Ultra-Low Delay for All: Live Experience, Live Analysis

Olga Bondarenko[1], Koen De Schepper[2], Ing-Jyh Tsang [2], Bob Briscoe[1],
Andreas Petlund[1], Carsten Griwodz[1]
[1]Media Performance Group, Simula Research Laboratory, Norway
[2]Nokia Bell Labs, Belgium
{olgabo, bob, apetlund, griff}@simula.no, {koen.de_schepper, ing-jyh.tsang}@nokia.com

ABSTRACT

This demo dramatically illustrates how replacing 'Classic' TCP congestion control (Reno, Cubic, etc.) with a 'Scalable' alternative like Data Centre TCP (DCTCP) keeps queuing delay ultra-low; not just for a select few light applications like voice or gaming, but even when a variety of interactive applications all heavily load the same (emulated) Internet access. DCTCP has so far been confined to data centres because it is too aggressive—it starves Classic TCP flows. To allow DCTCP to be exploited on the public Internet, we developed DualQ Coupled Active Queue Management (AQM), which allows the two TCP types to safely co-exist. Visitors can test all these claims. As well as running Web-based apps, they can pan and zoom a panoramic video of a football stadium on a touch-screen, and experience how their personalized HD scene seems to stick to their finger, even though it is encoded on the fly on servers accessed via an emulated delay, representing 'the cloud'. A pair of VR goggles can be used at the same time, making a similar point. The demo provides a dashboard so that visitors can not only experience the interactivity of each application live, but they can also quantify it via a wide range of performance stats, updated live. It also includes controls so visitors can configure different TCP variants, AQMs, network parameters and background loads and immediately test the effect.

CCS Concepts

•Networks → Transport protocols; Network performance analysis; Network management;

1. PROBLEM STATEMENT

Interactive latency-sensitive applications are becoming prevalent on the public Internet, e.g. Web, voice, conversational and interactive video, finance apps, online gaming, cloud-based apps, remote desktop. However, Choi *et al* [5] points out that there is probably more desire than measurable demand for interactive apps, because the lags in the general Internet prevent such apps even being developed. In the developed world, increases in access network bit-rate have been giving diminishing returns, as latency has become the critical bottleneck. Recently, much has been done to reduce propagation delay, e.g. by placing caches or servers closer to users. However, latency is a multi-faceted problem [4], so other sources of delay have become the bottleneck.

This demo focuses on dramatically cutting the queuing delay component. Queuing delay is often not apparent [8], but during periods of high load (whether brief or more persistent), it becomes a major but intermittent component of latency. This variability is either directly experienced by the user, e.g. as stalls in Web browsing, or real-time applications use buffers to absorb most of the jitter, which turns near-worst-case latency into the norm.

The demo does not follow the Diffserv approach [3], which only reduces queuing for some traffic at the expense of other more elastic traffic. That would be fine in the simplistic traditional view where elastic (capacity-seeking or 'greedy') TCP-like traffic is the culprit and other latency-sensitive traffic is the victim. However, this is no longer a sufficient model when some multimedia applications can be both capacity-seeking (culprit) and latency-sensitive (victim).

Instead, the demo (Fig. 1) is motivated by the idea that, often for each home, *all* apps will be latency-sensitive. For instance, visitors can use finger gestures or VR goggles to pan or zoom a sub-window of a panoramic video generated on servers representing 'the cloud' for the demo. Even though media delivery is over TCP, which continually seeks out capacity, and even though the base propagation delay to the cloud servers is typical of a broadband ISP's end-to-end network (7 ms, which is emulated using netem in the demo), visitors can verify that the queuing delay is so low that the interaction feels natural, even under high load—the video on the touchpad seems to stick to your finger. They can also relate their experience to a visualization of the distribution of queuing delay on the dashboard, even while they apply heavy load from numerous other delay-sensitive apps.

Our demo builds on the state-of-the-art in Active Queue Management (AQM), e.g. fq_CoDel [7], PIE [10], but shifts into a completely new design space. All AQMs at least ensure that a queue will not keep growing until it fills the buffer, which is important because buffers have often been mistakenly bloated to try to prevent loss. In general, AQMs introduce an increasing level of discard from the buffer the longer the queue persists above a shallow threshold. This gives sufficient signals to greedy flows to keep the buffer free for its intended purpose: absorbing bursts.

