Congestion-aware Path Selection for Tor*

Tao Wang, Kevin Bauer, Clara Forero, and Ian Goldberg

Cheriton School of Computer Science
University of Waterloo
{t55wang,k4bauer,ciforero,iang}@cs.uwaterloo.ca

Abstract. Tor, an anonymity network formed by volunteer nodes, uses the estimated bandwidth of the nodes as a central feature of its path selection algorithm. The current load on nodes is not considered in this algorithm, however, and we observe that some nodes persist in being under-utilized or congested. This can degrade the network's performance, discourage Tor adoption, and consequently reduce the size of Tor's anonymity set. In an effort to reduce congestion and improve load balancing, we propose a congestion-aware path selection algorithm. Using latency as an indicator of congestion, clients use opportunistic and lightweight active measurements to evaluate the congestion state of nodes, and reject nodes that appear congested. Through experiments conducted on the live Tor network, we verify our hypothesis that clients can infer congestion using latency and show that congestion-aware path selection can improve performance.

1 Introduction

Tor is an anonymity network that preserves clients' online privacy [6]. Today, it serves hundreds of thousands of clients on a daily basis [13]. Despite its popularity, Tor suffers from a variety of performance problems that result in high and variable delays for clients [7]. These delays are a strong disincentive to use Tor, reducing the size of the network's user base and ultimately harming Tor users' anonymity [5]. One reason why Tor is slow is due to the challenges of balancing its dynamic traffic load over the network's available bandwidth. In this work, we propose a new approach to load balancing that can reduce congestion, improve performance, and consequently encourage wider Tor adoption.

Path selection in Tor. The current path selection algorithm selects nodes based on the bandwidth of the nodes (adjusted by the current distribution of bandwidth in the network among entry guards, exits and other nodes), giving a higher probability of being chosen to nodes with higher bandwidth. It also takes into account a number of constraints designed to promote network diversity. However, peer-to-peer file sharing users, while discouraged from using Tor, may still do so and consume a significant portion of the available bandwidth [15]. Even though the number of such users is likely small, when these bulk downloaders

* An extended version of this paper is available [22].

use nodes with insufficient bandwidth, they may affect the performance of other clients using the nodes by introducing high delays due to congestion.

Latency as a congestion signal. Congestion occurs at the node level either when a node reaches its bandwidth rate limit configured in Tor, or when a node's connection to the Internet is congested. When a node is congested, outgoing cells must wait in the node's output queue. We find that this *node latency* is sometimes significantly larger than the *link latency*, which is dominated by the propagation delay between two nodes. Delays that do not originate from propagation effects have been found to be quite common [3]; they have also been found to be large [18]. From measurements and analysis of the live Tor network, we find that Tor's token bucket rate limiting implementation often contributes to congestion delays of up to one second per node. These delays are detrimental to interactive web browsing users, who are the most common type of Tor user [15].

Congestion-aware path selection. To reduce congestion and improve Tor's load balancing, we introduce *node latency* as a new metric to be used when selecting nodes to form a circuit. Our approach uses a combination of lightweight active and opportunistic methods to obtain this information. Clients measure the overall latency of their circuits and use an inference technique to extract the component latencies due to congestion for each individual node along the circuit. Live experiments indicate that a typical client's circuit latency can be reduced by up to 40% if congestion information is taken into account during path selection. We also argue that the security and anonymity implications of our scheme are minimal.

Contributions. This paper contributes the following:

1. We identify latency as a measure of node congestion and characterize how congestion varies across different types of nodes. We describe ways to observe and isolate this node congestion from other sources of delay (such as propagation delay) with lightweight tests.
2. We design and evaluate a latency inference technique that attributes congestion-related latencies to constituent nodes along a measured circuit.
3. We extend Tor's path selection algorithm to avoid congested relays. Our approach has low overhead, can be incrementally deployed, needs no additional infrastructure, and our live evaluation shows that it improves performance.

2 Tor Background

Tor is the third-generation onion routing design providing source and destination anonymity for TCP traffic. A client wanting to connect to an Internet destination through Tor first contacts a *directory server* to obtain the list of Tor nodes. Next, the client constructs a *circuit* of three *Tor routers* (or nodes) and forwards traffic through the circuit to a desired Internet destination using a layered encryption scheme based on onion routing [10]. To balance the traffic load across the routers' bandwidth, clients select routers in proportion to their bandwidth capacities. To mitigate the predecessor attack [23], the first router on the circuit (called an "entry guard") is selected among nodes with high stability and bandwidth.

Clients choose precisely three entry guards to use for all circuits and new entry guards are selected every 30 to 60 days. The last router (called an "exit router") is chosen to allow delivery of the client's traffic to the destination. All data is transmitted through Tor in fixed-size 512-byte units called *cells*. More details about Tor's design can be found in its design document [6] and its protocol specification [4].

