of current edges the node has as compared to the estimated ideal number of edges and then add some of the deleted links. For adding links, we would follow the same process as when adding a node. A node joining the network follows the process as explained above in the construction of the clustering model graph. After all the edges of the new node are connected, then based on the estimated edges and each cluster's average number of edges, we can determine to which cluster the new node should be added. The closest number between the clusters' average number of edges to the node's estimated edges will determine the cluster. Later the cluster ratio of all clusters are updated.

CHAPTER 4

EXPERIMENTS

In this chapter, we first discuss the details of our simulations of both the expander graph and the clustering model graph topologies. We then report the anonymity and performance results of both topologies.

4.1 Simulation design

For both of our topology simulations, we start with a set of 1500 nodes. Each node has a bandwidth value that is randomly assigned from a set of sample Tor bandwidth values. The Tor bandwidths are extracted from the Tor descriptor file, which is downloaded by a Tor client from the directory server and which contains all the information about the Tor relays. We assign each node an ID called the King ID, which corresponds to a node from King dataset. The King dataset is a collection of pairwise latency values of 1700 nodes [6], and assigning a King ID to a node is used to calculate latency values between nodes. The latency is measured by the round trip time (RTT)— the total time taken by a data packet to travel from the source to the destination and back.

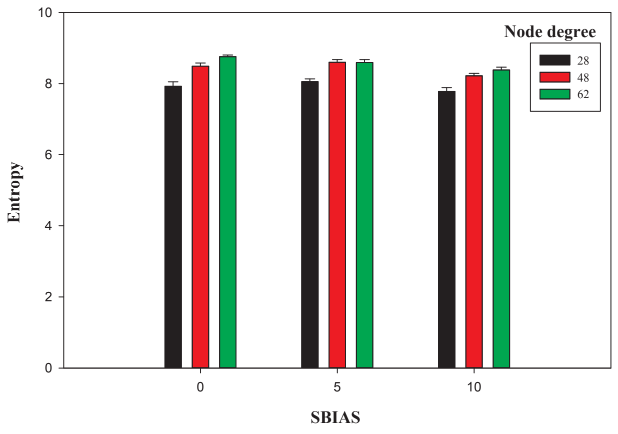
From the full set of 1500 nodes, 750 initial nodes are picked at random, and we construct the topology is constructed for the expander graph or the clustering model graph as described in Sections 3.1.1 and 3.2.1, respectively. The other 750 nodes are held in a *reserve bucket*, which is used in simulating the maintenance process.

The churn operations, in which nodes join and leave are implemented by adding and removing nodes from the topology. We add a node by picking a node randomly from the reserve bucket, adding it to the topology, and then applying the maintenance procedures described in Sections 3.1.2 and 3.2.2. Similarly, when a node leaves, it is placed back into the reserve bucket and the maintenance procedure handles the graph changes. In our experiments, we have evaluated for between 100 to 25,000 churn operations. 25,000 churn operations is a lot of turnover for real–world Tor.

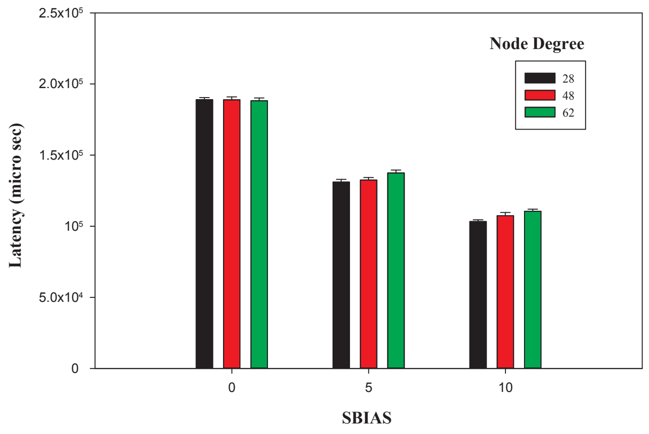
For measuring the anonymity of our topologies, we use the *entropy* metric described by Danezis [21]. Entropy is calculated based on the distribution probability for a packet entering through one node to exit through each of the other nodes. The highest entropy would be when a packet entering through one node has equal probability to exit through any other node. If the probabilities are unequal, in case of some nodes are not reachable in the maximum number of hops, then the entropy will be reduced. Hence the higher the entropy, better the connectivity. Better connectivity leads to better mixing of packets in the network, which in turn improves the anonymity. To measure the performance of our topologies, we use the average latency over possible circuits in the graph. Lower latency means better performance for the user.

