Peer Profiling and Selection in the I2P Anonymous Network

zzz, Lars Schimmer

The I2P Project

http://www.i2p2.de/

zzz@i2pmail.org

echelon@i2pmail.org

Abstract

I2P is an anonymous communication network that routes data through tunnels. It selects subsets of peers for tunnels using peer profiling mechanisms, including opportunistic measurement of per-tunnel bandwidth and the number of accepted tunnels. The profiling and selection system was implemented in early 2004 and its effectiveness has been conclusively demonstrated. It optimizes bandwidth while maintaining the anonymity of its clients. In this paper we document I2P's peer profiling and selection algorithms, discuss their strengths and weaknesses and describe directions for future work. We hope that it will be useful to guide improvements in I2P and other anonymous networks with similar features.

1 I2P Overview

I2P is an anonymous overlay network based on unidirectional encrypted tunnels. The project evolved from the Invisible Internet Project (IIP), which was a specialized IRC server transmitting data through a mix network. The project was formed to support the efforts of those trying to build a more free society by offering them an uncensorable, anonymous, and secure communication system able to operate successfully in arbitrarily hostile environments [Jr03].

Starting from a low-latency transport protocol for The Freenet Project, it grew into an independent project. I2P is primarily public domain software, but also includes code under GPL, BSD, MIT, and Apache licenses. The project is undergoing rapid development, with seven releases in 2008. The I2P software suite consists of the core router and several software packages (web server, BitTorrent clients etc.).

The routing method is called *garlic routing*. It is similar to the onion routing used by the Tor project [Tor] in that it transmits data through multiple peers, thus concealing the true IP address of the sender from the recipient and vice versa. Garlic routing allows multiple messages or *cloves* inside a single container. I2P employs a VPN-like approach that is designed primarily for communication within the network (hidden services in Tor terms). In I2P, only a few *outproxies* (exit nodes in Tor terms) operate as gateways to the standard Internet.

At the time of writing the I2P network consists of approximately 830 active peers. This number varies over time, as 50-100 new nodes join the network every day and others leave. Usually on weekends the network peaks at 1000 active nodes and on weekdays the minimum is about 700 active routers. The network has a stable base of approximately 550 active clients or *destinations*. Only one HTTP outproxy is publicly advertised and accessible (it is possible there may be private outproxies as well). There is also an IRC *inproxy* (gateway from the standard Internet) for I2P development discussion, and an email gateway. The current bandwidth used by all peers is roughly 11 MByte/sec with peaks at 16 MByte/sec and the peers build ~40,000 tunnels to route this traffic. In the last 6 months the network doubled in terms of running routers, bandwidth and tunnels. We project continued stable growth for the network.

1.1 Tunnel Overview

I2P transmits its payload through tunnels. A tunnel is a unidirectional encrypted connection through zero or more peers. Every router and destination has some incoming and outgoing tunnels. Tunnels are created for each router and destination upon initialization. Message transport may use one or more tunnels, and an application-level TCP connection is not correlated with a particular tunnel. We distinguish between *exploratory* and *client* tunnels.

1.2 Exploratory tunnels

Exploratory tunnels are generally low bandwidth, and are used by the router itself. These tunnels are used to test other tunnels, send network database (*netDb*) queries, and build client tunnels. The paragraph **Peer Selection** describes the process of tunnel building in more detail.

1.3 Client tunnels

Client tunnels transmit the payload for protocols like HTTP, IRC and others through I2P. These tunnels can be high bandwidth and reach up to 200 KByte/sec.

Each service in I2P has a *destination* as a representation of the location to reach the service. Examples are servers like eepsites (the I2P internal webpages), monotone server, IRC server or audio streaming servers, and the clients have a destination as a return address, e.g. IRC clients, bittorrent clients, monotone clients or else. Servers have stable destinations while clients destination are created new on every router restart.

Each destination has an associated set of tunnels to transport the data. The application and/or user can control the specifications of the tunnels:

- Length: The number of peers or *hops* in the tunnel; typically 2 or 3
- Length Variance: A range to be randomly added to the tunnel length at creation; typically [0-1]

- Pool Quantity: The number of tunnels in the *tunnel pool*; messages are distributed across tunnels in the pool
- Backup Tunnels: Additional tunnels may be specified in case the original tunnels die unexpectedly

The route from a client to server is divided into two tunnels: the *outgoing tunnel*, controlled by the client, and the *incoming tunnel*, controlled by the server. The connection between these two tunnels is direct; the *outbound endpoint* of the outgoing tunnel connects directly to the *inbound gateway* of the incoming tunnel. Tunnels provide anonymity by separating the destination's inbound gateway and outbound endpoint from the destination's router. By setting the length of his tunnels, every participant may control his anonymity level. Longer tunnels provide more anonymity at the cost of lower performance and reliability.

