Optimizing High Latency Links in the Developing World

Yaw Anokwa, Colin Dixon, Gaetano Borriello
Computer Science and Engineering
University of Washington
{yanokwa,ckd,gaetano}@cs.washington.edu

Tapan Parikh
Information School
University of California, Berkeley
parikh@ischool.berkeley.edu

ABSTRACT

Long distance Wi-Fi links, satellite connections, and other low-bandwidth, high-latency, intermittent options are becoming the norm for providing connectivity in the developing world. For network administrators who must manage these connections, providing users the "best" (or even adequate) service is not a trivial problem.

Previous work has focused on optimizing throughput and while we acknowledge the importance of this approach, we argue that latency is an important and often ignored component of network performance. The intrinsically high latencies seen in the developing world are exacerbated by excessive queueing from traffic which often swamp links with miss-sized queues. Current solutions to this problem tend to require resources (people, time and money) that are generally not available in developing environments. In this paper, we demonstrate that latency is a problem in real world deployments and propose an easy to deploy solution.

Categories and Subject Descriptors

H.2 [**Computer-Communication Networks**]: Internetworking; H.5 [**Information Interfaces and Presentation**]: Miscellaneous

General Terms

Design, Human Factors, Performance, Reliability

1. INTRODUCTION

Many areas in the developing world are quickly gaining broadband links to Internet. According to the World Bank [7] low-income countries saw a 120% increase in Internet bandwidth from 1999-2004 and a 143% increase in broadband subscribers from 2001-2005. From long-distance Wi-Fi to VSAT, many of these connections provide high-latency, low-bandwidth links for governments, businesses and schools. The connections, once provided, are often utilized by a large number of users whose traffic quickly saturates the link. This common occurrence was observed by the first author who worked as a network administrator for two hospitals and four health clinics in rural Rwanda.

WiNS-DR'08, September 19, 2008, San Francisco, California, USA.

Modern modems for typical thin access links like DSL, cable and VSAT are not the most carefully engineered devices and do a poor job at queue management [9]. Additionally, because TCP only backs off in response to loss, connections ignore the increased queuing delay leading to full queues which add seconds of extra delay. The result is that a single heavy TCP connection (like a large movie download) can increase the latency of all connections by multiple seconds when using such access links.

This latency increase not only makes interactive tasks nearly intolerable, it also directly affects the throughput of short-lived flows (like a webmail connection) that spend their entire lives in TCP slow start. In slow start, the bandwidth increase is directly affected by round trip times, so a significant increase in latency will mean that new flows will take longer to transfer even moderate amounts of data. This is especially damning given that most traffic in the developing world consists of HTTP requests for small objects [10].

2. PREVIOUS WORK

Of course, the observation that high latency, low bandwidth connections are problematic is not new. There is a large body of work ranging from tweaking protocols [6, 12, 4, 14, 18, 2, 3] to caching [17, 13] and delay tolerant networking [11].

Recently, Du et al. [10] analyzed trace data from Internet cafés and kiosks in Cambodia and Ghana. A strong case is made that because of the high costs of connection, low data rates and high demand, any improvements on the network bandwidth has a significant effect on user benefit. The authors propose a number of suggestions (including ad blocking, offline caching, time shifting, and compression) which could provide a better experience for users. While we agree with many of the claims the authors make, our work is aimed at latency. In our deployments, we have found users are relying on sites that serve dynamic content. For those users, their use of these services exacerbates the already high latencies and has a significant impact on user-perceived performance [8].

One could rely on options in the home or commercial domain and enable QoS on an off-the-shelf router. Some of the most popular routers use the built in Linux features `iptables` and `tc` to implement coarse-grained QoS for asymmetric links [1]. The approach is unfortunately outdated as one might expect from a shell script originally written in 2001. Coarse-grained filtering based on port and protocol is no longer sufficient to provide reasonable prioritization because, among other things, the confluence of services onto port 80. This confluence is especially pronounced in the developing world where computers are shared and webmail is one of the dominant applications. For these environments, the ability to make choices based on variables beyond ports and protocols is key.

