# Exploring the Potential Benefits of Expanded Rate Limiting in Tor: Slow and Steady Wins the Race With Tortoise

W. Brad Moore
Georgetown University
Washington, D.C. 20057
wbm@cs.georgetown.edu

Chris Wacek
Georgetown University
Washington, D.C. 20057
cwacek@cs.georgetown.edu

Micah Sherr
Georgetown University
Washington, D.C. 20057
msherr@cs.georgetown.edu

## ABSTRACT

Tor is a volunteer-operated network of application-layer relays that enables users to communicate privately and anonymously. Unfortunately, Tor often exhibits poor performance due to congestion caused by the unbalanced ratio of clients to available relays, as well as a disproportionately high consumption of network capacity by a small fraction of filesharing users.

This paper argues the very counterintuitive notion that *slowing down traffic on Tor will increase the bandwidth capacity of the network* and consequently improve the experience of interactive web users. We introduce Tortoise, a system for rate limiting Tor at its ingress points. We demonstrate that Tortoise incurs little penalty for interactive web users, while significantly decreasing the throughput for filesharers. Our techniques provide incentives to filesharers to configure their Tor clients to also relay traffic, which in turn improves the network's overall performance. We present large-scale emulation results that indicate that interactive users will achieve a significant speedup if even a small fraction of clients opt to run relays.

## Categories and Subject Descriptors

C.2.4 [**Computer-Communication Networks**]: Distributed Systems; C.2.0 [**Computer-Communication Networks**]: General—*Security and Protection*; C.2.1 [**Network Architecture and Design**]: [Distributed Networks]

## General Terms

Performance, Anonymity, Security

## Keywords

Anonymity, Tor, Performance

## 1. INTRODUCTION

Anonymity networks such as Tor [9] allow their users to privately communicate without revealing their identities. These systems are regularly used to enable private browsing, circumvent censorship firewalls, and provide unfettered access to information. In particular, Tor enables any application that communicates using TCP to tunnel its connections through the anonymity network.[1] This flexibility permits a variety of applications – web browsers, instant messaging clients, file sharing applications, and more – to achieve some degree of anonymity. Tor does not discriminate against any particular application, and moreover, its anonymity features aggravate efforts to distinguish between applications.

A consequence of Tor's versatility is that the anonymity network's capacity is disproportionately consumed by a small subset of users who run high-bandwidth applications. By analyzing the traffic that exited their exit relay in 2008, McCoy *et al.* found that while nearly 97% of observed connections could be classified as *interactive* (e.g., web browsing), approximately 40% of anonymous traffic belonged to *non-interactive* applications [17] such as BitTorrent. By itself, this disproportionate bandwidth utilization is not problematic: Tor is a general-purpose anonymity network, and file sharing has many legitimate uses.

Unfortunately, relative to unprotected communication, Tor suffers from high-latency and low-bandwidth. The network's poor performance not only negatively impacts the applications that it services, it also likely discourages the network's use as many would-be users may be unwilling to sacrifice so much performance for increased privacy. In their performance analysis of Tor, Dingledine and Murdoch identify BitTorrent as a major cause of Tor's slowness [10].

A simplistic approach to improving Tor's performance is to disallow BitTorrent on Tor. However, such a policy arguably runs counter to Tor's philosophy and mission as an anti-censorship technology. Additionally, it is unclear how client applications can be reliably differentiated (though notably, previous work has shown that applications can be probabilistically identified by examining their traffic patterns [13]).

An alternative approach to improving Tor's performance is to increase the number of relays that forward anonymous traffic [10, 14, 22]. A recent study of the Tor network estimates that the number of clients outnumbers the number of available relays by a factor of nearly 67 [15]. Increasing the number of relays diminishes this imbalance and consequently decreases congestion in the network.

This paper adopts this latter technique and proposes an incentive scheme to increase the number of relays on the Tor network and improve the network's overall performance and capacity. Our solution, which we call Tortoise, takes the counterintuitive and seem-

[1]Applying Tor without carefully considering the application's protocol and communication characteristics risks exposing the sender's identity [4, 7]. For example, certain BitTorrent clients annotate requests with their senders' network addresses [4]; similarly, improperly configured end-hosts may reveal receivers' identities by failing to anonymize DNS resolution requests.


*ACSAC* '11 Dec. 5-9, 2011, Orlando, Florida USA

ingly contradictory approach that *slowing down Tor will help achieve speedup*. In particular, Tortoise imposes strict rate limits on individual client connections at the network's ingress points but does not limit connections originating from Tor relays. By carefully tuning these limits, interactive clients such as web browsers see little effect, while bandwidth-intensive users (for example, filesharers) experience a significant decrease in throughput. We argue that this slowdown provides incentives for filesharers to operate their own relays through which they can bypass the network's rate limiting. We posit that some fraction of filesharers will become sufficiently frustrated to operate their own relays, which will in turn serve additional traffic and reduce congestion. Moreover, the savings in bandwidth from rate limiting filesharers and other high-throughput users who do not run relays can provide additional network capacity.

Unlike recently proposed incentive and e-cash approaches that require centralized *mints* [14] or *banks* [2, 5], our solution is fully decentralized, is backwards-compatible with the existing Tor network, and may be deployed incrementally. We demonstrate the feasibility of Tortoise and investigate its ability to improve the performance of interactive web clients through emulation using ExperimenTor [3], a large-scale network emulator that executes actual Tor binaries on a virtual network. We show that the performance of interactive web browsers will significantly increase if just a small number of clients opt to run their own relays.

**Threat Model and Limitations**. Importantly, Tortoise is not robust against determined adversaries who wish to circumvent its rate limiting through Sybil-style attacks [11]. For example, an attacker can achieve high throughput by connecting to many (rate limited) relays and aggregating their bandwidths. Although existing Sybil defenses may offer some protection, we note that Tortoise does not worsen the performance of the network (relative to unmodified Tor) in the presence of such determined attackers. Rather, Tortoise *is designed to provide incentives for ordinary users* – some of whom desire high throughputs – to also operate as relays. If a small fraction of such honest users opt to relay Tor traffic, then the network will improve. More sophisticated high-bandwidth users may evade the rate limits, but as we discuss below, doing so may be more costly than behaving correctly and operating a relay.

Additionally, as with other solutions that motivate users to run relays, Tortoise inherently trades off performance for anonymity: operating a relay increases performance, but decreases sender anonymity since the initiator of high-throughput traffic is likely the operator of a Tor relay. We discuss the security implications of using Tortoise in more detail below.

We begin by reviewing the Tor network and describing its current rate limiting features.

## 2. BACKGROUND

Tor is a volunteer-operated network of approximately 2,500 application-layer routers (also called *relays* or ORs). The network provides anonymity by forwarding traffic from clients (also called *proxies* or OPs) along a bidirectional *anonymous circuit* consisting of Tor routers. To conceal the identities of the communicants, Tor encrypts messages such that each relay can discern only the identities of the previous and next hops along the anonymous circuit. By default, Tor uses three-relay hops, consisting of a guard relay, a middle relay, and an exit relay.

**Increasing the number of relays provides greater anonymity and performance**. The number and configuration of Tor relays determine the network's performance and anonymity. A large number of (honest) relays provides strong anonymity since traffic has a lower likelihood of traversing only malicious relays. (If the guard and exit relays are malicious and colluding, then the adversary can identify the sender and receiver of intercepted communication [19, 32].) In addition, an increase in the number of Tor relays improves the performance of the network by providing additional capacity, which in turn decreases congestion.