3 Related Work

Tor requires a good path selection algorithm to effectively distribute its traffic load across its nodes. Currently, Tor uses an algorithm that chooses routers in proportion to their bandwidth capacities. Different criteria have been proposed as possible factors in the path selection algorithm, such as autonomous system awareness [8] and application awareness [20]. In this paper, we describe a modification to Tor's existing path selection algorithm to incorporate congestion information, which improves load balancing.

Using latency as a path selection criterion has been investigated by Sherr et al. [19]. In their paper, a case is made for link-based path selection, which uses link-based properties (e.g., latency, jitter, loss). Panchenko and Renner [17] propose using round-trip time as a link-based measure to choose paths. They give a technique to obtain round-trip time and roughly analyze the increase in performance by using this criterion. In this paper, however, we look into considering latency as a *node-based* property instead of a link-based property. Link-based latency includes propagation delay, so only using link-based latency as a measure may bias path selection against circuits with nodes that are geographically far apart or on diverse networks.

Latency in Tor has also been considered from other perspectives. Hopper et al. [12] looked into how network latency can be used to deanonymize clients. Evans et al. [9] investigate using long paths to congest routers, thus revealing the identities of those connected to the router due to the change in round-trip time. Since our congestion-informed path selection approach allows clients to detect congested routers, our proposal may be a defense against such attacks; we do not, however, focus on defense mechanisms in this paper, but rather on improving Tor's performance.

Lastly, in contrast to proposals that seek to reduce congestion by redesigning Tor's congestion control mechanisms [1, 18], our work is focused solely on identifying and avoiding congested routers.

4 Latency Measurement and Congestion Inference

We next present a technique for inferring node-level congestion using circuit measurements. In this section, we describe our latency model and our approach to measuring latency, and present a technique for identifying congestion-related delays.

4.1 Latency Model

We next define a latency model for nodes. Our latency measurements on the Tor network suggest that latency measurements on a node can be cleanly divided into a *non-congested component* and *congestion time*. When a node is not congested, the latency can be attributed to propagation delays, which are nearly constant. Non-congested measurements can therefore be defined as measurements that are very close to the minimum of all measurements on the same node. For many nodes, this accounts for most of the data. When a node is congested, an amount of congestion time is added to the round-trip time before it can reach the client. This amount of time is frequently much larger than the non-congested measurements.

t_{min}	the minimum round-trip time
t_c	the congestion time
t	the round-trip time
γ	a smoothing constant

Table 1. Node-based latency model

In Table 1 we define terms with respect to a node. t_{min} is the minimum round-trip time for all measurements of round-trip time t of a node. It is a constant, assuming all measurements are done from the same client; the chief component of t_{min} is the propagation delay. We define the congestion time $t_c = t - t_{min}$. By removing t_{min} from the round-trip time, we isolate the congestion time. γ is a small smoothing constant added to the measurements to allow for quick reactions to transient congestion, as detailed further in Section 4.4. Thus, the actual congestion time is $t_c = t - t_{min} + \gamma$.

4.2 Measuring the Latency

We next discuss how circuit-level latency is measured by the client. This measurement should fulfill the following criteria:

1. It should be lightweight. There should be little burden on the network even if all of Tor's estimated 300,000 clients use this scheme simultaneously.
2. It should be indistinguishable from non-measurement traffic. Otherwise, it may be possible for malicious routers to influence the measurements.
3. It should exclude the destination server's latency. We want the measurement to consider only the delays within the Tor network, as delays at the destination server may be experienced regardless of whether Tor is used.

To satisfy these criteria, measurements of a circuit can be done in two ways: we can *actively* probe the circuit, or we can perform measurements *opportunistically* so as not to create a burden on Tor.

Active probing. One way to measure the round-trip time is to tell the exit node to connect to *localhost*, which the exit node will refuse to. This scheme, used by Panchenko et al. [17], works by forcing the exit node to return an error message to the client, so the client obtains the round-trip time to the exit node. However, a potential disadvantage is that a malicious exit node can identify the measurement probes and attempt to influence the results.

In our experiments, we use a technique that is conceptually similar: we use circuit build cells to measure the circuit latency. To extend the circuit to the

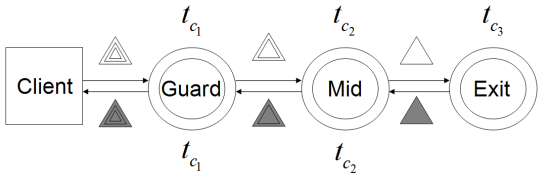


Fig. 1. A breakdown of congestion in testing. The test packet (colorless triangle) is sent to the exit node and a response packet (colored triangle) is returned without going through any destination server.

final exit router, the client sends a circuit `EXTEND` cell through the entry guard and the middle router. The middle router sends a `CREATE` cell to the exit router, which after performing public-key cryptography replies with a `CREATED` cell back through the circuit to the client. The time spent performing public-key cryptography can be considered a small constant, which will later be factored out of the latency measurement.

