Tor Development Roadmap, 2008-2011

Roger Dingledine

Contents

1 Introduction

This document describes Tor research and development items that should happen on a three-year timeframe to move Tor forward at being both a useful circumvention tool and a useful privacy tool.

There are two goals to the items in this roadmap. First, we want to make sure to continue adapting Tor to changing environments so it can continue to be a useful tool right now and for the next few years — people are trying to deploy it and use it today, and we want to make sure it stays working well. Second, we want to tackle some of the long-term issues that have been holding Tor back from being useful to a broad set of people — issues that will require some investment now, but will ultimately pay off in creating a more sustainable and automated tool. The tasks here aim to include the right mix between these near-term-useful items and the items that will need several years of design and analysis before they can become useful.

1.1 Outline

There are many pieces to this project. Section 2 describes the core development work that needs to be done to allow us to extend to new features and designs. Sections 3 and 4 cover performance and robustness improvements: load balancing so we can use the available Tor relays in a way that keeps all the traffic moving quickly, making Tor relays work on Windows without crashing, encouraging more users to become relays or bridges, continued scalability work, and work to continue reducing the overhead of directory information. Section 5 adds more circumvention features, focusing on the critical gaps in the current design. Section 6 works on client safety, that is, steps to make sure that educated and prepared users can possibly be secure while using Tor. Section 7 is then client usability: how to make it easier for more ordinary users to still have safety while using Tor, and how to make it more convenient for them to set it up. Last, Section 8 covers not-so-technical development and support work that needs to be done in parallel with the technical development.

1.2 What the various annotations mean

This roadmap aggregates funding and priorities from several different sources. The main sources are funding from IBB and Anonymous Sponsors to focus on Tor as a circumvention tool. The other smaller pieces are funding from Google to work on free software development, funding from NLnet to work on directory scalability and on hidden services, and funding from NRL to work on more basic research questions about anonymity and path selection. Getting funding from multiple sources who care about related goals lets us focus not just on the core development, but also on the future features and development that can move us past 'maintenance' level.

In the timelines below, "Year0.5" runs from the beginning of September 2008 to the end of February 2009; "Year1.0" runs from beginning of March 2009 to end of August 2009; etc. For tasks within the one year timeframe, we have also specified which key engineer will be taking the lead on that task. We'll choose key engineers to lead later tasks once we get closer to them.

2 Core development

In order for The Tor Project to continue putting out high-quality software, we need to decide on a roadmap for new features, improvements, and cleanups for each release; prioritize them; and periodically refine the roadmap[1]. Part of the iterative cleanup process is fixing bugs, doing testing and writing new unit tests, maintaining existing features, and interacting with users and relay operators to refine our requirements. Another part of the process is coordinating with volunteer and part-time developers to make good use of their energies. More generally, we must be ready to solve whatever other problems and related projects come up. This core development work is what keeps Tor moving forward and keeps new releases coming out.

We have two special focuses for our core development work over the course of this project.

[1] See e.g. https://svn.torproject.org/svn/tor/trunk/doc/TODO.021

The first focus is on users, especially users in blocked and partially censored environments. How can we give them reliable and adequate performance when they connect to the Tor network, including when they are using obsolescent hardware? This step also includes adding other features that don't warrant their own sections below, such as IPv6 support for destinations and various other Tor Proposals[2]. **(IBB: Year0.5 to Year1.5, high cost.)**

The second focus is on relays, especially relays and bridge relays run by ordinary Tor users on Windows who use Vidalia for configuration. We've added a lot of features lately that make this process simpler, but we need to make sure to keep these features working well as we work on the next components. For example, we must make sure to keep our resource load (especially memory and CPU) low so Tor relays remain practical on cheap and old hardware — as we add new features that demand more resources, we need to be constantly working on ways to maintain good efficiency and size. **(IBB: Year0.5 to Year1.5, high cost.)**

While this section is funded at an adequate level to keep everything moving, we could still make use of more core development funding: there are many good volunteer developers who would be much more productive if we could give them more attention and help them become core developers. Getting these people up to speed is also a necessary step of figuring out which ones we should pay if we end up with more money, or if we end up with sponsors with specific target results in mind. **(Needs a sponsor: Year1.0, high cost. Medium priority.)**

3 Improve performance through better use of the current network

3.1 Improve Path Selection and Load Balancing

There are many steps to the path selection and load balancing problem. Research needs to be done on integrating load balancing feedback with path selection. Johannes Renner did some initial work here during his 2007 Summer of Code project to extend the Torflow library, and Nikita Borisov of University of Illinois published a paper at NDSS 2008 in this direction [8]. Most recently, Steven Murdoch published a paper at PETS 2008 in Leuven refuting the load balancing aspect of Nikita's design [7].

We shouldn't expect immediate results on this research, because it first requires more measurements and more analysis. We could make some great progress on this in the next year or two though — and since making good use of available resources in the network is one of the critical next steps in making Tor scale better, we should focus on it.

First we need to examine all these proposed algorithms more closely, design new ones as necessary, and simulate the effects of the ones that seem most promising. One example here is looking at 2-hop paths vs 3-hop paths, and comparing the usability, performance, and average network load benefits against the potential for decreased anonymity. Another example is looking at choosing the middle hop of the path based on latencies, or geographical or network locality, to again trade off performance for anonymity. The goal is to identify strategies where we can make a small compromise in expected anonymity for a large expected performance gain. Part of the challenge here is to understand what sort of anonymity metrics we should use, so they both reflect reality and are practical to compute.

Good load balancing improvements could double or more the average throughput of Tor users — mainly because the variances that Tor clients are seeing right now are so high, perhaps because a small number of relays are extremely overloaded. Yet other topics to look at include a distributed bandwidth estimation protocol based on Nikita's NDSS paper, and ways to prioritize traffic or circuits to be more fair to as many users as possible.

Another design component we need to consider while we're working on these load balancing algorithms is susceptibility to attack. That is, right now Tor relays self-advertise whatever bandwidth they want. This has led to a variety of attack papers where an attacker signs up an allegedly high-resource relay and attracts a lot more traffic than he otherwise could have handled [2]. So while we design load balancing schemes above, we should aim for ones where the weight for each relay is measured rather than just declared – this could be accomplished for example via community consensus or via a threshold of authorities.

[2] https://svn.torproject.org/svn/tor/trunk/doc/spec/proposals/000-index.txt

Another topic to tackle is finishing Fallon Chen's circuit timeout work[3]. Her 2008 Google Summer of Code work helped characterize the distribution of circuit build times — the goal is to discard circuits that take more than a standard dev longer than they should, because we already know that they are going to be particularly crummy circuits. Alas, the implementation and verification remain to be done. This step could conceivably reduce the variance of circuit latency a whole lot.

3.1.1 Organize, sort, and prepare

For the Year0.5 deliverable, the main goal here is to lay out all the directions we need to explore, and for each item guess the amount of work it'll take and how useful we think it'll be. Put priorities on each, and for each one describe a path of how to get from here to there. We should also brainstorm about what other measurements and metrics we should gather over time that will let us make better decisions on this topic.

(IBB/Anonymous Sponsors: Year0.5, medium cost. Led by Steven.)

For the Year1.0 deliverable, we will work on the low hanging fruit from the Year0.5 report, and also get some longer term projects going.

(IBB/Anonymous Sponsors: Year1.0, medium cost. Led by Mike.)

(Anonymous Sponsors: Year1.5 and Year2.0, medium cost.)

3.1.2 Measure circuits and streams in the wild

At the same time we should build an automated infrastructure to measure and track performance (both bandwidth and latency) in the network over time, so we can observe trends and see what effects our modified algorithms are producing in the network once we deploy them.