Ideally, each person who uses Tor would also run a Tor relay, contributing some bandwidth to the network's overall capacity. Unfortunately, the current ratio of end-clients to relays is estimated to be 67:1 [15], leading to significant congestion and poor performance. This imbalance can be partially explained by the multiple costs of running a Tor relay: Operating a relay taxes both the hosting computer as well as its network connection. In order to provide a benefit to the network, a router must be continuously online for weeks before Tor clients will begin to use it. Additionally, when Tor is configured to operate as an exit router, the operator's computer may appear to law enforcement officials to be accessing illegal content. Unlike the two other relays that comprise a Tor circuit, the exit relay directly accesses the server requested by the sender; if this service serves illegal content, it will appear to the outside world that the request originated at the exit relay, putting the relay's operator at substantial risk. There are currently few incentives to operating a relay, and it is reasonable to assume that most current Tor relay operators volunteer their computer and network resources for altruistic reasons.

**Rate limiting in Tor**. Tor includes rate limiting features that allow relay operators to configure how much collective bandwidth they wish to delegate for serving Tor traffic. The existing functionality does not currently support per-connection rate limits, although such features are present in alpha releases. As described below, Tortoise extends Tor's rate limiting by throttling clients' inbound traffic.

## 3. DISMISSED: FILESHARER IDENTIFICATION AND FILTERING

A seemingly plausible and straightforward method of reducing the strain on the Tor network is to filter filesharing traffic at exit relays using standard port blocking. In fact, the Tor Project recommends that users running exit routers block BitTorrent's default ports [23]. However, such filtering does little to deter determined filesharers since users of these services can trivially switch to non-standard ports.

Alternatively, exit relays could apply more advanced techniques and perform deep packet inspection (DPI) and/or traffic fingerprinting [13] on the traffic that they forward. Recall that once a user's traffic has reached the exit relay, it is no longer protected by any layers of encryption that were applied by Tor (since the exit relay must interface with the destination server as if it were the original client). Hence, exit relays could examine outgoing traffic and discard any detected BitTorrent packets.

However, applying DPI and traffic fingerprinting at exit relays suffers from several shortcomings. First, and perhaps most importantly, such strategies are antithetical to the goals of the Tor project. Tor is an anonymity network whose principal purpose is to provide its users with unfettered Internet access without the fear that their traffic is being monitored. In order to be effective, the traffic blocking schemes described above would necessarily have to violate Tor's underlying philosophy by engineering eavesdropping into the system's design. Relatedly, another of Tor's goals is to allow its users to access content that would otherwise be unavailable to them; actively blocking content is incompatible with this goal. Additionally, if Tor were to attempt to identify and limit cer-

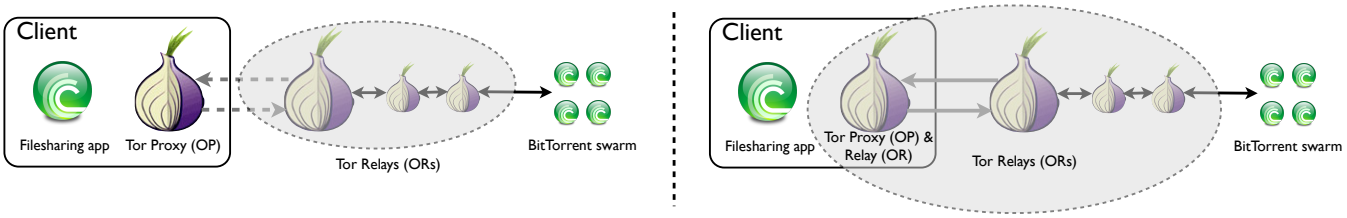


**Figure 1: Tortoise's universal rate limiting. Dashed lines indicate connections that are subject to Tortoise's universal rate limit. The shaded circle encompasses relays that comprise the Tor network. *Left:* Client operates a OP and is subject to the universal rate limit. *Right:* Client additionally operates an OR, increasing the size of the Tor network and becoming exempt from the universal rate limit.**

tain types of traffic, users generating that traffic could always shape their communication to resemble a type of traffic not easily discriminated against by Tor (for example, communication that is shaped like encrypted web traffic). Finally, performing either DPI or fingerprinting techniques imposes added complexity and increases the relays' computational costs.

At best, identifying filesharers is an arms race: detection approaches will likely be followed and countered by obfuscation techniques, *ad nauseam*. In the next section, we present Tortoise, a universal rate limiting approach that is applied to *all* communication, thereby evading this adversarial arms race.

# 4. TORTOISE

Tortoise modifies Tor's already-implemented (but not very utilized) token-bucket system to limit users' bandwidths at the network's ingress points. Our goal is to establish a *universal rate limit* that imposes a heavy throughput penalty for users who use the network for bulk transfers while not significantly degrading the experience of users who use the anonymity service for interactive web browsing.

By itself, a universal rate limit will do little to improve Tor's performance. Imposing bandwidth limits on bulk transfer users is unlikely to reduce their overall effect on the network, since their use of the Tor network already indicates their willingness to tolerate slow speeds. For instance, halving their speeds with a universal rate limit is likely only to double the time the bulk users spend on the network. In general, merely penalizing bulk transfer users is a zero-sum game.

An intuitive strategy is to apply a low universal rate limit to provide incentives to all users (both low- and high-bandwidth clients) to operate Tor relays. However, many users of the Tor network connect from totalitarian regimes where Internet access is severely limited and subject to strict censorship. Requiring these users to operate relays not only does not add significant capacity to the Tor network, such a policy may also physically endanger the operators. Instead, Tortoise is designed to place the burden on users who require large bandwidths. That is, anyone can access Tor and achieve bandwidth that is suitable for web browsing. Users who require greater bandwidths are incentivized to also offer their services as Tor relays.

Tortoise aims to improve the overall performance of the Tor network not by traffic shaping, but rather by increasing the capacity of the network by encouraging users to run routers. Tortoise achieves this goal by enforcing the universal rate limit only on Tor OPs (clients); the connections between Tor ORs (relays) are not impacted by Tortoise and are subject only to relay-specific bandwidth limits. (As described below, Tortoise requires ORs to meet certain conditions in order to be exempted from the universal rate limit.) Hence, clients who also run routers can use their routers as bridges to the Tor network, bypassing Tortoise's universal rate limit. We posit that some bulk transfer users who find their bandwidth on the Tor network severely limited will be motivated to bypass the bandwidth limits by running their own OR.

An illustration of Tortoise's universal rate limiting is presented in Figure 1. Initially (Figure 1, *Left*), a filesharing client tunnels his traffic through Tor and is subject to Tortoise's universal rate limit (indicated in the Figure with dashed lines). To achieve better performance, the client then opts to also run a Tor relay (Figure 1, *Right*). The new relay increases the size of the Tor network, which consequently decreases congestion and improves the network's overall performance.

## 4.1 Preventing Cheating

High-bandwidth users who wish to evade the universal rate limit may do so by operating their own relay. However, Tortoise should ensure that those relays are actually contributing to the performance of the Tor network *in toto*. For example, a user could attempt to game the system by running a relay only when it wants to download content at high speed, or it may operate a very low-bandwidth relay that has little effect on the network's overall capacity.

