Internet Latency: Causes, Solutions and Trade-offs

A digest of Reducing Internet Latency: A Survey of Techniques and their Merits[1]

David Hayes*, Ing-Jyh Tsang†, David Ros‡, Andreas Petlund‡, Bob Briscoe§
*University of Oslo, Norway, †Alcatel-Lucent Bell-Labs, Belgium ‡Simula Research Laboratory AS, Norway, §BT, UK

***Abstract*—This paper is a digest of [1], an extensive survey discussing the merits of over 300 techniques for reducing Internet latency. It gives a broad overview of the causes, solutions, and trade-offs involved in reducing latency in the Internet. The overview covers key sources of delays and proposed solutions: due to the structural arrangement of the network, how network end-points interact, along the end-to-end path, related to link capacities, within end-hosts, and those that address multiple sources. Trade-offs are discussed in terms of the latency reduction different techniques provide versus their deployability.**

I. Introduction

Latency is a measure of the responsiveness of an application ... the time it takes for a single critical bit to reach the destination, measured from when it was first required [1]. The exact meaning of this depends on the application. For real-time interactive applications, the critical bit is the last bit of the base chunk of information (ie a frame for interactive video). Start-up latency is important for video streaming, where the critical bit is the first bit of data. For applications that transfer blocks of information (ie email, instant messaging, downloads) the critical bit is the last bit in the message.

Latency is becoming a critical issue for application performance. It is important to understand the root causes, possible solutions, and the benefits and costs involved in deploying them. This paper digests the more substantial work [1] which addresses these questions.

The remainder of the paper starts by overviewing available techniques for reducing delays by the sources of delay as illustrated in Fig. 1. § VII looks at composite solutions that address several sources of delay. § VIII summarises the relative merit of different techniques versus the difficulty in deploying them, and § IX gives conclusions.

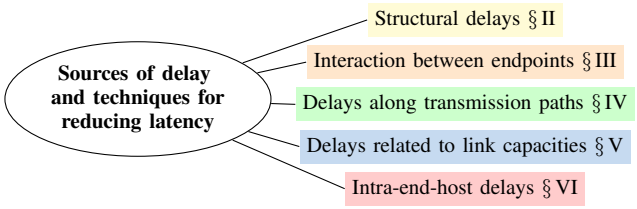


Fig. 1. Techniques for reducing latency organized by sources of delay. (See [1] for a more extensive outline)

II. Structural delays

Internet communication relies on interactions between a set of endpoint systems, such as those exemplified in Fig. 2. The latency experienced by clients is significantly impacted by the placement of the software components, such as servers, caches, databases and proxies in relation to the client endpoint. We look at this with respect to the following categories: sub-optimal routes/paths, name resolution, content placement and service architecture.

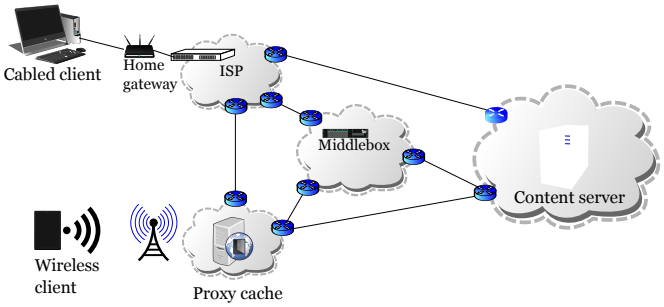


Fig. 2. Communication between Internet endpoints must traverse a wide range of different systems. The choice of routing paths with low latency depends on the technology residing in the nodes and on peering agreements between Autonomous Networks. Traffic may also be intercepted by a middlebox or served by a proxy cache. Buffering and processing in the nodes also impact the latency observed by the user [1].

a) Sub-optimal routes/paths: Choosing the right route between hosts can impact latency severely. Although methods for optimising for latency through routing changes are not widely deployed, several research results treat this subject. Detecting congested links when determining the route is possible, but not widely deployed. Another option is to exploit multiple paths between the hosts, either using interfaces connected to different networks or to use overlay hosts to find a low-latency path.