This data will be useful for this section in terms of advising us on how to change the designs, and it will also be used in the Metrics work in Section 5.7 so we can graph the health and progress of the network over time and notice trends.

(Anonymous Sponsors: Year1.5 to Year2.5, medium cost. Led by Mike.)

3.1.3 Implement and deploy

The final step would involve implementing and deploying new better load balancing and performance-improving algorithms that improve (or at least don't hinder) both performance and anonymity. **(Anonymous Sponsors: Year2.0 to Year3.0, high cost.)**

(Needs a sponsor: high cost. High priority.)

3.2 Tor over UDP, UDP over Tor

Moving to using UDP transport in Tor will provide huge advantages to performance, since user connections will do end-to-end congestion control and we should be able to fit many more connections onto a Tor network with a given capacity, and since we will tolerate dropped packets without slowing down every stream over the connection that dropped the packet. Moving to UDP transport in Tor will also provide scalability advantages, because each Tor relay doesn't need to hold open a TCP socket to each other Tor relay – meaning we can increase the network capacity, and thus again increase performance. Further, more relays (in particular more diversity of relays) leads to better anonymity for users.

Another advantage of moving to UDP transport is that Tor would now be able to handle connections for UDP-based applications like Skype (VoIP in general), OpenVPN, DNS, etc.

We've been working with Ian Goldberg at Waterloo. He was the Chief Scientist at Zero-Knowledge Systems, a company that deployed a UDP-based anonymity system called the Freedom network that was quite like Tor. He has several grad students who want to work on exactly this problem, and Joel Reardon has written his Master's thesis investigating it.

[3]https://svn.torproject.org/svn/tor/trunk/doc/spec/proposals/151-path-selection-improvements.txt

The first step would be to rederive how the Freedom network worked, combine that with ideas from the above two research groups, and produce a design and specification for how to pass Tor traffic over UDP. This step involves adding sequence numbers and MACs to each packet as it traverses the Tor network, handling and retrying dropped packets during the circuit-level crypto handshake, etc. The second step would be to analyze the design with respect to Tor's current security properties, including perfect forward secrecy from the current circuit handshake, not partitioning traffic across multiple connections, etc. Step three is to figure out a migration plan that allows us to move to the new design within a year or so, and doesn't harm user security or performance too much during migration. (For example, some designs we've seen involve a "flag day" where all users and servers stop using one network and start using a different one.) Then we iterate steps one, two, and three until we get a realistic design that still has adequate security properties.

(IBB: Year0.5, low cost. To fund Joel's thesis work with supervision by Ian (matched 4x by the Canadian government!))

(IBB: Year0.5, low cost. Brainstorming the roadmap here.)

However, there's still a big gap: since we're transporting TCP and UDP packets end-to-end and just writing them onto the network on each side, the details of the TCP stack used on the client side becomes relevant. Operating systems like Windows, OS X, and Linux choose sequence numbers, source ports, and other connection properties in a predictable way, meaning the exit relay, the destination site, or somebody in between can observe the traffic and discover that two connections are coming from the same user. This attack probably works across different exit relays, and probably works across time (e.g. users with a given timestamp skew will probably retain it later too[4]).

(IBB: Year1.5, medium cost. Take the work Joel and others have been doing and try to turn it into a realistic design and development roadmap. What are the critical missing pieces and what are the steps required to resolve them? Just how hard would it be to maintain a user-space TCP library? Etc.)

So step four is to find, adapt, and/or write a user-space TCP stack that can rewrite and normalize TCP and UDP packets and streams so they no longer contain these identifying properties. This is a huge task, and we shouldn't really plan it until the above phase has given us better intuition. **(Needs a sponsor: Very high cost. User-space TCP stack. Low priority.)**

At this point, we will have a convincing design for how to migrate to UDP for connections between Tor relays, and for Tor clients that are in a position to use UDP. But clients in censored areas may still not be able to use UDP for their first hop, because it will have an unusual network footprint. These users would still benefit from most of the above advantages (improved performance inside the Tor network, improved scalability and thus improved anonymity), but they wouldn't get what is perhaps the most important benefit for them, which is tolerating high packet loss to their first hop.

So step five is to reverse engineer Skype, or pick a different popular UDP-based app that is allowed through most firewalls, figure out what security properties it's missing (for example, I bet its crypto handshake doesn't provide perfect forward secrecy), and then either figure out how to achieve our security properties while looking like Skype traffic, or decide to have a second inferior handshake that would provide less security to these users and also partition them from the rest of the Tor user base. Then we iterate steps one-three above until our new protocol seems like the right one to deploy. **(Needs a sponsor: High cost. Design and papers and review. Low priority.)**

Step six is then to implement, deploy, and manage the migration. **(Needs a sponsor: High cost. Implement and deploy. Low priority.)**

3.3 Hidden service performance and reliability

Tor Hidden Services allow users to set up anonymous information services, like websites, that can only be accessed through the Tor network and are protected against identification of the host that runs the services. The most critical limitations of Tor Hidden Services are the time it takes until a Hidden Service is registered in the network and the latency of contact establishment when accessed by a user. Due to design issues in the

[4]https://wiki.torproject.org/noreply/TheOnionRouter/TorFAQ#PhysicalFingerprint

original Tor protocol, the connection to a new Hidden Service can take several minutes, which leads most users to give up before the connection has been established. Using Tor Hidden Services for direct interactive user-to-user communication (e.g. messaging) is nearly impossible due to the high latency of Hidden Service circuit setup.

This project aims at speeding up Tor Hidden Services by improving the way Tor circuits are set up between the user and the Hidden Service as well as the way a Hidden Service is registered in the Tor network. In a first step precise diagnostics of the behavior of the Hidden Services in lab setups and real world situations will be conducted to find the root causes of the bad timing effects. Based on these diagnostics, optimization strategies will be designed and verified for unwanted implications for the security and anonymity of the Tor network. Precise success metrics will be developed in the diagnostics phase, after it becomes clear where the time is lost and what improvements are realistic.

(NLnet: Year0.5, medium cost. Led by Karsten.)

Still need to add in authorization, many further steps to improving hidden service performance and reliability. [say more] **(Needs a sponsor: Year1.0, medium cost. Led by Karsten.)**

4 Improve performance through more capacity

Better performance comes from increased network capacity, and better security comes from increased network diversity.

4.1 Relay stability on Windows

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 many volunteers have been working lately on related modifications to the libevent library. The next steps are to A) finish that work, B) teach Tor to do OpenSSL calls on buffers rather than interacting directly with the network, and C) teach Tor to use the new libevent buffers approach. In particular, we can imagine the following roadmap:

- 1. Update libevent to include support for an IOCP backend for its buffering logic.
  - a. Update libevent's bufferevents code to support multiple backends.
  - b. Add a generic multithreading support mechanism for use by libevent backends that need multiple threads.
  - c. Make libevent's bufferevents base logic threadsafe, as needed.
  - d. Merge, adapt, and rewrite IOCP code.
  - e. Evaluate performance and optimize.
- 2. Update libevent's buffering logic so that it can support SSL-over-buffers, as Tor requires.
  - a. Determine appropriate APIs for buffering abstraction. Niels and Nick are discussing filtering versus function tables as an appropriate mechanism.
  - b. Implement these APIs.
  - c. Implement an OpenSSL wrapper with the flexibility we need.
- 3. Other libevent hacking as may be required.
  - a. Minimize uses of "pullup" function on libevent buffers.
  - b. Evaluate whether any of Tor's buffer RAM hacks (like freelists) are actually useful for libevent.
  - c. Make sure that all of the functions currently implemented in Tor's buffers can be trivially cloned by libevent's.
- 4. Update Tor to take advantage of new libevent features as available.
  - a. Rewrite Tor's basic networking layers so that it contains an element that behaves equivalently to libevent's buffers. This process will inform 3c and 2a above, and may require us to revisit those steps with iterative refinements.

