Adaptive throttling of Tor clients by entry guards

Roger Dingledine
arma@torproject.org

Tor Tech Report 2010-09-001
September 19, 2010

Abstract

Looking for a paper topic (or a thesis topic)? Here's a Tor research area that needs more attention. The short version is: if we prevent the really loud users from using too much of the Tor network, how much can it help?

We've instrumented Tor's entry relays so they can rate-limit connections from users, and we've instrumented the directory authorities so they can change the rate-limiting parameters globally across the network. Which parameter values improve performance for the Tor network as a whole? How should relays adapt their rate-limiting parameters based on their capacity and based on the network load they see, and what rate-limiting algorithms will work best?

We'd love to work with you to help answer these questions.

One of the reasons why Tor is slow is that some people use it for file-sharing or other high-volume transfers. That means if you want to get your instant message cell through, it sometimes needs to wait in line behind a pile of other cells—leading to high latency and, maybe even worse, highly variable latency.

One way to improve the situation is Can and Goldberg's "An Improved Algorithm for Tor Circuit Scheduling" [2], which was integrated into Tor as of 0.2.2.11-alpha. The idea is to track how many cells the relay has handled for each circuit lately, and give priority to cells from quieter circuits.

But while that puts some cells in front of others, it can only work so many miracles: if many cells have been placed in front of your cell it still has to wait, and the more overall load there is in the network, the more often that will happen.

Which leads to the research problem: if we work to keep the really loud flows off the network in the first place, how much can it help?

Tor 0.2.2.15-alpha lets you set the PerConnBWRate and PerConnBWBurst config options in your relay, to use token buckets[1] to rate limit connections from non-relays. Tor 0.2.2.16-alpha added the capability for the directory authorities to broadcast network-wide token bucket

[1] https://en.wikipedia.org/wiki/Token_bucket

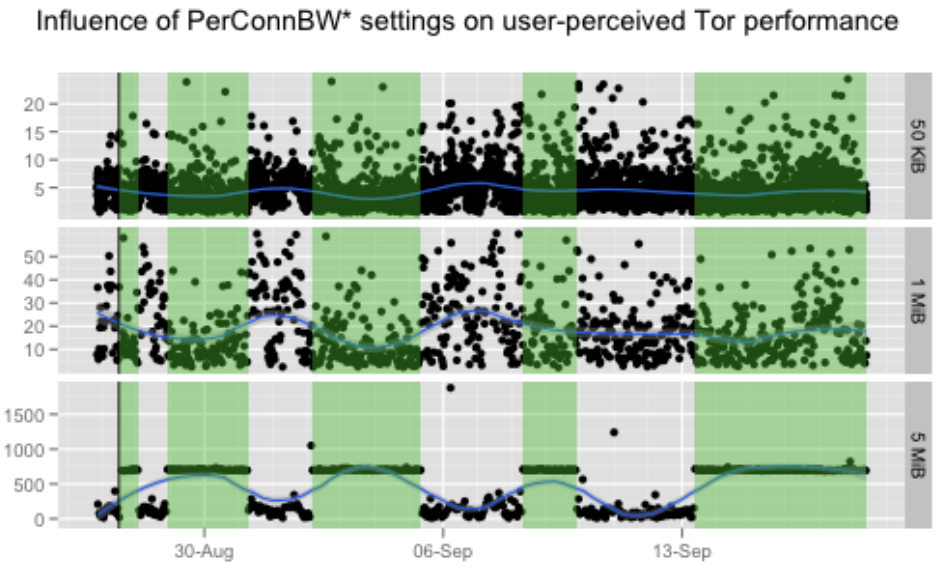

Figure 1: Influence of PerConnBW* settings on user-perceived Tor performance over time

parameters, so we can change how much throttling there is and then observe the results on the network.

