Tor Development Roadmap: Wishlist for 2008 and beyond

Roger Dingledine Nick Mathewson

January 10, 2008

1 Introduction

Tor (the software) and Tor (the overall software/network/support/document suite) are now experiencing all the crises of success. Over the next years, we're probably going to grow even more in terms of users, developers, and funding than before. This document attempts to lay out all the well-understood next steps that Tor needs to take. We should periodically reorganize it to reflect current and intended priorities.

2 Everybody can be a relay

We've made a lot of progress towards letting an ordinary Tor client also serve as a Tor relay. But these issues remain.

2.1 UPNP

We should teach Vidalia how to speak UPNP to automatically open and forward ports on common (e.g. Linksys) routers. There are some promising Qt-based UPNP libs out there, and in any case there are others (e.g. in Perl) that we can base it on.

2.2 "ORPort auto" to look for a reachable port

Vidalia defaults to port 443 on Windows and port 8080 elsewhere. But if that port is already in use, or the ISP filters incoming connections on that port (some cablemodem providers filter 443 inbound), the user needs to learn how to notice this, and then pick a new one and type it into Vidalia.

We should add a new option "auto" that cycles through a set of preferred ports, testing bindability and reachability for each of them, and only complains to the user once it's given up on the common options.

2.3 Incentives design

Roger has been working with researchers at Rice University to simulate and analyze a new design where the directory authorities assign gold stars to well-behaving relays, and then all the relays give priority to traffic from gold-starred relays. The great feature of the design is that not only does it provide the (explicit) incentive to run a relay, but it also aims to grow the overall capacity of the network, so even non-relays will benefit.

It needs more analysis, and perhaps more design work, before we try deploying it.

2.4 Windows libevent

Tor relays still don't work well or reliably on Windows XP or Windows Vista, because we don't use the Windows-native "overlapped IO" approach. Christian King made a good start at teaching libevent about overlapped IO during Google Summer of Code 2007, and next steps are to a) finish that, b) teach Tor to do openssl calls on buffers rather than directly to the network, and c) teach Tor to use the new libevent buffers approach.

2.5 Network scaling

If we attract many more relays, we will need to handle the growing pains in terms of getting all the directory information to all the users.

The first piece of this issue is a practical question: since the directory size scales linearly with more relays, at some point it will no longer be practical for every client to learn about every relay. We can try to reduce the amount of information each client needs to fetch (e.g. based on fetching less information preemptively as in Section 3.1 below), but eventually clients will need to learn about only a subset of the network, and we will need to design good ways to divide up the network information.

The second piece is an anonymity question that arises from this partitioning: if Tor's security comes from having all the clients behaving in similar ways, yet we are now giving different clients different directory information, how can we minimize the new anonymity attacks we introduce?

2.6 Using fewer sockets

Since in the current network every Tor relay can reach every other Tor relay, and we have many times more users than relays, pretty much every possible link in the network is in use. That is, the current network is a clique in practice.

And since each of these connections requires a TCP socket, it's going to be hard for the network to grow much larger: many systems come with a default of 1024 file descriptors allowed per process, and raising that ulimit is hard for end users. Worse, many low-end gateway/firewall routers can't handle this many connections in their routing table.

One approach is a restricted-route topology [1]: predefine which relays can reach which other relays, and communicate these restrictions to the clients. We would need to compute which links are acceptable in a way that's decentralized yet scalable, and we would need an efficient (compact) way to characterize the topology information so all the users could keep up to date.

Another approach would be to switch to UDP-based transport between relays, so we don't need to keep the TCP sockets open at all. Needs more investigation too.

2.7 Auto bandwidth detection and rate limiting, especially for asymmetric connections.

2.8 Better algorithms for giving priority to local traffic

Proposal 111 made a lot of progress at separating local traffic from relayed traffic, so Tor users can rate limit the relayed traffic at a stricter level. But since we want to pass both traffic classes over the same TCP connection, we can't keep them entirely separate. The current compromise is that we treat all bytes to/from a given connectin as local traffic if any of the bytes within the past N seconds were local bytes. But a) we could use some more intelligent heuristics, and b) this leaks information to an active attacker about when local traffic was sent/received.

2.9 Tolerate absurdly wrong clocks, even for servers

2.10 First a bridge, then a public relay?

Metrics for deciding when you're fast enough and stable enough to opt to switch from being a bridge relay to a public relay.

