Challenges in deploying low-latency anonymity (DRAFT)

Roger Dingledine[1], Nick Mathewson[1], and Paul Syverson[2]

[1] The Free Haven Project <{arma,nickm}@freehaven.net>
[2] Naval Research Laboratory <syverson@itd.nrl.navy.mil>

Abstract. There are many unexpected or unexpectedly difficult obstacles to deploying anonymous communications. Drawing on our experiences deploying Tor (the second-generation onion routing network), we describe social challenges and technical issues that must be faced in building, deploying, and sustaining a scalable, distributed, low-latency anonymity network.

1 Introduction

Anonymous communication is full of surprises. This paper discusses some unexpected challenges arising from our experiences deploying Tor, a low-latency general-purpose anonymous communication system. We will discuss some of the difficulties we have experienced and how we have met them (or how we plan to meet them, if we know). We also discuss some less troublesome open problems that we must nevertheless eventually address.

Tor is an overlay network for anonymizing TCP streams over the Internet [13]. It addresses limitations in earlier Onion Routing designs [17, 27, 35, 36] by adding perfect forward secrecy, congestion control, directory servers, data integrity, configurable exit policies, and location-hidden services using rendezvous points. Tor works on the real-world Internet, requires no special privileges or kernel modifications, requires little synchronization or coordination between nodes, and provides a reasonable trade-off between anonymity, usability, and efficiency.

We deployed the public Tor network in October 2003; since then it has grown to over a hundred volunteer-operated nodes and as much as 80 megabits of average traffic per second. Tor's research strategy has focused on deploying a network to as many users as possible; thus, we have resisted designs that would compromise deployability by imposing high resource demands on node operators, and designs that would compromise usability by imposing unacceptable restrictions on which applications we support. Although this strategy has drawbacks (including a weakened threat model, as discussed below), it has made it possible for Tor to serve many thousands of users and attract funding from diverse sources whose goals range from security on a national scale down to individual liberties.

In [13] we gave an overall view of Tor's design and goals. Here we describe some policy, social, and technical issues that we face as we continue deployment. Rather than providing complete solutions to every problem, we instead lay out the challenges and constraints that we have observed while deploying Tor. In doing so, we aim to provide a research agenda of general interest to projects attempting to build and deploy practical, usable anonymity networks in the wild.

2 Background

Here we give a basic overview of the Tor design and its properties, and compare Tor to other low-latency anonymity designs.

2.1 Tor, threat models, and distributed trust

Tor provides *forward privacy*, so that users can connect to Internet sites without revealing their logical or physical locations to those sites or to observers. It also provides *location-hidden services*, so that servers can support authorized users without giving an effective vector for physical or online attackers. Tor provides these protections even when a portion of its infrastructure is compromised.

To connect to a remote server via Tor, the client software learns a signed list of Tor nodes from one of several central *directory servers*, and incrementally creates a private pathway or *circuit* of encrypted connections through authenticated Tor nodes on the network, negotiating a separate set of encryption keys for each hop along the circuit. The circuit is extended one node at a time, and each node along the way knows only the immediately previous and following nodes in the circuit, so no individual Tor node knows the complete path that each fixed-sized data packet (or *cell*) will take. Thus, neither an eavesdropper nor a compromised node can see both the connection's source and destination. Later requests use a new circuit, to complicate long-term linkability between different actions by a single user.

Tor also helps servers hide their locations while providing services such as web publishing or instant messaging. Using "rendezvous points", other Tor users can connect to these authenticated hidden services, neither one learning the other's network identity.

Tor attempts to anonymize the transport layer, not the application layer. This approach is useful for applications such as SSH where authenticated communication is desired. However, when anonymity from those with whom we communicate is desired, application protocols that include personally identifying information need additional application-level scrubbing proxies, such as Privoxy [26] for HTTP. Furthermore, Tor does not relay arbitrary IP packets; it only anonymizes TCP streams and DNS requests (but see Section 4.1).

Most node operators do not want to allow arbitrary TCP traffic. To address this, Tor provides *exit policies* so each exit node can block the IP addresses and ports it is unwilling to allow. Tor nodes advertise their exit policies to the directory servers, so that client can tell which nodes will support their connections.

As of January 2005, the Tor network has grown to around a hundred nodes on four continents, with a total capacity exceeding 1Gbit/s. Appendix A shows a graph of the number of working nodes over time, as well as a graph of the number of bytes being handled by the network over time. The network is now sufficiently diverse for further development and testing; but of course we always encourage new nodes to join.

Tor research and development has been funded by ONR and DARPA for use in securing government communications, and by the Electronic Frontier Foundation for use in maintaining civil liberties for ordinary citizens online. The Tor protocol is one of the leading choices for the anonymizing layer in the European Union's PRIME directive to help maintain privacy in Europe. The AN.ON project in Germany has integrated an independent implementation of the Tor protocol into their popular Java Anon Proxy anonymizing client.

Threat models and design philosophy. The ideal Tor network would be practical, useful and and anonymous. When trade-offs arise between these properties, Tor's research strategy has been to remain useful enough to attract many users, and practical enough to support them. Only subject to these constraints do we try to maximize anonymity.[1] Because of our strategy, Tor has a weaker threat model than many designs in the literature. In particular, because we support

[1] This is not the only possible direction in anonymity research: designs exist that provide more anonymity than Tor at the expense of significantly increased resource requirements, or decreased flexibility in application support (typically because of increased latency). Such research does not typically abandon aspirations toward deployability or utility, but instead tries to maximize deployability and utility subject to a certain degree of structural anonymity (structural because usability and practicality affect usage which affects the actual anonymity provided by the network [1, 4]).

interactive communications without impractically expensive padding, we fall prey to a variety of intra-network [4, 24, 38] and end-to-end [8, 31] anonymity-breaking attacks.

Tor does not attempt to defend against a global observer. In general, an attacker who can measure both ends of a connection through the Tor network can correlate the timing and volume of data on that connection as it enters and leaves the network, and so link communication partners. Known solutions to this attack would seem to require introducing a prohibitive degree of traffic padding between the user and the network, or introducing an unacceptable degree of latency (but see Section 4.2). Also, it is not clear that these methods would work at all against a minimally active adversary who could introduce timing patterns or additional traffic. Thus, Tor only attempts to defend against external observers who cannot observe both sides of a user's connections.

Against internal attackers who sign up Tor nodes, the situation is more complicated. In the simplest case, if an adversary has compromised c of n nodes on the Tor network, then the adversary will be able to compromise a random circuit with probability $\frac{c^2}{n^2}$ (since the circuit initiator chooses hops randomly). But there are complicating factors: (1) If the user continues to build random circuits over time, an adversary is pretty certain to see a statistical sample of the user's traffic, and thereby can build an increasingly accurate profile of her behavior. (See Section 4.3 for possible solutions.) (2) An adversary who controls a popular service outside the Tor network can be certain to observe all connections to that service; he can therefore trace connections to that service with probability $\frac{c}{n}$. (3) Users do not in fact choose nodes with uniform probability; they favor nodes with high bandwidth or uptime, and exit nodes that permit connections to their favorite services. See Section 4.5 for discussion of larger adversaries and our dispersal goals.

