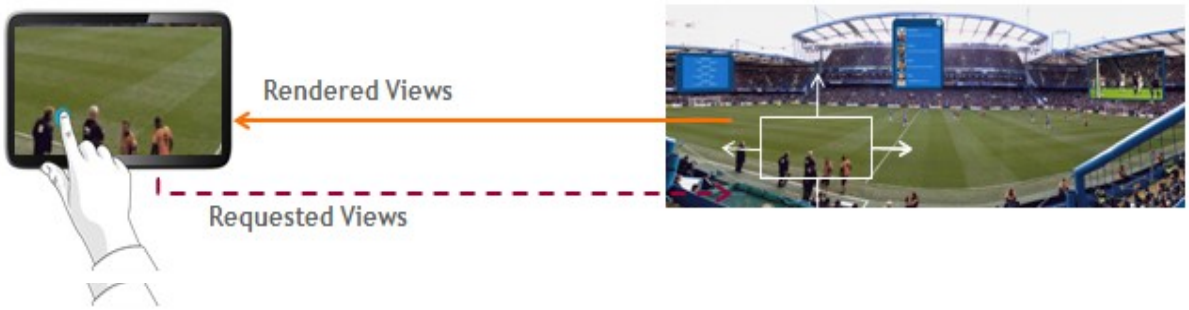
Ultra-Low Delay for All

By Bob Briscoe

At the Bits-N-Bites session at IETF-93 in Prague, something quite remarkable was demonstrated: a streamed football match that you could pan and zoom with finger gestures on a touch screen–and still get HD (high definition) at full zoom. The app in itself was pretty neat, but the responsiveness was the remarkable thing; it seemed to stick to your finger as you panned or pinched.



The video sticks to your finger as you pan and zoom with finger gestures and you still get HD at full zoom

As you'd expect, attendees had some pretty pointed questions for those responsible: Reducing Internet Transport Latency (RITE), a team of European Internet researchers whose goal it is to remove the root causes of unnecessary latency over the Internet. The initiative is funded by the European commission under the fp7-ICT programme.

Was it cached locally?

No. Each client was feeding the finger gestures to a remote proxy, which was generating the HD scene on the fly for that user, from a panoramic video of the whole stadium.

Was it just a short cable?

No. Earlier in the week, in the active queue management (AQM) Working Group (WG), the same technology had been demonstrated using remote login on Bell Labs' broadband testbed. They were streaming from a proxy in a data centre to a home network across real core, backhaul and digital subscriber line (DSL) access network equipment—overall 7ms base round trip delay—the sort of base delay you should get to your local content delivery network (CDN). For Bits-N-Bites, they were using netem to emulate the same delay.

Was this Diffserv quality of service (QoS)?

No. That was the remarkable thing. Diffserv only gives QoS to some at the expense of others. This was for all traffic, even under high load. Through a dashboard you could click to add up to 100 parallel Web flows per second, and you could start dozens of downloads to pile on even more load. Not only did the pan and zoom responsiveness stay 'finger-sticking' good, but a chart on the dashboard showed that all the other flows were seeing the same ultra-low queuing delay. It measured the queuing delay of each packet—not just the video but all the Web flows and downloads. The worst delay was so low you could hardly see the plot—just a couple of pixels.

Had they just configured very shallow buffers?

No. The dashboard showed the link was fully utilized.

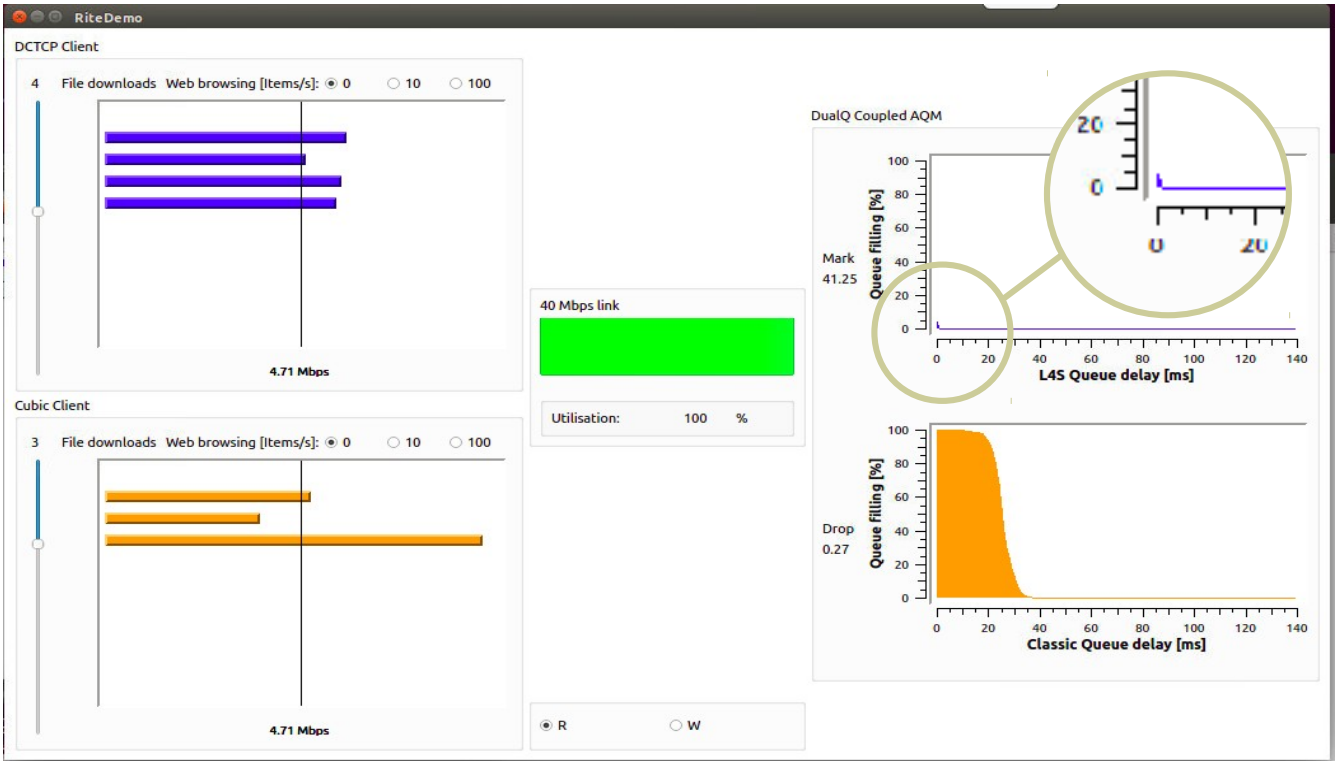
So what was the magic under the covers?