Opportunistic probing. If only active probing is used, our scheme might add too much measurement traffic into the Tor network, particularly if all clients were to perform such measurements frequently. Thus, we also use an opportunistic approach that leverages Tor's end-to-end control cells as the measurement apparatus. The stream-level and circuit-level `SENDME` cells are sent end-to-end in response to every 50 and 100 `DATA` cells, respectively. In addition, `BEGIN` and `CONNECTED` cells are sent whenever a new exit TCP stream is established, which for web browsing clients can happen several times per web page visited. As long as the client is using the circuit, we can obtain a number of measurements without any additional burden on the Tor network.

Note that if we want the exit node to immediately send a message back without spending time contacting a server, then the measurement is slightly skewed towards the first two nodes. To be precise, the message has to travel through each link among the client and the nodes twice, and it has to wait in the queue (if any) of the first two nodes twice, but it only needs to wait in the queue of the exit node once (see Figure 1).

Overhead. The opportunistic measurements have no overhead, as they leverage existing end-to-end control cells. However, it might be desirable to augment the opportunistic measurements with additional active measurements, at some communication cost. We can obtain one congestion time entry for each member of a circuit by sending just one cell (512 bytes). Suppose the client actively probes each circuit it builds 5 times over 10 minutes. This will add an average of 5 B/s of traffic to each node. If 300,000 users use this scheme together, they will add a total of 4.5 MB/s of traffic to Tor. This is currently around 0.5% of the total bandwidth offered by all Tor nodes, so our scheme will only add a small load to the Tor network. As will be seen in Section 4.4, a small number of measurements can be effective in detecting and avoiding congested circuits; the other measurements needed can be done opportunistically.

4.3 Isolating Circuit Congestion

When we obtain a measurement on the circuit, we want to highlight the congestion times t_{c_1}, t_{c_2}, t_{c_3} for each node along the circuit. First, it is necessary to separate the circuit's propagation delay from the delay due to congestion. We next describe this process.

For one round-trip of the entire circuit, the time T can be dissected this way:

$$T = T_{min} + (T_c - \gamma)$$
$$T_c = 2t_{c_1} + 2t_{c_2} + t_{c_3}$$

where T_{min} is an estimate of the circuit's end-to-end propagation delay and T_c is the circuit's delay due to congestion (γ is a small constant described in Section 4.4). The difference between T_{min} and T_c is that T_{min} should be constant for the same circuit, while T_c varies depending on the extent of the circuit's congestion. In addition, T_c only includes the last node once as in our tests, as our probes do not exit through the final node. In our tests, we find that the congestion term T_c is sometimes zero, but it is often non-zero.

For each measurement of T in this circuit, we save it in a list $\{T_1, T_2, ..., T_k\}$, and after all measurements of the circuit are done, we take the lowest measurement, and let this be T_{min}. Note that the number of measurements taken per circuit should be large to ensure that T_{min} converges to the circuit's actual end-to-end propagation delay.[1] Through experimental analysis, we find that T_{min} can be correctly determined within an error of 0.05 s with 80% probability by using only five measurements—in the case that T_{min} is not correctly identified, the circuit being considered is likely to be heavily congested.

The i^{th} measurement of congestion time ($0 \leq i < k$) is given by:

$$T_{c_i} = T_i - T_{min} + \gamma$$

In Figure 1, we summarize how a single end-to-end circuit round-trip time measurement is conducted and where the congestion occurs.

4.4 Attributing Circuit Congestion to Nodes

Now that we have isolated the delay due to congestion from the circuit's total delay, we need to attribute the congestion delay to the circuit's constituent nodes. Each client maintains a congestion list of all known relays paired with a number L of congestion times for each relay. This list is updated as new measurements are taken. Consider a three-hop circuit. Suppose the estimated congestion times of nodes r_1, r_2, r_3 in this circuit are respectively $t_{c_1}, t_{c_2}, t_{c_3}$. The entry guard is r_1, the middle router is r_2, and the exit router is r_3. Next, suppose the round-trip time taken for some cell to return across this circuit is T; then the total circuit's congestion time is $T_c = T - T_{min} + \gamma$. For r_1 and r_2, we assign the following congestion time:

[1] T_{min} can also be intelligently estimated using other methods. For instance, the King method [11] can be used to approximate the pairwise network latency between any two Tor nodes without probing either of the routers directly.

$$t_{c_i} \leftarrow T_c \cdot \frac{2t_{c_i}}{2t_{c_1} + 2t_{c_2} + t_{c_3}}$$

Here $i = 1$ for node r_1 and $i = 2$ for node r_2. For r_3, we assign the following congestion time:

$$t_{c_3} \leftarrow T_c \cdot \frac{t_{c_3}}{2t_{c_1} + 2t_{c_2} + t_{c_3}}$$

Details. A technical issue emerges when a node becomes congested after a long period of being non-congested. In this scenario, the estimated congestion time would be very close to zero and the algorithm would not respond fast enough to assign a high congestion time to this node. This is where the term γ comes into play. By ensuring that the minimum estimated congestion time is at least γ, we can guarantee that even nodes without a history of congestion will not be immune to blame when congestion occurs in a circuit with such a node. We empirically find $\gamma = 0.02\,\mathrm{s}$ to be a good value; this is not large enough to cover the differential between congested and non-congested nodes, yet it ensures that convergence will not take too long.

