Poster – PnP: Improving Web Browsing Performance over Tor Using Web Resource Prefetch-and-Push

Giang T. K. Nguyen† Xun Gong‡ Anupam Das† Nikita Borisov‡
†Department of Computer Science
‡Department of Electrical and Computer Engineering
University of Illinois at Urbana-Champaign
{nguyen59,xungong1,das17,nikita}@illinois.edu

ABSTRACT

Tor is a widely used network for anonymous communication. Its users frequently experience large communication delays, due to the high user-to-relay ratio, the bandwidth-intensive BitTorrent transfers of a small fraction of the user base, and the inherent latencies from routing traffic through multiple relay hops scattered around the world. These delays significantly degrade the user experience of web browsing, a dominant use of Tor.

Improving web browsing performance of Tor has been a subject of much research. Targeting the network and transport layers, prior work includes proposals to throttle bandwidth-intensive connections or to prioritize interactive traffic such as web browsing. We attack the problem at the application layer, noting that a typical web page consists of multiple resources, each of which requires a one-round-trip HTTP request-response cycle to load in the browser. Thus, for a page with many resources, these round trips are a major contributor to the page load time. We investigate PnP (for Prefetch-and-Push), where the Tor exit prefetches resources of the web page a client is visiting and pushes them to the client. Our experiments show a significant reduction in page load times as well as higher client's privacy from web page fingerprinting by a local attacker.

Categories and Subject Descriptors

C.2.0 [**Computer-Communication Networks**]: General—*Data communication*; C.2.1 [**Computer-Communication Networks**]: Network Architecture and Design; C.4 [**Computer Systems Organization**]: Performance of Systems

Keywords

Tor; web; latency; prefetch; fingerprinting

1. INTRODUCTION

Tor [5] is a widely used network for anonymous communication. Its users frequently experience large communication delays; in part, it is because the network comprises of about 4 000 volunteer-run relays supporting 500 000 daily users.[1] Moreover, Tor is popular among BitTorrent users who, while representing a small fraction of the overall users, consume a large fraction of the overall bandwidth [7]. Finally, some delays are inherent in anonymous routing, as paths traverse relays scattered around the world to better hide the true origin of a connection. These delays significantly degrade the user experience of web browsing, a dominant use of Tor.

Improving the delay performance of Tor has been a subject of much research. Specifically focusing on web browsing are proposals to throttle bandwidth-intensive applications such as BitTorrent [2, 6, 8] or to prioritize low-bandwidth traffic such as web browsing during scheduling [10]. These techniques can improve the delay performance, but users can still experience large delays.

We attack the problem at the application protocol layer (an approach Murdoch also considered in an early proposal [9]), noting that a typical web page consists of multiple embedded resources (e.g., images, cascading style sheets, etc.). Each embedded resource requires a one-round-trip HTTP request-response cycle to load in the browser. Thus, for a web page with many resources, these round trips, especially across a typical three-hop Tor circuit, is a major contributor to the delay in loading the page. In our design, the Tor exit prefetches the resources of the web page a client is visiting and pushes them to the client. Note that in contrast with previous work on link *prediction* [4], which prefetches *pages* that the user is likely to visit *next*, we instead prefetch *components* of the page that the user is visiting *now*.

Prefetching improves application-level performance even as the transport layer delays remain the same. It is complementary to other approaches to improve Tor performance. In particular, the priority scheduling proposed by Tang and Goldberg [10] is currently part of the Tor code base and is included in the baseline performance in our analysis.

2. DESIGN

Fig. 1 shows the overall architecture of our design. We make use of two proxies, one on the client host (client-side proxy, or CSP) that sits between the user's browser and the Tor client, and one on the Tor exit node (server-side proxy, or SSP) that sits between the Tor exit router and the website. This allows us to support PnP without modifying browser software or the Tor codebase.

When the browser requests a web page, the CSP forwards the HTTP GET request to the SSP, which proceeds to load the whole web page at the specified URL as a browser would: this entails *prefetching* resources embedded in the page. As the SSP receives these resources, it *pushes* them over the Tor circuit towards the CSP.