Unlike fq_CoDel, the demo avoids combining AQM with

MMSys'16 May 10-13, 2016, Klagenfurt, Austria

ACM ISBN 978-1-4503-4297-1/16/05.
DOI: http://dx.doi.org/10.1145/2910017.2910633

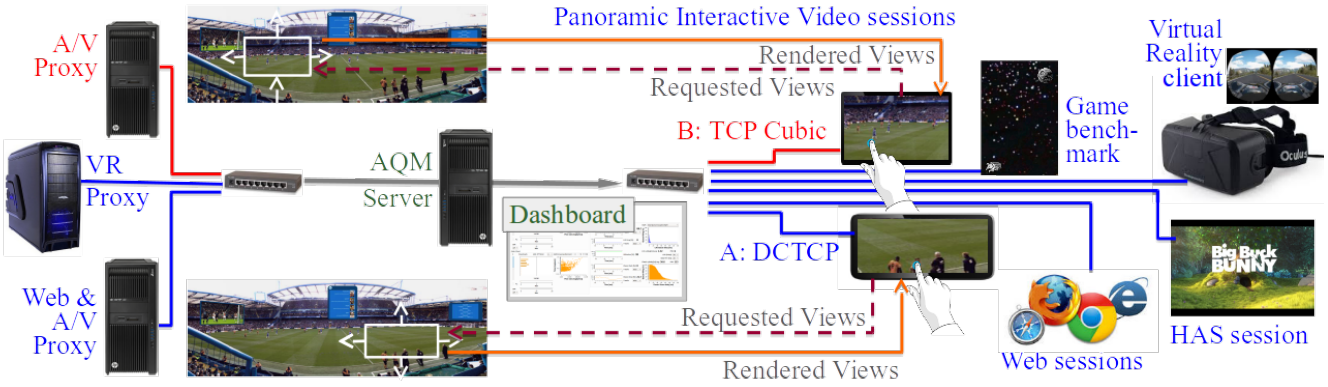

Figure 1: Demonstration of Ultra-Low Queuing Delay for *all* Applications using DCTCP over an Emulated Broadband Access Link from the Public Internet

per-flow queuing (fq), which has the following two flaws (amongst others). Firstly, fq isolates 'victim' interactive flows from the queuing of 'culprit' long-running greedy flows, but this does nothing for a greedy interactive media flow that is both culprit and victim. Secondly, whenever more than one large flow is running, fq_CoDel can only give every such flow constant bit-rate. This can seriously degrade multimedia applications with a continually varying information rate (e.g. VBR or constant quality video [9]). Indeed, visitors can use the dashboard of our demo to replace our AQM with fq_CoDel, to see how it makes HTTP Adaptive Streaming (HAS) degrade by one or two levels of encoding.

The demo attacks the inherent queuing variation of TCP itself. Even with a well-tuned AQM, a single Classic TCP flow (Reno or Cubic) will intermittently build up a queue of about one base round trip time (RTT) whenever a large-enough object is transferred, which doubles the total RTT [12].

2. SOLUTION: FIX THE ROOT CAUSE

As capacity has scaled, 'Classic' TCP has increased the scale of its saw-toothing rate variations. Our demo uses a scalable TCP, specifically Data Centre TCP (DCTCP [2]), which induces far smaller sawteeth and they stay small whatever the rate. They fit within a very shallow queue without compromising utilization, and with hardly any delay and throughput variation.

DCTCP does not only work in data centres. It was so-called because it starves Classic TCP flows, so it could only be deployed in private networks (like data centres) where all stacks could be switched over at once. DCTCP requires the network to mark the explicit congestion notification (ECN) field [11] in the IP header rather than dropping packets. Otherwise, if the peak of each little sawtooth induced a loss, the loss probability would be infeasibly high. DCTCP flows are designed to induce such frequent (aggressive) marking,

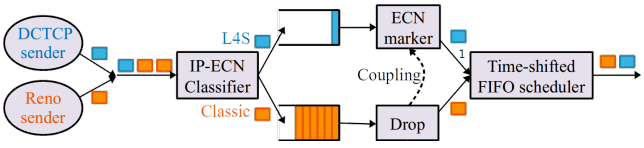

Figure 2: DualQ Coupled AQM: Two queues to protect low L4S latency; Coupling to balance flow rates

but Classic TCP flows mistake it for heavy congestion, so they keep yielding more capacity share.

The demo allows visitors to verify our solution to this co-existence problem: the DualQ Coupled AQM (DualQ for short). An overview is given here but an IETF draft [6] is available for details. The DualQ is deployed in the downstream bottleneck of an emulation of broadband Internet access (Fig. 2), where per-customer queues are typically located. It uses two queues, offering services called 'Classic' and 'Low-Latency, Low-Loss, Scalable' (L4S). The DualQ is like a semi-permeable membrane that stops the 'Classic' queue delaying the L4S, but allows bandwidth to be shared freely between them. So DCTCP flows give stunningly and predictably low queuing delay, but they do not starve the bandwidth of competing Classic traffic. These claims are quantified in a supporting paper, using a testbed built from real data centre, core, access and residential network equipment, including real DSL lines. The demo set-up has been verified to be comparable to this testbed.