In our evaluation process, we use the term *SBIAS* to refer to the value of s in Equation 2.1 used to tune the amount of bias towards lower latency links. The greater the *SBIAS* value, the more it biases towards low–latency links, which means better performance. If *SBIAS* is 0, then there is no bias, which is equivalent to selecting nodes uniformly at random.

In our default model for the expander graph, we set the SBIAS to 5, the average node degree to 48, and the number of hops to two. For the clustering model, the default model has SBIAS of 5, average node degree of 48, five clusters, and two hops in a circuit. We vary these parameters from the default model for different parts of our evaluation. In all of our graphs, all data points indicate the mean over 20 runs, and error bars indicate the standard error.



(a)



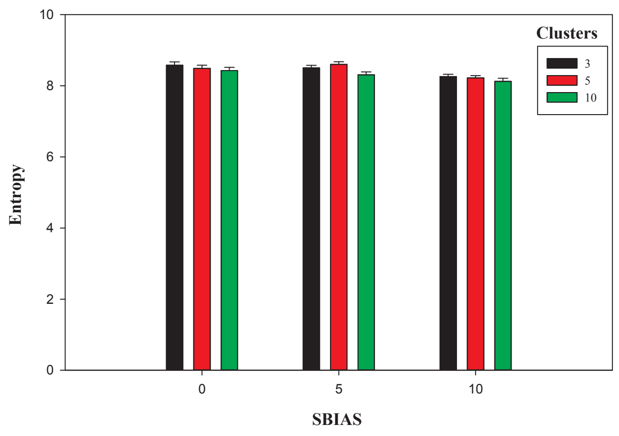
(b)

Figure 4.1. Clustering model topology performance with various node degrees and latency bias values: (a) entropy and (b) latency.

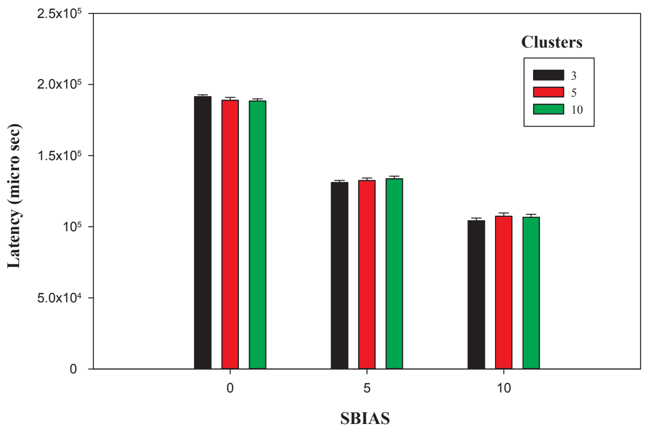
4.2 Clustering model topology performance

From our experiments, we found that the minimum entropy value was 8.9114 which 0.5% less the average entropy value. This small percentage difference shows that the clustering of nodes based on bandwidth and connecting to other nodes based on these clusters, makes sure that even the low–bandwidth nodes can reach all the nodes in the network with less number of hops. Hence, we choose to cluster the nodes rather than directly applying probability distribution function on all the nodes during the construction of the topology.

We evaluate the anonymity and the performance of the clustering model topology by varying three parameters: latency bias (SBIAS), average node degree, and



(a)



(b)

Figure 4.2. Clustering model topology performance with various number of clusters and latency bias values: (a) entropy and (b) latency.

number of clusters. We calculate the entropy value and average latency of the graph for 25,000 churn operations.

From Figure 4.1(a) and Figure 4.2(a), we can see that over all of the tested values of latency bias (SBIAS) and number of clusters, the entropy value is almost the same. The maximum entropy value was 9.45. It varies slightly with the average node degree. We get 8.7 bits of entropy for an average node degree of 62 versus 7.9 bits of entropy for an average node degree of 28 for cluster size of 5, but this variation is not that significant. These results indicate that the anonymity of the clustering model graph is not greatly affected by variations in latency bias or average node degree or the number of clusters.