More powerful attacks may exist. In [18] it was shown that an attacker who can catalog data volumes of popular responder destinations (say, websites with consistent data volumes) may not need to observe both ends of a stream to learn source-destination links for those responders. Similarly, latencies of going through various routes can be cataloged [4] to connect endpoints. It has not yet been shown whether these attacks will succeed or fail in the presence of the variability and volume quantization introduced by the Tor network, but it seems likely that these factors will at best delay rather than halt the attacks in the cases where they succeed. Along similar lines, the same paper suggests a "clogging attack." Murdoch and Danezis [24] show a practical clogging attack against portions of the fifty node Tor network as deployed in mid 2004. An outside attacker can actively trace a circuit through the Tor network by observing changes in the latency of his own traffic sent through various Tor nodes. These attacks only reveal the Tor nodes in the circuit, not initiator and responder addresses, so it is still necessary to discover the endpoints to complete the attacks. Increasing the size and diversity of the Tor network may help counter these attacks.

Distributed trust. In practice Tor's threat model is based on dispersal and diversity. Our defense lies in having a diverse enough set of nodes to prevent most real-world adversaries from being in the right places to attack users, by distributing each transaction over several nodes in the network. This "distributed trust" approach means the Tor network can be safely operated and used by a wide variety of mutually distrustful users, providing sustainability and security.

No organization can achieve this security on its own. If a single corporation or government agency were to build a private network to protect its operations, any connections entering or leaving that network would be obviously linkable to the controlling organization. The members and operations of that agency would be easier, not harder, to distinguish.

Instead, to protect our networks from traffic analysis, we must collaboratively blend the traffic from many organizations and private citizens, so that an eavesdropper can't tell which users are which, and who is looking for what information. The Tor network has a broad range of users, including ordinary citizens concerned about their privacy, corporations who don't want to reveal information to their competitors, and law enforcement and government intelligence agencies who need to do operations on the Internet without being noticed. Naturally, organizations will not want to depend on others for their security. If most participating providers are reliable, Tor tolerates

some hostile infiltration of the network. For maximum protection, the Tor design includes an enclave approach that lets data be encrypted (and authenticated) end-to-end, so high-sensitivity users can be sure it hasn't been read or modified. This even works for Internet services that don't have built-in encryption and authentication, such as unencrypted HTTP or chat, and it requires no modification of those services.

2.2 Related work

Tor differs from other deployed systems for traffic analysis resistance in its security and flexibility. Mix networks such as Mixmaster [23] or its successor Mixminion [9] gain the highest degrees of anonymity at the expense of introducing highly variable delays, making them unsuitable for applications such as web browsing. Commercial single-hop proxies [2] can provide good performance, but a single compromise can expose all users' traffic, and a single-point eavesdropper can perform traffic analysis on the entire network. The Java Anon Proxy [5] provides similar functionality to Tor but handles only web browsing rather than all TCP. Zero-Knowledge Systems' Freedom network [3] was even more flexible than Tor in transporting arbitrary IP packets, and also supported pseudonymity in addition to anonymity; but it has a different approach to sustainability (collecting money from users and paying ISPs to run Tor nodes), and was eventually shut down due to financial load. Finally, peer-to-peer designs that are intended to be more scalable, for example Tarzan [16] and MorphMix [28], have been proposed in the literature but have not been fielded. These systems differ somewhat in threat model and presumably practical resistance to threats. Note that MorphMix differs from Tor only in node discovery and circuit setup; so Tor's architecture is flexible enough to contain a MorphMix experiment. We direct the interested reader to [13] for a more in-depth review of related work.

3 Social challenges

Many of the issues the Tor project needs to address extend beyond system design and technology development. In particular, the Tor project's *image* with respect to its users and the rest of the Internet impacts the security it can provide. With this image issue in mind, this section discusses the Tor user base and Tor's interaction with other services on the Internet.

3.1 Communicating security

Usability for anonymity systems contributes to their security, because usability affects the possible anonymity set [1, 4]. Conversely, an unusable system attracts few users and thus can't provide much anonymity.

This phenomenon has a second-order effect: knowing this, users should choose which anonymity system to use based in part on how usable and secure *others* will find it, in order to get the protection of a larger anonymity set. Thus we might supplement the adage "usability is a security parameter" [4] with a new one: "perceived usability is a security parameter." From here we can better understand the effects of publicity on security: the more convincing your advertising, the more likely people will believe you have users, and thus the more users you will attract. Perversely, over-hyped systems (if they are not too broken) may be a better choice than modestly promoted ones, if the hype attracts more users [12].

So it follows that we should come up with ways to accurately communicate the available security levels to the user, so she can make informed decisions. JAP aims to do this by including a comforting 'anonymity meter' dial in the software's graphical interface, giving the user an impression of the level of protection for her current traffic.

However, there's a catch. For users to share the same anonymity set, they need to act like each other. An attacker who can distinguish a given user's traffic from the rest of the traffic will not be distracted by anonymity set size. For high-latency systems like Mixminion, where the threat model is based on mixing messages with each other, there's an arms race between end-to-end statistical attacks and counter-strategies [7, 9, 22, 30]. But for low-latency systems like Tor, end-to-end *traffic correlation* attacks [8, 21, 31] allow an attacker who can observe both ends of a communication to correlate packet timing and volume, quickly linking the initiator to her destination.

Like Tor, the current JAP implementation does not pad connections apart from using small fixed-size cells for transport. In fact, JAP's cascade-based network topology may be more vulnerable to these attacks, because its network has fewer edges. JAP was born out of the ISDN mix design [25], where padding made sense because every user had a fixed bandwidth allocation and altering the timing pattern of packets could be immediately detected. But in its current context as an Internet web anonymizer, adding sufficient padding to JAP would probably be prohibitively expensive and ineffective against a minimally active attacker.[2] Therefore, since under this threat model the number of concurrent users does not seem to have much impact on the anonymity provided, we suggest that JAP's anonymity meter is not accurately communicating security levels to its users.

On the other hand, while the number of active concurrent users may not matter as much as we'd like, it still helps to have some other users on the network. We investigate this issue next.

3.2 Reputability and perceived social value

Another factor impacting the network's security is its reputability: the perception of its social value based on its current user base. If Alice is the only user who has ever downloaded the software, it might be socially accepted, but she's not getting much anonymity. Add a thousand activists, and she's anonymous, but everyone thinks she's an activist too. Add a thousand diverse citizens (cancer survivors, privacy enthusiasts, and so on) and now she's harder to profile.

Furthermore, the network's reputability affects its operator base: more people are willing to run a service if they believe it will be used by human rights workers than if they believe it will be used exclusively for disreputable ends. This effect becomes stronger if node operators themselves think they will be associated with their users' disreputable ends.

So the more cancer survivors on Tor, the better for the human rights activists. The more malicious hackers, the worse for the normal users. Thus, reputability is an anonymity issue for two reasons. First, it impacts the sustainability of the network: a network that's always about to be shut down has difficulty attracting and keeping adequate nodes. Second, a disreputable network is more vulnerable to legal and political attacks, since it will attract fewer supporters.