- b. Have this layer use new libevent code when it's present and efficient.
- c. Evaluate performance; continue to optimize.

(Anonymous Sponsors: Year1.0, medium cost. For a first cut of the above steps.)
(Anonymous Sponsors: Year1.5 to Year2.5, medium cost. Make sure it works smoothly. Led by Nick.)

4.2 Tor clients that find themselves reachable and reliable should automatically become a bridge or relay.

We've made a lot of progress towards letting an ordinary Tor client also serve as a Tor relay, and we will continue to make progress as we move forward. There are several more topics that need investigation still:

4.2.1 Risks from being a relay

Understand the risks from letting the attacker send traffic through your relay while you're also initiating your own anonymized traffic. Three different research papers [1, 5, 6] describe ways to identify the relays in a circuit by running traffic through candidate relays and looking for dips in the traffic while the circuit is active. These clogging attacks are not that scary in the Tor context so long as relays are never clients too. But if we're trying to encourage more clients to turn on relay functionality too (whether as bridge relays or as normal relays), then we need to understand this threat better and learn how to mitigate it.

One research direction is to investigate the RelayBandwidthRate feature that lets Tor rate limit relayed traffic differently from local traffic. Since the attacker's "clogging" traffic is not in the same bandwidth class as the traffic initiated by the user, it may be harder to detect interference. Or it may not be.

We aren't really comfortable setting users up en masse as bridges or relays until we understand these issues more.

At the end of year1, we're going to start on the proposal in Section 4.2.2 below for how exactly Tors will measure themselves and decide that they should elect to be a bridge relay and/or normal relay. That is, at the end of year1 we'd like to have a clear plan for how users who become relays will be safe. We want to be confident that we can build this plan.

This means the main tasks for the research in year1.0 are: a) evaluate all the various attacks that are made possible when an attacker can use you as a relay, and b) identify ways to make them not a big deal. Then we should c) pick the right way or ways, and spec them out such that we believe they will work and be possible to implement.

This work might involve simulation, might involve analysis, will probably involve messing with Tor's round-robin read/write algorithms, etc.

(IBB/Anonymous Sponsors: Year1.0, medium cost. Led by Steven.)
(Anonymous Sponsors: Year2.0 to Year2.5, high cost.)

4.2.2 First a bridge, then a relay.

I want all clients to start out automatically detecting their reachability and opting to be bridge relays. Then if they realize they have enough consistency and bandwidth, they should automatically upgrade to being non-exit relays.

(Anonymous Sponsors: Year1.5, low cost. Write a proposal. Led by Roger.)
(IBB: Year1.5, low cost. Build a development roadmap for what internal Tor pieces will need to change and how they should change.)
(Anonymous Sponsors: Year2.0 and Year2.5, medium cost.)
(Needs a sponsor: Lots.)

4.3 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.

However, the current incentives design we invented has a serious flaw, which is that the set of gold-starred relays is known to the adversary, and over time he can narrow down which gold-star users are always the ones online when a certain activity (e.g. posting to a blog) happens. We need to revamp the design so the set of high-priority users and the set of currently online relays is less clearly related.

Also under this heading are 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 connection 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.

(IBB/Anonymous Sponsors: Year0.5, medium cost. Revise and publish the current draft and start gathering comments and new designs. Led by Roger.)

(Needs a sponsor: We could move up the timeline here with more funding and more attention. This is potentially a very large (but very useful) area to tackle.)

4.4 Continue research on how to splinter the network as it grows so we can maintain a good balance of both anonymity and scalability.

This topic is probably the hardest open research problem in the field right now. We need to enumerate and analyze the various solutions we've come up with already, and work on new better solutions.

Step one is to specify the details for the simple version: partition the networkstatus documents as they get too large, and have clients fetch and use only a single partition, and have mirrors only cache descriptors from within their partition. Analyze the scalability, performance, and anonymity properties therein. We've made a start in the recent PET 2008 paper by Danezis and Syverson [3].

(Anonymous Sponsors: Year1.5 to Year2.0, medium cost. Write a proposal, led by Peter.)

(Needs a sponsor: We'll look for more money in Year3 to revise and/or build it.)

4.4.1 Blending the network partitions

Step two is to research variants of this design that "blend" multiple partitions together. [more detail here]
(Needs a sponsor: High cost. Medium priority.)

4.5 Clients download less directory info. Especially useful for clients on modems.

See "piece one" in https://www.torproject.org/svn/trunk/doc/spec/proposals/ideas/xxx-grand-scaling-plan.txt

The challenge here is that many of the design decisions for this topic impact other scalability and partitioning decisions down the road: that is, what we do here will decide what options we have for all the other designs. So we need to think very carefully so we don't introduce new vulnerabilities and so we don't preclude other design changes.

The start of this design is Proposal 138[5] which will cut the size of the networkstatus from about 2500 relays to 1500 relays by removing the ones that most clients won't download; Proposal 140[6] which will let clients only download a diff of one consensus to the next, rather than downloading a whole new consensus;

[5] https://svn.torproject.org/svn/tor/trunk/doc/spec/proposals/138-remove-down-routers-from-consensus.txt
[6] https://svn.torproject.org/svn/tor/trunk/doc/spec/proposals/140-consensus-diffs.txt

and Proposal 141[7] which lays out a plan for the just-in-time descriptor download design, which would cut out a huge amount of the directory overhead.

(NLnet: Year0.5, medium cost. Analysis of the various designs and tradeoffs. Led by Peter.)

(Anonymous Sponsors: Year0.5 to Year1.5, medium cost. Support of the analysis in Year0.5, then figure out a transition plan in Year1.0, and deploy in Year1.5. Led by Peter.)

4.6 Advocacy for running more relays

There are several components here.

First, we need to reach out to the right communities. There are lots of technical organizations out there who would be happy to help if only they realized the need and understood how straightforward it is.

Second, we need to streamline the instructions, including coming up with an easy set of screenshots for configuring a relay with Vidalia. Right now the web pages we point people to are years old. They're still correct, but in the mean time we have made many of the steps easier, and we should update the docs to reflect these changes. As an example, Tor on Debian now starts as root, so it can bind to port 443 directly without needing any clunky iptables rules to do port forwarding — but we haven't written this down anywhere useful.

Third, we need to maintain connections to the people who have set up the relay. I think the number one reason relays disappear is because the people who set them up don't think anybody cares anymore. To this end, Jacob has been working on a "Tor weather" website[8] that lets people sign up to get email when a relay disappears. This could be used either by the relay operator, or by our advocates to get reminders when somebody needs a nudge.

Fourth, we need to work on ways to inform people about our progress at growing the network. Some of this task will be started in the Metrics work in Section 5.7. We need to come up with flashier ways to show our growth, perhaps through Tor network maps that visualize the data in a slick way, through Facebook reputation for relay operators, or through other ideas that come along.

We also need to investigate how well our "get people to offer their ORPort on 443" strategy is working, because those relays are most useful for folks in firewalled areas.

(IBB: Year0.5, medium cost. Get Tor weather up, stable, and in use by some relay operators and some relay advocates.)

(IBB: Year1.0 and Year1.5, medium cost. For the network maps work and also maintenance. Led by Jacob.)

5 Suitability for circumvention

5.1 Normalize our network fingerprint even more

Play the TLS handshake arms race as needed.

The Year0.5 milestone here is to produce a list of the likely avenues we anticipate for blocking, and for each avenue build a plan for how we should respond to get Tor unblocked again. We don't intend to deploy fixes until pushed by the arms race to do so, but we should work towards having the fixes ready or close to ready so we don't look like idiots for a month.