Each router selects the peers with which it builds the tunnel based on the profile of its capabilities described below. It also determines the order of peers within a tunnel, using the XOR distance of a peer's router ID from a random key, to inhibit predecessor attacks [Wr04, Wr08]. Also, two or more I2P peers within the same /16 subnet cannot be used within the same tunnel, to frustrate simple forms of collusion.

Each tunnel in I2P has a fixed lifetime of 10 minutes and will be discarded after this time. This enhances anonymity by having all tunnels look the same, to reduce the chance of correlation with a router or destination. A new tunnel will be created and added to the pool before the existing one times out. Data will be sent on one or more of the tunnels of the pool associated with the specific destination.

Peers can reject or drop tunnel build requests sent by other peers for a number of reasons (overload, shutdown in progress, limit reached, out of sync). Even within the lifetime of a tunnel these can be discarded due to overload or unexpected shutdown of one of the peers in the tunnel. To detect tunnel failure, each tunnel is tested on a periodic basis. The test sends a single 1 Kbyte message per minute, which adds a small overhead to the router traffic that is inconsequential for all but the lowest-bandwidth routers. Test messages are designed to verify tunnel connectivity and not to test tunnel bandwidth.

The tunnels on a peer are sorted into different classes:

- Exploratory tunnels: Tunnels used by the router itself
- Client tunnels: the tunnel starts or ends in this peer, bound to a server or client on the peer
- Participating tunnels: this peer is a member of one of the tunnels and the participating tunnels come in from a peer and continue through to another peer.

Three cases are possible: outbound endpoint of an outgoing tunnel, entrance or "inbound gateway" of an incoming tunnel, and an inner peer of a participating tunnel.

1.4 Network Database Overview

The NetDB is a collection of information stored about each peer and destination. It contains a *RouterInfo* structure for each peer and a *LeaseSet* (similar to the hidden service descriptor in Tor) for each known destination. LeaseSets will not be described here, for more information on this topic see the I2P website [I2P].

The RouterInfo is a small collection of all vital information describing a peer. The IP address, port, peer ID, I2P stable version number, network version, transport capabilities and some statistical data are included. The statistical data are for network diagnostics and are not trusted by routers. Each peer is set by default to 48KBps input and 24KBps output bandwidth with 90% share percentage (90% of maximum bandwidth is allowed to be used up by participating tunnels).

Each peer is sorted into a bandwidth class based on the user-configured shared bandwidth: these classes are named K, L, M, N and O. Limits are <=12 KBps for K, to >=128 KBps for O. This classification is just for network diagnostics, except that peers of class K are not used at all for routing participating tunnels. Statistics show only a small number of K routers (96 % are class L-O), so almost all peers are configured to route tunnels for others.

1.5 Tunnel Building

Each router in I2P selects a subset of peers for its tunnels. Ideally, the routers should select the fastest peers available. A simple implementation would be to allocate tunnels in proportion to the self-reported bandwidth values for each peer. This allows a simple low-resource attack where malicious nodes can report a high bandwidth so that a larger fraction of tunnels are routed through them [SB08]. As such an attack can easily attract a large number of tunnels and thus compromise anonymity [Ba07], I2P implements *peer profiling*.

2 Peer Profiling and Tiers

Peer profiling was proposed for the I2P network in December 2003 by jrandom [Jr03a]. It was introduced in the 0.3 release in March 2004 [Jr04].

Peer selection within I2P is the process of selecting the path, or sequence of other routers, for locally generated messages and their replies. This path is an ordered set of peers in a tunnel. The router maintains a database of each peer's performance, called the *profile*. The router uses that data to estimate the bandwidth of a peer, how often a peer will accept tunnel build requests, and whether a peer seems to be overloaded or otherwise unable to reliably perform. The profiling system includes mechanisms similar to the "opportunistic bandwidth measurement algorithm" proposed for Tor [SB08]. It does not require "active bandwidth probing" or generation of special-purpose traffic for measurement. Direct measurement, such as transport layer latency or congestion, is not used as part of the profile as it can be manipulated and associated with the measuring router, exposing the router to trivial attacks.