When a new estimated congestion time has been assigned to a node, the node's mean estimated congestion time should be updated. We maintain a list of congestion time measurements for each node, L; when this amount of data has been recorded, we push out old data whenever new data comes in. If L is chosen to be large, then the client's preference for a node will not change as quickly, and vice versa.[2]

5 Techniques for Mitigating Congestion

Congestion can be either short term (e.g., a file sharer decides to use a certain node for their activities) or long term (e.g., a node's bandwidth is consistently overestimated or its flags and exit policy are too attractive). For short-term congestion, we want to provide an instant response to switch to other circuits. For long-term congestion, we propose a path selection algorithm that takes congestion time into account.

5.1 Instant Response

We provide two ways in which clients can perform instant on-the-spot responses to high congestion times in a circuit.

Choosing the best pre-built circuits. Tor automatically attempts to maintain several pre-built circuits so that circuit construction time will not affect the user's experience. Two circuits are built that are capable of exiting to each port used in the past hour (a circuit can count for multiple ports). Only one of those circuits is chosen as the next circuit when the user's circuit times out or breaks. A reasonable scheme, therefore, is to test all of those circuits before choosing

[2] Alternatively, an exponentially weighted moving average (EWMA) of congestion delay would reduce the space necessary to store historical congestion data.

which to use. As stated above, those tests can be done quickly and with minimal overhead using active probing. We suggest that five active probing measurements per pre-built circuit is sufficient to choose the best, as we observe in our experiments (in Section 6). Since the circuits are pre-built, these measurements will not cause the client any further delay.

Switching to another circuit. While using the circuit, a client may continue to measure the circuit and obtain congestion times. This can be done with no overhead to the Tor network by opportunistically leveraging Tor's stream-level and circuit-level `SENDME` cells, or the stream `BEGIN` and `CONNECTED` cell pairs (as described in Section 4.2). This gives us the round-trip time T, from which we can follow the procedure given in Section 4.3 to isolate the nodes' congestion time. If the estimated congestion time is large, the client should stop using this circuit and choose another circuit instead.

Comparison. Tor currently takes into account the circuit build time adaptively and drops circuits that take too long to build [2]. This approach, however, cannot identify circuits that may become congested after they are constructed, and the client will not learn to avoid attempting to build circuits over nodes that are consistently congested. Furthermore, propagation delays are included in the circuit building time, which is undesirable. Our two schemes improve upon Tor's circuit building timeout mechanism.

5.2 Path Selection

In addition to an instant response, we also want a long-term response where clients can selectively *avoid* certain nodes if they often receive poor service from these nodes. This can be helpful when there are nodes with poorly estimated bandwidth, when bulk downloaders customize their clients to use only specific relays, or when there are other unexpected load balancing issues that have not been resolved. Our congestion-aware path selection works as follows.

Each client will keep a list of all routers, each of which will be recorded with a list of their measured congestion times. The list of measured values is of size L; when new data comes in, old data is pushed out.

Node selection. Our scheme is designed to be built atop the current path selection algorithm in this way: when we wish to extend a circuit by one node, we pick a few nodes from the list according to the original scheme (e.g., 10 nodes), and then choose one of them in negative correlation to their estimated congestion times. Estimated congestion times should be obtained by leveraging both the active and opportunistic measurements done for the instant response schemes. Suppose that node r's estimated congestion time is t_{c_r}. We define a base constant $\alpha > 0$, and use it to obtain the probability of selecting the node r for a circuit:

$$P(C_r) \propto \frac{1}{\alpha + t_{c_r}}$$

where C_r is the event of node r being chosen. α is a constant that prevents very low congestion nodes from dominating the path selection algorithm.

The effect of this scheme on the user's experience and the Tor network itself depends in part on the choice of the constant α. A smaller α will impact the load balancing more as nodes with less estimated congestion become more likely to be chosen.

Advantages. Our approach is simple and efficient. Furthermore, this scheme requires no further infrastructure to support, and it is incrementally deployable in the sense that any client who chooses to use this scheme can immediately do so. The long term path selection scheme adds no further overhead on the network over the instant response scheme, as it can share the few measurements used to support the instant response scheme.