While people therefore have an incentive for the network to be used for "more reputable" activities than their own, there are still trade-offs involved when it comes to anonymity. To follow the above example, a network used entirely by cancer survivors might welcome file sharers onto the network, though of course they'd prefer a wider variety of users.

Reputability becomes even more tricky in the case of privacy networks, since the good uses of the network (such as publishing by journalists in dangerous countries) are typically kept private, whereas network abuses or other problems tend to be more widely publicized.

The impact of public perception on security is especially important during the bootstrapping phase of the network, where the first few widely publicized uses of the network can dictate the types of users it attracts next. As an example, some U.S. Department of Energy penetration testing engineers are tasked with compromising DoE computers from the outside. They only have a limited number of ISPs from which to launch their attacks, and they found that the defenders

[2] Even if JAP could fund higher-capacity nodes indefinitely, our experience suggests that many users would not accept the increased per-user bandwidth requirements, leading to an overall much smaller user base. But see Section 4.2.

were recognizing attacks because they came from the same IP space. These engineers wanted to use Tor to hide their tracks. First, from a technical standpoint, Tor does not support the variety of IP packets one would like to use in such attacks (see Section 4.1). But aside from this, we also decided that it would probably be poor precedent to encourage such use—even legal use that improves national security—and managed to dissuade them.

3.3 Sustainability and incentives

One of the unsolved problems in low-latency anonymity designs is how to keep the nodes running. Zero-Knowledge Systems's Freedom network depended on paying third parties to run its servers; the JAP project's bandwidth depends on grants to pay for its bandwidth and administrative expenses. In Tor, bandwidth and administrative costs are distributed across the volunteers who run Tor nodes, so we at least have reason to think that the Tor network could survive without continued research funding.[3] But why are these volunteers running nodes, and what can we do to encourage more volunteers to do so?

We have not formally surveyed Tor node operators to learn why they are running nodes, but from the information they have provided, it seems that many of them run Tor nodes for reasons of personal interest in privacy issues. It is possible that others are running Tor for their own anonymity reasons, but of course they are hardly likely to tell us specifics if they are. Tor exit node operators do attain a degree of "deniability" for traffic that originates at that exit node. For example, it is likely in practice that HTTP requests from a Tor node's IP will be assumed to be from the Tor network. More significantly, people and organizations who use Tor for anonymity depend on the continued existence of the Tor network to do so; running a node helps to keep the network operational.

Since Tor is run by volunteers, the most crucial software usability issue is usability by operators: when an operator leaves, the network becomes less usable by everybody. To keep operators pleased, we must try to keep Tor's resource and administrative demands as low as possible.

Because of ISP billing structures, many Tor operators have underused capacity that they are willing to donate to the network, at no additional monetary cost to them. Features to limit bandwidth have been essential to adoption. Also useful has been a "hibernation" feature that allows a Tor node that wants to provide high bandwidth, but no more than a certain amount in a giving billing cycle, to become dormant once its bandwidth is exhausted, and to reawaken at a random offset into the next billing cycle. This feature has interesting policy implications, however; see the next section below. Exit policies help to limit administrative costs by limiting the frequency of abuse complaints. (See Section 3.5.)

3.4 Bandwidth and file-sharing

Once users have configured their applications to work with Tor, the largest remaining usability issue is performance. Users begin to suffer when websites "feel slow." Clients currently try to build their connections through nodes that they guess will have enough bandwidth. But even if capacity is allocated optimally, it seems unlikely that the current network architecture will have enough capacity to provide every user with as much bandwidth as she would receive if she weren't using Tor, unless far more nodes join the network.

Much of Tor's recent bandwidth difficulties have come from file-sharing applications. These applications provide two challenges to any anonymizing network: their intensive bandwidth requirement, and the degree to which they are associated (correctly or not) with copyright infringement.

[3] It also helps that Tor is implemented with free and open source software that can be maintained by anybody with the ability and inclination.

High-bandwidth protocols can make the network unresponsive, but tend to be somewhat self-correcting as lack of bandwidth drives away users who need it. Issues of copyright violation, however, are more interesting. Typical exit node operators want to help people achieve private and anonymous speech, not to help people (say) host Vin Diesel movies for download; and typical ISPs would rather not deal with customers who draw menacing letters from the MPAA. While it is quite likely that the operators are doing nothing illegal, many ISPs have policies of dropping users who get repeated legal threats regardless of the merits of those threats, and many operators would prefer to avoid receiving even meritless legal threats. So when letters arrive, operators are likely to face pressure to block file-sharing applications entirely, in order to avoid the hassle.

But blocking file-sharing is not easy: popular protocols have evolved to run on non-standard ports to get around other port-based bans. Thus, exit node operators who want to block file-sharing would have to find some way to integrate Tor with a protocol-aware exit filter. This could be a technically expensive undertaking, and one with poor prospects: it is unlikely that Tor exit nodes would succeed where so many institutional firewalls have failed. Another possibility for sensitive operators is to run a restrictive node that only permits exit connections to a restricted range of ports that are not frequently associated with file sharing. There are increasingly few such ports.

Other possible approaches might include rate-limiting connections, especially long-lived connections or connections to file-sharing ports, so that high-bandwidth connections do not flood the network. We might also want to give priority to cells on low-bandwidth connections to keep them interactive, but this could have negative anonymity implications.

For the moment, it seems that Tor's bandwidth issues have rendered it unattractive for bulk file-sharing traffic; this may continue to be so in the future. Nevertheless, Tor will likely remain attractive for limited use in file-sharing protocols that have separate control and data channels.

3.5 Tor and blacklists

It was long expected that, alongside legitimate users, Tor would also attract troublemakers who exploit Tor to abuse services on the Internet with vandalism, rude mail, and so on. Our initial answer to this situation was to use "exit policies" to allow individual Tor nodes to block access to specific IP/port ranges. This approach aims to make operators more willing to run Tor by allowing them to prevent their nodes from being used for abusing particular services. For example, all Tor nodes currently block SMTP (port 25), to avoid being used for spam.

Exit policies are useful, but they are insufficient: if not all nodes block a given service, that service may try to block Tor instead. While being blockable is important to being good netizens, we would like to encourage services to allow anonymous access. Services should not need to decide between blocking legitimate anonymous use and allowing unlimited abuse.

This is potentially a bigger problem than it may appear. On the one hand, services should be allowed to refuse connections from sources of possible abuse. But when a Tor node administrator decides whether he prefers to be able to post to Wikipedia from his IP address, or to allow people to read Wikipedia anonymously through his Tor node, he is making the decision for others as well. (Wikipedia has blocked all posting from all Tor nodes based on IP addresses.) If the Tor node shares an address with a campus or corporate NAT, then the decision can prevent the entire population from posting. This is a loss for both Tor and Wikipedia: we don't want to compete for (or divvy up) the NAT-protected entities of the world.