Tortoise mitigates these "cheats" by relying on status flags maintained by the Tor directory servers. To prevent a user from taking advantage of Tortoise by running a router only at times when they want increased performance, Tortoise requires that a router be listed as STABLE in directory servers; connections from all other routers are subject to the universal rate limit. We note that applying rate limits to non-STABLE routers will not significantly impact the performance of the network, since Tor's default relay selection strategy biases selection in favor of STABLE relays. In order to appear as STABLE, a router must have a mean-time-between-failures greater than that of the median of all other routers [30]. At the time of this writing, the 50th percentile of Tor routers had an uptime of approximately four days.

Additionally, to prevent rewarding a user who operates a STABLE relay that offers very little bandwidth to the Tor network, only relays that are marked as FAST in the directory servers are excluded from the universal rate limit. FAST routers are defined as those that offer at least 20KBps or have bandwidths that are in the top 87.5% of known relays [30]. Tor's default relay selection strategy also heavily biases selection towards FAST relays, and hence applying the universal rate limit to non-FAST relays will not significantly degrade the performance of the network.

In summary, relays that are marked as STABLE and FAST are exempt from the universal rate limit. Currently, these are exactly the relays that are selected by Tor's relay selection algorithms, and consequently, are the relays that forward Tor's traffic.

## 4.2 Anonymity Considerations

At first blush, it may appear that Tortoise allows eavesdroppers to distinguish between encrypted traffic that belongs to a filesharer and that which belongs to a web client by measuring the monitored connection's throughput. However, upon inspection, this tactic will be less effective than anticipated. While filesharers have the most to gain by running a router under Tortoise, the benefits are not confined to filesharers alone. The incentive to run a Tor relay applies to all users who desire faster speeds through Tor, and hence high-bandwidth traffic may belong to any user who runs a relay.

Admittedly, since the goal in rate limit selection is to avoid adversely affecting web browsing clients (see Section 5.1), it is likely that Tortoise will more adversely affect filesharers, placing greater incentives on that population. However, because there are an order of magnitude more web users than file sharers using Tor [17], distinguishing between web and filesharer traffic remains difficult. For example, even if the participation rate (i.e., the rate of users who decide to run routers as a result of Tortoise) is ten times higher for filesharers than it is for web browsing clients, the number of web browsing clients that decide to participate will still be three times that of filesharers.

As with other incentive schemes that reward relay operators with additional bandwidth [2, 5, 14], Tortoise reduces anonymity by forcing a smaller *sender anonymity set* – the set of potential senders for a given anonymous connection. With both standard Tor and Tortoise, any Internet-connected device can use the anonymity system, and hence the sender anonymity set is quite large. However, Tortoise's "differentiated services" (that is, the use of rate limited as well as non-rate limited traffic classes) permit the attacker to reason that intercepted high-bandwidth data originates from a node that is also a router. As more users run relays, the size of the anonymity set will similarly increase, as will the anonymity offered by Tortoise. Of course, any reduction in anonymity can be entirely avoided since a relay operator may always choose to not take advantage of his increased bandwidth.

# 5. EVALUATION

We evaluate Tortoise using *ExperimenTor* [3], a large-scale network emulator that uses ModelNet [31] to model a network topology. Our ExperimenTor deployment consists of two machines: a *edge node* that runs all Tor relays, directory servers, clients, and web servers, and a *core* that emulates the actual network. The edge node has a 12-core 2.8GHz Xeon processor and runs Linux 2.6.35 and Tor version 0.2.1.28. Our default configuration consists of five authoritative directory servers, 15 relays, 900 clients, and 40 web/file servers. Our setup does not utilize guard relays, all relays are potential exit relays, and directory servers do not relay traffic. Tor was extended to include Tortoise's universal rate limiting extensions.

The core machine has a 2.8GHz Pentium D processor and runs FreeBSD 6.3. We assign bandwidth capacities to the Tor relays by randomly sampling bandwidths that were advertised in the live Tor network's directory servers in May 2011. Since Tor's default relay selection strategy is biased in favor of relays that offer the most bandwidth [9], we select from only the highest 300 bandwidths listed by the Tor directory server in order to achieve speeds that closely resemble those of the real Tor network. (We note that in the live Tor network, the first 300 relays advertise 86.5% of the network's total capacity.) The core node simulates a topology in which latencies between all nodes are less than 10ms.

We assign client bandwidths using two classes of client connections: *residential* and *institutional*. Residential connections are asymmetric – download bandwidth is greater than upload bandwidth (as is the case, for example, with most broadband services). Institutional connections are those where bandwidth is symmetric. We assign residential connections to 90% of the clients. The capacity distribution for client connections was selected by cross referencing 2011 country of origin data for Tor clients (gathered by the Tor Project) with average upload and download speeds for those countries as measured by `NetIndex` [21]. For example, since 20% of Tor client connections originate in the United States, we assign 20% of links the mean U.S. upload and download speeds for 2011. Both residential and institutional clients receive capacities from this distribution, but clients designated as institutional are assigned symmetric connections using the download speed for both upstream and downstream traffic.

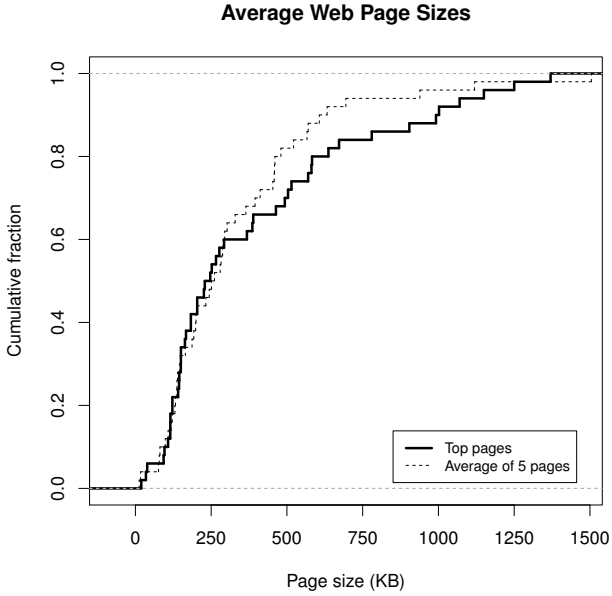


**Figure 2: Average page sizes for the top 60 websites as reported by Alexa.**

We instantiate two types of Tor users: **web clients** periodically request web pages. To model a realistic workload, we determined the sizes of the frontpages of the top 60 websites as reported by Alexa [1] (see Figure 2). Sampling from this distribution, web clients request pages of sizes 106KB, 150KB, 238KB, 496KB, and 992KB, each with equal probability. These values represent the 10th, 30th, 50th, 70th, and 90th percentile of web page sizes, respectively. Additionally, web clients are configured to pause an average of 11 seconds, the median "think time" between requests as measured in a study of web browsing behavior [12], and will take between 9 and 11 minute breaks after 15 minute browsing sessions. In contrast, **bulk clients** model high-bandwidth users and continually download files whose sizes are chosen uniformly at random from 1MB, 2MB, 3MB, 4MB, and 5MB. Keeping with the percentages observed by McCoy *et al.* [17], we select 3% (30) of our clients to be bulk; the remaining 870 are configured to be web clients.