6 Experiments

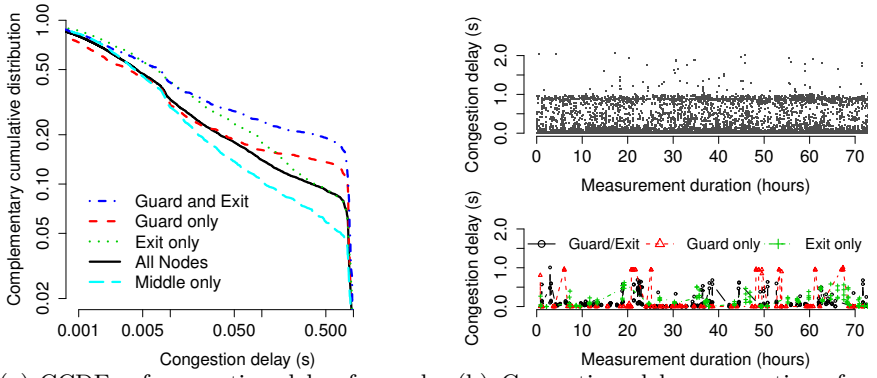
We designed a number of experiments that aim to validate our assertions about latency and congestion in Tor. For all experiments, we use the Tor control protocol to instrument Tor. We use the final pair of circuit construction cells to measure the round-trip time of a circuit (as described in Section 4.2). In the remainder of this section, we present experiments and results that show that congestion is a property of Tor nodes, explore the relationship between a node's consensus bandwidth and its estimated congestion, and evaluate the performance improvements offered by our congestion-aware router selection proposal.

6.1 Node Congestion

We first seek to demonstrate that congestion is a property of Tor routers. For 72 hours in August 2011, we collected round-trip time data for all Tor routers that can be used on a circuit by measuring the time to construct one-hop circuits. For each node, we subtracted the node's minimum measurement (e.g., the propagation delay) to isolate the congestion delays t_c.

Figure 2(a) shows the distribution of congestion delays for entry guards, exits, guard/exits, and middle-only nodes. The median congestion delay is minimal (3–5 ms) across all node types; however, the tails of the distributions tell a different story. For the most congested ten percent of the measurements, nodes marked as both guard and exit experience congestion delays greater than 866 ms, and guard-only nodes have at least 836 ms of congestion delay. Exit-only and middle-only nodes tend to be the least congested. Guard nodes may be the most congested because the stability and bandwidth criteria for the guard flag is too high. Relaxing the requirements for the guard flag would enable some middle-only nodes to become guards, reducing congestion among guards.

Figure 2(b) shows congestion delays over the duration of our measurements for all routers (top) and for three representative high-bandwidth (10 MiB/s) routers (bottom). We note that these delays tend to be low. However, there exists noticeable variability regardless of a node's flags or bandwidth, and many of the delays are close to one second (Figure 2(a) also illustrates these one second delays where the CCDF lines become vertical). These one second delays are the



(a) CCDFs of congestion delay for nodes with different flags (log-log scale)

(b) Congestion delays over time for all routers measured (top) and the variability for three high-bandwidth routers (bottom)

Fig. 2. Analysis of congestion delays

result of Tor's token bucket rate limiting with a once-per-second token refilling policy (see the extended version of this manuscript [22] for more details).[3] These one-second delays indicate that nodes are being asked to forward more traffic than they are configured to handle, resulting in congestion. Thus, we conclude that congestion is a property of the Tor router itself, motivating the need for clients to consider congestion when selecting nodes for a circuit.

To investigate the possible relationship between a node's bandwidth and its congestion, we analyze the nodes' consensus bandwidth as reported by Tor's directory servers. We observe no correlation between a node's bandwidth and congestion (the Pearson's r value between log of the bandwidth and the congestion time is -0.00842).[4] This implies that considering only a node's bandwidth during path selection may not be sufficient to achieve optimal load balancing.

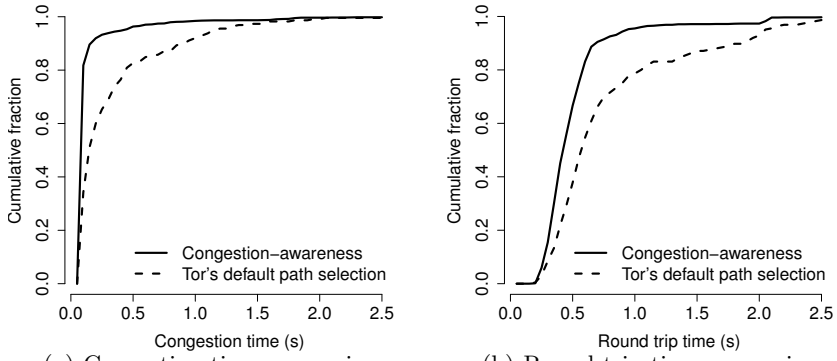
6.2 Performance Improvements of Our Schemes

We next present experiments that seek to quantify clients' latency improvements when using our scheme. Experiments are performed on both the instant response and long-term path selection components.