Worse, many IP blacklists are coarse-grained: they ignore Tor's exit policies, partly because it's easier to implement and partly so they can punish all Tor nodes. One IP blacklist even bans every class C network that contains a Tor node, and recommends banning SMTP from these networks even though Tor does not allow SMTP at all. This strategic decision aims to discourage the operation of anything resembling an open proxy by encouraging its neighbors to shut it down to get unblocked themselves. This pressure even affects Tor nodes running in middleman mode (disallowing all exits) when those nodes are blacklisted too.

Problems of abuse occur mainly with services such as IRC networks and Wikipedia, which rely on IP blocking to ban abusive users. While at first blush this practice might seem to depend on the anachronistic assumption that each IP is an identifier for a single user, it is actually more reasonable in practice: it assumes that non-proxy IPs are a costly resource, and that an abuser can not change IPs at will. By blocking IPs which are used by Tor nodes, open proxies, and service abusers, these systems hope to make ongoing abuse difficult. Although the system is imperfect, it works tolerably well for them in practice.

Of course, we would prefer that legitimate anonymous users be able to access abuse-prone services. One conceivable approach would require would-be IRC users, for instance, to register accounts if they want to access the IRC network from Tor. In practice this would not significantly impede abuse if creating new accounts were easily automatable; this is why services use IP blocking. To deter abuse, pseudonymous identities need to require a significant switching cost in resources or human time. Some popular webmail applications impose cost with Reverse Turing Tests, but this step may not deter all abusers. Freedom used blind signatures to limit the number of pseudonyms for each paying account, but Tor has neither the ability nor the desire to collect payment.

We stress that as far as we can tell, most Tor uses are not abusive. Most services have not complained, and others are actively working to find ways besides banning to cope with the abuse. For example, the Freenode IRC network had a problem with a coordinated group of abusers joining channels and subtly taking over the conversation; but when they labelled all users coming from Tor IPs as "anonymous users," removing the ability of the abusers to blend in, the abuse stopped.

4 Design choices

In addition to social issues, Tor also faces some design trade-offs that must be investigated as the network develops.

4.1 Transporting the stream vs transporting the packets

Tor transports streams; it does not tunnel packets. It has often been suggested that like the old Freedom network [3], Tor should "obviously" anonymize IP traffic at the IP layer. Before this could be done, many issues need to be resolved:

1. *IP packets reveal OS characteristics.* We would still need to do IP-level packet normalization, to stop things like TCP fingerprinting attacks. This is unlikely to be a trivial task, given the diversity and complexity of TCP stacks.
2. *Application-level streams still need scrubbing.* We still need Tor to be easy to integrate with user-level application-specific proxies such as Privoxy. So it's not just a matter of capturing packets and anonymizing them at the IP layer.
3. *Certain protocols will still leak information.* For example, we must rewrite DNS requests so they are delivered to an unlinkable DNS server rather than the DNS server at a user's ISP; thus, we must understand the protocols we are transporting.
4. *The crypto is unspecified.* First we need a block-level encryption approach that can provide security despite packet loss and out-of-order delivery. Freedom allegedly had one, but it was never publicly specified. Also, TLS over UDP is not yet implemented or specified, though some early work has begun [29].
5. *We'll still need to tune network parameters.* Since the above encryption system will likely need sequence numbers (and maybe more) to do replay detection, handle duplicate frames, and so on, we will be reimplementing a subset of TCP anyway—a notoriously tricky path.
6. *Exit policies for arbitrary IP packets mean building a secure IDS.* Our node operators tell us that exit policies are one of the main reasons they're willing to run Tor. Adding an Intrusion

Detection System to handle exit policies would increase the security complexity of Tor, and would likely not work anyway, as evidenced by the entire field of IDS and counter-IDS papers. Many potential abuse issues are resolved by the fact that Tor only transports valid TCP streams (as opposed to arbitrary IP including malformed packets and IP floods), so exit policies become even *more* important as we become able to transport IP packets. We also need to compactly describe exit policies so clients can predict which nodes will allow which packets to exit.

7. *The Tor-internal name spaces would need to be redesigned.* We support hidden service `.onion` addresses (and other special addresses, like `.exit` which lets the user request a particular exit node), by intercepting the addresses when they are passed to the Tor client. Doing so at the IP level would require a more complex interface between Tor and the local DNS resolver.

This list is discouragingly long, but being able to transport more protocols obviously has some advantages. It would be good to learn which items are actual roadblocks and which are easier to resolve than we think.

To be fair, Tor's stream-based approach has run into stumbling blocks as well. While Tor supports the SOCKS protocol, which provides a standardized interface for generic TCP proxies, many applications do not support SOCKS. For them we already need to replace the networking system calls with SOCKS-aware versions, or run a SOCKS tunnel locally, neither of which is easy for the average user. Even when applications can use SOCKS, they often make DNS requests themselves before handing an IP address to Tor, which advertises where the user is about to connect. We are still working on more usable solutions.

4.2 Mid-latency

Some users need to resist traffic correlation attacks. Higher-latency mix-networks introduce variability into message arrival times: as timing variance increases, timing correlation attacks require increasingly more data [22]. Can we improve Tor's resistance without losing too much usability?

We need to learn whether we can trade a small increase in latency for a large anonymity increase, or if we'd end up trading a lot of latency for only a minimal security gain. A trade-off might be worthwhile even if we could only protect certain use cases, such as infrequent short-duration transactions. We might adapt the techniques of [22] to a lower-latency mix network, where the messages are batches of cells in temporally clustered connections. These large fixed-size batches can also help resist volume signature attacks [18]. We could also experiment with traffic shaping to get a good balance of throughput and security.

We must keep usability in mind too. How much can latency increase before we drive users away? We've already been forced to increase latency slightly, as our growing network incorporates more DSL and cable-modem nodes and more nodes in distant continents. Perhaps we can harness this increased latency to improve anonymity rather than just reduce usability. Further, if we let clients label certain circuits as mid-latency as they are constructed, we could handle both types of traffic on the same network, giving users a choice between speed and security—and giving researchers a chance to experiment with parameters to improve the quality of those choices.

4.3 Enclaves and helper nodes

It has long been thought that users can improve their anonymity by running their own node [13, 17, 36], and using it in an *enclave* configuration, where all their circuits begin at the node under their control. Running Tor clients or servers at the enclave perimeter is useful when policy or other requirements prevent individual machines within the enclave from running Tor clients [27, 35].

Of course, Tor's default path length of three is insufficient for these enclaves, since the entry and/or exit themselves are sensitive. Tor thus increments path length by one for each sensitive endpoint in the circuit. Enclaves also help to protect against end-to-end attacks, since it's possible

that traffic coming from the node has simply been relayed from elsewhere. However, if the node has recognizable behavior patterns, an attacker who runs nodes in the network can triangulate over time to gain confidence that it is in fact originating the traffic. Wright et al. [37] introduce the notion of a *helper node*—a single fixed entry node for each user—to combat this *predecessor attack*.

However, the attack in [24] shows that simply adding to the path length, or using a helper node, may not protect an enclave node. A hostile web server can send constant interference traffic to all nodes in the network, and learn which nodes are involved in the circuit (though at least in the current attack, he can't learn their order). Using randomized path lengths may help some, since the attacker will never be certain he has identified all nodes in the path, but as long as the network remains small this attack will still be feasible.