The profile contains several statistics and is continuously updated. For each statistic, averages, event counts, and maximums for several periods (for example 1 minute, 10 minute, 1 hour, and 24 hour) are available. Example statistics are: how long it takes for the peer to reply to a network database query, how often a tunnel through the peer fails, and how many new router references the peer sends. The profiles are also stored persistently on disk, so the statistics are not reset at router initialization. Also, the profile stores several timestamps, including the last time a peer was heard from, the last time it accepted a tunnel request, and the last communication error.

Much of the profile data is unused in the current software. It remains a "reserve" defense that can be easily used to enhance the router's resistance to theoretical or actual attacks, such as denial-of-service attempts, selective packet drops, network database (floodfill) disruption, and others. The profiles are a router's unique assessment of each peer's capabilities, and are not distributed to other peers or published, for this would easily compromise anonymity.

The profiles are periodically coalesced and sorted into tiers of peers that are used for various functions, as described further below.

2.1 Speed

The speed calculation simply estimates how much data can be sent or received on a single tunnel through the peer in a minute based on past performance (this may also be termed bandwidth or capacity, but we use the I2P terminology here). Specifically, it is the average of the bandwidth of the fastest three tunnels, measured over a one-minute period, through that peer. Previous algorithms used a longer measurement time and weighed recent data more heavily. Another previous calculation used total (rather than per-tunnel) bandwidth, but it was decided that this method overrated slow, high-capacity peers (that is, slow in bandwidth but high-capacity in number of tunnels). Even earlier methods were much more complex. The current speed calculation has remained unchanged since early 2006.

Only a router's locally generated and received traffic is used for these measurements - transit or "participating" traffic is not used. As it is not known if the peer before or after the router in a tunnel is a true participant or the originator or destination, that data would not be valid. Also, this could be exploited to get a router to rank some peer of their choosing as quite fast. Having a remote peer influence the rankings in this way could be dangerous to anonymity.

2.2 Capacity

An estimate of peer tunnel capacity, defined as the *number of successful tunnel builds* through the peer in a time period, is crucial to the smooth operation of an I2P router. As tunnel building is expensive, it is important to rate peers based on their willingness to accept tunnels. Peers may reject or drop tunnel requests for any number of reasons. Generally the rejection or drop is caused by a bandwidth limit (from participating tunnels or locally generated traffic), a processing (CPU) limit which manifests itself as large request processing delay, or an absolute limit on participating tunnel count.

The capacity calculation simply estimates how many tunnels the peer would agree to participate in over the next hour. The actual capacity rating, before adjustments (see below), is as follows: Let r(t) be the successful builds per hour, over a certain time period *t*:

$$R = 4*r(10m) + 3*r(30m) + 2*r(1h) + r(1d)$$

If there are no successful builds in a given time period, the value is zero. Build requests are never sent for the sole purpose of constructing the rating. The rating also includes a 'growth factor' that adds a small amount each time, so that builds through new peers are periodically attempted.

2.3 Manual Adjustments and Unused Statistics

For both the speed and capacity metrics, *bonuses* may be used to manually adjust preferences for individual peers.

There are several other statistics available in the profile that are not currently used for peer selection. These include latency for client messages, tunnel build request response time, communication error frequency, and network database lookup success rate and response time. The potential for improving peer selection based on these statistics is a topic for further research.

The router also maintains an "integration" metric reflecting the number of other peer references received from that peer. The integration metric is used to qualify a peer as a floodfill router (directory authority in Tor terms) but is not used for peer selection.

2.4 Capacity: Crime, Blame, and Punishment

Raw tunnel build capacity is not a sufficient measurement - it is essential to decrement measured capacity as a form of "punishment" for bad behavior, because tunnel build attempts are expensive, and malicious peers must be avoided. A tunnel build request is about 4KB, and the reply is identically sized. Generation of the build request also consumes significant CPU and entropy to create the cryptographic keys. As an example of the factors that must be considered:

- A peer that accepts 10 out of 10 requests is better than one that accepts 10 out of 100.
- A peer that explicitly rejects a request is better than one that drops it.
- A peer that accepts a request but later drops data through the tunnel should be avoided.