Then at each milestone we'll revisit our list and revise our plans as needed.

(IBB/Anonymous Sponsors: Year0.5 to Year3.0, medium cost. Periodic adjustments as Smartfilter and Websense do their thing. Led by Nick, with red teaming from Steven.)

[7] https://svn.torproject.org/svn/tor/trunk/doc/spec/proposals/141-jit-sd-downloads.txt
[8] https://weather.torproject.org/

5.2 More bridge address distribution strategies

Assess more bridge address distribution strategies, based on a broader set of technologies like SMS, radio, World ofWarcraft, etc. Many of these approaches will require more manual ongoing attention than our first few approaches.

In the meantime, we need to design and deploy other bridge address distribution strategies that make the process a bit more automated for the users. In particular, we're thinking about a "bridge loop" design where bridge identities form a "loop" at the bridge directory authority, and if you know any bridge in the loop you can learn all the others. This approach will allow Tor clients who know a few bridges to be updated with new bridges as their old ones rotate, without opening up the list to full enumeration.

(Anonymous Sponsors: Year2.5, medium cost.)

5.3 Layered guard nodes for bridge users

Decide whether bridge users need to choose a second "layer" of entry guards, so it's harder for an ordinary Tor middle server to enumerate bridge relays just by seeing who connects. Start solving this problem somehow, for example by making bridge users do the above.

I think this is going to be necessary in the near term, since if it turns out to be a real attack, it is a very practical one. **(Anonymous Sponsors: Year2.0, low cost. Analysis led by Roger.)**

(Anonymous Sponsors: Year3.0, medium cost. For deployment.)

5.4 Tracking bridge reachability

Better and more automated measurement tools for whether bridges are actually up, and actually reachable from inside target countries.

"Actually up" is quite straightforward: we already do simple reachability testing from the bridge authority. Tracking reachability from inside target countries will be a statistical game based on how many geoip details we can collect from the bridges themselves. See the "better user metrics" item below.

5.5 Email auto-responder

Email auto-responder so for example gmail users can fetch the Tor software via email. Social network distribution techniques. Continue beating on this problem.

For Year0.5, we'd like a design proposal that describes what commands it responds to and how it responds; how it interacts with DKIM; and a plan for supporting many languages. We should put out a prototype so experts can use it, and to get more intuition about our requirements.

For Year1.0, we should polish it some more and get some outside help with translation, how to phrase the messages, and how to make it more intuitive to users.

(IBB/Anonymous Sponsors: Year0.5 and Year1.0, medium cost. For design and setup, led by Jacob.)

(Anonymous Sponsors: Year2.0 and Year3.0, low cost. For operation, led by Jacob.)

5.6 Research: how many bridges do you need to know to maintain reachability?

We need to track the churn of bridges over time and then analyze how many bridges are smart to know, or how often it is smart to learn new bridges, in order to stay connected. If a few bridges are likely to last a long time, we can focus on other problems. On the other hand, if even a half dozen bridges are likely to all vanish soon, we need to work both on encouraging bridge stability and on better algorithms for learning about new bridges.

Then we need to watch for trends and changes over time to see if our job is getting easier or harder.

(Anonymous Sponsors: Year1.5 to Year3.0, medium cost. In Year1.5 and Year2.0, assess what data to store and how, and then store it. In Year3.0, operation and analysis. Led by Karsten.)

5.7 Better user metrics and measurements

Keeping ongoing metrics about the Tor network can help us in many different ways. First, they can help with the resource allocation challenges from Section 3: we need to know how the current network is doing before we can think usefully about how to change it. Second, ongoing metrics can help with evaluating whether our changes are having the effects we want. Third, they can help with sponsor and community outreach: many people and organizations who are interested in funding The Tor Project's work want to know that we're successfully serving parts of the world they're interested in, and that efforts to expand our user base are actually succeeding.

There are a number of ways to measure and demonstrate progress, but most of them have anonymity implications, so they must be approached carefully.

Item one is to automate the collection of current network traffic statistics: volume of data over time, number of available relays each day, total advertised capacity for the day, etc. We've been collecting some of this data already so we can have some baselines, but we need to collect it in a more reliable fashion (i.e. pay somebody to be sure to do it), and we need to redesign and rewrite our scripts for processing it into useful graphs and figures since we're talking dozens of gigabytes of input data.

Item two is to let websites measure how many of their users are coming from Tor. To do this, we should clean up and maintain the "exitlist" service we are offering, so it's easy for other sites to check if a given IP address was a Tor exit relay at a given point. We will also need to provide some glue, and maybe training, to let people convert apache logs into useful information about the fraction (and variety?) of Tor users. To be most effective, we will want to answer not just "is this IP address a Tor exit relay now?", but also "was it an exit relay during this time period?"

Item three is to implement the "geoip lookup" designs that Nick has been designing[9]. These should give us a better estimate of the overall set of current Tor users; and as we migrate to the directory guards design, it should still be able to give us some rough numbers.

Item four is to try to handle the skew in our user geoip stats from dynamic IP addresses. Currently we're forced to just measure a rolling 24 hour window, since most users in Germany and China get a new IP address daily. Thus our metrics leave out most of the people who use Tor only infrequently – which could be a big part of our target population. Can we find or generate a database of Internet address blocks that are associated with frequent cycling? If so, we could discard the sightings of those users after 24 hours, while accumulating the other sightings over a longer time period. We might be able to bypass this step if we can do the geoip lookup designs based on measuring something we know users do with a certain frequency, like fetching a new copy of the consensus networkstatus – in this way we are measuring the number of Tor clients running rather than trying to measure the number of IP addresses in use. Note that if we do a good enough job with the "user counting" designs in item three above, we may not need to worry about item four.

Item five is to investigate whether we can safely switch the current website logs over to doing a GeoIP lookup before discarding the IP address. (Right now we immediately discard the IP address, because we don't want to become a target for external queries[10].) Being able to track downloads or page views over time is not a perfect metric, because it doesn't include downloads from other sites (e.g. Debian repositories) or downloads shared among friends, but it is still a metric. Do we need to randomize or otherwise lock down the logs to make this increased data gathering acceptably safe?

Item six is to track and graph the countries and number of users that bridge relays report[11]. Right now we're accumulating a big pile of "extrainfo" descriptors from bridge relays, but we aren't doing anything with them. We should write scripts to parse them, remember the important details, and maintain periodic snapshots so we can look for trends over time of how many users we're seeing from different countries.

Item seven is to do some analysis about how to detect country-wide blocking events based on trends in the above data. For example, if the website sees a sharp drop-off in hits from a given country, perhaps the site is blocked by that country's firewall or ISPs. Similarly, a sharp drop-off in reported GeoIP stats from bridges or directory guards could show that the country is starting to block access to bridges and/or block

[9] https://svn.torproject.org/svn/tor/trunk/doc/spec/proposals/ideas/xxx-geoip-survey-plan.txt
[10] http://seclists.org/nmap-hackers/2004/0016.html
[11] https://svn.torproject.org/svn/tor/trunk/doc/spec/proposals/126-geoip-reporting.txt

Tor's network signature. The challenge here is that when there are very few data points, it's hard to learn something statistically significant from losing a few. Is there a way to aggregate the overall data points in a way that makes it easier to notice changes?

Since Karsten will be finishing his PhD at the end of 2008, he will be ramping up on the overall metrics topic in 2009.

(Anonymous Sponsors: Year0.5 to Year3.0, high cost. Led by Karsten.)

(Needs a sponsor: The current funding level is enough for a bit less than half of Karsten. We could easily make use of a full-time smart researcher on this topic.)