b) Name resolution: Name resolution for mapping Internet Protocol (IP) addresses to domain names is necessary for all new connections. This will create extra delay whenever connecting to an ”unknown” host name. Common practice is to store locally all resolved names to save future lookups. Caching domain names in the gateway is also commonly done to save time for everyone residing within the gateway’s domain. Determining the best size of the cache in such cases has been researched, showing that due to Zipf distribution of lookup frequency of names, arbitrarily large increases in cache size will not have significant effect. A way of hiding the latency of name resolution from the user is to use Domain Name System (DNS) prefetching. When loading a web page, the links embedded in the page can be resolved to save delay when the user clicks the link. Using DNS to redirect the client to Content Delivery Networks (CDNs) close to their physical connection is also commonly applied to reduce the user’s latency.

c) Content placement: Choosing the right strategy for where to place the content is of critical importance to the user's latency experience. Placing copies of the data (caches) in between the server and the client is a common way of saving resources and reducing latency [2]. To offload servers that experience periods of heavy traffic, *load balancing* is applied, redirecting traffic to different copies of the content in a round-robin manner. It is also possible (at some cost) to actively push content to the client in anticipation of the data being needed. CDNs apply many of the known caching and replication techniques to provide copies of their content close to users all over the world. Web browsers cache content locally to be used when similar pages are loaded. Using push technology, modern webpages often fill such local caches with data for later use. In real-time applications, like online games, it is common to hide the actual latency connected to the placement of the content by using prediction based on the physical models in the system to give the appearance of smooth performance.

d) Service architecture: Alternatives to the Client-Server model can be used to reduce latency. Peer to Peer (P2P) architectures are generally bad for latency, but *structured p2p* can achieve good results while keeping the benefits of the P2P distribution. Virtualisation can be used to run a chain of services in a single memory space to avoid serialisation delays between them. Similarly, caches can be dynamically spawned in strategic locations using Cloud services.

III. Interaction between endpoints

Transport protocols control the transfer of data between endpoints over a network. This control may involve multiple control interactions before data communication starts. Higher layers may also require setup interaction (e.g. for security). During a session further control interactions may be required, for instance to recover lost packets to provide reliable transfer. In small data transactions these interactions may dominate the total latency, so reducing them can dramatically reduce the latency of the session.

a) Transport initialisation: At transport layer the startup latency is a result of protocol handshake (e.g. at the start of a TCP, SCTP or DCCP connection). Each sequentially completed handshake incurs a minimum of one Round Trip Time (RTT) of delay, whereas larger delays are imposed when the packet is lost (due to congestion, or to unsupported features in the network and/or a middlebox), requiring retransmission or a timeout to restart the exchange. Parameters such as communication or even latency reducing options, may need to be negotiated at the start of the session, for example: selection of alternate transport protocols, mechanisms such as Explicit Congestion Notification (ECN), use of relays to overcome the limitations of middleboxes, etc. Several techniques can be used to mitigate the impact of startup latency, such as by redesign and reducing the number of sequential protocol exchanges, by multiplexing data over an existing session or by persistent use of a session (instead of for each transfer opening and closing a session). For example, TCP Fast Open (TFO [3]) circumvents

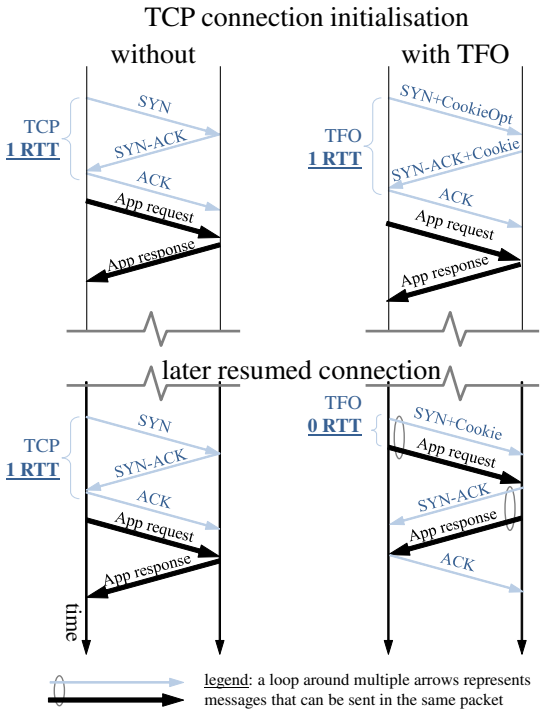


Fig. 3. TCP Fast Open (TFO) saves a round trip when resuming a connection.