CCS'13, November 4–8, 2013, Berlin, Germany.
ACM 978-1-4503-2477-9/13/11.
http://dx.doi.org/10.1145/2508859.2512519.

[1]Figures obtained from the Tor Metrics Portal in August 2013: https://metrics.torproject.org/

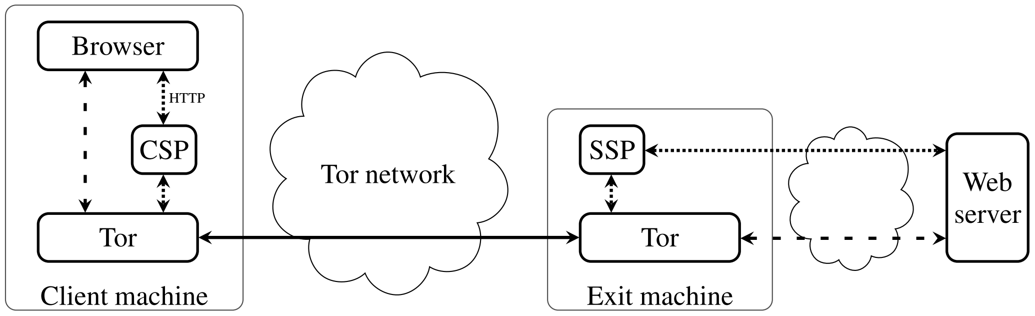

Figure 1: PnP overall architecture, with the SSP co-located with the Tor exit relay. The CSP provides an HTTP interface to the browser. The dotted and dashed arrows represent communication channels when using and when not using PnP, respectively.

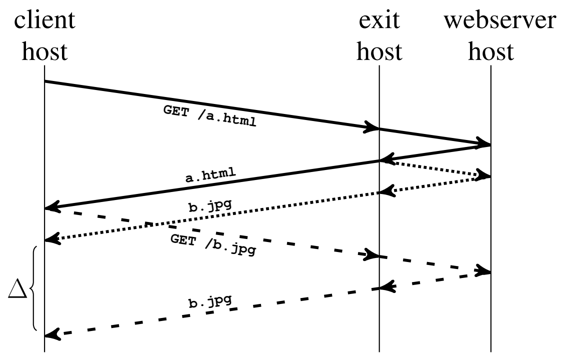

Figure 2: Timing diagram illustrating the improved page load time for a web page with a single resource b.jpg. The dotted and dashed arrows represent messages exchanged when using and when not using PnP, respectively, and Δ represents the difference in page load time.

When the CSP receives a pushed resource, it can deliver the resource to the browser immediately if it has already received a request from the browser for that resource. Otherwise, the CSP temporarily saves the resource in memory and later delivers it upon the browser's request. We note that, if the browser requests a resource that the CSP has yet to receive from the SSP, then the CSP queues the request but does not forward it to the SSP, under the assumption that the SSP will soon push the resource. This means that browsing a web page involves sending only one request over the network. Fig. 2 shows the exchanged messages and the resulting improved page load time for a toy web page with a single resource.

3. PROTOTYPE AND EXPERIMENTS

3.1 Prototype Implementation

We have implemented a PnP prototype. Part of the SSP has to load whole web pages; for this purpose, we use `QWebPage` from the Qt application development framework.[2] `QWebPage` uses the rendering engine `QtWebKit`[3] to fetch the page's HTML and other embedded resources. For the CSP-SSP communication, we use Google's SPDY[4] protocol, which is designed to efficiently multiplex multiple HTTP "connections" over a single TCP connection, supports "server-push," and has open-source library implementations. The CSP is a modified `shrpx` proxy from the `spdylay` project,[5] and the SSP builds on `spdylay`'s SPDY API.

[2] http://qt-project.org/
[3] http://trac.webkit.org/wiki/QtWebKit
[4] http://www.chromium.org/spdy/spdy-whitepaper

3.2 Modeling web pages

Previous work on Tor performance used a single file to represent a web page. To evaluate PnP, we need a more realistic representation of web pages with multiple resources. Instead of experimenting with real web pages, which change over time, we create simple static models that preserve the numbers and sizes of resources in those web pages: for each resource in the real page, we embed a JPEG image of the same size in the model. We note that our tests lack the complex dependency structure exhibited by real web pages [11]; we plan to improve the web page models in future work.