5.7.1 Year 0.5

The deliverable that we imagine for year 0.5 is a report that answers the following questions:

a) What information do we want to present? What are useful statistics? This includes all metrics mentioned above, and also the metrics described in Sections 3 and 5.4.

b) What data are available for these metrics? This includes public data (network consensus, router descriptors, extrainfos) as well as non-public data (directory requests, website requests, bridge usage).

c) What data are missing that are required to provide these statistics? Can this data be provided publicly or not? Is it a good idea anonymity-wise to collect that data?

d) What systems are available that present some of these statistics? Can we extend one or more of them, or should we start from scratch? (How likely is it that we'll be able to work together with Kasimir Gabert on his TorStatus project? Further, it might be a good move to have Sebastian Hahn, one of our successful Google Summer of Code 2008 students, work on the PHP part.)

5.7.2 Year 1.0

The idea for the remaining six months of the first year is to implement the metrics concerning public data. These are 1) network statistics and 2) the exit script and log aggregator.

a) The current plan is to extend the implementation of torstatus to provide more information about network statistics. Torstatus currently takes all of its information from current network documents (consensus, router descriptors, extrainfos), but does not keep a history itself. The first step would be to extend the database to keep a history. (The previous analysis should help in estimating what kind of data is required.) Further, there should be some basic statistics that display the historical collected data. [This is a huge milestone, and our original idea was to separate collecting data and creating statistics; but a deliverable that just fills a database isn't as shiny as one that actually displays new information.]

b) We should try to integrate historical data (e.g. weasel's descriptor archives) into the database to make the network statistics complete.

c) Extend the exit list and write a script to locally analyze webserver logs. It *might* be better to use and extend the exitlist functionality of torstatus for this instead of our own exitlist. Otherwise, there is a certain overhead for getting comfortable with two systems and putting in a history database. We will know more about that after finishing the analysis.

d) Integrate historical data into the exitlist database.

It's hard to estimate how much time these tasks will take before actually having performed the analysis from Year0.5. We may end up moving some of them to the Year1.5 milestone. Item d is the one most likely to be punted.

5.7.3 Year 1.5 and 2.0

The plan for the second year is to tackle the metrics based on non-public data: 1) directory requests, 2) website requests, 3) bridge usage. This includes collecting data at their source (if not already done), aggregating them, and transferring them to the statistics portal. The portal would keep a history of these data in its own database and display the data nicely. Handling non-public data is more complicated than

public data, because we can't just grab the data ourselves, but need to preprocess and collect them at different places.

This phase would also involve gathering the stream and circuit measurement data from Section 3.1.2, so we can look for trends over time.

Yet another part for year two is researching how many bridges are needed to maintain reachability (Section 5.6).

5.7.4 Year 2.5 and 3.0

Year three would be used to handle the more advanced topics: 1) Filter out dynamic IP addresses from logs and 2) detect country-wide blocking events.

The other task for year three is operation and analysis of Section 5.6.

5.8 Distributed bridge authorities

Make our 'single bridge authority' design into a 'redundant bridge authorities' design, so bridge publish/lookup can't be knocked over or broken into by attacking a single location. ...

(Needs a sponsor: This will be hard to do right.)

5.9 Research: scanning-resistance

5.10 Research: hiding whether the user is reading or publishing?

6 Client safety

6.1 Automatic update

Tor currently has no mechanism for updating clients in the event of security vulnerabilities and changes to blocking mechanisms. To be effective in the arms race, we will need one.

Step one is to work on auto-update-of-Tor features inside Vidalia:

- looking at the majority-signed networkstatus consensus to decide when to update and to what version (Tor already lists what versions are considered safe, in each networkstatus document),
- doing the update either via Tor or via the directory mirror update protocol (proposal 127) when possible, for additional privacy,
- checking package signatures,
- giving the user an interface for these updates, including letting her opt to migrate from one major Tor version to the next.

This should work on both Windows and OS X. Ideally we would adapt some third-party lib to do parts of this for us – but we haven't found any good free-software security-oriented auto-update lib out there yet. Then we need to work on auto-update for Vidalia itself too, as well as for other supporting applications like Polipo. We also need to produce better plans around security issues like signing key rotation so we can be sure to provide safe and secure service over time. **(Google: Year0.5, medium cost.)**

(IBB: Year1.0, medium cost. Get it rolled out to experimental users.)

Then we should take a step back and revise our design into a thin client that's separate from Vidalia. It should know how to fetch new versions of the components it wants and check their signatures appropriately. We could imagine this approach as a tiny bundle stub that bootstraps itself into a full Tor bundle; it would also replace our current stopgap workaround for modem users who can't fetch the entire Tor Browser Bundle without losing their connection. We would still want to ship the full bundle too for folks who want to get the whole set and carry it around with them, though. **(Anonymous Sponsors: Year1.5 and Year2.0, medium cost.)**

Then we need to look at how Firefox's automatic update scheme works with our automatic update scheme in the context of the Tor Browser Bundle. What about other FF extensions inside the bundle? Which app(s) should be the one managing the automatic update? Does Firefox do anything smarter than check the SSL certificate of the site that it updates from? **(Anonymous Sponsors: Year2.5, medium cost.)**

6.2 Vidalia development work

a) Integration for PlaintextPorts warnings and other status events. Vidalia should walk the user through recognizing that connections to plaintext ports (e.g. 110, 143) probably indicate a bad move on the user's part.

b) More generally, there are a variety of status events that Tor currently sends to Vidalia (e.g. for reachability testing as a server), but Vidalia has no way to present them usefully to the user. We should come up with a generalized way to interact with the user when errors or warnings occur.

c) We should put Polipo into the main bundle, and teach Vidalia to launch it and close it.

d) Bridge usage display. If you're a bridge relay, then you know what countries you've got users from — you publish the aggregated lists in your extrainfo descriptor. We should also put that in the Vidalia display, so you can get a warm fuzzy feeling about saving the world.

e) Look into better error and crash reporting mechanisms for Vidalia under Windows. Google had a nice-looking one called Breakpad[12] that we should try out.

f) Let Vidalia change languages without needing to quit and start again.

g) Other development items and GUI support items as they come up.

(IBB/Anonymous Sponsors: Year0.5, medium cost. For a-d, led by Matt.)
(IBB/Anonymous Sponsors: Year1.0, medium cost. For e-g, led by Matt.)
(Anonymous Sponsors: Year1.5 and Year2.0, medium cost. Led by Matt.)

Later plans might include working on an improved and more usable network map in Vidalia, perhaps based on Marble or some other way to present a higher resolution map without shipping an enormous graphics file with Vidalia.

6.3 Node/Network scanning

We've written a prototype node scanner called 'SoaT' to scan the Tor network for malfunctioning and malicious relays; we presented it at Black Hat and Defcon 2007. Currently, it scans for malicious content injection at the exit relays by checking MD5 sums of documents. It would be nice to produce a more flexible fingerprint of a page, perhaps based on the javascript/object/image content only. Additionally, it would be nice to integrate scan results into the Tor directory consensus, so clients can use the information in their routing decisions to avoid malicious or failing relays. Passive scanning and reliability reporting from clients and relays to the directory servers is also a possibility, but some of this may need experimentation and research[13].

There are several components here. The big first steps are A) a more automated scanning mechanism, B) coming up with plausible tests that are hard to distinguish from "normal" fetches, and C) integrating the results into the directory authorities so Tor clients get quick feedback about problems we discover.