We took precautions to ensure that our physical emulation setup did not introduce any processing or network effects. The edge and the core machines are connected via a dedicated 1Gbps link; the total aggregate network throughput did not exceed 1Gbps, and our experiments are not bound by our ExperimenTor configuration's bandwidth or CPU resources.

## 5.1 Rate Limit Selection

The universal rate limit should be sufficiently large to not significantly impact the experience of web users. We can compute a reasonable minimum rate limit by considering both the amount of time that users are willing to wait for a web page to be retrieved, as well as the distribution of web page sizes.

## 5.2 Effects of Rate Limiting

At the other extreme, the rate limit should not be so high as to allow high-bandwidth users to consume an unfair share of the network's resources. That is, the universal rate limit should be chosen to degrade the performance of high bandwidth users, and consequently provide incentives for them to operate their own relays.

Figure 2 shows the cumulative distribution of the sizes of web pages from the top 60 Alexa [1] web sites. Reported page sizes include embedded images and Javascript but exclude Flash and other content that would not be included in a web page loaded through Tor. The Figure plots both the front page web sizes as well as the average of four additional randomly selected pages on each site.

We select our rate limits based on the expected load time of current web pages. Using the (somewhat dated) heuristic that users tolerate web page load times of eight seconds or less [33], we select a 200KBps limit, which covers all pages from the Alexa dataset. Additionally, we also evaluate Tortoise when using a more stringent 100KBps limit, which covers approximately 80% of the top 60 Alexa websites.

Figure 3 shows the effects of rate limiting at 100KBps (left) and 200KBps (right) on web and bulk clients, when no clients opt to run relays. Web clients are largely unaffected by the rate limit: even with the more severe 100KBps limit, the mean transfer speed drops only 15%, from 40KBps to 34KBps. Bulk transfer clients, on the other hand, are severely affected: even with the less severe 200KBps limit, the mean transfer speed seen by these clients drops 31%, from 70KBps to 49KBps. As we show below, if even a small percentage of the effected clients is sufficiently motivated to operate relays, the overall performance and capacity of the network will improve.

**Computational overhead**. Tortoise requires that each relay maintain additional token buckets to perform the added rate limiting. Consequently, Tortoise incurs a computational cost relative to unmodified Tor. However, as shown in Figure 4, Tortoise's computational overhead is fairly modest. The Figure plots the CPU utilization of the edge node that executes all 960 Tor instances. Although the edge node's CPU usage is a coarse-grained measure, comparing the processing cost when using unmodified Tor and Tortoise provides a useful estimate of the latter's overhead. With unmodified Tor, the median CPU utilization on Experimen-Tor is 24.7%; with Tortoise, the usage increases slightly to 26.4%.

## 5.3 Performance Improvements

**Benefits of adding more relays**. We begin by examining the effects of adding more relays to a Tor network. Figure 6 plots the cumulative distribution of client bandwidths when no rate limiting is applied and relays are added to the network. Here, we model an idealized setting in which clients altruistically decide to become relays. It is important to note that such clients generally contribute *less* to the network than the original 15 ORs since (i) the former are generally on slower (e.g., consumer broadband) connections than the dedicated ORs and (ii) as clients, they also use their capacities for their own purposes.

As expected, the more relays that are added, the greater the bandwith available to nodes on the network. In the base case with 15 ORs (1.7% of all nodes), the mean client bandwidth is 41KBps. To determine the effect of clients who also opt to become ORs, we examine two scenarios: in the first case, 10% (2) of bulk transfer clients and 2% (18) of web browsing clients opt to run routers, adding an additional 20 routers to the network. In the second case, these percentages double for an additional 40 routers on the network. In the first configuration, the mean bandwidth experienced by clients increases by 27% to 52KBps. With 40 additional relays, the mean bandwidth grows by 66% to 68KBps.

Though the results of having client Tor instances also run as relays are (unsurprisingly) positive, it is unlikely that, in practice, so many clients will suddenly and unselfishly choose to become routers. *The challenge is to motivate clients to also act as relays*, despite the costs involved. Applying the universal rate limit supplies such motivation, as bulk clients witness their average bandwidths decrease by 31% and 30% with respective limits of 200KBps and 100KBps. (In contrast, web clients incur only 15% and 17% decreases with the two rate limits.) Given the option of achieving greater speeds by running relays, we anticipate that at least a small fraction of bulk (and potentially web) clients will also run ORs. We investigate the advantages of behaving as both an OP and OR below.

**Benefits to clients that become relays**. Operating an OR exempts a client from Tortoise's universal rate limit. As illustrated in Figure 5, clients that choose to additionally run as an OR will experience better bandwidth than those in the same network that do not. Here, Tortoise utilizes a 100KBps universal rate limit; the low and high adoption rates correspond to the scenarios described above in which 10% (2%) and 20% (4%) of bulk (web) clients opt to run ORs. Although all clients who run ORs experience increased bandwidth, bulk relays gain the greatest benefit from choosing to run a relay since they are most affected by the universal rate limit. Under a 100KBps limit, the mean bandwidth experienced by all Tor clients who chose to run routers increased by 38% and 78% when 10% and 20% of bulk clients became routers, respectively. Most of this improvement was realized by the bulk transfer clients.

**Benefits of adding more relays with rate limiting**. Figure 7 examines the impact of running Tortoise on network performance. The grey line denotes the average bandwidth seen on a network running unmodified instances of Tor. The solid black line shows the bandwidth of a network using Tortoise. If no clients opt to run as a relay, adopting a 200KBps or 100KBps rate limit will lead to decreased capacity seen by the network. However, given the motivation to run as a relay (see Figure 5), we argue that at least some small fraction of clients will decide to additionally operate as ORs. If only 10% of bulk and 2% of web clients (2.22% of all clients) elect to run relays, the mean bandwidth increases slightly by 3.1% and 0.7% with 100KBps and 200KBps limits, respectively. The addition of another 20 client relays (4.44% of all clients) produces more significant gains, providing 36% and 31% greater mean bandwidths under the 100KBps and 200KBps rate limits respectively, than the unmodified network. As more clients choose to become relays (and hence gain better performance themselves), the network achieves greater speedups.

**Handling increasing capacity**. Since the "freed" capacity that is not being used due to rate limiting may be applied to serve other clients, a network running Tortoise will be better able to handle additional clients than a network which uses unmodified Tor. Figure 8 depicts the change in network capacity when the sizes of both a Tortoise network and an unmodified Tor network are increased by 20%. (Here, we conservatively assume that no additional nodes run ORs.) The average bandwidth across the network decreases from 45KBps to 30KBps (35%) for the unmodified network, and from 50KBps to 36KBps (28%) on a network running Tortoise with a