Finally, previous work assumes a level of expertise that is not easily found in the developing world. Alternatives to the approaches

listed above are the free and practical guides [16, 15] which cover everything from policy for users to troubleshooting links. Unfortunately, at a couple hundred pages of fairly technical text, the accessibility of information is still a problem. So while some researchers are aware of the issue of latency, widely deployed and cost effective solutions are hard to find in the field.

In the rest of this paper, we provide data which details the extent of the problem and a simple to deploy and maintainable solution.

3. THE PROBLEM

To demonstrate the problem, we measured latencies from a VSAT sourced network in rural Rwanda. The network supports a district hospital and sees two way web and email traffic in addition to occasional VoIP traffic. Available bandwidth is capped at 700 Kbps down and 200 Kbps up and uses no caching or quality of service tools.

We gathered data using simple UDP probes in each direction at a rate of approximately one probe every 2 seconds over 3 days starting on March 4th, 2008. To separate the effects of queueing in each direction, we sampled the uplink and downlink separately. This results in some effects of clock skew in the data, but it falls squarely in the noise when compared to the larger-scale phenomenon we are interested in.

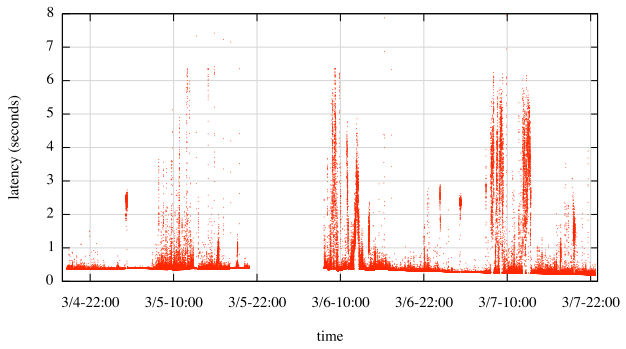

Figure 1: Uplink latencies from Rwanda to Seattle as seen by periodic small UDP probes.

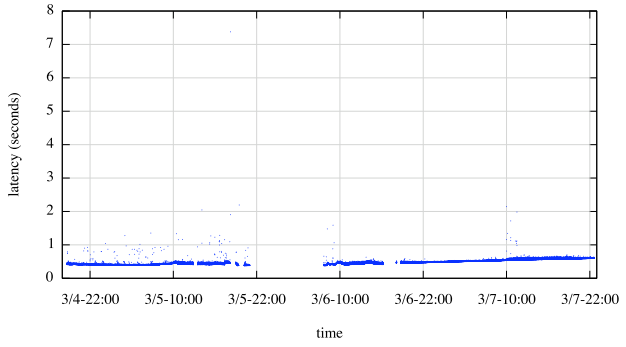

Figure 2: Downlink latencies from Seattle to Rwanda as seen by periodic small UDP probes.

The results of probes run over 3 days are presented in Figure 1 for the uplink and Figure 2 for the downlink. As can be seen, the majority of the congestion occurs on the uplink with sustained latency spikes well over 2000 ms. The downlink is comparatively quiescent which we attribute to its comparatively larger throughput.

The average delay seen on the downlink is 478 ms while the minimum is 376 ms. The 100 ms difference is due to a few scattered high latency measurements and consistent clock skew increasing latency for the last day of the measurements. On the other hand, the uplink has an average latency of 572 ms with the same minimum. This increase in the average latency is understated by the opposite effect of the clock skew in this direction, so in practice, there is more than a 200 ms or 50% increase in average latency on the uplink due to congestion.

While it might seem as though simply resizing the buffer would solve the problem, in practice this can at most mitigate the effects. Reducing buffer sizes to below the bandwidth-delay product of a connection results in being unable to fully utilize the available bandwidth of a connection, and without knowing the delay of connections a priori, queues must be sized to match the longest possible delay.