2.11 Risks from being a relay

3 Tor on low resources / slow links

3.1 Reducing directory fetches further

3.2 AvoidDiskWrites

3.3 Using less ram

3.4 Better DoS resistance for tor servers / authorities

4 Blocking resistance

4.1 Better bridge-address-distribution strategies

4.2 Get more volunteers running bridges

4.3 Handle multiple bridge authorities

4.4 Anonymity for bridge users: second layer of entry guards, etc?

4.5 More TLS normalization

4.6 Harder to block Tor software distribution

4.7 Integration with Psiphon

5 Packaging

5.1 Switch Privoxy out for Polipo

- Make Vidalia able to launch more programs itself

5.2 Continue Torbutton improvements

especially better docs

5.3 Vidalia and stability (especially wrt ongoing Windows problems)

5.4 Polipo support on Windows

5.5 Auto update for Tor, Vidalia, others

5.6 Tor browser bundle for USB and standalone use

5.7 LiveCD solution

5.8 VM-based solution

5.9 Tor-on-enclave-firewall configuration

5.10 General tutorials on what common applications are Tor-friendly

5.11 Controller libraries (torctl) plus documentation

5.12 Localization and translation (Vidalia, Torbutton, web pages)

6 Interacting better with Internet sites

6.1 Make tordnsel (tor exitlist) better and more well-known

6.2 Nymble

6.3 Work with Wikipedia, Slashdot, Google(, IRC networks)

6.4 IPv6 support for exit destinations

7 Network health

7.1 torflow / soat to detect bad relays

7.2 make authorities more automated

7.3 torstatus pages and better trend tracking

7.4 better metrics for assessing network health / growth

- geoip usage-by-country reporting and aggregation (Once that’s working, switch to Directory guards)

8 Performance research

8.1 Load balance better

8.2 Improve our congestion control algorithms

8.3 Two-hops vs Three-hops

8.4 Transport IP packets end-to-end

9 Outreach and user education

9.1 ”Who uses Tor” use cases

9.2 Law enforcement contacts

- ”Was this IP address a Tor relay recently?” database

9.3 Commercial/enterprise outreach. Help them use Tor well and not fear it.

9.4 NGO outreach and training.

- ”How to be a safe blogger”

9.5 More activist coordinators, more people to answer user questions

9.6 More people to hold hands of server operators

9.7 Teaching the media about Tor

9.8 The-dangers-of-plaintext awareness

9.9 check.torproject.org and other ”privacy checkers”

9.10 Stronger legal FAQ for US

9.11 Legal FAQs for other countries

10 Anonymity research

10.1 estimate relay bandwidth more securely

10.2 website fingerprinting attacks

10.3 safer e2e defenses

10.4 Using Tor when you really need anonymity. Can you compose it with other steps, like more trusted guards or separate proxies?

10.5 Topology-aware routing; routing-zones, steven’s pet2007 paper.

10.6 Exactly what do guard nodes provide?

Entry guards seem to defend against all sorts of attacks. Can we work through all the benefits they provide? Papers like Nikita’s CCS 2007 paper make me think their value is not well-understood by the research community.

11 Organizational growth and stability

11.1 A contingency plan if Roger gets hit by a bus

- Get a new executive director

11.2 More diversity of funding

- Don't rely on any one funder as much - Don't rely on any sector or funder category as much

11.3 More Tor-funded people who are skilled at peripheral apps like Vidalia, Torbutton, Polipo, etc

11.4 More coordinated media handling and strategy

11.5 Clearer and more predictable trademark behavior

11.6 More outside funding for internships, etc e.g. GSoC.

12 Hidden services

12.1 Scaling: how to handle many hidden services

12.2 Performance: how to rendezvous with them quickly

12.3 Authentication/authorization: how to tolerate DoS / load

13 Tor as a general overlay network

13.1 Choose paths / exit by country

13.2 Easier to run your own private servers and have Tor use them anywhere in the path

13.3 Easier to run an independent Tor network

14 Code security/correctness

14.1 veracode

14.2 code audit

14.3 more fuzzing tools

14.4 build farm, better testing harness

14.5 Long-overdue code refactoring and cleanup

15 Protocol security

15.1 safer circuit handshake

15.2 protocol versioning for future compatibility

15.3 cell sizes

15.4 adapt to new key sizes, etc

References

[1] George Danezis. Mix-networks with restricted routes. In Roger Dingledine, editor, *Privacy Enhancing Technologies (PET 2003)*. Springer-Verlag LNCS 2760, 2003.