Overhead from directory info: past, present, future

Roger Dingledine
arma@torproject.org

Tor Tech Report 2009-02-001
February 16, 2009

1 Introduction

A growing number of people want to use Tor in low-bandwidth contexts (e.g. modems or shared Internet cafes in the Middle East) and mobile contexts (start up a Tor client, use it for a short time, and then stop it again). Currently Tor is nearly unusable in these situations, because it spends too many bytes fetching directory info. This report summarizes the steps we've taken so far to reduce directory overhead, and explains the steps that are coming next. First, what do I mean by "directory info"? Part of the Tor design is the *discovery* component: how clients learn about the available Tor relays, along with their keys, locations, exit policies, and so on. Tor's solution so far uses a few trusted directory authorities that sign and distribute official lists of the relays that make up the Tor network.

2 History of v1, v2, v3 dir protocols

Over the years we've had several different "directory protocols", each more bandwidth-friendly than the last, and often providing stronger security properties as well. In Tor's first directory design[1] (Sept 2002), each authority created its own list of every relay descriptor, as one flat text file. A short summary of relay status at the top of the file told clients which relays were reachable. Every Tor client fetched a copy from an authority every 10 minutes.

Tor 0.0.8 (Aug 2004) introduced "directory caches", where normal relays would fetch a copy of the directory and serve it to others, to take some burden off the authorities. Tor 0.0.9 (Dec 2004) let clients download "running routers" status summaries separately from the main directory, so they could keep more up-to-date on reachability without needing to refetch all the descriptors. It also added zlib-style compression during transfer. At this point everybody fetched the whole directory every hour and the running-routers document every 15 minutes.

There were two big flaws with the v1 directory scheme: a security problem and an overhead problem. The security problem was that even though there were three authorities, you just went

[1] https://gitweb.torproject.org/torspec.git/blob/HEAD:/attic/dir-spec-v1.txt

with the most recent opinion you could find, so a single evil authority could screw everybody. The overhead problem was that clients were fetching a new directory even when very little of it had changed. In Dec 2004 there were 57 relays and the uncompressed directory was 172KB; but by May 2006 we were up to 749 relays and the full directory was almost 1MB compressed. Even though we'd lowered the period for fetching new copies to 2 hours (20 minutes for caches), this was not good.

We introduced the v2 directory design[2] in Tor 0.1.1.20 in May 2006. Each authority now produced its own "network status" document, which listed brief summaries of each relay along with a hash of the current relay descriptor. Clients fetched all the status documents (there were 5 authorities by now) and ignored relays listed by less than half of them. Clients only fetched the relay descriptors they were missing. Once bootstrapped, clients fetched one new status document (round robin) per hour. Peter Palfrader produced a graph of bandwidth needed to bootstrap and then keep up-to-date over a day.[3]

All of the improvements so far were oriented toward saving bandwidth at the server side: we figured that clients had plenty of bandwidth, and we wanted to avoid overloading the authorities and caches. But if we wanted to add more directory authorities (a majority of 5 is still an uncomfortably small number), bootstrapping clients would have to fetch one more network status for every new authority. By early 2008, each status document listed 2500 relay summaries and came in around 175KB compressed, meaning you needed 875KB of status docs when starting up, and then another megabyte of descriptors after that. And we couldn't add more authorities without making the problem even worse.

The v3 directory design[4] in Tor 0.2.0.30 (Jul 2008) solved this last problem by having the authorities coordinate and produce an hourly "consensus" network status document, signed by a majority of the six v3 authorities. Now clients only have to fetch one status document.

3 Next fixes

Based in part on sponsorship by NLnet[5] to reduce Tor's directory overhead, there are two directions we're exploring now: reducing relay download overhead and reducing consensus networkstatus download overhead.

We've cut relay descriptor size by 60% by moving some of the descriptor info to separate "extra-info" descriptors that clients don't need to fetch,[6] and we've cut the consensus size by 40% by leaving out non-Running relays[7] (since clients won't fetch descriptors for them anyway).