the three-way handshake, by using a TFO cookie to inform previous connected servers (See Fig. 3).

b) Secure session initialisation: In the present times security mechanisms are an essential requirement for Internet communication. Security protocols without modifications can impose significant latency to setup an application flow, as in general these protocols were not designed with any focus on latency. Thus there are several proposals to update security protocol interactions to reduce the number of RTTs, with the potential to provide significant latency gains for short sessions. For example, faster Transport Layer Security (TLS) negotiation (TLS False Start combined with TFO, TLS Snap Start) and bootstrapping security from the DNS (Minimal latency networking, Snap Start client).

c) Packet loss recovery delays: When data traverses a link with appreciable packet loss/corruption due to link errors and/or a heavily loaded network bottleneck suffering congestion, the transport-layer error/loss control can induce a substantial amount of delay. For example, Flach *et al.* [4] show that Web flows experiencing loss can increase fivefold in average completion time, making loss recovery delays a dominating factor for Web latency.

There are three types of loss recovery techniques: retransmission, redundancy and loss concealment. Retransmission technique uses a control loop to identify loss/corruption and retransmit the missing packets. If the mechanism requires packet ordering, loss implies Head-Of-Line (HOL) blocking at the receiver to reorder the retransmitted data. In case retransmission fails a further delay is incurred as a retransmission timer needs to be triggered. Redundancy methods can be as simple as packet duplication at the sender (e.g. sending multiple copies of a single control packet) or by coding a

combination of packets using Forward Error Correction (FEC). Fundamentally, FEC imposes a trade-off between decreased capacity and enhanced reliability, and additional processing at the sender and receiver. A combination of the following loss recovery methods may be needed to achieve a tradeoff between processing, reliability and delay: applications can be tolerant to loss; techniques to reduce packet loss detection times can be used (e.g. updating the TCP and SCTP retransmission timeout (RTO) timer); a combination of redundancy and retransmission (for example 'hybrid ARQ'); finally, ECN can be used to propagate an unambiguous congestion signal, marking packets instead of dropping, thus avoiding loss and need for retransmission.

d) Message aggregation delays: TCP uses two complementary aggregation methods to reduce the number of IP packets to be transported, i.e the *Nagle algorithm* [5] and the *delayed ACK algorithm* [6]. The *Nagle algorithm* delays the sending of a small segment while a previously-sent segment has not been acknowledged. It coalesces small blocks of application data into a larger block, which then can be sent in a single packet, instead of sending as many packets as data blocks. The *delayed ACK algorithm* reduces the amount of pure ACK messages (i.e. containing no data), either by piggybacking an ACK signal onto a data-carrying segment, or by sending a pure ACK only for every two full-sized data segments. In case the ACKs are lost or no data is flowing in the reverse direction, a timer as high as 200 ms, ensures that an ACK is always eventually sent. These mechanisms trade latency for bandwidth efficiency, but used in combination they may give rise to severe additional delay [7]. To avoid this latency penalty the Nagle algorithm can be turned off, or a variant [8] can be used, where transmission of a small segment is delayed *only* if the previously-sent, unacknowledged segment is *also* small.