Figure 4.1(b) and Figure 4.2(b) show that average node degree and number of clusters do not affect the latency of circuits. As might be expected, however, the SBIAS value has a major effect on the latency of the graph. For an average node degree of 28 and cluster size of 5, the latency value for SBIAS 0 is 188.873 ms and for SBIAS 10 is 103.301 ms, which is 45.3% decrease in latency. The latency

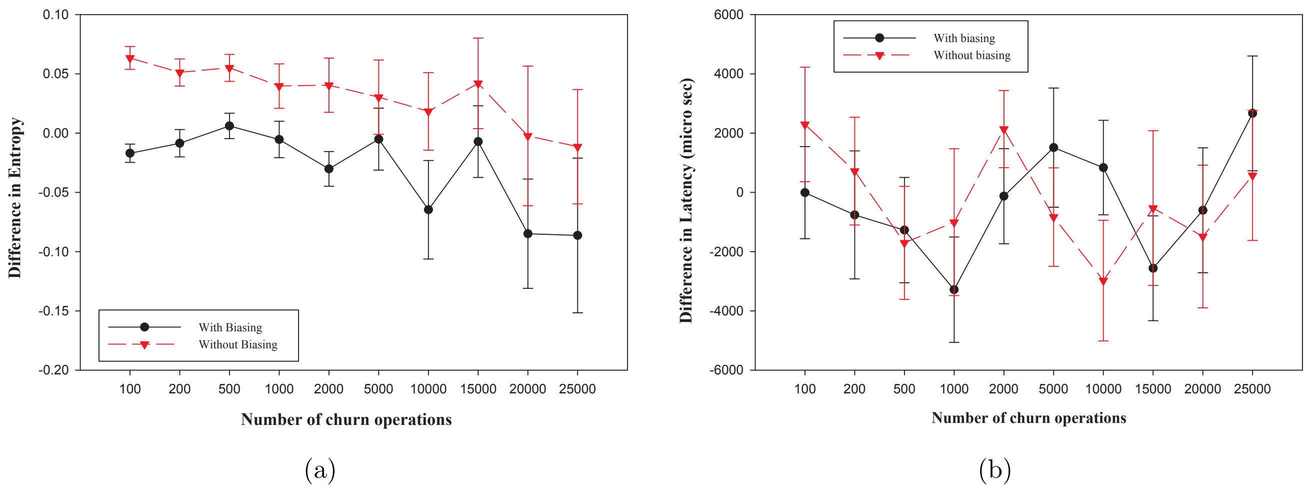


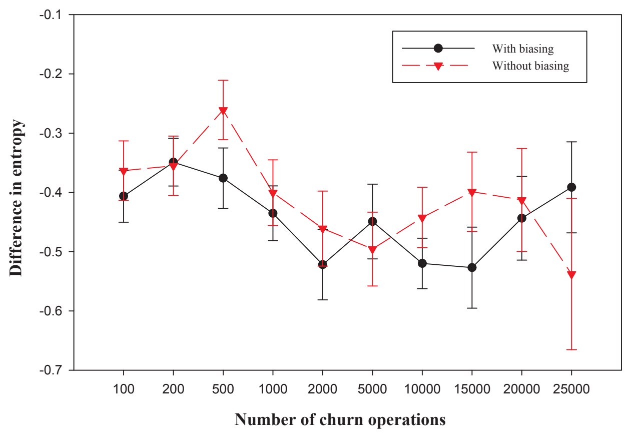
Figure 4.3. Expander graph maintenance with and without latency biasing: (a) entropy and (b) latency.

has significantly reduced, which means that there is a significant increase in system performance.This shows that the clustering model topology's performance can be significantly increased without affecting anonymity by biasing towards low–latency links.

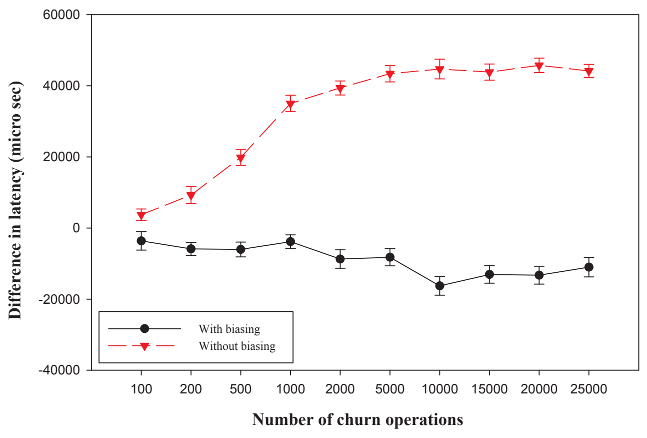
4.3 Expander graph and Clustering model graph maintenance

In this section, we evaluate the effects of latency bias on the maintenance process. We bias the construction process of both the expander graph and the clustering model graph with SBIAS of 5. Then we compare the entropy and latency values of the churn process with and without latency bias. We evaluate our maintenance process of both the graphs for 100 to 25,000 churn operations.