Ideally, capacity should be decremented for build request rejections, build request timeouts, and tunnel test failures. Unfortunately, a router does not know which of the tunnel peers to blame when a request or tunnel test message is dropped. Tunnel builds requests are handed off from peer to peer along the path, and since tunnels are unidirectional, a tunnel cannot be tested in isolation. What should the router do with such incomplete information?

- Naive solution: Do not blame any peer.
- Better solution: Blame each peer with equal weight.
- Best solution: Blame each peer, but use a weighted value if there is partial information available on the probability of a particular peer's fault.

The naïve solution was used in I2P for many years. However in mid-2008, we implemented the weighted blame system, as it became apparent that recognizing and avoiding unreliable and unreachable peers is critically important.

As an example, assume a tunnel build request (4 outbound hops through peers A-B-C-D) has expired. The reply was due back through the inbound exploratory tunnel (2 hops E-F). The following options are among the possibilities:

1. Any of the peers could be at fault, so blame no one.
2. Blame each of the 6 peers equally with weight 1/6.
3. Weight each tunnel equally, and distribute the blame equally in each tunnel, so blame outbound peers A-D with weight 1/8, and blame inbound peers E-F with weight 1/4.
4. Knowing that the usual failure point in I2P is an unreachable inbound gateway[1] (E in this case), blame E with weight 1/2 and the other peers with weight 1/10.

I2P now uses option 3 for build request timeouts, and option 4 for tunnel test failures in most cases. The effectiveness of these changes has been demonstrated by significant improvement in tunnel build success rates and network bandwidths in the latter half of 2008. The system works because consistently "bad" peers are discovered and avoided fairly quickly, while peers that are falsely blamed are blamed with roughly equal frequency, which does not hurt their relative ranking.

Finally, we multiply the test failures by 4 to increase the punishment for agreeing to a tunnel but then dropping data. The current calculation for capacity, r(t) for each time *t*, is:

R(t) = accepts – rejects – weighted timeouts – 4*weighted failures + growth factor

[1] Consider a test of previously built outbound tunnel A-B-C-D and inbound tunnel E-F build by router X. Tunnel build requests follow the path of the tunnel itself, therefore the act of router X building these tunnels establishes and verifies the transport connections X-A, A-B, B-C, C-D, E-F, and F-X. In addition, due to the 10-minute tunnel lifetime, and transport idle timeouts that are generally longer than that, those connections will probably remain up for the lifetime of the tunnel. The only connection that is not necessarily established in advance is the connection between the outbound endpoint (D) and the inbound gateway (E). Therefore, if a tunnel test fails, it is usually due to configuration or firewall issues at the inbound gateway.

2.5 Sorting Profiles Into Tiers

To review, exploratory tunnels are generally low-bandwidth, and are used for router operations, including building and testing other tunnels. Client tunnels are used for all user client and server traffic, including accessing internal I2P network destinations or "hidden services" such as *eepsites*, connection to external gateways (*inproxies* and *outproxies*) and other uses.

Every 30 seconds, all the profiles are sorted into three tiers:

- The *Not Failing* tier contains all peers with whom communication was attempted in the last few hours, including the following two tiers. Typical size is 300-500 peers.
- The *High-Capacity* tier includes peers with above average rating for accepting tunnel build requests, and the following tier. Typical size is 10-30 peers.
- The *Fast* tier includes peers from the High-Capacity tier whose speed rating (i.e. peak bandwidth per tunnel) is above average for all peers in the Not Failing tier. Typical size is 8-15 peers.

Both the speed and capacity metrics are skewed, with a small number of high ratings and a "long tail". By using the average and not the median, we select a small number of peers for the top two tiers.

2.6 Peer Selection

Candidate peers for tunnel builds are selected as follows:

- Client tunnels are built from peers in the Fast tier.
- Exploratory tunnels are built from peers either in the Not Failing tier or the High Capacity tier.

For exploratory tunnels, the tier selected is chosen on a per-build basis, using a weighted random function. The proportion of builds using the High Capacity tier is

(client success rate – exploratory success rate) / client success rate

As exploratory build success declines, the router builds more tunnels from the high capacity tier, to limit the amount of effort spent on the expensive tunnel build request operation. Therefore the selection maintains a balance between minimizing tunnel build requests and the need to explore peers.

It may seem inadvisable to use the Not Failing tier (generally the lowest-bandwidth, lowest-capacity peers) for exploratory tunnels, since these tunnels are required to function for a router to build client tunnels. However, failing exploratory tunnels are recognized quickly, so this is not a significant limitation. Using all peers for exploratory tunnels provides I2P a system of opportunistic bandwidth and capacity measurement.