IV. Delays along transmission paths

The flight latency of a packet is the time it takes from transmission endpoint to reception endpoint. This includes delays due to signal propagation speed, gaining access to the transmission media, serialisation of the sending data, switching and forwarding, error recovery and queueing.

a) Signal propagation delays: Electromagnetic waves travel at the speed of light, but this is slower in more dense media such as optical fibres than it is in less dense media such as air. The delay encountered depends on this speed of propagation and the length of media. Propagation delays can be reduced by: (i) reducing the path length by making it straighter, and (ii) increasing the speed by using a medium with faster propagation.

b) Medium acquisition delays: Where communicating entities share a common medium, access to the media has to be managed to avoid mutual interference. On demand contention based mechanisms, such as in Wifi, can cause significant and unpredictable delays since access to the media is not guaranteed. They can also aid latency-sensitive traffic by providing prioritised access to the medium.

c) Serialization, switching and forwarding delays: It takes time to get data out of a network card and onto the medium and into the receiver, a process often repeated at every junction along the network path. Junctions often also add switching and forwarding delays. Increasing line rates reduces the serialization delays, while reducing the number of hops reduces the number of times it is added to the flight latency. Cut-through and wormhole switching can reduce switching and forwarding delays.

d) Link error recovery delays: Recovering errors at the links where they occur can avoid longer end-to-end recovery delays. Managing link recovery delays is a compromise between reducing utilisation, adding a constant coding delay to avoid errors, adding a variable delay to retransmit locally even if the application doesn't need it, and avoiding variable end-to-end retransmission delays to repair losses. Local loss repair is especially useful on error-prone links, such as microwave and cellular, where a good compromise can improve the average latency and make it more consistent.

e) Queueing delay: Queueing delays are often the largest contributors to the flight latency. The issue has been compounded by an over-abundance of network buffering, an issue that has been termed *bufferbloat* [9]. The following paragraphs outline efforts to reduce these delays.

Flow and circuit scheduling can avoid queueing delays in the network by pre-scheduling a circuit or flow path for packets to traverse from end-to-end through the network. This adds an initial setup delay, however, once the path has been established the queueing delay is zero, so the flight latency reduces to the sum of the propagation delays through the network.

Reducing buffers sizes at network switches and in the MAC layer prevents excessive queue build up. IP layer switches have been designed to provide roughly one Bandwidth Delay Product (BDP) of buffering. Buffer sizing is often a compromise between latency and link utilisation, however, switches for highly aggregated traffic require only a fraction of a BDP sized buffer to achieve high utilisation. At the edges of the network where there is a lower degree of flow multiplexing transport layer burstiness causes queueing. Packet pacing can reduce the edge network buffer requirements.

Packet scheduling has been an area of active research for decades. It does not reduce the average delay of all packets, but instead can give selected packets precedence over others, thus reducing the latency of those selected packets. Efforts can be categorised into (i) class based mechanisms which give priority to particular classes of traffic, (ii) flow based mechanism which segregate flows such that competing flows cannot inflict latency on other flows, (iii) latency targeting mechanisms, such as schemes that schedule packets according to their delay deadlines, and (iv) hierarchical mechanism which combine the previously listed mechanisms in a hierarchical manner to achieve particular Quality of Service (QoS) objectives.

Queue management and in particular Active Queue Management (AQM) can be used to help keep average queueing delays low by preemptively marking or dropping packets to prevent greedy transport protocols like TCP from keeping the

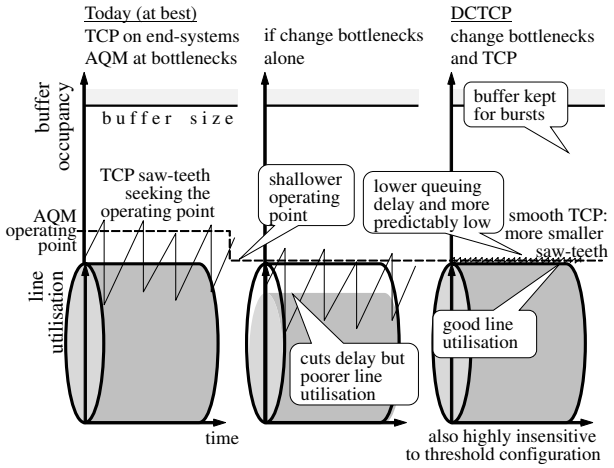


Fig. 4. How Data Centre TCP (DCTCP) reduces delay without losing utilisation [1].