We spent the second half of 2008 working on a much more radical design change[8] where we move all the descriptor components that are used for path selection into the consensus, and then clients would download each relay descriptor "just in time" as part of circuit extension. This design would have two huge benefits: a) clients download zero descriptors at startup,

[2] https://gitweb.torproject.org/torspec.git/blob/HEAD:/dir-spec-v2.txt

[3] As of August 30, 2012, this graph file cannot be found anymore.

[4] https://gitweb.torproject.org/torspec.git/blob/HEAD:/dir-spec.txt

[5] http://www.nlnet.nl/

[6] https://gitweb.torproject.org/torspec.git/blob/HEAD:/proposals/104-short-descriptors.txt

[7] https://gitweb.torproject.org/torspec.git/blob/HEAD:/proposals/138-remove-down-routers-from-consensus.txt

[8] https://gitweb.torproject.org/torspec.git/blob/HEAD:/proposals/141-jit-sd-downloads.txt

greatly speeding bootstrapping, and b) the total number of relay descriptor downloads would be based on the number of circuits built, regardless of how many relays are in the network.

Alas, we have backed off on this proposal: Karsten Loesing's work on hidden service performance concluded that the circuit extension is the main performance killer, and our "just in time" download proposal would add more round-trips and complexity into exactly that step. So we have a new plan: we're still going to move all the path-selection information into the consensus, but then we're going to put the remaining pieces into a new microdescriptor[9] that will hopefully change at most weekly. That means the initial bootstrap costs will still be there (though the microdescriptor is maybe a third the size of the normal descriptor), but so long as you can keep a disk cache, descriptor maintenance will be reduced to roughly zero. We're aiming to roll this change out in Tor 0.2.3.x, in late 2010.

We also have plans to further reduce the consensus download overhead. Since the consensus doesn't actually change that much from one hour to the next, clients should fetch consensus diffs[10] rather than fetching a whole new consensus. We could expect another 80% reduction in size here. We hope to roll this step out in mid to late 2009. Alas, this goal is stymied by the fact that we haven't found any small portable BSD-licensed C diff libraries.[11] Anybody know one?

So these two changes together mean an initial bootstrap cost of maybe 100KB+300KB, and then a maintenance cost of maybe 20KB/hour. But actually, once we've gotten the maintenance level so low, we should think about updating the consensus more often than once an hour. The goal would be to get relays that change IP addresses back into action as soon as possible—currently it takes 2 to 4 hours before a new relay (or a relay with a new location) gets noticed by clients. Since one-third of Tor relays run on dynamic IP addresses [1], bringing that level down to 30 to 60 minutes could mean a lot more network capacity.

Down the road, there are even more radical design changes to consider. One day there will be too many relays for every client to know about all of them, so we will want to partition the network (see Section 4.4 of Tor's development roadmap[12]). When we do that, we can bound the amount of directory info that each client has to maintain. Another promising idea is to figure out ways to let clients build paths through the network while requiring less information about each relay. There's definitely a tradeoff here between centralized coordination (which is easier to design, and can more easily provide good anonymity properties) and scaling to many tens of thousands of relays. But that, as they say, is a story for another time.

References

[1] Karsten Loesing. Measuring the tor network, evaluation of relays from public directory data. Technical Report 2009-06-001, The Tor Project, June 2009. https://research.torproject.org/techreports/dirarch-2009-06-22.pdf.

[9] https://gitweb.torproject.org/torspec.git/blob/HEAD:/proposals/158-microdescriptors.txt
[10] https://gitweb.torproject.org/torspec.git/blob/HEAD:/proposals/140-consensus-diffs.txt
[11] https://lists.torproject.org/pipermail/tor-dev/2008-June/001533.html
[12] https://svn.torproject.org/svn/projects/roadmaps/2008-12-19-roadmap-full.pdf