In these experiments, an unmodified Tor client used the current path selection algorithm in Tor. At the same time, a modified client uses the instant response components of our scheme (from Section 5.1) to determine which circuit it should use. The original client builds 225 circuits and measures each one precisely 30 times to obtain round-trip times. The modified client determines which circuits it should use based on the same data.

[3] Increasing the frequency with which the tokens are refilled may reduce or eliminate these one second delays. This design change is currently being discussed [21].

[4] Dhungel et al. report no significant correlation between bandwidth and *overall* delay [3].



(a) Congestion time comparison (b) Round-trip time comparison

Fig. 3. Improvements in congestion time and round-trip time for instant response

Choosing the best pre-built circuits. We first tested how much of an improvement we would see if the client simply tested each circuit five times when building them preemptively and chose the one with the lowest congestion. For simplicity we assumed that the client always had three circuits to choose from. The original client tested each of its circuits 30 times, and took the mean of the congestion times as its experience with the circuit. The modified client chose the best among every three circuits to use by only looking at the first five measurements; after choosing the best out of three, all 30 measurements of that circuit are revealed to the modified client and it is taken as its experience of the circuit. Without the scheme, the mean circuit congestion time was about 0.276 s. With the scheme, it was about 0.119 s. We find that this large improvement was because most circuits were non-congested, except a minority where the congestion time was very high. Those circuits also clearly exhibited congestion in the first five measurements. This experiment demonstrates that just a few measurements are needed to effectively identify congested circuits.

Switching to another circuit. We next tested how much of an improvement we would get if the client switches to a better circuit when the current one becomes too congested. This time both the original client and the modified client can see all measurements. The modified client dropped a circuit if the last five measurements had a mean of more than 0.5 s of congestion; 73 of the 225 circuits were eventually dropped. This sufficed to improve the mean congestion experienced from 0.276 s to 0.137 s.

Finally, we combined the two instant response schemes. 75 of the 225 circuits were chosen using the first scheme, and later 11 of the 75 circuits chosen were eventually dropped using the second scheme. We achieved a mean congestion time of 0.077 s, compared to the original 0.276 s. The total round-trip time was reduced from a mean of 0.737 s to 0.448 s. Figure 3(a) shows the distribution of congestion times for the client when it used our improvements compared to the original selection scheme, and Figure 3(b) shows the distribution of round-trip

time for the same comparison. Note that such a reduction in congestion delays would result in a faster time-to-first-byte for interactive clients (e.g., web browsing clients), which positively affects the users' perceived quality of service [16].

Overhead. One may worry that these schemes will add too much overhead because they drop existing circuits and build new ones. With the first scheme we are not dropping any circuits. With the second scheme, in our experiment we found that we would need to build about 26% more circuits, which is a relatively modest increase.[5]

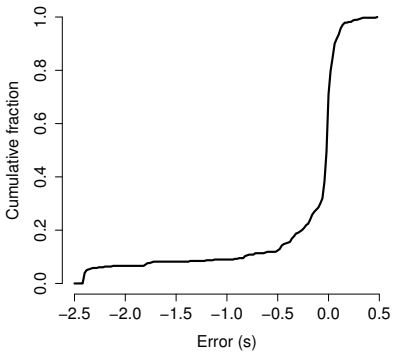


Fig. 4. Distribution of errors when learning individual node congestion over a large number of trials

Long-term path selection. We evaluate the long-term path selection algorithm as follows. We ran a client that builds many circuits over the entire Tor network using the original path selection scheme. In total 13,458 circuits were built, for which the round-trip time was obtained 5 times each. One-third of the circuit build times were used as testing data; the rest were used in training the client to learn the estimated congestion times for each relay. By using the long-term path selection scheme, we observed a decrease in the mean congestion time for this experiment from 0.41 s to 0.37 s over the testing data. The improvement is not as large as in the instant response schemes, because the long-term path selection scheme tackles more persistent factors which adversely affect node performance rather than short-term bursts of congestion.

The long-term path selection scheme offers an improvement nonetheless because it is capable of deducing the congestion time of individual nodes while only measuring the congestion times of random circuits, allowing it to choose uncongested nodes. We performed a total of 379 trials where we compared deduced congestion (by building three-hop circuits) to directly measured congestion (by building one-hop circuits). Figure 4 shows the distribution of errors. We found that nearly 90% of the errors were within -0.5 s to 0.5 s, and 65% of the errors were within -0.1 s to 0.1 s. The scheme very rarely overestimated node congestion, but sometimes underestimated it, as shown by the large number of negative errors. The mean error was therefore -0.2 s. This may be because high congestion is somewhat random in nature, so the scheme is less accurate in predicting the extent of a node's congestion while only given a previous record.