Figure 4.3(a) and Figure 4.3(b) show the evaluation of maintenance of expander graph. Figure 4.3(a) shows that with latency bias, there is a slight decrease in anonymity. On the other hand, in Figure 4.3(b), although there is some variation, the latency values for both with bias and without bias are almost the same, falling within



(a)

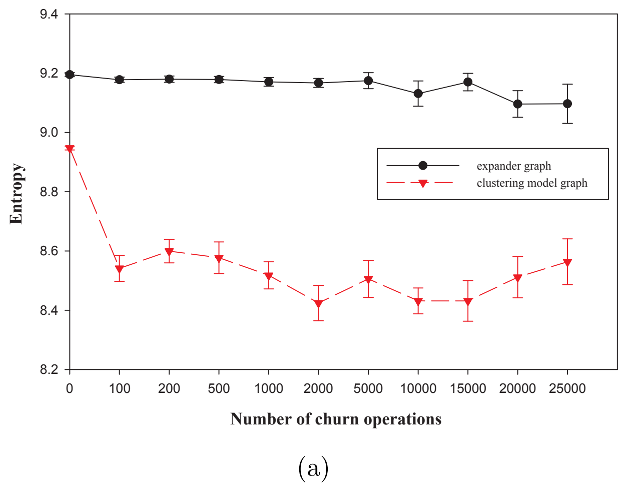


(b)

Figure 4.4. Clustering model topology maintenance with and without latency biasing: (a) entropy and (b) latency.

their respective error ranges. We conclude that with latency bias, the performance of the maintenance process remains the same but with a slight reduction in anonymity. Hence for the expander graph, it yields better results if we carry out the maintenance process without latency bias.

Figure 4.4(a) shows that there is no significant change in the entropy for clustering model topology, with latency bias or without bias. From Figure 4.4(b), however, we can see that latency bias in the maintenance process keeps latency low, whereas not biasing increases latency over time. For example, for 25,000 churn operations, the difference in latency between before and after maintenance process increases by 80.03%, if carried without biasing, whereas with biasing, the difference in latency decreases by 19.96%. Hence, for the clustering model topology, latency bias for the maintenance process yields better results than maintaining the graph without biasing. Since, entropy does not change significantly but latency decreases with biasing.



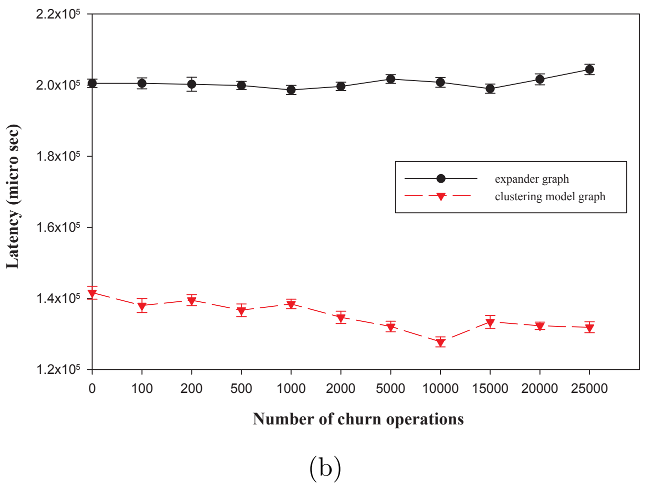
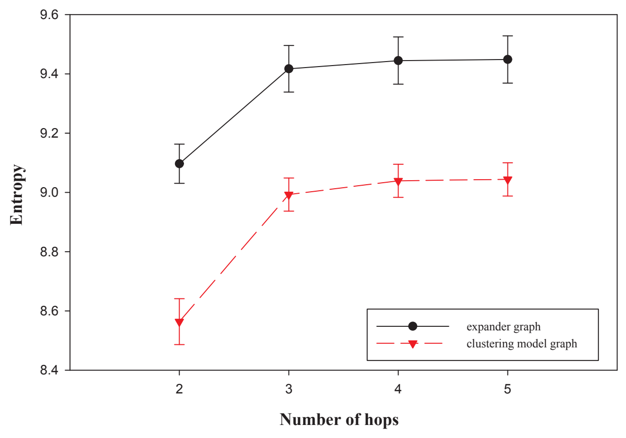


