DOI:10.1145/3330338

To view the accompanying paper, visit doi.acm.org/10.1145/3330336

Technical Perspective Do You Know Why Your Web Pages Load Faster?

By Costin Raiciu

TO KEEP UP with demand and ensure users get quick access to information on the World Wide Web, Internet Service Providers have been adding capacity continuously, interconnecting more users and companies and at faster speeds. For home users, the progression has seen capacity increase from dial-up (56kbps) to fiber (1Gbps), while for mobile users cellular speeds have increased from GPRS (~100Kbps) to LTE (~100Mbps).

As with Moore's Law for computing, and despite continuous investment in capacity, we have reached a point where adding more capacity will not necessarily make the Web faster. The fundamental reason is that *propagation latency*, the time it takes information to travel from one point to another on the Internet, is lower bounded by the time it takes light to travel the same distance, and thus cannot be lowered.

The time required to download a small Web page is dominated by propagation latency between the client and the server, and not throughput. If a client from Bucharest wishes to visit a Web page hosted in Silicon Valley, the download time will be lower bounded by round-trip time, which is the latency to cross the Atlantic twice and cannot be faster than 100ms. In practice, latency is quite a bit higher than this theoretical optimum.

To reduce this latency, content distribution networks (such as Akamai) appeared around 2000 that placed servers all around the globe to move content physically closer to users. In my example, the content hosted in Silicon Valley would be replicated on CDN servers in Romania such that the client can reach the content in tens instead of hundreds of milliseconds. CDNs are now ubiquitous, but they do not solve the latency problem completely: they work really well for static content, but less so for dynamically generated one.

More importantly, the protocols sending information over the Internet have not been optimized for latency, and require many round-trip times between the client and the server to download a Web page. In fact, to download a small Web page over the prevalent transport protocol stack (HTTP2 running over TLS version 1.2 over TCP) requires at least four RTTs, severely inflating Web latency. Higher latencies lead to disgruntled users and less business, so there is a strong push to reduce Web latency.

To reduce the number of RTTs and thus Web latency, non-trivial changes to the base protocols (HTTP, TLS, and TCP) are required. While capacity enhancements or CDN deployments were implemented by a single entity (for example, ISPs), protocol changes require multiple stakeholders to agree as they first require standardization, then implementation by multiple operating systems and finally deployment on user devices. Following this approach, changes to TCP were introduced over the past six years to allow zero-RTT connection setup and TLS version 1.3 is significantly faster than 1.2. Unfortunately, such changes to existing protocols have limited impact because they must obey the layered architecture (HTTP/TLS/TCP), they need to support legacy applications, and require huge development resources and many years to get deployed.

QUIC is a novel protocol proposed by Google that reduces latency by replacing the entire HTTP/TLS/TCP stack with a single protocol that runs on top of UDP. The key benefit of running atop UDP is the protocol stack can be implemented as a user-space application, rather than in the kernel as needed when changing TCP, for instance. This implies that QUIC protocol changes can be pushed as easily as changing an application.

Google's QUIC approach is radical because it bypasses all the hurdles faced by incremental protocol changes: as Google controls both the servers and the client stack it can simply implement the protocol and deploy it both on its servers and the clients (through Chrome), as often as it wishes, without external factors delaying the process. QUIC was first deployed in 2012 and has since been continuously updated. Today, QUIC is widely used and it carries a large fraction of Google's traffic; it is also undergoing standardization to enable other companies to use it, but standardization follows deployment, not the reverse.

QUIC's organic development has left heads scratching both in the research and standardization communities. QUIC's advocates point to impressive performance numbers in its favor, mostly reported by Google. Its detractors complain about the lack of justification for the chosen protocol mechanisms, and in general the lack of understanding of the reasons *why* QUIC outperforms TCP; the argument is that without such understanding, QUIC's gains could prove elusive when the network evolves in the future.

The following paper is a bold attempt to unearth the reasons why QUIC works better than TCP. The authors provide a unique and comprehensive insight into QUIC's behavior and how it compares to HTTP2/TLS/TCP. In contrast to many other studies of QUIC's performance, the work by Kakhki et al. does not only focus on the latest version of QUIC, but examines all versions comparatively, contrasting code changes to varying performance. Furthermore, the paper fights the lack of documentation by extracting the QUIC state machine from the code itself. The work is interesting because it sets the basis for a thorough understanding of why QUIC works so well and it should be equally interesting for computer science researchers outside networking.

Costin Raiciu is an associate professor in the Computer Science Department at the University Politehnica of Bucharest, Romania.