7 Anonymity and Security Implications

We consider if our schemes may be open to attacks which cause a loss of anonymity for the client. To be specific, we consider sender anonymity, which is achieved if a message and its origin cannot be traced. It is known that sender

[5] Circuit building cells are much rarer than data transfer cells; further, the Tor Project is working to decrease the computation required for circuit building by a factor of four [14].

anonymity is lost in Tor if the entry guard and the exit node in a circuit are both compromised. The possibility of such depends on the attacker's control of the network. We therefore focus on the possibility of an attacker increasing their control of the network through our schemes.

We consider a particular attack called the *smearing attack*. The attacker first uses all of his available bandwidth to deploy malicious nodes. These malicious nodes attempt to give the appearance of congestion by artificially delaying cells. If a client measures a circuit containing both innocuous and malicious nodes, the innocuous nodes will be "smeared" with high estimated congestion times. The clients are then less likely to choose these nodes under the long-term path selection scheme. After a certain amount of time, these malicious nodes will be estimated to have a very high congestion as well, so the smearing becomes less effective. Once a malicious node becomes rarely selected, it is taken down, and a new one is created in order to maintain the attack. This attack is continued until all innocuous nodes can no longer be smeared further (this is bounded by the amount of bandwidth available to the attacker). After all nodes are maximally smeared, the attacker can stop the attack and enjoy a larger control of the network for a while, as his nodes will now seem more attractive.[6]

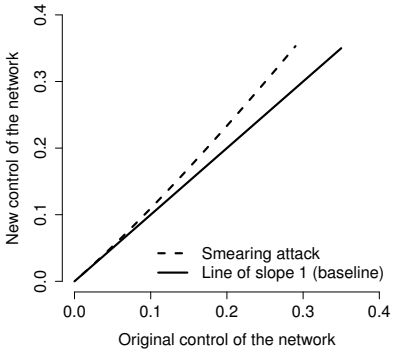


Fig. 5. An estimate of the how much control an attacker can gain through the smearing attack. We chose $t_{max} = 5000\,\text{ms}$, $t_c = 500\,\text{ms}$, $L = 20$, $C = 30$.

A parameter of the attack is C, which indicates for how long each malicious node will attempt to smear other nodes before being replaced. If $C = 5$, for example, the attacker will attempt to keep each malicious node up for as long as it takes to smear other nodes five times for each client measuring the nodes, then take it down and replace it with another node. We take t_c as the mean performance of the nodes (including the malicious node) and t_{max} as the maximum time the client performing the latency measurement will wait for before timing out. The estimation is done by running a simulation with the simplifying assumption that all nodes can be selected in all positions.

Figure 5 shows how much bandwidth the malicious nodes must possess in order to affect the measurements of the congestion time of the non-malicious nodes. The attacker can indeed smear other nodes and gain an advantage by coming up with fresh, non-smeared nodes. We also note that the advantage gained is temporary; when the adversary stops performing the attack and uses all their bandwidth to acquire control of the network, clients will start measuring the other nodes' non-smeared congestion times as well, so their estimated congestion times will slowly return to their non-smeared levels.

[6] Note that nodes are less likely to be chosen if they do not have the "stable" and "fast" flags. The stable flag is a barrier for malicious nodes, as it requires the node to demonstrate high stability before they can be effective. We neglect this barrier in the following analysis, giving more power to the attacker.

Information leakage could also cause anonymity loss. The list of latencies stored on a user's computer may compromise anonymity if divulged. If the list of latencies for all users is known to an attacker, he can perform an attack by only controlling the exit node, and using the lists to probabilistically guess who is connecting by checking the frequency of connections; this will give him some amount of information. Our scheme, however, gives no reason to directly divulge the list of latencies at any point. Furthermore, this list is updated based on client behavior and measurements, which the attacker cannot easily observe or manipulate without controlling a substantial portion of the network.

While we recognize that our scheme introduces a small risk due to the smearing attack, we believe that reducing congestion would result in increased resilience to attacks that utilize congestion to identify the set of routers used by a client [9]. Due to space constraints, a full investigation is future work.

8 Conclusion and Future Work

Many different metrics for path selection in Tor have been proposed, some of which consider the use of latency. However, previous work treats latency as a property of a link and focuses on the delays that occur primarily due to propagation. We assume a different approach: we identify the importance of latency as an indicator of a node's congestion. To reduce congestion, improve load balancing and, ultimately, improve clients' quality of service, we propose an improved path selection algorithm based on inferred congestion information that biases path selection toward non-congested nodes. We also propose ways for clients to respond to short-term, transient congestion that improve on Tor's adaptive circuit building timeout mechanism.