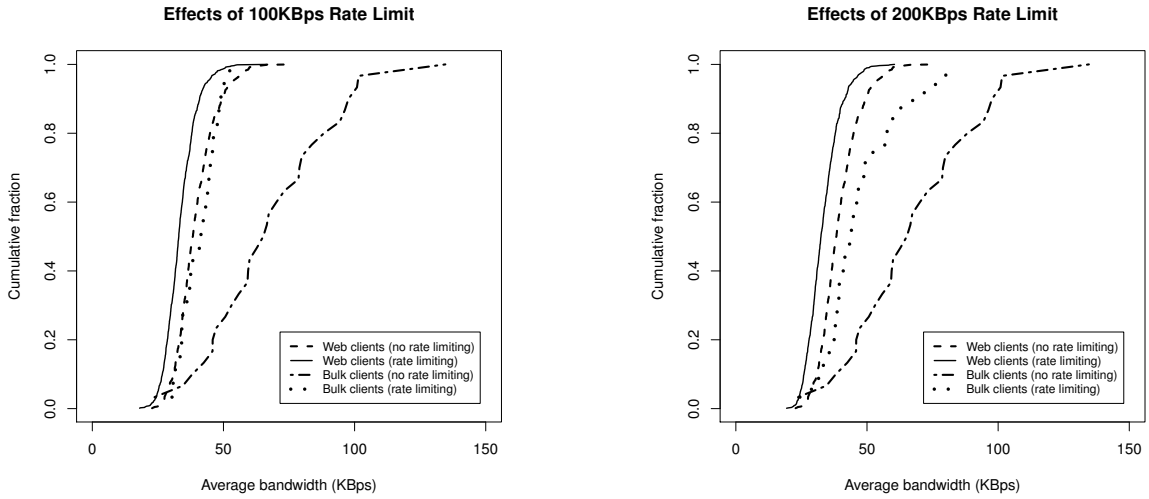


**Figure 3: The effects of 100KBPs (left) and 200KBps (right) universal rate limits on web and bulk clients.**

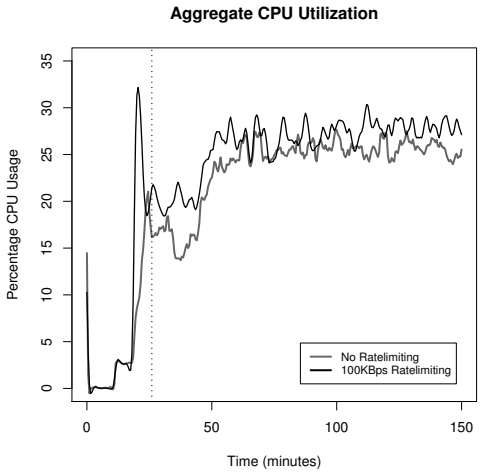


**Figure 4: Aggregate CPU usage of the entire network. The vertical dotted line represents the approximate time at which all nodes have joined the network.**

100KBps limit. Similarly, with a 200KBps limit, Tortoise's average bandwidth is reduced from 57KBps to 40KBps (30%) when extra clients are added. In other words, the Tortoise-based network is better able to tolerate a sudden and large increase in (non-relay) clients; when the number of clients increases by 20%, the percentage decreases in performance for Tortoise are 20% and 14% less than that of regular Tor when 100KBps and 200KBps rate limits are respectively applied.

**Effect of partial participation**. To evaluate Tortoise's efficacy when not all relays apply the universal rate limit, we simulated a Tor network in which only 50% of the relays rate limit the clients. Figure 10 shows the cumulative distribution of bandwidths when 0, 20 ("low client adoption"), and 40 ("high client adoption") clients opt to also run relays.

This network's performance was similar to a network with 100% Tortoise adoption in cases where significant numbers of clients chose to run routers: with 20 client routers, this network had a mean bandwidth of 44KBps, vs. 43KBps for the 100% Tortoise network; with 40 routers, these numbers were 54KBps and 56KBps, respectively. The only case where a network with 50% Tortoise adoption showed any significant difference from a 100% Tortoise network was when no clients chose to run routers, in which case the network with 50% adoption exhibited a mean bandwidth of 42KBps, while the 100% Tortoise network averaged 35KBps. While the above data might seem to imply that a network with less than 100% adoption of Tortoise performs better than one with full adoption, it is important to note that a network with less than 100% adoption is less likely to lead to clients choosing to run routers. Figure 9 illustrates the fact that in a network with 50% adoption of Tortoise, there is less motivation to become a router simply because there is less difference between speeds achieved by clients running relays and those not running relays.

## 5.4 Summary

The above emulation results indicate that Tortoise has the potential to significantly increase Tor's performance. As highlighted above, the challenge of our technique is selecting a universal rate limit that properly motivates clients to operate as relays. If no additional clients serve as relays, then applying a rate limiting trivially slows down the network. However, our emulation results suggest that the addition of even a few relays improves the network's overall performance and capacity, even if the vast majority of clients are subjected to rate limits. Assuming that at least a small number of clients are sufficiently motivated to operate as relays, Tortoise's performance gains can be felt even with partial use of Tortoise rate limits. For instance, the mean client bandwidth increased by 6.5% over standard Tor even in the conservative case when the universal rate limit was 100KBps, only 50% of the relays applied the rate limit, and just 2.2% of clients opted to run relays.

# 6. DISCUSSION AND LIMITATIONS

**Tortoise's relay exemption policy is currently incompatible with bridges**. Tortoise may cause problems for *bridges* – unlisted Tor relays that allow users to connect to the Tor network in locations where the public relays are inaccessible. Although bridges may forward traffic from multiple clients, Tortoise will subject them

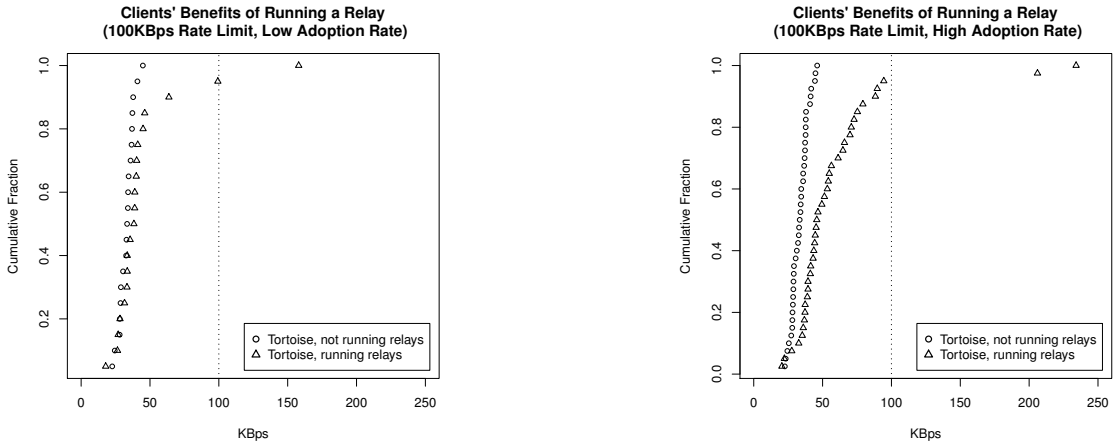


**Figure 5: Bandwidth improvements seen by clients that choose to run routers, in a network running** Tortoise**. The experiments represented in the two graphs differ only in the number of clients who choose to run routers.**

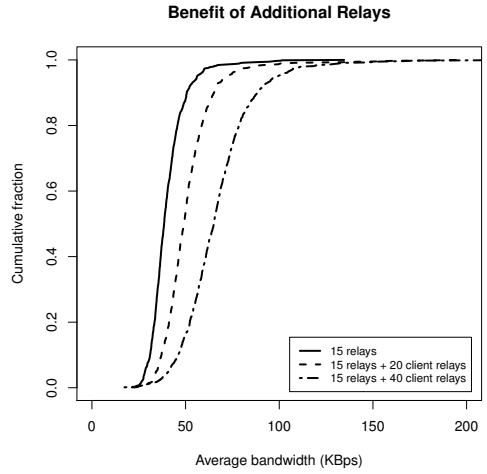


**Figure 6: Average bandwidth seen by all clients in an unmodified network, and unmodified networks with additional client relays.**

to the universal rate limit since the bridges are not listed in Tor directories.