3.3 Experiment over Live Tor Network

Setup. We deploy a public Tor relay that allows exit connections only to our SSP and web servers. To reduce variability, the web servers are on the same local network as the exit, but we add an 80 ms RTT to emulate the wide-area network. The web servers host page models of 17 top news websites such as CNN and the BBC (with JavaScript enabled). These page models range from 60 to 250 resources and from 400 KB to 2.5 MB in total sizes.

The client is a desktop machine at our lab. It is configured to use only one guard entry (though due to network churn, it used three different guards during the experiment) and use only our exit relay for all connections. The client loads each of the 17 page models 40 times, alternating between using and not using PnP (20 times each). The client browser is a minimal browser based again on `QWebPage`. To ensure the integrity of each page load, the browser is given a manifest (with SHA-1 digests) of the page it is loading and verifies that it correctly loads only resources specified in the manifest. We also collect packet captures on the client machine for fingerprinting analysis. In all, we have 680 successful page loads.

Page load time results. As Fig. 3 shows, using PnP achieves a significant improvement over the baseline: the median and the 90th percentile page load times improve by $\sim$9 seconds, or 62% and 47% reductions, respectively. Also, we note the lower variability and the shorter tail in page load times if using PnP.

Mitigation of website fingerprinting. Recent work showed that it is possible to learn the identity of a web page being retrieved over Tor by searching for web page fingerprints inside the client's traffic patterns. Cai et al. use the Damerau-Levenshtein distance as well as support vector machines to achieve 84% accuracy in classifying a set of 100 web pages retrieved over Tor [3].

[5] http://sourceforge.net/projects/spdylay/

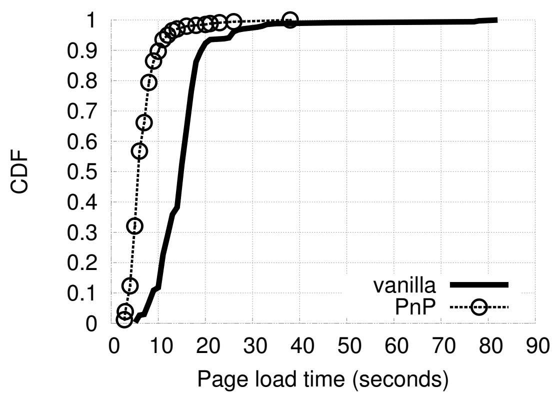

Figure 3: Page load time of 680 total page loads of 17 page models over the live Tor network on July 11, 2013, with and without using PnP.

We apply their techniques to the traces of browsing the 17 web page models produced by our experiments and compare web page classification tasks with and without PnP enabled. We estimate the classification accuracy by performing 10-fold cross validation tests. The average success rate is 99% when training on and classifying vanilla traces, and drops to 62% when training on and classifying PnP traces. This suggests PnP is a promising technique to defend against website fingerprinting attacks.

4. CHALLENGES AND FUTURE WORK

We discuss some of the challenges in designing PnP, possible solutions, and future research directions.

Time-space trade-off. Due to caching or customizations (e.g., by extensions), the client browser might not request some resources when loading a web page. Thus, PnP will waste bandwidth if it prefetches and pushes these resources. We plan to conduct a measurement study on real Tor exits to estimate the expected wasted bandwidth. For example, when an exit observes a request for a web page, it separately loads the page and sees that `b.jpg` is part of the page. If the exit does not subsequently observe the client's request for `b.jpg`, then `b.jpg` counts towards wasted bandwidth. Note, though, that the client might in fact request `b.jpg` through a different exit, in which case, `b.jpg` would not have been wasted because with PnP in use, the CSP would request the web page through the same SSP, exit policies permitted.