The first cut at this tool would focus on making it easy to configure and run on a variety of platforms, giving it a good suite of plausible-looking tests, and feeding the results into a directory authority. **(Anonymous Sponsors: Year0.5 to Year2.0, medium cost. Led by Mike.)**

Then we should make the tool more automated and easier to run for long periods unattended, including ways for users to submit tests directly to the database ("I'm having problems with the following page or website, please add it to the test suite") and other ways to add the client reliability reporting described above. **(Anonymous Sponsors: Year2.5 and Year3.0, medium cost. Led by Mike.)**

[12] http://code.google.com/p/google-breakpad/
[13] http://fscked.org/transient/SecuringTheTorNetwork.pdf

6.4 Torbutton development

We need to stabilize and then continue the "Torbutton-dev" work so we can give people a robust Firefox extension that not only lets them toggle Tor on/off, but also protects them from many application-level threats on the web.

We've made a huge amount of progress already[14], but many more issues remain — in particular, while we've been focusing lately on making sure that Torbutton *can* provide safe browsing, it probably is at odds with usability and useful browsing in many ways. We will need to work with users to find the right balance between safety and usefulness.

Further, Torbutton will need constant attention to maintain the current features as Firefox's internals change. We have filed several dozen Firefox bugs, and are working with Mozilla to resolve them. Alas, there will continue to be more.

Look into ways for Vidalia and Torbutton to communicate. For example, some link seems necessary for the "new identity" button to function properly when both Firefox and Polipo are determined to do keepalive on their current connections.

(Anonymous Sponsors: Year0.5 to Year2.5, high cost. Led by Mike.)

6.5 Torbutton equivalent for Thunderbird

We're hearing from an increasing number of users that they want to use Thunderbird with Tor. However, there are plenty of application-level concerns – for example, by default Thunderbird will put your hostname in the outgoing mail that it sends. At some point we should start a new push to build a Thunderbird extension similar to Torbutton.

(Needs a sponsor: The complexity here may be on the same order of magnitude as the Torbutton work for Firefox. We should wait until more users and more funders care about this.)

6.6 Evaluate new anonymity attacks

There have been a variety of new anonymity attacks published recently, and more on the path to publication.

We need to work with the authors of these papers to help them show the efficacy of their attacks in ways that we can grasp / believe / reproduce; and we need to brainstorm practical solutions or mitigations for the scenario where the attacks do in fact work (or can be made to work).

(Anonymous Sponsors: Year1.0 to Year3.0, medium cost. Led by Steven.)

(IBB: Year1.5, medium cost.)

(Needs a sponsor: This is an immense but critical task, and we will need to get most of the funding for it from other sources, but this item will make sure that it gets at least some attention.)

6.7 Evaluating traces from Tor Browser Bundle, Tor VM, etc

We've made a good start at enumerating the traces we see from the current Tor Browser Bundle running Firefox 2. As we note in Section 7, it will need ongoing attention as we move to Firefox 3 and as other changes are made. Further, other mechanisms for launching Tor, such as the QEMU approaches for Incognito and Tor VM, will leave still different sets of traces.

Beyond just enumerating the traces and evaluating their severity, we also need to work to reduce the impact of each trace. How many of them are impossible to get rid of? How many issues are resolved by running most of the sensitive programs inside a VM? What about if we moved to Vista?

(IBB: Year0.5, low cost. Led by Steven.)

(Needs a sponsor: We could probably make more progress on this, but we really need to bring in outside forensic experts. And even then, the prognosis doesn't look good about

[14] https://www.torproject.org/torbutton/design/

actually being able to do anything about the traces. Windows is bad like that. Maybe this is something some other organization wants to pick up?)

6.8 Is your browser config safe?

We need test websites that can evaluate at a glance whether the user is vulnerable to the wide variety of web-based exploits we've already discovered. This "check" website would be useful for several goals:

- We could use it for regression tests as we make new versions of Torbutton, and as Firefox makes upgrades that might reopen new problems.
- We could use it to evaluate other browsers and other extensions that people might try to use instead of the recommended Firefox + Torbutton combination.
- We could use it to help users confirm that they are in fact using Torbutton in the correct configuration.

Some of these steps, especially the last, could even be done by shipping a tiny webserver inside Tor, and letting the browser interact with it locally. This approach has the benefit of being a much more controlled environment, where we can avoid confusion due to outside variables; but it has the drawback that we can't automatically update it for new or better tests in the way that we can update a central test website.

(Needs a sponsor: We'd like to get to this in the Year2 range; but doing it right will require a lot of attention.)

6.9 Documentation and explanations of how and when to use Tor

There are a few starts to these docs out there, like Ethan's GV tutorial[15]. But I bet people would like some more detailed, more updated walkthroughs too — also covering other activities like setting up instant messaging. Some of this is hopefully going to be covered by the NGO-In-A-Box folks, but I don't know how indepth they go.

(Needs a sponsor: Maybe some other organization should help on this?)

6.10 Recommended configurations and applications

Figuring out recommended configurations for various apps and figuring out which apps in each category to recommend. We're doing better at this now that we have the Tor IM Browser Bundle, since it handles IM, IRC, etc. But many people still want to use their own apps, or at least apps that aren't quite as klunky at Pidgin. We need to investigate the default configurations of apps that users want to use in the wild, and see if we can either a) come up with some instructions for how to use them safely, and/or b) come up with some recommendations for which ones we better than others. Some explanations ("here's why you shouldn't use IE") would probably help give people some intuition.

(Needs a sponsor: Maybe some other organization should help on this?)

6.11 Safe path selection when you know more than other users

NRL Research and development project. [say more]

(NRL: Year0.5, medium cost. Led by Nick.)

7 Client usability

7.1 Transparently intercepting connections

Currently, Tor clients need to actively configure their applications to use Tor as a SOCKS proxy. This step results in many confused users, as well as (probably) many users who are using Tor in an unsafe

[15] http://advocacy.globalvoicesonline.org/projects/guide/

configuration. Even for web browsing, many of the challenges that Torbutton attempts to solve come from ways that websites can trick users into bypassing their proxy settings. And simply disabling all of these avenues results in usability problems, e.g., preventing users from watching Flash videos at Youtube.

Some tools like Xerobank VM and JanusVM use a virtual machine (generally VMWare running a Linux OS) to run the Tor client and web proxy, and they use the VM's ability to intercept outgoing connections so they can redirect them into Tor.

It might be that the VM approach is necessary, in which case we should work on adapting QEMU so our bundles are not subject to VMWare's restrictive redistribution licenses. Or it might be that we can just reuse drivers like the ones OpenVPN uses to transparently intercept the outgoing connections and pass them into the Tor client.

We need to research the options and implications, and document the problems that can come up — for example, trying to pass outgoing email connections into Tor will not work very well, since few Tor relays have an exit policy that will allow outgoing email. We may end up needing an administrative interface for which addresses and ports should be sent into Tor; but we'd like to find a more usable solution than that.

Ultimately we need to deploy a Windows Tor client that's wrapped in a VM with transparent proxying. Done right, this should be the default way that Windows users interact with Tor.

(IBB/Anonymous Sponsors: Year0.5 and Year1.0, high cost. Research, analysis, and prototyping. Led by Martin.)

(IBB/Anonymous Sponsors: Year1.5 and Year2.0, medium cost. Deployment, led by Martin.)

7.2 Tor Browser Bundle

Lock down the Tor Browser Bundle, make it more robust, etc. Deal with deployment issues, updating features and maintenance for new versions of the component software, translations, etc.

Year 0.5 items:

- Work on a new launcher out of Vidalia rather than using Portable Firefox. This new launcher would let Vidalia recognize better when Firefox has closed.
- We'd like to make it possible to run a TBB Firefox and a normal Firefox in parallel, but without putting a lot more money in this item, we can't promise it in case there is some weird FF bug that blocks us. We will either do it or determine exactly what weird FF bug it is that blocks us.
- Switch TBB to Firefox 3, once Torbutton 1.2 is stable enough on it.
- Gather a new set of traces, once we have a new launcher and once we switch to Firefox 3.