DualQ applies congestion signals to L4S packets more aggressively than it applies drop signals to Classic packets, to exactly counterbalance the more aggressive response of DCTCP to congestion signals. Visitors can start different numbers of each type of flow and see live on the dashboard that they all end up with similar rates, as if they were all the same type of TCP. The DualQ AQM does not identify transport layer flows to do this; it inspects nothing deeper than the IP header.

The low delay service can only be enjoyed where all the pieces are in place: DCTCP on the client & server and the DualQ at the bottleneck queue. But every piece is incrementally deployable. For instance, if there is no DualQ at the bottleneck to generate the aggressive ECN markings, DCTCP falls back to 'Classic' TCP behaviour. The reduction in queuing delay is so remarkable that we expect ISPs to be keen to deploy the DualQ at known bottlenecks—primarily the buffers feeding the access links in each direction. And the DCTCP code already in Windows and Linux is good enough for testing, even though it could use some improvements for the general Internet.

3. INTERACTIVE EXPERIENCE

The demo (Fig. 1) consists of 9 physical machines and extra equipment for two types of latency sensitive applications:

- Cloud-based Interactive Video applications: Panoramic

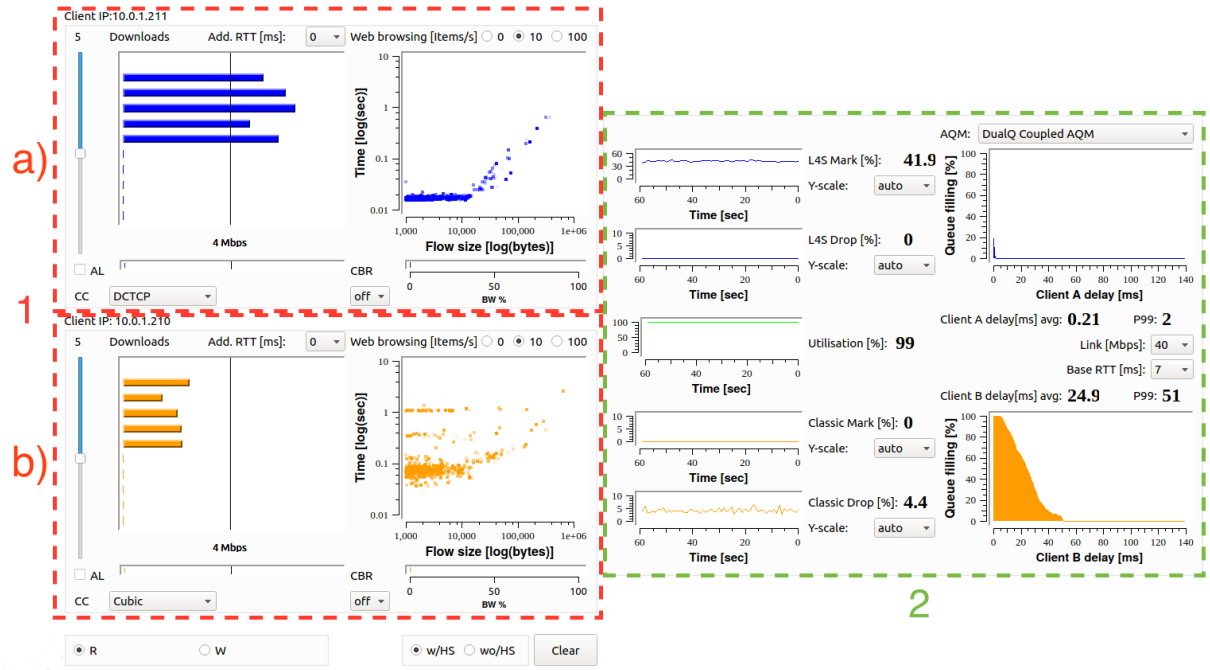

Figure 3: Dashboard for Live Configuration & Analysis

Interactive Video (PIV), Virtual Reality (VR);

- Web-based interactive applications: HTTP adaptive streaming (HAS), game-suitability benchmarking tool, web-browsing.

The Web-based gaming and browsing are very sensitive to extra delay, since there is a message exchange between the client and the server during every user interaction. They are also sensitive to the packet losses experienced with 'Classic' TCP, because they are prone to long time-out delays whenever a loss hit the tail of an object transfer. HAS video is also sensitive to delay during interactions (rewind, etc.) with the additional challenge of bursty bit-rate.