Helper nodes also aim to help Tor clients, because choosing entry and exit points randomly and changing them frequently allows an attacker who controls even a few nodes to eventually link some of their destinations. The goal is to take the risk once and for all about choosing a bad entry node, rather than taking a new risk for each new circuit. (Choosing fixed exit nodes is less useful, since even an honest exit node still doesn't protect against a hostile website.) But obstacles still remain before we can implement them. For one, the literature does not describe how to choose helpers from a list of nodes that changes over time. If Alice is forced to choose a new entry helper every d days and c of the n nodes are bad, she can expect to choose a compromised node around every dc/n days. Statistically over time this approach only helps if she is better at choosing honest helper nodes than at choosing honest nodes. Worse, an attacker with the ability to DoS nodes could force users to switch helper nodes more frequently, or remove other candidate helpers.

4.4 Location-hidden services

Tor's *rendezvous points* let users provide TCP services to other Tor users without revealing the service's location. Since this feature is relatively recent, we describe here a couple of our early observations from its deployment.

First, our implementation of hidden services seems less hidden than we'd like, since they build a different rendezvous circuit for each user, and an external adversary can induce them to produce traffic. This insecurity means that they may not be suitable as a building block for Free Haven [11] or other anonymous publishing systems that aim to provide long-term security, though helper nodes, as discussed above, would seem to help.

Hot-swap hidden services, where more than one location can provide the service and loss of any one location does not imply a change in service, would help foil intersection and observation attacks where an adversary monitors availability of a hidden service and also monitors whether certain users or servers are online. The design challenges in providing such services without otherwise compromising the hidden service's anonymity remain an open problem; however, see [34].

In practice, hidden services are used for more than just providing private access to a web server or IRC server. People are using hidden services as a poor man's VPN and firewall-buster. Many people want to be able to connect to the computers in their private network via secure shell, and rather than playing with dyndns and trying to pierce holes in their firewall, they run a hidden service on the inside and then rendezvous with that hidden service externally.

News sites like Bloggers Without Borders (www.b19s.org) are advertising a hidden-service address on their front page. Doing this can provide increased robustness if they use the dual-IP approach we describe in [13], but in practice they do it to increase visibility of the Tor project and their support for privacy, and to offer a way for their users, using unmodified software, to get end-to-end encryption and authentication to their website.

4.5 Location diversity and ISP-class adversaries

Anonymity networks have long relied on diversity of node location for protection against attacks—typically an adversary who can observe a larger fraction of the network can launch a more effective attack. One way to achieve dispersal involves growing the network so a given adversary sees less. Alternately, we can arrange the topology so traffic can enter or exit at many places (for example, by using a free-route network like Tor rather than a cascade network like JAP). Lastly, we can use distributed trust to spread each transaction over multiple jurisdictions. But how do we decide whether two nodes are in related locations?

Feamster and Dingledine defined a *location diversity* metric in [15], and began investigating a variant of location diversity based on the fact that the Internet is divided into thousands of independently operated networks called *autonomous systems* (ASes). The key insight from their paper is that while we typically think of a connection as going directly from the Tor client to the first Tor node, actually it traverses many different ASes on each hop. An adversary at any of these ASes can monitor or influence traffic. Specifically, given plausible initiators and recipients, and given random path selection, some ASes in the simulation were able to observe 10% to 30% of the transactions (that is, learn both the origin and the destination) on the deployed Tor network (33 nodes as of June 2004).

The paper concludes that for best protection against the AS-level adversary, nodes should be in ASes that have the most links to other ASes: Tier-1 ISPs such as AT&T and Abovenet. Further, a given transaction is safest when it starts or ends in a Tier-1 ISP. Therefore, assuming initiator and responder are both in the U.S., it actually *hurts* our location diversity to use far-flung nodes in continents like Asia or South America.

Many open questions remain. First, it will be an immense engineering challenge to get an entire BGP routing table to each Tor client, or to summarize it sufficiently. Without a local copy, clients won't be able to safely predict what ASes will be traversed on the various paths through the Tor network to the final destination. Tarzan [16] and MorphMix [28] suggest that we compare IP prefixes to determine location diversity; but the above paper showed that in practice many of the Mixmaster nodes that share a single AS have entirely different IP prefixes. When the network has scaled to thousands of nodes, does IP prefix comparison become a more useful approximation? Second, we can take advantage of caching certain content at the exit nodes, to limit the number of requests that need to leave the network at all. What about taking advantage of caches like Akamai or Google [32]? (Note that they're also well-positioned as global adversaries.) Third, if we follow the recommendations in [15] and tailor path selection to avoid choosing endpoints in similar locations, how much are we hurting anonymity against larger real-world adversaries who can take advantage of knowing our algorithm? Fourth, can we use this knowledge to figure out which gaps in our network most affect our robustness to this class of attack, and go recruit new nodes with those ASes in mind?

4.6 The Anti-censorship problem

Citizens in a variety of countries, such as most recently China and Iran, are blocked from accessing various sites outside their country. These users try to find any tools available to allow them to get-around these firewalls. Some anonymity networks, such as Six-Four [33], are designed specifically with this goal in mind; others like the Anonymizer [2] are paid by sponsors such as Voice of America to encourage Internet freedom. Even though Tor wasn't designed with ubiquitous access to the network in mind, thousands of users across the world are now using it for exactly this purpose.

Anti-censorship networks hoping to bridge country-level blocks face a variety of challenges. One of these is that they need to find enough exit nodes—servers on the 'free' side that are willing to relay traffic from users to their final destinations. Anonymizing networks including Tor are well-

suited to this task, since we have already gathered a set of exit nodes that are willing to tolerate some political heat.

The other main challenge is to distribute a list of reachable relays to the users inside the country, and give them software to use them, without letting the censors also enumerate this list and block each relay. Anonymizer solves this by buying lots of seemingly-unrelated IP addresses (or having them donated), abandoning old addresses as they are 'used up,' and telling a few users about the new ones. Distributed anonymizing networks again have an advantage here, in that we already have tens of thousands of separate IP addresses whose users might volunteer to provide this service since they've already installed and use the software for their own privacy [19]. Because the Tor protocol separates routing from network discovery [13], volunteers could configure their Tor clients to generate node descriptors and send them to a special directory server that gives them out to dissidents who need to get around blocks.

Of course, this still doesn't prevent the adversary from enumerating and preemptively blocking the volunteer relays. Perhaps a tiered-trust system could be built where a few individuals are given relays' locations, and they recommend other individuals by telling them those addresses, thus providing a built-in incentive to avoid letting the adversary intercept them. Max-flow trust algorithms [20] might help to bound the number of IP addresses leaked to the adversary. Groups like the W3C are looking into using Tor as a component in an overall system to help address censorship; we wish them success.

5 Scaling

Tor is running today with hundreds of nodes and tens of thousands of users, but it will certainly not scale to millions.