So the first question is to model how this should work to improve performance at a single entry guard. Say we do periodic performance tests fetching files of size 50KB, 1MB, and 5MB using that relay as our first hop, and say the relay sets PerConnBWRate to 5KB/s and PerConnBWBurst to 2MB. That is, bursts of up to 2 megabytes on the connection are unthrottled, but after that it's squeezed to a long-term average of 5 kilobytes per second, until the flow lets up at which point the allowed burst slowly builds back up to 2MB. We would expect the 5MB tests to show horrible performance, since they'll need at least $3000/5 = 600$ seconds to fetch the last 3MB of the file. We would expect the 50KB and 1MB tests to look *better*, though: if we're squeezing out the really big flows, there's more space for what's left.

We actually performed this experiment, using Sebastian's relay fluxe3. You can see his performance graphs over time in Figure 1. The black dots are individual fetches, the y axis is how many seconds it took to fetch the file, and the x axis is time (the green-shaded areas are when the feature is turned on). Don't pay too much attention to the blue smoothing line; it's probably not matching the actual distribution well.

Figure 2 has the cumulative distribution functions for the same data. The performance gets significantly better both for the 50KB and the 1MB downloads, and as expected gets significantly worse for the 5MB downloads.

So far so good; we've done the proof of concept, and now we need a research group to step in, make it rigorous, and help tackle the real research questions.

The next question is: how well does this trick work under various conditions? It would seem that if there's plenty of spare bandwidth on the relay, it should have little effect. If we choose parameters that are too lenient, it also would have little effect. But choosing parameters that are too low will hurt normal web browsing users too. Are there rate and burst values

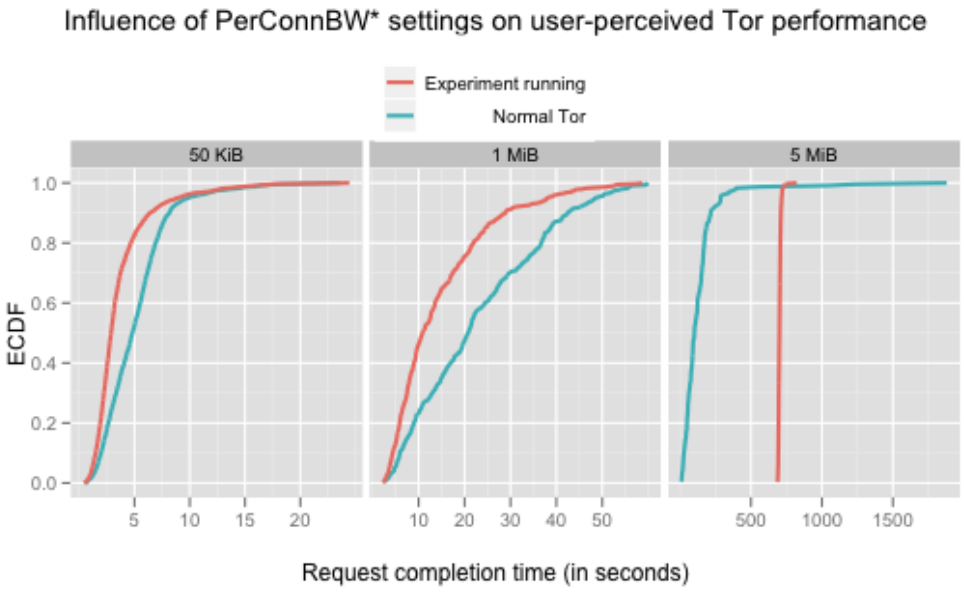

Figure 2: Empirical cumulative distribution functions of the influence of PerConnBW* settings on user-perceived Tor performance

that would cleanly separate web browsers from bulk downloaders? Will certain relays (that is, relays with certain characteristics) provide clearer performance improvements than others?

How often is it the case, for various relay capacities and various user loads, that at least one connection is being throttled? The CDFs appear to show improved performance spread out pretty evenly relative to download time. What statistics should we make relays track over time so we can get a better intuition about the load they're really seeing?

Now is where it gets tricky. If we're only thinking about one relay turning on this feature in a vacuum, then if that relay has enough capacity to handle all its flows, it should—no sense slowing down *anybody* if you can handle all the traffic. But instead think of the entire Tor network as a system: your Tor requests might also slow down because a loud Tor flow entered at some other relay and is colliding with yours somewhere along the path. So there's a reason to squeeze incoming connections even if you have enough capacity to handle them: to reduce the effects of bottlenecks elsewhere in the system. If all relays squeeze client connections, what changes in the performance test results do we expect to see for various rate limiting parameter values?