We propose two methods for adapting Tortoise to better support bridges. For both solutions, we consider a relay that wishes to determine whether an upstream connection is from a client or a bridge: clients should be subject to rate limits, while bridges should be exempt, but only if they are actually forwarding traffic.

One potential approach is for the relay to use a separate bridge to attempt to create a Tor circuit through the node in question. If the circuit is successfully extended, then the node must be a bridge. If the circuit cannot be extended, the node can safely be assumed to be a client. Importantly, since a bridge (known only to the relay) is used to extend the circuit, the node in question cannot behave as a bridge only to the relay (i.e., to be falsely detected as a bridge in order to evade the rate limit). The difficulty with this approach is that if the node is a bridge, its bridge listening port will not be immediately known to the relay. However, the relay can attempt to extend a circuit (via the bridge) using commonly chosen ports.[2] Additionally, Tor would have to be slightly modified to allow bridges to extend circuits to other bridges.

Alternatively, bridges could reveal their identities to independently-chosen trusted relays. These relays would be aware of the bridges' status and will not subject them to the rate limit. The challenge with this approach is that since clients select the anonymous path, bridges would additionally have to recommend a second hop (the trusted relay) that is not subject to rate limiting. We leave the study of these and other potential strategies for supporting Tor bridges as a future research direction.

**Tortoise relies on accurate directories**. Only the relays that are marked STABLE and FAST in the Tor directory are exempt from Tortoise's universal rate limit. A client may attempt to cheat the system by advertising a relay that is neither FAST nor STABLE, but is marked as such by the directory. Since the directories periodically poll the relays to measure their failure rates, a dishonest client cannot easily fake a STABLE rating. It can, however, report a false (high) bandwidth to cause a directory to rate it as FAST.

Several techniques have been recently proposed to avoid the reliance on relays' self-reported capacities. For example, Snader and Borisov introduce an *opportunistic measurement* system in which relays report the observed bandwidth of their peers. Directory servers then advertise the median of these measurements [28]. Perry has suggested an alternative technique in which *measurement authorities* perform empirical measurements of relays' bandwidths [24].

Additionally, a node that wishes to gain exemption from the universal rate limit may contribute only the minimum amount of bandwidth such that they receive the FAST flag. The FAST tag is applied to the fastest 87.5% of routers (as of this writing, this requires a bandwidth of only 15KBps). To better ensure that clients who also run relays are meaningfully contributing to the network, a potential refinement to Tortoise's approach is to additionally apply a rate limit *on relays*. Here, relays' bandwidths would be capped based on the amount of bandwidth that they provide to the network. Hence, the improvement in client bandwidth will be proportional to the amount of bandwidth that the client's OR serves the network.

**Tortoise is susceptible to Sybil-style attacks**. Tortoise is vul-

[2]Many bridges choose to run on port 443, since HTTPS traffic is often allowed through firewalls, and like HTTPS, Tor uses SSL/TLS to provide confidentiality.

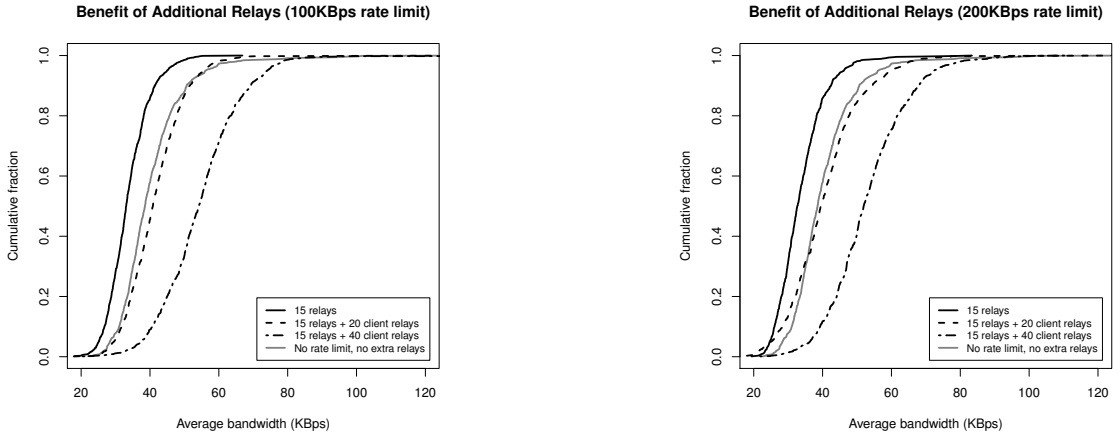


**Figure 7: Average bandwidth rates when additional relays join the network when a 100KBps (left) or 200KBps (right) universal rate limit is imposed.**

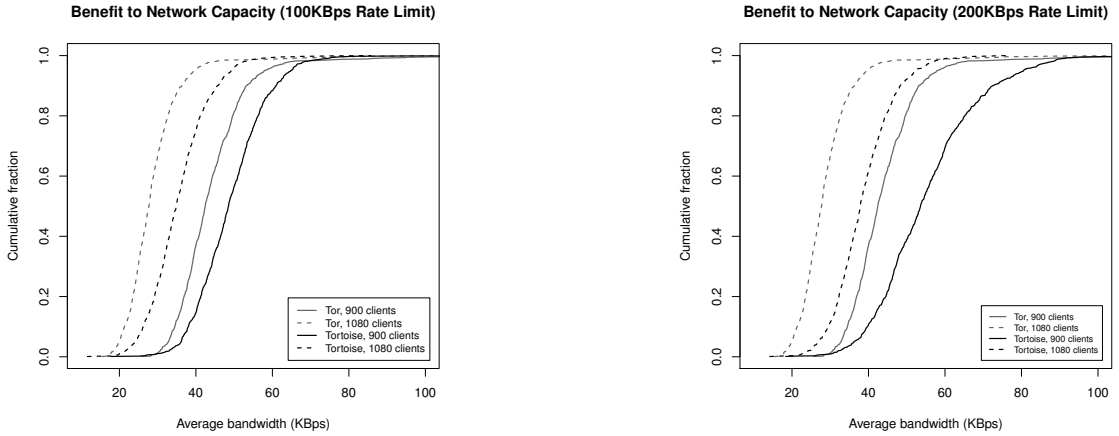


**Figure 8: Increased capacity with 100KBps (left) and 200KBps (right) universal rate limits.**

nerable to Sybil-style [11] attacks in which a client achieves high bandwidth by multiplexing connections over many Tor circuits. Although each circuit may be individually rate limited, the combined bandwidth may allow the client to surpass the universal rate limit.

Existing Sybil countermeasures may be applied to mitigate such attacks against Tortoise. In particular, guard relays may require clients to complete periodic *cryptopuzzles* [18] in order to continuously forward their traffic. Solving occasional cryptopuzzles will add only a modest burden to standard clients, but could be very computationally expensive for misbehaving clients that connect through many guard nodes. Here, the goal is not to disallow a client from establishing a large number of connections to the Tor network, but rather to shift the incentives to better motivate compliance with Tortoise's envisioned model. That is, applying periodic cryptopuzzles may make it more cost effective to operate an OR rather than to evade the system's universal rate limit.