The cloud-based apps are perfect examples of the potential of ultra-low latency Internet access. With the PIV app, visitors use finger-gestures on a touch-screen to control panning and zooming. The client sends control messages to a proxy in the cloud that encodes a personalised HD scene on the fly as a sub-window of a panoramic video feed from a football match stitched together from multiple cameras. The encoded sub-window is streamed to the client using TCP over the Internet. In the VR case, the arrangement is similar. The visitors head movements make the Oculus Rift Virtual Reality (VR) goggles send scene selection messages to a proxy in the cloud, which encodes the appropriate scene from a spherical camera in a racing car and sends it back to the goggles for rendering. Depending on the video activity, these cloud-based apps alternate between capacity-seeking and app-limited behaviour.

With DCTCP and the DualQ AQM, the round trip delay from gesture to rendering is so low that the football video seems to stick to your finger, even under high load. In the immersive VR case, the delay is nearly imperceptible too, although the experience is tainted by the frame-rate. With other congestion controls and AQMs the extra queuing introduces noticeable lag that becomes unusable under load.

4. INTERACTIVE CONFIG & ANALYSIS

A dashboard has been developed that visitors can use to interactively configure different network conditions, TCP congestion controls and extra traffic load. It also allows visitors to visualize a wide range of resulting performance metrics (updated every second), which they can correlate live with the actual performance of the applications. The dashboard (Fig. 3) consists of two main parts:

1. Client blocks (labelled as "1" in the figure), which are sections allowing visitors to visualize and configure server-to-client traffic load in our testbed. Each block (1a), (1b) represents a different client-server pair, with their own TCP stacks.
2. AQM/Link block (labelled as "2" in the figure), allowing visitors to configure the network and visualize its response to the traffic load.

4.1 Client block

Each client block can be configured to select a TCP congestion control, e.g. DCTCP, Reno or Cubic, with or without ECN support. Extra emulated RTT delay can also be added. It includes the following graphical elements reflecting contributions to network load:

- long-running TCP flows (bar plot on the left);
- short/web TCP flows (scatter plot on the right);
- application-limited (AL) TCP flows (single bar plot directly below long-running flows on the left);
- constant bitrate (CBR) flow (single bar plot directly below web traffic on the right).

Long-running flows simulate the behaviour of capacity-seeking flows that generate bulk traffic. Visitors can choose the number of such long-running file downloads with the slider (for demonstration purposes the dashboard remotely controls the client's downloads using the secure copy Linux command (scp). AL flows are generated by all interactive applications we demonstrate - PIV, VR, HAS, emulated online gaming and web browsing (described in Section 3). CBR flow is emulated with iperf [1]. The bar plot next to the slider displays

throughput for each file download flow, while a similar bar plot below file downloads shows the sum of the throughput for AL flows. The throughput for file downloads and AL flows is shown relative to the "fair rate", which is marked with a line and calculated as link capacity excluding the bandwidth taken by CBR and AL flows, divided by the number of downloads. If all file downloads get a fair share of the bandwidth, their rate will be close to this line. The AL flows might use less bandwidth than the file downloads at times (being, by definition, application limited). However, for completeness their rate is also shown relative to the calculated fair share of the bandwidth. A CBR flow can be turned on by selecting the desired rate, which is represented as a percentage of the link capacity.

The short-running flows emulate web browsing, consisting of 10 or 100 flows per second (selected with the radio button). The plot below the radio buttons shows completion time for each of the flows in seconds. Both X and Y axes of the plot are on a log scale. The completion time is displayed as a measurement that includes TCP handshake by default, which can be switched to the measurement excluding the handshake (toggled by the radio buttons below both clients blocks: "w/HS", meaning "with HandShake", and "wo/HS", meaning "without HandShake"). The "Clear" button below the client blocks clears the completion time plots, removing the old data points.

4.2 Link/AQM block

Within the link/AQM block there are three combo boxes that allow visitors to configure the AQM (DualQ, RED, PIE or fq_CoDel), link capacity, and base RTT. The performance visualization aspects of the block consists of 2 parts: mark/drop probability and link utilization (on the left); and queue delay (on the right). All the measurements are displayed separately for each client (A and B). Depending on the selected AQM, the traffic shown as Client A and Client B will go through the same or different queues. Nonetheless, we always show them separately, so that it is clear how much delay is experienced by each of the end systems. Mark/drop probability and utilization measurements are shown both as a per 1-second sample (right) and as a 60-second history (left). The Y-scale for the history plots is adjusted automatically by default, but can also be changed to a fixed value, selected from the respective combo-boxes. Queue delay is shown in milliseconds per sample, as an inverted cumulative probability distribution function (CDF) in the plots and the average (avg) and 99^{th} percentile (P99) values per sample.