buffer full. The focus of recent work, such as the Proportional Integral Controller Enhanced (PIE) AQM [10], and the Controlled Delay (CoDel) AQM [11] (and fq-CoDel), has been to reduce parameter sensitivity which was a problem with earlier proposals. These algorithms buffer transitory bursts of packets, but try to keep the average queue length to a target delay.

f) Transport-based queue control: Data Centre TCP (DCTCP) [12] works together with AQMs along the path using ECN signals to maintain high throughput with low flight latency (see Fig. 4). Apart from using signals from an AQM, packet delay can be used as a congestion signal. Delay-based congestion control mechanisms are able to maintain low queueing delays along the path, but only when not competing with protocols that use loss as a congestion signal.

V. Delays related to link capacities

Link capacities along a path can have an impact on latency in several ways.

a) Insufficient capacity: The available capacity may be scarce. This could simply be due to a link with an inherently low capacity, which will lead to high packet serialisation times. There are a number of different techniques that can reduce latency due to insufficient capacity.

Increasing capacity is most obvious one, increasing link speeds, yields smaller serialisation times and helps with alleviating persistent congestion, but it may not always be a feasible solution. Aggregate use of several links or end-to-end paths results in an effective higher capacity; this can be done at different layers of the protocol stack (e.g., bundling of parallel links at the network layer, or use of multipath transport protocols [13]).

Reducing redundant information sent across the network makes it possible to utilise scarce capacity in a more efficient way. The use of multicast or protocol-header compression [14] are approaches that fall into this category.

b) Underutilised capacity: Capacity may be abundant yet be poorly utilised. Inefficiencies are typically related to how end-to-end protocols share a link's capacity. Congestion control algorithms such as TCP's have to sense available capacity as quickly as possible (to avoid inefficiency). TCP congestion control mechanisms have well-known limitations in this regard. Probing for capacity when a flow starts (either at the beginning of a connection or after a long idle period) can be improved by e.g. allowing larger initial bursts of packets, since this can noticeably reduce the transfer time of small to medium volumes of data.

c) Collateral damage: The goal of sensing and using capacity as quickly as possible is often at odds with trying to share it in a fair manner. Allowing larger initial bursts may induce severe loss—either on the flow sending the burst, or on other flows, or both—and, therefore, higher delays. Diverse techniques have been proposed to sense available capacity as rapidly as possibly while avoiding increased losses and delays.

VI. Intra-end-host delays

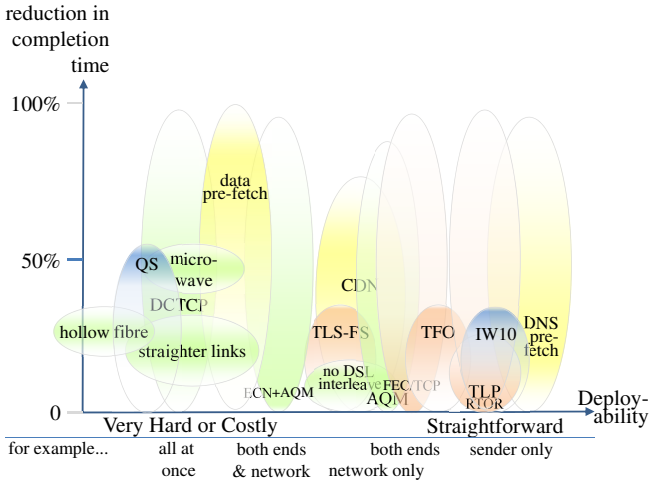
Delay caused by bloated buffers within end-hosts needs to be solved with the same techniques as for buffers in the network; correct buffer sizing and active queue management (see §.IV). HOL blocking adds delay when packets are lost or reordered during transmission. HOL blocking is particularly harmful when a loss in one ordered stream holds back another that has not experienced loss, merely because it is multiplexed into a common sequence with the blocked stream. In addition, ECN and FEC can remove or hide the losses that are the underlying cause of the blocking, respectively.