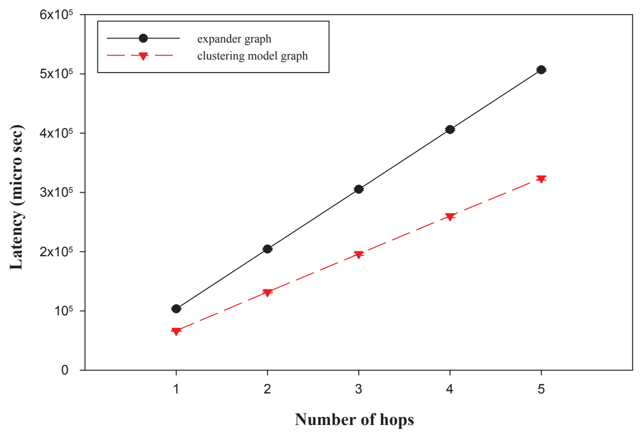
Figure 4.5. Expander graph verses Clustering model over number of churn operations: (a) entropy and (b) latency.

4.4 Expander graph verses Clustering model graph

In this section, we compare the expander graph with the clustering model. We evaluate for between 0 to 25,000 churn operations. Figure 4.5(a) shows that the expander graph's anonymity is significantly better than clustering model's anonymity. In this comparison, we get 8.947 bits of entropy for clustering model, whereas 9.195 bits of entropy for expander graph, which is 2.7% more than clustering model, without churn. With 25,000 churn operations, we get 8.564 bits of entropy for clustering model and 9.097 bits of entropy for expander graph, which is 6.2% more than clustering model. Figure 4.5(b) shows, however, that the clustering model's performance is significantly better than the expander graph's performance. For example, with no churn, we get latency of 200.492 ms for expander graph and 141.604 ms for clustering model, which is 29.37% decrease in latency compared to expander graph. For 25,000 churn operations, we get latency of 204.380 ms for expander graph and 131.860 ms for clustering model, which is 35.48% decrease in latency. We also compare these



(a)



(b)

Figure 4.6. Expander graph verses Clustering model over number of hops: (a) entropy and (b) latency.

topologies with varying number of hops in Figure 4.6(a) and Figure 4.6(b). We can see that the expander graph provides better anonymity compared to the clustering model, since the expander graph's entropy value is more than clustering model's and the clustering model topology provides better performance compared to the expander graph, since the clustering model's latency value is less than the expander graph's, with similar graph size and node degree.

CHAPTER 5

CONCLUSION AND FUTURE WORK

In this thesis, we examine the construction and the maintenance of two restricted edge network topologies for low–latency anonymity system, an expander graph topology and a novel clustering model topology. We construct expander graphs with constant degree and a bias towards low–latency edges. This not only increases system performance but also limits the throughput fingerprinting attack. The clustering model topology considers both latency between nodes and node bandwidth during construction, and this provides us to utilize the available bandwidth and faster access to this node bandwidth. We conducted simulation experiments on both topologies to evaluate their performance and anonymity

From our experiments, we conclude that the clustering model's performance can be significantly increased without affecting anonymity by biasing the construction and maintenance process towards low–latency links. In terms of maintenance of graphs, expander graph maintenance should be carried out without latency bias and the clustering model graph maintenance should be carried out with latency bias for better performance. We also saw that, for comparable graphs, the expander graph had significantly better anonymity while the clustering model had significantly better performance.

We now describe some possible future work. Our proposed topologies should to be tested on real–world network or at least real–world–like simulations such as ExperimenTor [22] or the PlanetLab–based testbed created by Chowdhuri [23]. These tests can help us evaluate the structure and properties of our proposed topologies in real–world environment.

In Tor, the first node chosen during the path selection process is from a set of nodes called *guard nodes*. These nodes are the most reliable and trust–worthy nodes in the network. Considering guard nodes during building topologies could be one more area to explore. We can also explore how our approach integrates with proposed congestion control mechanisms [24].

REFERENCES

[1] R. Dingledine, N. Mathewson, and P. Syverson, "Tor: The second–generation onion router," in *Proceedings of the USENIX Security Symposium*, 2004.