5. CONCLUSIONS AND FUTURE WORK

In our demo visitors can load up an emulated broadband Internet access with numerous interactive multimedia applications and enjoy remarkably low and predictably low queuing delay, giving unprecedented interactivity for *all* applications at the same site, even under heavy load. This demonstrates the following contributions:

1. the insight that all applications running at any one time can be latency sensitive, so solutions like Diffserv that give low delay for some at the expense of others are not sufficient;
2. that replacing classic TCP with a scalable congestion control, such as Data Centre TCP (DCTCP), addresses the root cause of this problem—'Classic' TCP itself;
3. that our DualQ AQM enables DCTCP to be released on the public Internet so that:
   - DCTCP flows are isolated from the delay induced by any Classic TCP flows, without making 'Classic' TCP delay any worse;
   - they both share out the capacity as if they were the same type of TCP, without any configuration or flow inspection, even though DCTCP would normally starve 'Classic' TCP flows (demonstrable in our demo by disabling the DualQ);
4. a GUI/tool that allows visitors to select TCP congestion controls, network configurations, AQMs and load models, then correlate the live experience of all the interactive applications with live quantifiable visualization of the impact on performance of each choice.

Future work is planned to extend the functionality of the tool to measure the performance of other applications and mechanisms and to distribute the entire toolkit as an open-source package.

6. REFERENCES

[1] iPerf - The network bandwidth measurement tool.

[2] M. Alizadeh, A. Greenberg, D. A. Maltz, J. Padhye, P. Patel, B. Prabhakar, S. Sengupta, and M. Sridharan. Data Center TCP (DCTCP). In *Proceedings of the ACM SIGCOMM 2010 Conference*, SIGCOMM '10, pages 63–74, New York, NY, USA, 2010. ACM.

[3] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, and W. Weiss. An Architecture for Differentiated Services. RFC 2475, RFC Editor, Dec. 1998.

[4] B. Briscoe, A. Brunstrom, A. Petlund, D. Hayes, D. Ros, I.-J. Tsang, S. Gjessing, G. Fairhurst, C. Griwodz, and M. Welzl. Reducing Internet Latency: A Survey of Techniques and their Merits. *Communications Surveys Tutorials, IEEE*, PP(99):1–1, 2014.

[5] Y. Choi, J. A. Silvester, and H.-c. Kim. Analyzing and Modeling Workload Characteristics in a Multiservice IP Network. *Internet Computing, IEEE*, 15(2):35–42, March 2011.

[6] K. De Schepper, B. Briscoe, O. Bondarenko, and I.-J. Tsang. DualQ Coupled AQM for Low Latency, Low Loss and Scalable Throughput. Internet Draft draft-briscoe-aqm-dualq-coupled-00, IETF, Aug. 2015. (Work in Progress).

[7] T. Hoeiland-Joergensen, P. McKenney, D. Täht, J. Gettys, and E. Dumazet. Flowqueue-codel. Internet Draft draft-ietf-aqm-fq-codel-04, work in progress, June 2014.

[8] O. Hohlfeld, E. Pujol, F. Ciucu, A. Feldmann, and P. Barford. A QoE Perspective on Sizing Network Buffers. In *Proc. Internet Measurement Conf (IMC'14)*, pages 333–346. ACM, Nov. 2014.

[9] M. Nilsson, B. Crabtree, P. Mulroy, and S. Appleby. Equitable quality video streaming for IP networks. *International Journal of Internet Protocol Technology*, 4(1):65–76, Mar. 2009.

[10] R. Pan, P. Natarajan, C. Piglione, M. S. Prabhu, V. Subramanian, F. Baker, and B. VerSteeg. PIE: A lightweight control scheme to address the bufferbloat problem. In *Proc. IEEE 14th Int'l Conf. on High Performance Switching and Routing (HPSR)*, pages 148–155, 2013.

[11] K. K. Ramakrishnan, S. Floyd, and D. Black. The Addition of Explicit Congestion Notification (ECN) to IP. RFC 3168, RFC Editor, Sept. 2001.

[12] L. Stewart, D. A. Hayes, G. Armitage, M. Welzl, and A. Petlund. Multimedia-unfriendly TCP Congestion Control and Home Gateway Queue Management. In *Proc. 2nd annual Conference on Multimedia Systems (MMSys'11)*, pages 35–44. ACM, 2011.