The gap in data overnight from March 5th to 6th is the result of failure on the machine where the probe was executing. The cause of the problem is currently unknown, but was solved by a reboot when the staff arrived in the morning. Frequent downtime is one of the hazards of working with technology in the developing world.

4. PROPOSED SOLUTION

We propose that the simple QoS prioritization used in [1] be extended to provide more fine-grained, adaptive and behavioral classification of flows rather than purely based upon protocol and port. This approach is motivated by both the increasing ineffectiveness of port-based classification and the need for behavior-based classification to ease the burden of creating new heuristics whenever new protocols come into use.

Unfortunately, while port- and protocol-based classification are well supported in the Linux kernel, behavioral classification currently lacks any noticeable support from the usual networking tools. A likely solution involves making the assumption that high-bandwidth flows will be more tolerant of high-latency, whereas low-bandwidth flows will likely better benefit from low-latency.

This approach will likely work because of the workload we see in the developing world. The workload is dominated by HTTP traffic where object sizes are heavy-tailed, which in turn implies that flow duration will be heavy tailed. Thus, flows which have used a considerable amount of bandwidth are likely to continue dominating the links. Such flows should be tolerant of extra delay.

On the other hand, there will be many short flows for which the latency, and not the throughput, will dominate performance. These short flows will not tolerate extra delay well. This approach mirrors previous work [14] which used similar heuristics along with TCP window tweaks to classify and prioritize downstream flows.

In the spirit of simplifying network management, we advocate a solution involving two classes for flows: fast and slow. The fast class will contain a number of the smallest flows such that they will not cause any noticeable queueing among themselves and can thus be guaranteed to achieve near the physical link latency. The rest of the flows will be classified as slow and given no guarantees of reasonable latency. An example of a fast flow is the individual HTTP requests for text on webpages, while an example of a slow flow is requests for large file downloads.

The slow flows are prevented from interfering with the fast flows by enforcing that packets not reach the access link fast enough to cause queueing delay at the link itself, but instead all queueing is managed at a single point under our control. Similarly, slow flows must be protected from starvation, which is done by guaranteeing them some fraction of the link at all times. The rest of the link not used by the fast flows will also be available to slow flows. Fast

flows which consume more than expected bandwidth are eventually migrated to the slow class and vice versa. This is accomplished with a sliding window estimation of each flow's current throughput.

In the past, it has also been typical to use the source address, source port, destination address, destination port 4-tuple to identify flows, but many download accelerators and some web browsers open multiple TCP connections to improve performance. These connections should likely all be considered part of the same flow. Classifying flows based purely on source and destination addresses should improve fairness.

While not adversary-proof, flow-classification also prevents a clever user (often the network administrator) from repeatedly opening short-lived connections to the same server to download a larger file more quickly by exploiting that classification of each of the smaller connections into the fast class. A potential downside is that some protocols (like FTP or even some new Web 2.0 applications) will have both control and data connections open to the same server and the control connection may be unduly punished.

5. NEXT STEPS

In our experience, network administrators in the developing world do not have the training required to deploy many of the solutions proposed in previous work. Our contribution in this work is the push to put a larger emphasis on providing solutions on deployable platforms.

The first platform is a standard desktop Linux distribution. Our final product will be a package which can be installed on any distribution or a custom small distribution which can be downloaded and installed on any x86 machine. Our goal will be to provide an easy to use GUI which hides much of the complexities of the system. While we would like to support Windows, the networking environment in Windows makes it difficult to reroute traffic. It is our hope that with the progression of virtual machines, Windows network traffic can eventually be routed through a virtual machine where we place our solution.

The second platform is OpenWRT [5], the popular Linux based firmware that runs on a variety of routers from the Linksys WRT54G to the Asus WL-500G Premium. These routers can be found for under $80 and from suppliers that can source components to developing world retailers. In the case of the WL-500G, the router is equipped with USB 2.0 ports which can be used for external hard drives or flash disks. This storage capacity can be used to implement services such as file sharing and Squid caching.