Year 1.0 items:

- Port Tor Browser Bundle to OS X.
- Make TBB the recommended Tor download for Win / OSX.
- Make sure it's easy to switch to an unbranded Firefox (e.g. called "Tor Browser") on short notice, since we might find ourselves in exactly that situation.
- Evaluate CCC's "Freedom Stick" version of TBB. Decide if we like the approach they took, and if we want to recommend it or suggest some changes.

Year 1.5 should include investigating what else to put on a USB image for TBB. Can we do some disk encryption, so if we decide to put out USB images with a different bridge relay for each image, simply swiping the USB key isn't enough to learn about the bridge? This would also be useful if we want to give out per-key hidden service authorization.

Another item to tackle is locking down Pidgin in the "Tor Browser IM bundle". Right now it's possible that it broadcasts the local username when using IRC, or otherwise gives away private data. We need to at least document these issues, and ideally fix some of them.

(Anonymous Sponsors: Year0.5, low cost. Led by Steven.)

(IBB/Anonymous Sponsors: Year1.0 to Year2.5, medium cost. Led by Jacob.)

7.2.1 Deploying USB keys with Tor Browser Bundle on them

Now that we've got the software itself working pretty well, we should figure out all the details like where to buy USB keys in bulk, how to silk-screen a logo onto them, what size they should be, whether we should aim for those cool USB form factors that look like smart cards or buttons or whatever. Should we include an auto-start file for the Windows users when they pop the key in? Should we include a few tutorials on how to stay safe on the Internet? What else? I wouldn't recommend burning tens of thousands of these yet though, since we're hoping to have improved versions in a year or two (e.g. using QEMU and/or with an auto updater).

(Needs a sponsor: Maybe some other organization should help on this (with periodic advice from us)?)

7.3 Playing nicely with websites

Right now Wikipedia and some other services block posts from Tor users, due to a few abusers. Tools like Nymble[4] allows these services to recognize the abusers later without ever needing to learn who or where they are – thus allowing the non-abusers to start using the service like usual again.

The first step here is to write down a clear explanation of why anonymity is not at odds with open access to sites like Wikipedia and Slashdot. Roger gave a talk at Wikimania 2006 to explain to the Mediawiki developers about other options they have besides blacklisting IP addresses[16]. Once we have this essay ready, we can start to educate other people in the community about the fact that practical options do exist. The Wikipedia community is diverse though, and can be stubborn, so this will not be accomplished quickly. At the same time we need to work with the legitimate Tor users (e.g. the ones who would like to edit Wikipedia through Tor) and help them to adopt some interim solutions, if any. **(Anonymous Sponsors: Year2.0 to Year3.0, medium cost. For advocacy and operations, led by Roger.)**

The next step (after this project) is to help deploy a system like Nymble, which is an infrastructure for letting websites blacklist abusive users without needing to (or being able to) unveil their location or identity. They are apparently working on an implementation right now, but it will definitely need help with deployment, usability, and sustainability.

7.4 Let users specify their exit country

As the network grows, and censorship gets more varied and widespread, exiting from a censored country becomes more of a hassle.

The first steps are a) changing the Vidalia interface to let you communicate which country you want to exit from, b) turning countries into sets of Tor servers that we're pretty sure are in those countries, and c) making sure that the Tor client does in fact do the right thing with the set of preferred servers (the basic features have been in for a while, but nobody uses them for much and they likely have some problems). **(Anonymous Sponsors: Year2.5, medium cost.)**

The next steps after this contract are to tackle making the Tor client scale well when you specify really large lists of server IDs either in the "use these" or the "don't use these" side, work on more complex interface approaches ("use these two countries but not this one"), specify other constraints the user has in mind ("please only pick servers with certain uptime, certain bandwidth, certain operating system, etc"), look into the anonymity issues with choosing a smaller set of options at each step, look into the security questions of using geoip data vs whois data for country codes, let us build a more efficient interface (that doesn't require interacting with the GUI) by specifying country properties in the url (like www.google.com.cn.exit) without actually broadcasting this url to the destination website in the Host: http header, and figure out various other issues that come up as we start deploying a solution. We may get follow-on (external) funding to move this part forward.

[16] `http://freehaven.net/~arma/slides-wiki-tor.pdf`

7.5 LiveCD

We need a nice bootable LiveCD containing a minimal OS and a few applications configured to use it correctly. The Incognito project has demonstrated that this is quite feasible, and there is a nice student in Sweden who is currently trying to maintain it for free.

It needs more documentation, more analysis about whether its configuration choices are the right ones, and more thought for what applications to include so it has enough that it's useful — and what apps to *exclude* to keep its complexity from dragging it down.

Most appealingly, Incognito has recently added a feature allowing it to be booted from inside Windows as a standalone system that provides a set of self-contained correctly-configured applications chosen for their good security. This approach could provide the best combination of the transparently intercepted connections item above with the Tor Browser Bundle item. The tradeoff is download size.

(IBB: Year0.5 to Year1.5, low cost. Responsive support and feedback for the volunteer maintainer.)

(Needs a sponsor: High cost. At some point we should ramp up development and support for Incognito. Medium priority.)

7.6 Translation coordination and automation

We've set up a prototype online translation website[17] based on Pootle. We've also set up some preliminary documentation[18] for how to interact with the website and provide translations.

First, we need to continue to coordinate and maintain our current volunteer translators. They do a tough job basically for free, so we need to keep making sure their questions get answered promptly and making them know they're appreciated. We should keep working on ways to let them stay notified when new translations are needed, and to let them track their translation progress. **(IBB: Year0.5 to Year1.5, low cost.)**

Second, while we've successfully transitioned most of our file formats over to the Pootle interface, we still haven't finished our "wml2po" (website to .po) converter, so there's no way in the Pootle web interface currently to translate the Tor website pages. We were hoping to get a good start on that from one of our Google Summer of Code 2008 students, but that didn't work out. So, we should get that going ourselves. The really tricky parts are: A) we need to automatically break up wml files (basically html files) in such a way that we get one 'idea' per piece: we're currently thinking that splitting on the 'p' tag and similar tags should do it. B) If we change a paragraph just a little bit, we would like some way to recognize that the previously translated string is a good place to start when translating the new paragraph. In the obvious way to break up a web page into strings, the new string would not be the same as the old string, so we'd have no way of knowing they're related. We need to either label the paragraphs in a way such that the label stays the same as we edit the paragraph, or do something even smarter to recognize when a new paragraph is 'likely' to be a derivative of an old one. **(IBB: Year0.5 to Year1.5, low cost.)**

(Needs a sponsor: medium cost. Adding more funding would get this second step done more reliably.)

7.7 Usability testing of Tor

Especially the browser bundle, ideally amongst our target demographic. That would help a lot in knowing what needs to be done in terms of bug fixes or new features. We get this informally at the moment, but a more structured process would be better.

(Needs a sponsor: Maybe some other organization should help on this?)

[17] https://translation.torproject.org/

[18] https://www.torproject.org/translation-portal

7.8 Better Debian/Ubuntu Packaging for Tor+Vidalia

Right now Vidalia works well on Windows and OS X, where it launches Tor itself and manages the other programs. On Linux, though, the correct way to install Tor is by adding the Tor package, which causes the Tor program to start at boot under a separate user.

The current solution for running Vidalia and Tor on Linux involves either telling the user to stop the existing Tor daemon (and prevent it from being launched on boot) and let Vidalia start its own Tor process, or explaining to the user how to set a control port and password in their torrc. The former is bad because Tor would run with the same privileges as the user; the latter is bad because it involves way too many hard steps.

A better solution on Debian would be to use Tor's ControlSocket feature, which allows Vidalia to talk to Tor via a Unix domain socket, and could possibly be enabled by default in Debian's Tor packages. Vidalia can then authenticate to Tor using filesystem-based (cookie) authentication if the user running Vidalia is also in the debian-tor group.