Peers are selected with equal weight within each tier. If a sufficient number of peers for a tunnel are not found within a given tier, the peer selection moves on to the next-lower tier.

3 Performance Evaluation and Further Work

As shown in available network statistics, the recent improvements to the I2P profiling and peer selection system have significantly improved bandwidth and other performance metrics [Stats]. The core profiling system, including the "opportunistic bandwidth measurement algorithm" proposed in [SB08], has been in place since early 2004.

I2P does not use claimed bandwidth, which eliminates a class of low-resource attacks. While we have not included here experimental data demonstrating that the selection system is effective, it is readily apparent to the authors that in a broad range of experimental conditions, the members of the Fast and High Capacity tiers are those peers with high claimed bandwidth (class O). In our opinion the current bandwidth constraints within I2P are not caused by poor peer selection, but lie elsewhere[2]. The basic peer selection method has been in place for five years, and has been tuned only modestly in the last two years.

The algorithm is stable; the Fast and High Capacity tier members do not change rapidly. When a router uses a peer for tunnels, it tends to increase that peer's speed and capacity metric, thus keeping that peer in the High Capacity tier. This is desirable for anonymity, as using a large or rapidly varying set of peers for tunnels would increase vulnerability to predecessor attacks by increasing the odds that an attacker will eventually be a participant in a tunnel [Wr04, Wr08].

The tier system reacts quickly to individual peer failure or overload, and to increased local demand for bandwidth. The speed and capacity metrics are strongly weighted for recent performance. When a peer starts to drop test messages, or fails to respond to tunnel build requests, it will quickly be demoted out of the high-capacity pool. As bandwidth demand increases, the speed metric for individual peers will rapidly increase, and the fast tier will quickly become reorganized to include the newly recognized fast peers.

The tier system tends to use the highest-bandwidth peers when the network is not congested. As congestion increases, the total network traffic "spreads out" to lower-capacity peers. From an overall network perspective, this is optimal as it maintains a similar level of service for all routers.

The profiling system does not over-optimize. The router uses its own, normally generated traffic for peer profiling. No high-bandwidth test messages are required or used. When a router does not require high bandwidth or a high number of tunnels,

[2] Lock contention, memory usage, issues in the internal TCP implementation (*streaming library*), network database inefficiencies, protocol overhead, message fragmentation, message dropping strategies, UDP transport issues, and others – some of which may be topics for future papers.

the metrics for each peer are correspondingly lower. Therefore, even a low-bandwidth peer may be classified as "fast" and used for tunnels. This tends to, at least partially, spread low-bandwidth tunnels broadly across the network, and leaves the high-capacity peers for high-speed traffic. However, even very-low-bandwidth routers tend to accurately find a few fast peers and thus are well prepared when higher bandwidth is demanded.

Also, there is no need for a complete global optimum. I2P routers know only a subset of the active peers in the network, generally 20% to 80%. Through exploratory tunnel building and other peer discovery mechanisms[3], routers have no difficulty finding a sufficient portion of peers, and preventing network partitioning. As the I2P network grows the percentage of peers known to each router will decline as we implement additional mechanisms to limit memory usage, TCP and UDP connections, and other resources in the router. This poses no threat to the profiling system.

The profiling system is persistent across restarts, and maintains measurements for the last 24 hours. This allows a recently started router to quickly re-integrate to the network, whether the router was just restarted or has been down for some time.

The network performance is sensitive to adjustments of the parameters, weighting, and calculations. It is difficult to test and debug in a distributed network, and may be impossible to fully optimize. The I2P router contains a framework for local network simulation and testing; however, we have not used this framework for profiling and selection testing. As described above, the routers include bandwidth and tunnel build success statistics in the network database entry they publish to the floodfill routers. While this information is not trusted or used in the router, it is gathered by the stats.i2p website [Stats]. On that website, several network performance graphs are presented, and the I2P developers rely on this facility to monitor the network and judge the effectiveness of software changes in each release.

The basic measurements have been greatly simplified in the process of development. The speed calculation, for example, was at one time over 400 lines of code, and it is now only a few lines.