We think these routers are a great platform because they use less power than desktops, can be easily protected from power glitches, are robust to the elements and are easy to configure. Moreover, there is a large and responsive open source community that supports the platform. In addition to the software and hardware platforms, we will also be including extensive documentation and examples to assist network admins in deploying these solutions.

The first author is returning to Rwanda soon and rather than deploying the solution, he will present the local network administrators with the documentation, software and hardware needed to manage the high latency link for themselves. We will then evaluate the change in performance observed over the link as well as the interface in order to verify the efficiency of our approach.

6. CONCLUSION

Latency is an important and often ignored component of optimizing the performance of Internet connections in the developing world. Excessive queueing driven by traffic patterns often swamp thin access links with miss-sized queues. Current solutions to this problem tend to require resources that are generally not available in developing environments. In this paper, we have presented data that demonstrates the problem and proposed a solution that we plan to deploy and evaluate.

7. ACKNOWLEDGMENTS

The authors thank Bowei Du for providing data from his work.

8. REFERENCES

[1] ADSL bandwidth management. `http://www.faqs.org/docs/Linux-HOWTO/ADSL-Bandwidth-Management-HOWTO.html`.

[2] Background intelligent transfer service. `http://en.wikipedia.org/wiki/Background_Intelligent_Transfer_Service`.

[3] Differentiated services. `http://en.wikipedia.org/wiki/Differentiated_services`.

[4] Enhancing TCP over satellite channels using standard mechanisms. `http://www.ietf.org/rfc/rfc2488.txt`.

[5] OpenWRT. `http://openwrt.org/`.

[6] Performance enhancing proxies. `http://www.ietf.org/proceedings/99jul/I-D/draft-ietf-pilc-pep-00.txt`.

[7] World Bank. *Global Economic Prospects 2008: Technology Diffusion in the Developing World*. 2008.

[8] Tom Beigbeder, Rory Coughlan, Corey Lusher, John Plunkett, Emmanuel Agu, and Mark Claypool. The effects of loss and latency on user performance in Unreal Tournament 2003. In *NetGames*, 2004.

[9] Mark Claypool, Robert Kinicki, Mingzhe Li, James Nichols, and Huahui Wu. Inferring queue sizes in access networks by active measurement. In *PAM*, 2004.

[10] Bowei Du, Michael Demmer, and Eric Brewer. Analysis of WWW traffic in Cambodia and Ghana. In *WWW*, 2006.

[11] Kevin Fall. A delay-tolerant network architecture for challenged internets. In *SIGCOMM*, 2003.

[12] T.R. Henderson and R.H. Katz. Transport protocols for internet-compatible satellite networks. *Selected Areas in Communications, IEEE Journal on*, 17(2):326–344, Feb 1999.

[13] Sean C. Rhea, Kevin Liang, and Eric Brewer. Value-based web caching. In *WWW*, 2003.

[14] Neil T. Spring, Maureen Chesire, Mark Berryman, Vivek Sahasranaman, Thomas Anderson, and Brian N. Bershad. Receiver based management of low bandwidth access links. In *INFOCOM*, 2000.

[15] BMO Book Sprint Team. *How to Accelerate Your Internet*. Hacker Friendly LLC, 2006.

[16] Limehouse Book Sprint Team. *Wireless Networking in the Developing World*. INASP/ICTP, 2006.

[17] W. Thies, J. Prevost, T. Mahtab, G. Cuevas, S. Shakhshir, A. Artola, B. Vo, Y. Litvak, S. Chan, S. Henderson, M. Halsey, L. Levison, and S. Amarasinghe. Searching the world wide web in low-connectivity communities. In *WWW*, 2002.

[18] Arun Venkataramani, Ravi Kokku, and Mike Dahlin. Tcp nice: A mechanism for background transfers. In *Proceedings of the 2002 USENIX Operating Systems Design and Implementation (OSDI) conference*, December 2002.