Our experiments show that a single client can expect to experience up to a 40% decrease in delay when considering congestion during node selection. As future work, we plan to investigate the potential benefits and other effects when this scheme is deployed at scale through whole-network experiments.

Acknowledgements. This work was funded in part by NSERC, MITACS, and The Tor Project. We also thank Jean-Charles Grégoire, Angèle Hamel, Ryan Henry, Femi Olumofin, and Rob Smits for their valuable suggestions.

References

1. AlSabah, M., Bauer, K., Goldberg, I., Grunwald, D., McCoy, D., Savage, S., and Voelker, G. M. DefenestraTor: Throwing out windows in Tor. In *PETS* (2011).
2. Chen, F., and Perry, M. Improving Tor path selection. `https://gitweb.torproject.org/torspec.git/blob_plain/HEAD:/proposals/151-path-selection-improvements.txt`, July 2008.
3. Dhungel, P., Steiner, M., Rimac, I., Hilt, V., and Ross, K. W. Waiting for anonymity: Understanding delays in the Tor overlay. In *Peer-to-Peer Computing* (2010), IEEE, pp. 1–4.
4. Dingledine, R., and Mathewson, N. Tor Protocol Specification. `https://gitweb.torproject.org/tor.git/blob_plain/HEAD:/doc/spec/tor-spec.txt`. Accessed August 2011.

5. Dingledine, R., and Mathewson, N. Anonymity loves company: Usability and the network effect. In *WEIS* (June 2006).
6. Dingledine, R., Mathewson, N., and Syverson, P. Tor: The second-generation onion router. In *USENIX Security* (2004).
7. Dingledine, R., and Murdoch, S. Performance improvements on Tor or, why Tor is slow and what we're going to do about it. `http://www.torproject.org/press/presskit/2009-03-11-performance.pdf`, March 2009.
8. Edman, M., and Syverson, P. F. AS-awareness in Tor path selection. In *Proceedings of CCS* (2009), pp. 380–389.
9. Evans, N., Dingledine, R., and Grothoff, C. A practical congestion attack on Tor using long paths. In *Proceedings of the 18th USENIX Security Symposium* (August 2009).
10. Goldschlag, D. M., Reed, M. G., and Syverson, P. F. Hiding routing information. In *Proceedings of Information Hiding: First International Workshop* (May 1996), Springer-Verlag, LNCS 1174.
11. Gummadi, K. P., Saroiu, S., and Gribble, S. D. King: Estimating latency between arbitrary Internet end hosts. *SIGCOMM Comput. Commun. Rev. 32*, 3 (2002).
12. Hopper, N., Vasserman, E. Y., and Chan-Tin, E. How much anonymity does network latency leak? In *CCS* (2007).
13. Loesing, K. Measuring the Tor network: Evaluation of client requests to the directories. *Tor Project Technical Report* (2009).
14. Mathewson, N. New paper by Goldberg, Stebila, and Ostaoglu with proposed circuit handshake. `https://lists.torproject.org/pipermail/tor-dev/2011-May/002641.html`. Accessed June 2011.
15. McCoy, D., Bauer, K., Grunwald, D., Kohno, T., and Sicker, D. Shining light in dark places: Understanding the Tor network. In *PETS* (2008).
16. Nah, F. F.-H. A study on tolerable waiting time: How long are web users willing to wait? *Behaviour Information Technology 23*, 3 (2004), 153–163.
17. Panchenko, A., and Renner, J. Path selection metrics for performance-improved onion routing. In *Proceedings of the 2009 Ninth Annual International Symposium on Applications and the Internet* (Washington, DC, USA, 2009), IEEE Computer Society, pp. 114–120.
18. Reardon, J., and Goldberg, I. Improving Tor using a TCP-over-DTLS tunnel. In *USENIX Security* (2009).
19. Sherr, M., Blaze, M., and Loo, B. T. Scalable link-based relay selection for anonymous routing. In *PETS* (2009).
20. Sherr, M., Mao, A., Marczak, W. R., Zhou, W., Loo, B. T., and Blaze, M. A3: An Extensible Platform for Application-Aware Anonymity. In *17th Annual Network and Distributed System Security Symposium (NDSS)* (February 2010).
21. Tschorsch, F., and Scheuermann, B. Proposal 182: Credit bucket. `https://gitweb.torproject.org/torspec.git/blob_plain/HEAD:/proposals/182-creditbucket.txt`. Accessed August 2011.
22. Wang, T., Bauer, K., Forero, C., and Goldberg, I. Congestion-aware path selection for Tor. Technical Report CACR 2011-20. `http://www.cacr.math.uwaterloo.ca/techreports/2011/cacr2011-20.pdf`, December 2011.
23. Wright, M. K., Adler, M., Levine, B. N., and Shields, C. The predecessor attack: An analysis of a threat to anonymous communications systems. *ACM Trans. Inf. Syst. Secur. 7*, 4 (2004), 489–522.