Quite simply, all they were doing was *not* using regular transmission control protocol (TCP) (no New Reno, no Cubic). Instead, they had switched the stacks at both ends to what they called a scalable TCP, with no need to change the apps. They said any scalable TCP will work, as long as it uses explicit congestion notification (ECN) as well. For scalable TCP, they were using Data Centre TCP (DCTCP) unmodified, which Microsoft deployed since Windows Server 8, and there's a Linux version too.

Then they had set the bottleneck queue to ECN-mark packets above a shallow step threshold. Access link capacity is typically the bottleneck for DSL, cable and cellular. So, for their downstream DSL case they only needed this marking at the Broadband Network Gateway (BNG, aka. BRAS or MSE). The same in the home gateway ought to sort out the upstream as well.

Incremental Deployment?

Could we have this Nirvana on the public Internet? Surely any 'classic' TCP flows from older machines would introduce queuing delay that would ruin everyone else's perfect day. Also, 'scalable' TCPs are much more aggressive than 'classic' TCPs. So whenever the two competed, you would expect 'classic' TCPs to get only a small share of the capacity.



The dashboard. The tiny blue pixels (top-right) showed queuing delay remained ultra-low as 'scalable' TCP flows were added (top-left). As orange 'classic' TCP flows were added (bottom-left) their delay profile was no worse than today (bottom-right), but it didn't affect the low delay of blue traffic. Nonetheless, all flows shared the bandwidth roughly equally (compare top and bottom left), but with no flow inspection.

This was where the demo got really interesting. Using the dashboard, you could add 'classic' flows

as well. But you couldn't get it to affect the ultra-low queuing delay of the 'scalable' packets at all. And the queuing delay of the 'classic' flows wasn't compromised either—it was no worse than it would have been if all the load had been 'classic'.

Most impressive, all the flows still shared out the capacity roughly equally, as if they were all the same type of TCP. But there was no per-flow scheduling—indeed, they weren't inspecting anything above the Internet Protocol (IP) layer.

How did they do that?

They classified 'classic' traffic into a separate queue to prevent it delaying the 'scalable' traffic. Then, they coupled dropping and ECN-marking between the two queues, marking scalable flows more aggressively to exactly counterbalance their more aggressive response to the marks. This required a square relationship, which they coded really neatly; they just compared the queuing time against one random number for marking and against two for drop. They have a nice *aide-mémoire* for this: “Think twice before dropping.”

Sunsetting TCP?

Back in 2012, when the IETF had embarked on the real-time comms in Web browsers (RTCWEB) effort, it was known that queuing delay and jitter would often degrade rtcweb. An Internet Architecture Board (IAB) workshop led to the birth of the RTP Media Congestion Avoidance Techniques (RMCAT) and the active queue management (AQM) WGs. RMCAT would avoid real-time traffic adding to the problem, and AQM would at least remove unnecessarily long queues by tackling so-called 'buffer-bloat'. But the elephant in the room was TCP. Per-flow queuing was included in the AQM charter as a way to isolate a delay-sensitive flows from TCP, but it was hedged round with caveats, given the implication that the network would have to identify transport-layer flows, and decide on their relative capacity shares, not to mention the extra cost—a thousand-odd queues for a typical residential access.

The technology shown in Prague gives us a new component, using just two queues; a sort-of semi-permeable membrane that partitions off the harmful *delay* of 'classic' TCP, without prejudging where to partition the *bandwidth*.

The demo showed that the Internet could be so much better without 'classic' TCP. It demonstrated that a superior 'scalable' class of TCP algorithms already exists. And it showed the path to get from here to there. It was the IETF at its best.

For more information, see: http://riteproject.eu/dctth