This project will first involve adding support for Tor's ControlSocket to Vidalia. Then we should develop and test Debian and Ubuntu packages for Vidalia that conform to Debian's packaging standards, and make sure they work well with the existing Tor packages. We can also set up an apt repository to host the new Vidalia packages.

The next design challenge would be to find an intuitive usable way for Vidalia to be able to change Tor's configuration (torrc) file even though it is located in /etc/tor/torrc and thus immutable. The best idea we've come up with so far is to feed Tor a new configuration via the ControlSocket when Vidalia starts, but that's bad because Tor starts each boot with a different configuration than the user wants. The second best idea we've come up with is for Vidalia to write out a temporary torrc file and ask the user to manually move it to /etc/tor/torrc, but that's bad because users shouldn't have to mess with files directly.

(Needs a sponsor: Wait for more funding.)

7.9 Tor for Java

It would be great to reanimate one of the approaches to implement a Tor client in Java, e.g. the OnionCoffee project[19], and make it run in a Java environment. The first step would be to port the existing code and execute it in a current Java environment. Next, the code should be updated to support the newer Tor protocol versions, like the new v2 handshake. Further, support for requesting or even providing Tor hidden services would be neat.

(Needs a sponsor: Wait for more funding.)

7.10 Tor for mobile phones

Cell phones are an exciting environment for Tor. Android, iPhone, Windows Mobile, or other J2ME environments generally have hardware capable of supporting Tor clients. The first step is to research the challenges inherit in a mobile environment. Asessment of resource constraints, limited bandwidth, and mobile OS anonymity leaks are all first steps. Next steps could be porting the code to the chosen mobile environment and developing a usable user interface. Further, support for requesting or even providing Tor hidden services would be neat.

(Needs a sponsor: Wait for more funding.)

8 Non-development supporting tasks

8.1 Making our website more usable and useful

This is especially important in the context of circumvention. Chris had some suggestions, and there are plenty more suggestions where those came from. We also have a big pile of neat stuff on our wiki (including

[19] http://onioncoffee.sourceforge.net/

the faq), but nobody will find it there. Coordinating the translators comes into this somewhere. One of the things I keep noticing is that we have screenshots for things like Tor Browser Bundle, but the Farsi TBB page still points to English screenshots.

(Needs a sponsor: Maybe Anonymous Sponsors should help on this, especially the parts that matter to them?)

(Needs a sponsor: medium cost Even if Anonymous Sponsors picks up a lot of the website reworking, we need to get the common knowledge from inside Roger's head to the people doing the rework. So we should allocate some funding for this.)

8.2 Making the website more available in blocked countries

This involves keeping up with the steps on the "finding tor" page[20] and we plan to be working on some technical solutions such as the email autoresponder, but it also involves taking a more community-oriented approach, e.g. setting up mirrors only known by certain groups. More generally, we should coordinate the community of people running our mirrors – make them feel appreciated, help answer their questions, etc. We have some volunteers[21] already doing that a bit but it sure isn't as much attention as it could be. We also need to figure out how to make better use of mirrors inside censored areas, e.g. http://tor.anonymity.cn/

What ways would people who can't reach the Tor website like to get Tor? We've talked of putting Tor Browser Bundle on USB sticks and distributing them manually. We're working on an email auto responder. We also have an IRC auto responder, but we're concerned that no blocked users know what IRC is. What are they used to using?

(Needs a sponsor: Anonymous Sponsors should tackle this too.)

8.3 Legal advocacy for anonymity

Tor relies upon volunteers running relays, so advocacy that can improve the legal climate for those volunteers strengthens the diversity and capacity of the network. Several countries, mainly in the EU but soon in the US too, are pondering new laws about "data retention". The goal is to record "traffic headers", in the hopes that when something bad happens they'll have all the data just waiting to be pieced together. The reality is that this is just another huge database waiting to be leaked or broken into, while at the same time the bad guys are already using enough stepping stones and decoys that they won't be found. "The trail will lead back to my grandmother's computer, and then what?"

(Needs a sponsor: Ultimately if we ignore this it will impact our ability to get relays, in ways we can't easily predict right now.)

8.4 Training the trainers

All around the world there are people teaching other people how to safely use Tor and related applications. This training will be ramping up with projects like NGO-in-a-box and the Global Voices seminars.

We should help train the trainers about Tor, so they better understand the technology, issues, and tradeoffs and can then do a better job of training the users. **(Anonymous Sponsors: Year1.0 to Year3.0, high cost. Led by Roger and Jacob.)**

8.5 Teaching journalists about Tor

Figuring out how to phrase all this for the media, so they understand Tor. Crafting a message that the media can understand is a critical piece of this, especially because of how many different angles Tor has. This isn't so much about getting good press about Tor as it is about preparing journalists so if they see bad press and consider spreading it further, they'll stop and think "hey, our guy down the hall got hassled less

[20] https://www.torproject.org/finding-tor
[21] https://www.torproject.org/mirrors

in Thailand because of Tor." While crafting Tor's image in the media is clearly up to Tor to do, figuring out how it fits in the bigger picture of privacy and circumvention is something we should all do together.

(Needs a sponsor: Maybe other organizations can help with this?)

8.6 Teaching other human rights groups about Tor

Groups like Human Rights Watch, HRIC, and many others really need to understand how Tor works so they can recognize situations in which it is smart to recommend it, and situations in which it is not so smart. More generally, we'd like to give them a better intuition about the limitations of the various circumvention and/or anonymity models out there. Hopefully with this knowledge they'll be better prepared to deal with the next snake oil circumvention tool they run across.

This goal means we need to teach them, or write clear documents that teach them, or teach other people who teach them, or something like that.

(Needs a sponsor: Low to high cost. Get Tor people and documents in the right places to teach people.)

References

[1] 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.

[2] Kevin Bauer, Damon McCoy, Dirk Grunwald, Tadayoshi Kohno, and Douglas Sicker. Low-resource routing attacks against tor. In *Proceedings of the Workshop on Privacy in the Electronic Society (WPES 2007)*, Washington, DC, USA, October 2007.

[3] George Danezis and Paul Syverson. Bridging and fingerprinting: Epistemic attacks on route selection. In Nikita Borisov and Ian Goldberg, editors, *Proceedings of the Eighth International Symposium on Privacy Enhancing Technologies (PETS 2008)*, pages 133–150, Leuven, Belgium, July 2008. Springer.

[4] Peter C. Johnson, Apu Kapadia, Patrick P. Tsang, and Sean W. Smith. Nymble: Anonymous IP-address blocking. In *Privacy Enhancing Technologies (PET 2007)*. Springer-Verlag, LNCS 4776, 2007.

[5] Jon McLachlan and Nicholas Hopper. Don't clog the queue: Circuit clogging and mitigation in P2P anonymity schemes. In *Proceedings of Financial Cryptography (FC '08)*, January 2008.

[6] Steven J. Murdoch and George Danezis. Low-cost traffic analysis of Tor. In *IEEE Symposium on Security and Privacy*. IEEE CS, May 2005.

[7] Steven J. Murdoch and Robert N. M. Watson. Metrics for security and performance in low-latency anonymity networks. In Nikita Borisov and Ian Goldberg, editors, *Proceedings of the Eighth International Symposium on Privacy Enhancing Technologies (PETS 2008)*, pages 115–132, Leuven, Belgium, July 2008. Springer.

[8] Robin Snader and Nikita Borisov. A tune-up for Tor: Improving security and performance in the Tor network. In *Proceedings of the Network and Distributed Security Symposium - NDSS '08*. Internet Society, February 2008.