Scaling Tor involves four main challenges. First, to get a large set of nodes in the first place, we must address incentives for users to carry traffic for others. Next is safe node discovery, both while bootstrapping (how does a Tor client robustly find an initial node list?) and later (how does a Tor client learn about a fair sample of honest nodes and not let the adversary control his circuits?). We must also detect and handle node speed and reliability as the network becomes increasingly heterogeneous: since the speed and reliability of a circuit is limited by its worst link, we must learn to track and predict performance. Finally, we must stop assuming that all points on the network can connect to all other points.

5.1 Incentives by Design

There are three behaviors we need to encourage for each Tor node: relaying traffic; providing good throughput and reliability while doing it; and allowing traffic to exit the network from that node.

We encourage these behaviors through *indirect* incentives: that is, by designing the system and educating users in such a way that users with certain goals will choose to relay traffic. One main incentive for running a Tor node is social: volunteers altruistically donate their bandwidth and time. We encourage this with public rankings of the throughput and reliability of nodes, much like seti@home. We further explain to users that they can get deniability for any traffic emerging from the same address as a Tor exit node, and they can use their own Tor node as an entry or exit point and be confident it's not run by an adversary. Further, users may run a node simply because they need such a network to be persistently available and usable, and the value of supporting this exceeds any countervening costs. Finally, we can encourage operators by improving the usability and feature set of the software: rate limiting support and easy packaging decrease the hassle of maintaining a node, and our configurable exit policies allow each operator to advertise a policy describing the hosts and ports to which he feels comfortable connecting.

To date these incentives appear to have been adequate. As the system scales or as new issues emerge, however, we may also need to provide *direct* incentives: providing payment or other resources in return for high-quality service. Paying actual money is problematic: decentralized e-cash systems are not yet practical, and a centralized collection system not only reduces robustness, but also has failed in the past (the history of commercial anonymizing networks is littered with failed attempts). A more promising option is to use a tit-for-tat incentive scheme, where nodes provide better service to nodes that have provided good service for them.

Unfortunately, such an approach introduces new anonymity problems. There are many surprising ways for nodes to game the incentive and reputation system to undermine anonymity because such systems are designed to encourage fairness in storage or bandwidth usage not fairness of provided anonymity. An adversary can attract more traffic by performing well or can provide targeted differential performance to individual users to undermine their anonymity. Typically a user who chooses evenly from all options is most resistant to an adversary targeting him, but that approach hampers the efficient use of heterogeneous nodes.

A possible solution is a simplified approach to the tit-for-tat incentive scheme based on two rules: (1) each node should measure the service it receives from adjacent nodes, and provide service relative to the received service, but (2) when a node is making decisions that affect its own security (such as building a circuit for its own application connections), it should choose evenly from a sufficiently large set of nodes that meet some minimum service threshold [14]. This approach allows us to discourage bad service without opening Alice up as much to attacks. All of this requires further study.

5.2 Trust and discovery

The published Tor design uses a deliberately simplistic design for authorizing new nodes and informing clients about Tor nodes and their status. All nodes periodically upload a signed description of their locations, keys, and capabilities to each of several well-known *directory servers*. These directory servers construct a signed summary of all known Tor nodes (a "directory"), and a signed statement of which nodes they believe to be operational then (a "network status"). Clients periodically download a directory to learn the latest nodes and keys, and more frequently download a network status to learn which nodes are likely to be running. Tor nodes also operate as directory caches, to lighten the bandwidth on the directory servers.

To prevent Sybil attacks (wherein an adversary signs up many purportedly independent nodes to increase her network view), this design requires the directory server operators to manually approve new nodes. Unapproved nodes are included in the directory, but clients do not use them at the start or end of their circuits. In practice, directory administrators perform little actual verification, and tend to approve any Tor node whose operator can compose a coherent email. This procedure may prevent trivial automated Sybil attacks, but will do little against a clever and determined attacker.

There are a number of flaws in this system that need to be addressed as we move forward. First, each directory server represents an independent point of failure: any compromised directory server could start recommending only compromised nodes. Second, as more nodes join the network, directories become infeasibly large, and downloading the list of nodes becomes burdensome. Third, the validation scheme may do as much harm as it does good. It not only can't prevent clever attackers from mounting Sybil attacks, but it may deter node operators from joining the network, if they expect the validation process to be difficult, or they do not share any languages in common with the directory server operators.

We could try to move the system in several directions, depending on our choice of threat model and requirements. If we did not need to increase network capacity to support more users, we could simply adopt even stricter validation requirements, and reduce the number of nodes in the network

to a trusted minimum. But, we can only do that if can simultaneously make node capacity scale much more than we anticipate to be feasible soon, and if we can find entities willing to run such nodes, an equally daunting prospect.

In order to address the first two issues, it seems wise to move to a system including a number of semi-trusted directory servers, no one of which can compromise a user on its own. Ultimately, of course, we cannot escape the problem of a first introducer: since most users will run Tor in whatever configuration the software ships with, the Tor distribution itself will remain a single point of failure so long as it includes the seed keys for directory servers, a list of directory servers, or any other means to learn which nodes are on the network. But omitting this information from the Tor distribution would only delegate the trust problem to each individual user.

5.3 Measuring performance and capacity

One of the paradoxes with engineering an anonymity network is that we'd like to learn as much as we can about how traffic flows so we can improve the network, but we want to prevent others from learning how traffic flows in order to trace users' connections through the network. Furthermore, many mechanisms that help Tor run efficiently require measurements about the network.

Currently, nodes try to deduce their own available bandwidth (based on how much traffic they have been able to transfer recently) and include this information in the descriptors they upload to the directory. Clients choose servers weighted by their bandwidth, neglecting really slow servers and capping the influence of really fast ones. This is, of course, eminently cheatable. A malicious node can get a disproportionate amount of traffic simply by claiming to have more bandwidth than it does. But better mechanisms have their problems. If bandwidth data is to be measured rather than self-reported, it is usually possible for nodes to selectively provide better service for the measuring party, or sabotage the measured value of other nodes. Complex solutions for mix networks have been proposed, but do not address the issues completely [10, 14].

Even with no cheating, network measurement is complex. It is common for views of a node's latency and/or bandwidth to vary wildly between observers. Further, it is unclear whether total bandwidth is really the right measure; perhaps clients should instead be considering nodes based on unused bandwidth or observed throughput. And even if we can collect and use this network information effectively, we must ensure that it is not more useful to attackers than to us. While it seems plausible that bandwidth data alone is not enough to reveal sender-recipient connections under most circumstances, it could certainly reveal the path taken by large traffic flows under low-usage circumstances.

5.4 Non-clique topologies

Tor's comparatively weak threat model may allow easier scaling than other designs. High-latency mix networks need to avoid partitioning attacks, where network splits let an attacker distinguish users in different partitions. Since Tor assumes the adversary cannot cheaply observe nodes at will, a network split may not decrease protection much. Thus, one option when the scale of a Tor network exceeds some size is simply to split it. Nodes could be allocated into partitions while hampering collaborating hostile nodes from taking over a single partition [14]. Clients could switch between networks, even on a per-circuit basis.