Correctness. Web servers use HTTP cookies to authenticate users, to provide user-specific content, etc. Without access to the client browser's cookies, the SSP might not be able to retrieve the correct resources. A possible solution is for the client, when loading a new page, to guess based on previous visits to the page which of its stored cookies will be relevant for loading embedded resources, and to transfer those cookies to the SSP, which can use them when prefetching the page. There is little additional security risk (besides over-guessing and sending extraneous sensitive cookies) because without PnP, the client will still send those cookies along subsequent requests for embedded resources.

HTTPS support. If the client communicates with the web server over HTTPS, then PnP is unable to perform prefetch-and-push. To support HTTPS, the client must trust PnP to be a man-in-the-middle. One way to achieve this is for the owner of the web server to deploy a Tor exit/SSP securely under his control and give the SSP access to his web server's TLS private key. However, HTTPS support might not be crucial at this time because according to a recent study, HTTPS is <5% of all web traffic [1].

Acknowledgments

We would like to thank Roger Dingledine and Robert Johnson for initial discussions about using SPDY in Tor. This material is based upon work supported by the National Science Foundation under Awards Nos. 093655 and 1054937, by the Army Research Office under Award No. W911NF–13–1–0086, by the Air Force under Award No. FA 9550–11–1–0016, and by the Boeing Trusted Software Center at the University of Illinois at Urbana-Champaign.

5. REFERENCES

[1] B. Ager, N. Chatzis, A. Feldmann, N. Sarrar, S. Uhlig, and W. Willinger. Anatomy of a large european IXP. In *Proceedings of the ACM SIGCOMM 2012 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication*, SIGCOMM '12, pages 163–174. ACM, 2012.

[2] M. AlSabah, K. Bauer, and I. Goldberg. Enhancing Tor's performance using real-time traffic classification. In *Proceedings of the 2012 ACM Conference on Computer and Communications Security*, CCS '12, pages 73–84. ACM, 2012.

[3] X. Cai, X. C. Zhang, B. Joshi, and R. Johnson. Touching from a distance: Website fingerprinting attacks and defenses. In *Proceedings of the 2012 ACM conference on Computer and Communications Security*, CCS '12, pages 605–616. ACM, 2012.

[4] X. Chen and X. Zhang. A popularity-based prediction model for web prefetching. *Computer*, 36(3):63–70, Mar. 2003.

[5] R. Dingledine, N. Mathewson, and P. Syverson. Tor: the second-generation onion router. In *Proceedings of the 13th conference on USENIX Security Symposium - Volume 13*, SSYM'04, pages 21–21, Berkeley, CA, USA, 2004. USENIX Association.

[6] R. Jansen, P. Syverson, and N. Hopper. Throttling Tor bandwidth parasites. In *Proceedings of the 21st USENIX Security Symposium*. USENIX Association, 2012.

[7] D. Mccoy, K. Bauer, D. Grunwald, T. Kohno, and D. Sicker. Shining light in dark places: Understanding the Tor network. In *Proceedings of the 8th Privacy Enhancing Technologies Symposium*, PETS '08, pages 63–76. Springer-Verlag, 2008.

[8] W. B. Moore, C. Wacek, and M. Sherr. Exploring the potential benefits of expanded rate limiting in Tor: slow and steady wins the race with Tortoise. In *Proceedings of the 27th Annual Computer Security Applications Conference*, ACSAC '11, pages 207–216. ACM, 2011.

[9] S. J. Murdoch. Using the SPDY protocol to improve Tor performance. `https://gitweb.torproject.org/torspec.git/blob/HEAD:/proposals/ideas/xxx-using-spdy.txt`. Accessed August 20, 2013.

[10] C. Tang and I. Goldberg. An improved algorithm for Tor circuit scheduling. In *Proceedings of the 17th ACM Conference on Computer and Communications Security*, CCS '10, pages 329–339. ACM, 2010.

[11] X. S. Wang, A. Balasubramanian, A. Krishnamurthy, and D. Wetherall. Demystifying page load performance with WProf. In *Proceedings of the 10th USENIX Symposium on Networked Systems Design and Implementation*, NSDI '13. USENIX Association, 2013.