The punishment for bad behavior keeps the network running well, but also is an area for further research. How heavily a router punishes determines how fast the load spreads out across the network as the load increases, and how quickly an overloaded peer is avoided. The implementation contains an implicit estimate of the cost of a tunnel build request, as it rates the relative performance of a rejected requests and a dropped request. It also weighs the costs of an accepted request vs. a request not made at all. One possibility is to establish a baseline of a peer that has never been asked to participate in a tunnel, then consider percentage (or absolute number) of

[3] A newly installed router downloads RouterInfo structures out-of-network through a process called *reseeding*. A router accepting tunnel build requests learns about the previous and next peers in the tunnel. A router acting as an outbound endpoint learns about the inbound gateway when it must route a message to that gateway, and vice versa. A router also periodically queries the floodfill peers for a random key, and the reply will contain routers close to that key, a process called *exploration*.

request rejections, dropped requests, and test failures is required to drop the capacity rating below the baseline.

If peers are punished too heavily, the network will tend to congestion collapse as most peers are driven to negative capacity ratings, tunnel load spreads quickly throughout the network, and routers attempt to route tunnels through very-low-capacity peers. If peers are punished too lightly, routers will be slow to react to overloaded peers, and maintain the same high capacity peers for too long by accepting poor performance from a peer even when better peers may be available.

We recommend that those wishing to implement a profiling and selection system start with relatively simple algorithms, and add complexity later if necessary. I2P’s development has sometimes taken the reverse path; for example, I2P's speed calculation used to be several pages of code, now it is quite simple.

The evaluation of a distributed anonymous network’s performance is difficult but not impossible. A more formal measurement of I2P’s peer selection algorithms, either on the real network, an isolated test network, or simulation, would be a valuable extension to the analysis in this paper.

4 Conclusions

I2P routers accurately discover fast peers for tunnel routing without trusting claimed bandwidth or generating large amounts of traffic for testing. When a router requires little bandwidth, the precision of its peer selection is unimportant. When the router does require more bandwidth, the selection will be correspondingly better. Even very-low-bandwidth routers tend to accurately find fast peers and thus are well prepared when higher bandwidth is demanded.

To use the terms of [SB08], I2P's peer profiling and selection system is an opportunistic bandwidth measurement algorithm that is sensitive to network load and client demand. It does not use self-reported values. However it does not provide a "tunable" mechanism for users to trade off anonymity and performance. I2P provides alternate method (tunnel length configuration) for the user to make that adjustment. Not only is active bandwidth probing (i.e. generating large amounts of special-purpose data for testing is not practical, as [SB08] states, it is not necessary. In addition to the bandwidth measurements proposed in [SB08], I2P measures tunnel build acceptance rate, with adjustments for various bad behavior by peers. I2P's profiling and selection system has been in continuous use for approximately five years.

While the system works well, several improvements are possible. The authors will continue to research, evaluate, and tune I2P's peer profiling and selection system in the coming months.

References

[Ba07] K. Bauer, D. McCoy, D. Grunwald, T. Kohno, and D. Sicker: Low-resource routing attacks against anonymous systems, Proceedings of the 2007 Workshop on Privacy in the Electronic Society (WPES), 2007.

[I2P] This paper includes material from the I2P website http://www.i2p2.de/ by jrandom and others.

[IIP] http://invisibleip.sourceforge.net/iip/, http://en.wikipedia.org/wiki/Invisible_IRC_Project

[Jr03] jrandom: Invisible Internet Project (I2P) Project Overview, August 28, 2003 http://www.i2p2.de/_static/pdf/i2p_philosophy.pdf

[Jr03a] jrandom: I2P Development Meeting 68, December 9, 2003 http://www.i2p2.de/meeting68

[Jr04] jrandom: I2P Development Meeting 82, March 23, 2004 http://www.i2p2.de/meeting82

[SB08] Snader, R.; Borisov, N.: A Tune-up for Tor: Improving Security and Performance in the Tor Network, Proceedings of the Network and Distributed Security Symposium - NDSS '08, February 2008.

[Stats] http://stats.i2p/ or http://stats.i2p.to/

[Tor] http://www.torproject.org/

[Wr04] Wright, M.; Adler, M.; Levine, B.N.; Shields, C.: The Predecessor Attack: An Analysis of a Threat to Anonymous Communications. In ACM Transactions on Information and System Security (TISSEC) **4**(7), November 2004, pages 489-522.

[Wr08] Wright, M.; Adler, M.; Levine, B.N.; Shields, C.: Passive-Logging Attacks Against Anonymous Communications Systems.