More conservatively, we can try to scale a single Tor network. Likely problems with adding more servers to a single Tor network include an explosion in the number of sockets needed on each server as more servers join, and increased coordination overhead to keep each users' view of the network consistent. As we grow, we will also have more instances of servers that can't reach each other simply due to Internet topology or routing problems.

We can address these points by reducing the network's connectivity. Danezis [6] considers the anonymity implications of restricting routes on mix networks, and recommends an approach based

on expander graphs (where any subgraph is likely to have many neighbors). It is not immediately clear that this approach will extend to Tor, which has a weaker threat model but higher performance requirements: instead of analyzing the probability of an attacker's viewing whole paths, we will need to examine the attacker's likelihood of compromising the endpoints. Tor may not need an expander graph per se: it may be enough to have a single subnet that is highly connected, like an internet backbone. There are many open questions: how to distribute connectivity information (presumably nodes will learn about the center nodes when they download Tor), whether center nodes will need to function as a 'backbone', and so on. As above, this could create problems for the expected anonymity for a mix-net, but for a low-latency network where anonymity derives largely from the edges, it may be feasible.

6 The Future

Tor is the largest and most diverse low-latency anonymity network available, but we are still in the beginning stages of deployment. Several major questions remain.

First, will our volunteer-based approach to sustainability work in the long term? As we add more features and destabilize the network, the developers spend a lot of time keeping the server operators happy. Even though Tor is free software, the network would likely stagnate and die at this stage if the developers stopped actively working on it. We may get an unexpected boon from the fact that we're a general-purpose overlay network: as Tor grows more popular, other groups who need an overlay network on the Internet are starting to adapt Tor to their needs. Second, Tor is only one of many components that preserve privacy online. For applications where it is desirable to keep identifying information out of application traffic, someone must build more and better protocol-aware proxies that are usable by ordinary people. Third, we need to gain a reputation for social good, and learn how to coexist with the variety of Internet services and their established authentication mechanisms. We can't just keep escalating the blacklist standoff forever. Fourth, the current Tor architecture does not scale even to handle current user demand. We must find designs and incentives to let some clients relay traffic too, without sacrificing too much anonymity.

These are difficult and open questions, yet choosing not to solve them means leaving most users to a less secure network or no anonymizing network at all.

References

1. Alessandro Acquisti, Roger Dingledine, and Paul Syverson. On the economics of anonymity. In Rebecca N. Wright, editor, *Financial Cryptography*. Springer-Verlag, LNCS 2742, 2003.
2. The Anonymizer. `http://anonymizer.com/`.
3. Adam Back, Ian Goldberg, and Adam Shostack. Freedom systems 2.1 security issues and analysis. White paper, Zero Knowledge Systems, Inc., May 2001.
4. Adam Back, Ulf Möller, and Anton Stiglic. Traffic analysis attacks and trade-offs in anonymity providing systems. In Ira S. Moskowitz, editor, *Information Hiding (IH 2001)*, pages 245–257. Springer-Verlag, LNCS 2137, 2001.
5. Oliver Berthold, Hannes Federrath, and Stefan Köpsell. Web MIXes: A system for anonymous and unobservable Internet access. In H. Federrath, editor, *Designing Privacy Enhancing Technologies: Workshop on Design Issue in Anonymity and Unobservability*. Springer-Verlag, LNCS 2009, 2000.
6. George Danezis. Mix-networks with restricted routes. In Roger Dingledine, editor, *Privacy Enhancing Technologies (PET 2003)*. Springer-Verlag LNCS 2760, 2003.
7. George Danezis. Statistical disclosure attacks. In *Security and Privacy in the Age of Uncertainty (SEC2003)*, pages 421–426, Athens, May 2003. IFIP TC11, Kluwer.
8. George Danezis. The traffic analysis of continuous-time mixes. In David Martin and Andrei Serjantov, editors, *Privacy Enhancing Technologies (PET 2004)*, LNCS, May 2004. `http://www.cl.cam.ac.uk/users/gd216/cmm2.pdf`.

9. George Danezis, Roger Dingledine, and Nick Mathewson. Mixminion: Design of a type III anonymous remailer protocol. In *2003 IEEE Symposium on Security and Privacy*, pages 2–15. IEEE CS, May 2003.
10. Roger Dingledine, Michael J. Freedman, David Hopwood, and David Molnar. A Reputation System to Increase MIX-net Reliability. In Ira S. Moskowitz, editor, *Information Hiding (IH 2001)*, pages 126–141. Springer-Verlag, LNCS 2137, 2001.
11. Roger Dingledine, Michael J. Freedman, and David Molnar. The free haven project: Distributed anonymous storage service. In H. Federrath, editor, *Designing Privacy Enhancing Technologies: Workshop on Design Issue in Anonymity and Unobservability*. Springer-Verlag, LNCS 2009, July 2000.
12. Roger Dingledine and Nick Mathewson. Anonymity loves company: Usability and the network effect. In *Designing Security Systems That People Can Use*. O'Reilly Media, 2005.
13. Roger Dingledine, Nick Mathewson, and Paul Syverson. Tor: The second-generation onion router. In *Proceedings of the 13th USENIX Security Symposium*, August 2004. `http://tor.eff.org/tor-design.pdf`.
14. Roger Dingledine and Paul Syverson. Reliable MIX Cascade Networks through Reputation. In Matt Blaze, editor, *Financial Cryptography*. Springer-Verlag, LNCS 2357, 2002.
15. Nick Feamster and Roger Dingledine. Location diversity in anonymity networks. In *Proceedings of the Workshop on Privacy in the Electronic Society (WPES 2004)*, Washington, DC, USA, October 2004. `http://freehaven.net/doc/routing-zones/routing-zones.ps`.
16. Michael J. Freedman and Robert Morris. Tarzan: A peer-to-peer anonymizing network layer. In *9th ACM Conference on Computer and Communications Security (CCS 2002)*, Washington, DC, November 2002.
17. David M. Goldschlag, Michael G. Reed, and Paul F. Syverson. Hiding routing information. In R. Anderson, editor, *Information Hiding, First International Workshop*, pages 137–150. Springer-Verlag, LNCS 1174, May 1996.
18. Andrew Hintz. Fingerprinting websites using traffic analysis. In Roger Dingledine and Paul Syverson, editors, *Privacy Enhancing Technologies (PET 2002)*, pages 171–178. Springer-Verlag, LNCS 2482, 2002.
19. Stefan Köpsell and Ulf Hilling. How to achieve blocking resistance for existing systems enabling anonymous web surfing. In *Proceedings of the Workshop on Privacy in the Electronic Society (WPES 2004)*, Washington, DC, USA, October 2004. `http://freehaven.net/anonbib/papers/p103-koepsell.pdf`.
20. Raph Levien. Advogato's trust metric. `http://www.advogato.org/trust-metric.html`.
21. Brian N. Levine, Michael K. Reiter, Chenxi Wang, and Matthew Wright. Timing analysis in low-latency mix-based systems. In Ari Juels, editor, *Financial Cryptography*. Springer-Verlag, LNCS (forthcoming), 2004.
22. Nick Mathewson and Roger Dingledine. Practical traffic analysis: Extending and resisting statistical disclosure. In David Martin and Andrei Serjantov, editors, *Privacy Enhancing Technologies (PET 2004)*, LNCS, May 2004. `http://freehaven.net/doc/e2e-traffic/e2e-traffic.pdf`.
23. Ulf Möller, Lance Cottrell, Peter Palfrader, and Len Sassaman. Mixmaster Protocol — Version 2. Draft, July 2003. `http://www.abditum.com/mixmaster-spec.txt`.
24. Steven J. Murdoch and George Danezis. Low-cost traffic analysis of tor. In *IEEE Symposium on Security and Privacy*. IEEE CS, May 2005.
25. Andreas Pfitzmann, Birgit Pfitzmann, and Michael Waidner. ISDN-mixes: Untraceable communication with very small bandwidth overhead. In *GI/ITG Conference on Communication in Distributed Systems*, pages 451–463, February 1991.
26. Privoxy. `http://www.privoxy.org/`.
27. Michael G. Reed, Paul F. Syverson, and David M. Goldschlag. Anonymous connections and onion routing. *IEEE Journal on Selected Areas in Communications*, 16(4):482–494, May 1998.
28. Marc Rennhard and Bernhard Plattner. Practical anonymity for the masses with morphmix. In Ari Juels, editor, *Financial Cryptography*. Springer-Verlag, LNCS (forthcoming), 2004.
29. E. Rescorla and N. Modadugu. Datagram Transport Layer Security. IETF Draft, December 2003. `http://www.ietf.org/internet-drafts/draft-rescorla-dtls-02.txt`.
30. Andrei Serjantov, Roger Dingledine, and Paul Syverson. From a trickle to a flood: Active attacks on several mix types. In Fabien Petitcolas, editor, *Information Hiding (IH 2002)*. Springer-Verlag, LNCS 2578, 2002.