We have the capability of doing network-wide experiments to validate your theory, by putting the rate and burst in the hourly networkstatus consensus and letting relays pick it up from there. That is, you can modify the global network-wide parameters and then observe the effects in the network. Note that the experiment will be complicated by the fact that only relays running a sufficiently recent version of Tor will honor the parameters; that fraction should increase significantly when Tor 0.2.2.x becomes the new stable release (sometime in late 2010).

And finally, once you have a good intuition about how throttling flows at point X affects flow performance at point Y, we get to the really hard research questions. It's probably the case that no fixed network-wide rate limiting parameters are going to provide optimal behavior: the

parameters ought to be a function of the load patterns on the network and of the capacity of the given relay. Should relays track the flows they've seen recently and adapt their throttling parameters over time? Is there some flow distribution that relays can learn the parameters of, such that we e.g. throttle the loudest 10% of flows? Can we approximate optimal behavior when relays base their parameters on only the local behavior they see, or do we need a more global view of the system to choose good parameters? And what does "optimal" mean here anyway? Or said another way, how much flexibility do we have to define optimal, or do the facts here railroad us into prefering certain definitions?

What are the anonymity implications of letting the behavior of user A influence the performance of user B, in the local-view case or the global-view case? We could imagine active attacks that aim to influence the parameters; can those impact anonymity?

As a nice side effect, this feature may provide a defense against Sambuddho's bandwidth-based link congestion attack [1], which relies on a sustained high-volume flow—if the attack ever gets precise enough that we can try out our defense and compare.

Here are some other constraints to keep in mind:

- We need the Burst to be big enough to handle directory fetches (up to a megabyte or two), or things will get really ugly because clients will get slowed down while bootstrapping.
- Mike's bandwidth authority[2] measurement scripts[3] send traffic over relays to discover their actual capacity relative to their advertised capacity. The larger the claimed capacity, the more they send—up to several megabytes for the fast relays. If relays throttle these bandwidth tests, the directory authorities will assign less traffic to them, making them appear to provide better performance when in fact they're just being less useful to the network. This is an example of a case where it would be easy to misinterpret your results if you don't understand the rest of the Tor system. A short-term workaround would be to turn the bandwidth authority Tors into relays so they don't get throttled.
- You'll want to do experiments on an established relay with the Guard flag, or it probably won't see many client connections except for clients fetching directory updates. Note that the longer a relay has had the Guard flag, the more users it will have attracted; but after a month this effect falls off.
- Thinking from an economic perspective, it may turn out that the performance for a given user doesn't actually get better if we throttle the high-volume users, yet we've still improved things. That is, if the available capacity of the Tor network improves and thus it is supporting more users, we've made a better Tor network even if we didn't make it faster for individual users. Fortunately, this feedback effect shouldn't happen for short-term experiments; but it's something to keep in mind for the long term.
- It sure would be nice if whatever rate limiting algorithm you recommend is efficient to calculate and update over time. Some relays are seeing tens of thousands of users, and can't afford much processing time on each.

[2] https://blog.torproject.org/blog/torflow-node-capacity-integrity-and-reliability-measurements-hotpets

[3] https://gitweb.torproject.org/torflow.git/blob/HEAD:/NetworkScanners/BwAuthority/README.BwAuthorities

References

[1] Sambuddho Chakravarty, Angelos Stavrou, and Angelos D. Keromytis. Traffic analysis against low-latency anonymity networks using available bandwidth estimation. In *Proceedings of the 15th European conference on Research in computer security*, ESORICS'10, pages 249–267, Berlin, Heidelberg, 2010. Springer-Verlag.

[2] Can Tang and Ian Goldberg. An improved algorithm for Tor circuit scheduling. In Angelos D. Keromytis and Vitaly Shmatikov, editors, *Proceedings of the 2010 ACM Conference on Computer and Communications Security, CCS 2010, Chicago, Illinois, USA, October 4–8, 2010*. ACM, 2010.