# 7. RELATED WORK

There are a number of existing approaches that aim to increase Tor's performance. We categorize and outline some of these techniques below.

**Prioritizing techniques**. Tang and Goldberg recently proposed replacing Tor's round-robin circuit scheduler with one that considers a circuit's recent usage [29]. Using the exponential weighted moving average (EWMA), their technique favors bursty circuits over constantly busy circuits, and consequently lowers the latency of more interactive applications. Their scheduler has been integrated into Tor.

Unlike their approach in which interactive traffic is given precedence over busy clients by rearranging the schedule in favor of the former, Tortoise's traffic shaping is done by active throttling. The two techniques are orthogonal, and EWMA and Tortoise can be simultaneously applied to increase performance.

**Improved multiplexing**. Reardon and Goldberg note that Tor's TCP multiplexing techniques significantly contribute to network latency. They suggest tunneling TCP connections of DTLS (Datagram Transport Layer Security) packets to alleviate the effects of interference among multiplexed Tor circuits [25]. Similarly, Mathewson has investigated using SCTP (Stream Control Transmission Protocol) for Tor multiplexing [16]. In contrast to these approaches, Tortoise imposes strict rate limits on all non-contributing clients. Applying Tortoise in conjunction with the above multiplexing strate-

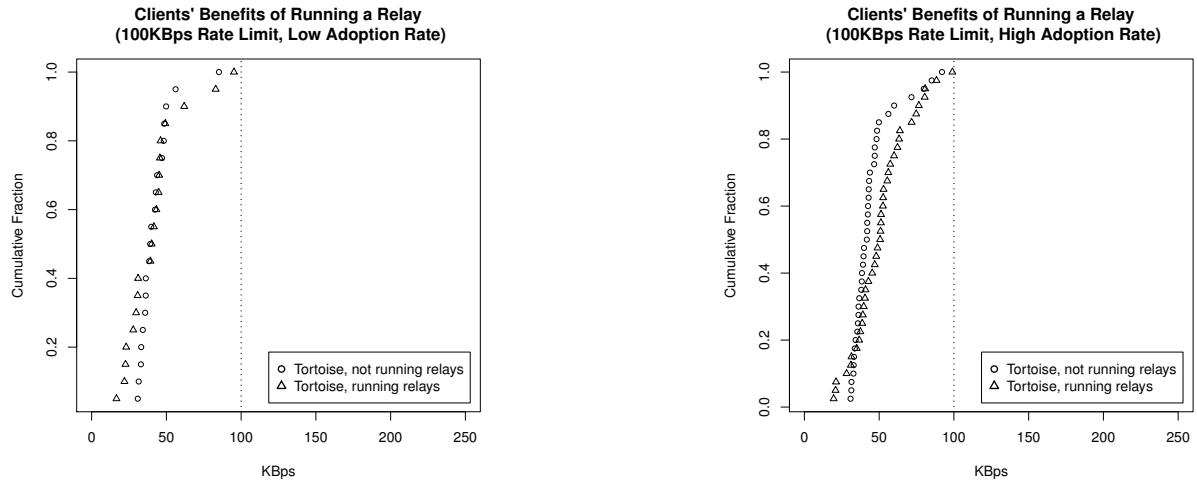


**Figure 9: Bandwidth improvements seen by clients that choose to run routers, in a network with only 50% of routers running** Tortoise**.**

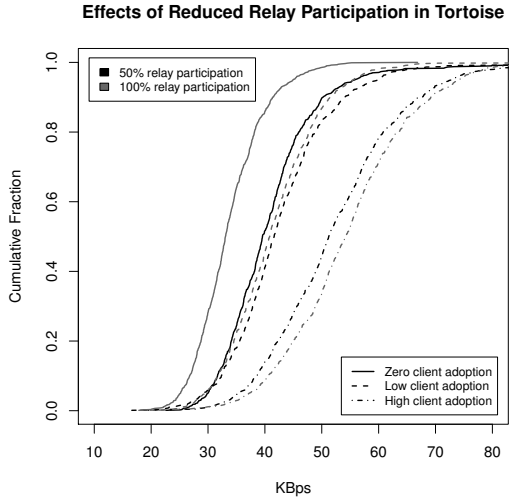


**Figure 10: The effect of sub-100% participation in** Tortoise **when a 100KBps rate limit is applied.**

gies should provide added benefit, since the reduced traffic load will lighten congestion.

**Relay selection**. Tor uses a bandwidth-weighted relay selection strategy in which the probability that a relay is chosen as a member of an anonymous path is proportional to the bandwidth advertised by that relay [8]. Snader and Borisov proposed a refinement to Tor that allows the sender to increase the performance of her paths at the expense of anonymity [28]. Murdoch at Watson later showed that Tor's current bandwidth-weighted strategy provides both good anonymity and performance [20]. Sherr *et al.* argue that path selection based on other non-bandwidth metrics (such as latency or loss) offers performance and anonymity benefits [26, 27].

However, as has been pointed out by Dingledine and Murdoch [10], the primary cause of Tor's slowness is likely due to congestion caused by file sharing. While the above relay selection techniques may provide better performance or stronger anonymity on less congested networks, we believe such approaches are unlikely to provide good performance on the current Tor network. Tortoise is agnostic to clients' relay selection algorithms, and may complement the above approaches by alleviating congestion and permitting higher performing and more flexible anonymous routes.

**Incentive schemes**. Tortoise is most similar to techniques that attempt to provide incentives to operate Tor relays. In PAR [2], relays earn coins that they can spend on fast paths. However, the need to frequently authenticate coins with a central bank limits the approach's scalability.

Jansen *et al.* introduce a number of mechanisms called BRAIDS that encourage Tor users to run relays [14]. BRAIDS implements a form of differentiated service by segregating traffic into three service classes, each of which has particular performance properties (e.g., high latency, high throughput, etc.). Similar to e-cash systems, BRAIDS uses a *ticket* model in which tickets may be exchanged for higher performing anonymous paths. By rewarding relay operators with tickets (and consequently, better performing paths), BRAIDS encourages users to run Tor relays.

Similar to BRAIDS, Tortoise offers a form of differentiated service by enforcing different rate limits for users depending on whether they run a Tor router or not, providing an incentive for users to operate relays. However, while BRAIDS requires a partially trusted offline bank to manage tickets, Tortoise is fully backwards compatible with the existing Tor network, can be incrementally deployed, and requires no centralized structures.

Ngan *et al.* proposed a system [22] in which cells from circuits are marked with a "gold star" if they originate from a Tor instance that is also a STABLE router. Tests on an experimental Tor network showed that using the system, cooperating nodes (nodes running a router, and thus receiving priority) gained a significant advantage over other nodes when the network was under heavy load. While the gold star system considers a cell's priority at each hop, Tortoise checks only at network ingress points, and gives all traffic equal priority once it is past the first hop. Additionally, the ability to prioritize traffic at each hop in the gold star scheme requires that each hop be running the modified software; in Tortoise, only the guard nodes' are aware of the rate limiting.

**Adaptive throttling of Tor clients by entry guards**. A September 2010 Tor blog post [6] addressed the efficacy of using per-connection rate limits to reduce the impact of bandwidth-intensive connections on the network. The author confirmed that one could

set rate limits in a manner such that only large transfers would be significantly throttled. However, some commenters claimed such measures would not help the network because a client could avoid rate limiting by running a relay from the same IP as the client transferring large amounts of data. Our work takes the opposite viewpoint: a bulk client running a relay in order to achieve faster speeds is not only acceptable, it is also desirable as it benefits the network as a whole.