31. Andrei Serjantov and Peter Sewell. Passive attack analysis for connection-based anonymity systems. In *Computer Security – ESORICS 2003*. Springer-Verlag, LNCS 2808, October 2003.
32. Anna Shubina and Sean Smith. Using caching for browsing anonymity. *ACM SIGEcom Exchanges*, 4(2), Sept 2003.
33. The Six/Four System. `http://sourceforge.net/projects/sixfour/`.
34. Angelos Stavrou, Angelos D. Keromytis, Jason Nieh, Vishal Misra, and Dan Rubenstein. Move: An end-to-end solution to network denial of service. In *ISOC Network and Distributed System Security Symposium (NDSS05)*. Internet Society, February 2005.
35. Paul Syverson, Michael Reed, and David Goldschlag. Onion Routing access configurations. In *DARPA Information Survivability Conference and Exposition (DISCEX 2000)*, volume 1, pages 34–40. IEEE CS Press, 2000.
36. Paul Syverson, Gene Tsudik, Michael Reed, and Carl Landwehr. Towards an Analysis of Onion Routing Security. In H. Federrath, editor, *Designing Privacy Enhancing Technologies: Workshop on Design Issue in Anonymity and Unobservability*, pages 96–114. Springer-Verlag, LNCS 2009, July 2000.
37. Matthew Wright, Micah Adler, Brian Neil Levine, and Clay Shields. Defending anonymous communication against passive logging attacks. In *IEEE Symposium on Security and Privacy*, pages 28–41. IEEE CS, May 2003.
38. Ye Zhu, Xinwen Fu, Bryan Graham, Riccardo Bettati, and Wei Zhao. On flow correlation attacks and countermeasures in mix networks. In *Proceedings of Privacy Enhancing Technologies workshop (PET 2004)*, LNCS, May 2004. `http://students.cs.tamu.edu/xinwenfu/paper/PET04.pdf`.

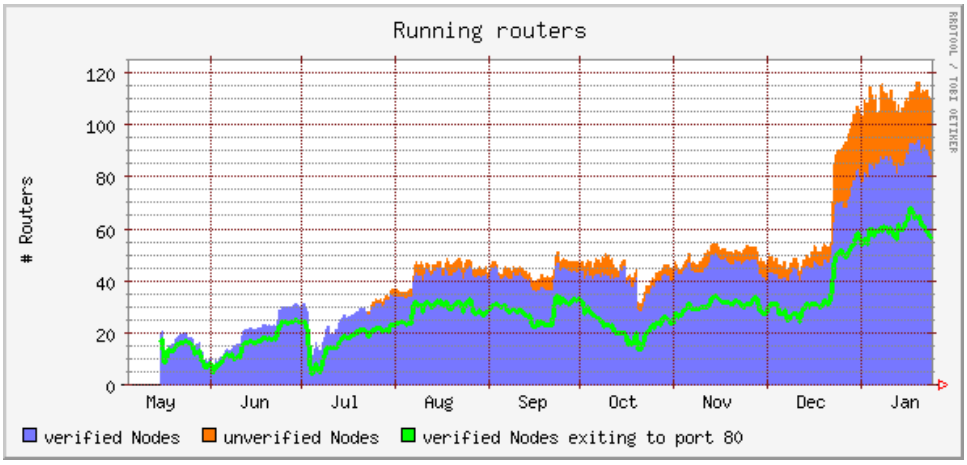

Fig. 1. Number of Tor nodes over time, through January 2005. Lowest line is number of exit nodes that allow connections to port 80. Middle line is total number of verified (registered) Tor nodes. The line above that represents nodes that are running but not yet registered.

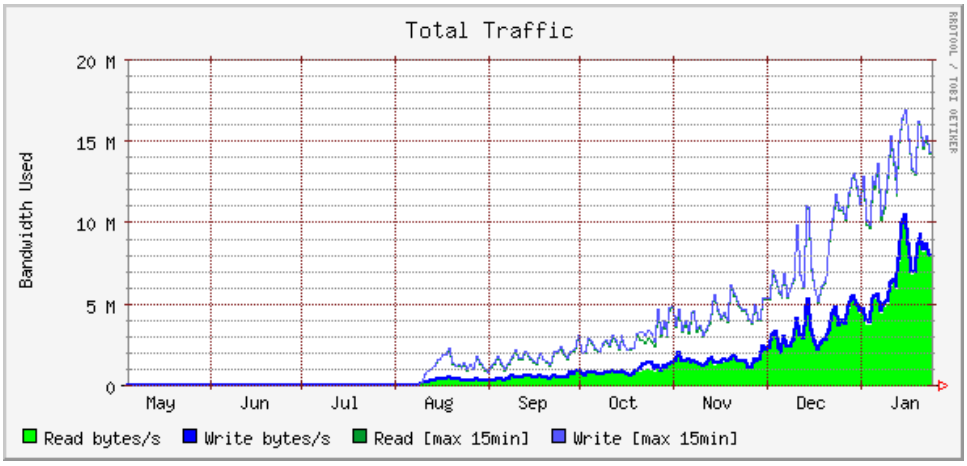

Fig. 2. The sum of traffic reported by each node over time, through January 2005. The bottom pair show average throughput, and the top pair represent the largest 15 minute burst in each 4 hour period.