[2] (2012) Tor metrics — number of users. [Online]. Available: https://metrics.torproject.org/users.html

[3] (2012) Tor metrics — number of relays. [Online]. Available: https://metrics.torproject.org/network.html

[4] P. Mittal, A. Khurshid, J. Juen, M. Caesar, and N. Borisov, "Stealthy traffic analysis of low–latency anonymous communication using throughput fingerprinting," in *Proceedings of the ACM Conference on Computer and Communications Security (CCS)*, 2011.

[5] N. Mallesh and M. Wright, "Poster: Shaping network topology for privacy and performance," in *Proceedings of the ACM Conference on Computer and Communications Security (CCS)*, 2011.

[6] "King dataset," 2005. [Online]. Available: http://pdos.csail.mit.edu/p2psim/kingdata/

[7] K. Bauer, J. Juen, N. Borisov, D. Grunwald, D. Sicker, and D. Mccoy, "On the optimal path length for Tor," in *Proceedings of the Hot Topics in Privacy Enhancing Technologies Symposium (HotPETS)*, 2010.

[8] T.-W. Ngan, R. Dingledine, and D. S. Wallach, "Building incentives into Tor," in *Proceedings of the International Conference*, 2010.

[9] R. Dingledine and N. Mathewson. (2012) Tor path specification. [Online]. Available: https://git.torproject.org/checkout/tor/master/doc/spec/path-spec.txt

[10] 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 Electronic Society (WPES)*, 2007.

[11] 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 System Security Symposium (NDSS)*, 2008.

[12] M. Sherr, M. Blaze, and B. T. Loo, "Scalable link–based relay selection for anonymous routing," in *Proceedings of the Privacy Enhancing Technologies Symposium (PETS)*, 2009.

[13] M. Akhoondi, C. Yu, and H. V. Madhyastha, "LASTor: A low–latency AS–aware Tor client," in *Proceedings of the IEEE Symposium on Security and Privacy (S&P)*, 2012.

[14] M. AlSabah, K. Bauer, T. Elahi, and I. Goldberg, "The path less travelled: Overcoming Tor's bottlenecks with multipaths," University of Waterloo, Tech. Rep., 2011.

[15] H.-C. Hsiao, T. H.-J. Kim, A. Perrig, A. Yamada, S. C. Nelson, M. Gruteser, and W. Meng, "LAP: Lightweight anonymity and privacy," in *Proceedings of the IEEE Symposium on Security and Privacy (S&P)*, 2012.

[16] S. Rhea, D. Geels, T. Roscoe, and J. Kubiatowicz, "Handling churn in a DHT," in *Proceedings of the USENIX Annual Technical Conference (ATC)*, 2004.

[17] P. B. Godfrey, S. Shenker, and I. Stoica, "Minimizing churn in distributed systems," in *Proceedings of the Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications (SIGCOMM)*, 2006.

[18] C. Law and K.-Y. Siu, "Distributed construction of random expander networks," in *Proceedings of the IEEE International Conference on Computer Communications (INFOCOM)*, 2003.

[19] S. Hoory, N. Linial, A. Wigderson, and A. Overview, “Expander graphs and their applications,” 2006.

[20] G. A. Wilkin and X. Huang, “K–means clustering algorithms: Implementation and comparison,” in *Proceedings of the International Multi–Symposiums on Computer and Computational Sciences (IMSCCS)*, 2007.

[21] G. Danezis, “Mix–networks with restricted routes,” in *Proceedings of the Privacy Enhancing Technologies Workshop (PET)*, 2003.

[22] K. Bauer, M. Sherr, D. McCoy, and D. Grunwald, “ExperimenTor: a testbed for safe and realistic Tor experimentation,” in *Proceedings of the Workshop on Cyber Security Experimentation and Test (CSET)*, 2011.

[23] S. D. Chowdhuri, “Measurements of a latency–biased expander topology in the Tor anonymity system,” Master’s thesis, The University of Texas at Arlington, 2012.

[24] T. Wang, K. Bauer, C. Forero, and I. Goldberg, “Congestion–aware path selection for Tor,” in *Proceedings of the International Conference in Financial Cryptography and Data Security (FC)*, 2012.

BIOGRAPHICAL STATEMENT

Harsha Doreswamy was born in Bangalore, India in 1988. He received his B.E. (Computer Science and Engineering) degree from the BMS College of Engineering, Visvesvaraya Technological University, India in 2009. From 2009 to 2010, he was with Core Objects, India as an Associate Trainee Engineer. He has been a part of ISec, the Information Security Lab, from 2011. From May 2012, he has been working at Copper labs, Irving, TX as a Mobile application developer.