# 8. CONCLUSION

This paper proposes Tortoise, a backwards-compatible extension to Tor that applies per-connection rate limits at Tor's ingress points. By carefully tuning these limits, our results indicate that Tortoise imposes little performance penalty to most web clients while simultaneously providing incentives to high-bandwidth clients to operate their own relays.

Tortoise's benefits hinge on the ability to attract additional relays. By enforcing strict rate limits and giving exemptions only to relay operators, we argue that clients who demand high bandwidths will be sufficiently motivated to contribute a fraction of their bandwidth to the Tor network. Emulation results demonstrate that even if a small percentage of clients opt to run relays, the network not only achieves significant performance gains, but also an increased capacity to handle additional load. For instance, if 4% of the clients are motivated to operate a relay, then the network experiences a 32% improvement in effective capacity and is significantly better able to tolerate a sudden influx of additional clients than the current Tor network.

## Acknowledgements

We thank the anonymous reviewers for their helpful feedback, and Kevin Bauer for several thoughtful discussions about this work. This research was supported by DARPA SAFER award N66001-11-C-4020 and NSF grant CNS-1064986. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the funding agencies.

## References

[1] Alexa: The Web Information Company. Top Sites. http://www.alexa.com/topsites. Retrieved May 13, 2011.

[2] E. Androulaki, M. Raykova, S. Srivatsan, A. Stavrou, and S. Bellovin. PAR: Payment for Anonymous Routing. In *Privacy Enhancing Technologies Symposium (PETS)*, 2008.

[3] K. Bauer, M. Sherr, D. McCoy, and D. Grunwald. ExperimenTor: A Testbed for Safe and Realistic Tor Experimentation. In *USENIX Workshop on Cyber Security Experimentation and Test (CSET)*, 2011.

[4] S. L. Blond, P. Manils, A. Chaabane, M. A. Kaafar, A. Legout, C. Castellucia, and W. Dabbous. De-anonymizing BitTorrent Users on Tor (poster). In *USENIX Symposium on Networked Systems Design and Implementation (NSDI)*, 2010.

[5] Y. Chen, R. Sion, and B. Carbunar. XPay: Practical Anonymous Payments for Tor Routing and Other Networked Services. In *ACM Workshop on Privacy in the Electronic Society (WPES)*, 2009.

[6] R. Dingledine. Research Problem: Adaptive Throttling of Tor Clients by Entry Guards. http://preview.tinyurl.com/3tcyaem. Retrieved May 24, 2011.

[7] R. Dingledine. Bittorrent Over Tor Isn't a Good Idea. https://blog.torproject.org/blog/bittorrent-over-tor-isnt-good-idea, April 2010.

[8] R. Dingledine and N. Mathewson. Tor Path Specification. http://www.torproject.org/svn/trunk/doc/spec/path-spec.txt, January 2008.

[9] R. Dingledine, N. Mathewson, and P. Syverson. Tor: The Second-Generation Onion Router. In *USENIX Security Symposium (USENIX)*, 2004.

[10] R. Dingledine and S. Murdoch. Performance Improvements on Tor, or, Why Tor is Slow and What We're Going to Do About It. https://svn.torproject.org/svn/projects/roadmaps/2009-03-11-performance.pdf, March 2009.

[11] J. R. Douceur. The Sybil Attack. In *International Workshop on Peer-to-Peer Systems (IPTPS)*, 2002.

[12] F. Hernández-Campos, K. Jeffay, and F. Smith. Tracking the Evolution of Web Traffic: 1995-2003. In *Modeling, Analysis and Simulation of Computer Telecommunications Systems (MASCOTS)*, 2003.

[13] A. Hintz. Fingerprinting Websites Using Traffic Analysis. In *Privacy Enhancing Technologies Symposium (PETS)*, 2003.

[14] R. Jansen, N. Hopper, and Y. Kim. Recruiting New Tor Relays with BRAIDS. In *ACM Conference on Computer and Communications Security (CCS)*, 2010.

[15] K. Loesing. Measuring the Tor Network: Evaluation of Client Requests to the Directories. Technical report, Tor Project, June 2009.

[16] N. Mathewson. Evaluating SCTP for Tor. http://archives.seul.org/or/dev/Sep-2004/msg00002.html, September 2004. Listserv posting.

[17] D. McCoy, K. Bauer, D. Grunwald, T. Kohno, and D. Sicker. Shining Light in Dark Places: Understanding the Tor Network. In *Privacy Enhancing Technologies Symposium (PETS)*, 2008.

[18] R. C. Merkle. Secure Communications over Insecure Channels. *Communications of the ACM*, 21:294–299, April 1978.

[19] S. J. Murdoch. Hot or Not: Revealing Hidden Services by Their Clock Skew. In *ACM Conference on Computer and Communications Security (CCS)*, 2006.

[20] S. J. Murdoch and R. N. M. Watson. Metrics for Security and Performance in Low-Latency Anonymity Systems. In *Privacy Enhancing Technologies Symposium (PETS)*, 2008.

[21] NetIndex Source Data. http://netindex.com/source-data/. Retrieved May 26, 2011.

[22] T.-W. J. Ngan, R. Dingledine, and D. Wallach. Building Incentives into Tor. In *Financial Cryptography and Data Security*, 2010.

[23] M. Perry. Tips for running an exit node with minimal harassment. https://blog.torproject.org/blog/tips-running-exit-node-minimal-harassment. Retrieved May 16, 2011.

[24] M. Perry. Computing Bandwidth Adjustments. Proposal 161, Tor Project, 2009.

[25] J. Reardon and I. Goldberg. Improving Tor using a TCP-over-DTLS Tunnel. In *USENIX Security Symposium (USENIX)*, 2009.

[26] M. Sherr, M. Blaze, and B. T. Loo. Scalable Link-Based Relay Selection for Anonymous Routing. In *Privacy Enhancing Technologies Symposium (PETS)*, August 2009.

[27] M. Sherr, A. Mao, W. R. Marczak, W. Zhou, B. T. Loo, and M. Blaze. A3: An Extensible Platform for Application-Aware Anonymity. In *Network and Distributed System Security Symposium (NDSS)*, 2010.

[28] R. Snader and N. Borisov. A Tune-up for Tor: Improving Security and Performance in the Tor Network. In *Network and Distributed System Security Symposium (NDSS)*, 2008.

[29] C. Tang and I. Goldberg. An Improved Algorithm for Tor Circuit Scheduling. In *ACM Conference on Computer and Communications Security (CCS)*, 2010.

[30] Tor Project, Inc. Tor Directory Protocol, Version 3, 2010. https://git.torproject.org/checkout/tor/master/doc/spec/dir-spec.txt.

[31] A. Vahdat, K. Yocum, K. Walsh, P. Mahadevan, D. Kostić, J. Chase, and D. Becker. Scalability and Accuracy in a Large-scale Network Emulator. *SIGOPS Oper. Syst. Rev.*, 36:271–284, December 2002.

[32] S. Zander and S. J. Murdoch. An Improved Clock-Skew Measurement Technique for Revealing Hidden Services. In *USENIX Security Symposium (USENIX)*, 2008.

[33] Zona Publishing. The Need for Speed II. *Zona Market Bulletin*, 5, April 2001.