How the host CPU, memory and I/O devices are designed to exchange and process data has a fundamental impact on the latency experienced by applications. Techniques such as parallelization, pipelining and zero-copy can increase performance and reduce latency. Bit-level, instruction-level, data or task parallelism can reduce latency, however this is not always straightforward [15]. Addressing latency in the host is not just about having faster hardware, it requires novel architectures looking at latency as a fundamental system parameter. In this respect, chip manufacturers already optimize and integrate frame control, scheduler, memory and data path logic in the silicon fabric [16].

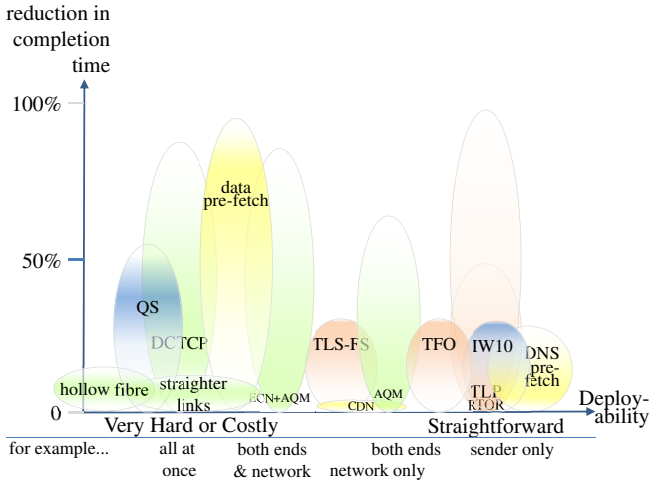
VII. Composite solutions

Some mechanisms "bundle" different techniques to tackle several delay sources. Wide Area Network (WAN) accelerators and Performance-Enhancing Proxies (PEPs) are two widely-deployed examples. The former may employ techniques that are either generic (e.g., removal of duplicate data transfers), or focused on the transport layer (e.g., Secure Socket Layer (SSL) acceleration or quickly filling available capacity) or on the application layer (e.g., reducing round trips of application-layer protocols that behave inefficiently on a WAN). PEPs are typically found in wireless links, and may leverage cross-layer information besides implementing mechanisms similar to those found in WAN accelerators.

SPDY [14] is an application-layer protocol that combines techniques intended to speed up the load time of web pages. It employs mechanisms such as protocol-header compression, stream multiplexing and stream prioritisation.



(a) Small session (∼20 kB) flows over WAN



(b) Small session (∼20 kB) flows over LAN

Fig. 5. Bubble plots of rough latency gains against ease of deployment for a selection of techniques. The heights of the captions of each bubble represent typical values, and the vertical extent of the bubble represents variance [1]. QS (Quick-Start), New-CWV (New Congestion Window Validation), IW10 (Initial Window 10), TFO (TCP Fast Open), TLS-FS (Transport Layre Security/False-start), DNS (Domain Name Service),RTOR (Retransmission TimeOut Restart), TLP (Tail Loss Probe), CDN (Content Distribution Network)

VIII. Gain vs. deployability

This section summarizes the benefits of the various techniques by comparing their potential gains versus the difficulty involved in deploying them. The two independent variables that affect the impact of all approaches are: the relative session size and relative distance between end-points. Fig. 5 visualises the merits of a subset of techniques for small session sizes (∼ 20 kB) illustrating two representative distances: small (Local Area Network (LAN)) and large WAN RTTs.

The bubbles in Fig. 5 are positioned according to the approximate gain vs pain of the different techniques. The vertical position of the caption representing the typical expected latency reduction of the technique. The further to the right a technique, the less complex it is to deploy, so in general high bubbles on the right hand side of the graph are better. That stated, some research is improving the deployability of techniques shifting them towards the right side of the graph.

IX. Conclusions

Reducing network latency is a key challenge for the future Internet. It is a *multifaceted undertaking and will require combining the various different competencies in the scientific and industrial communities in a collective effort.* [1]. This paper provides a succinct overview of the issues involved in addressing this problem. For a more thorough treatment with extensive references we refer readers to [1].

References

[1] 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," *IEEE Communication Surveys and Tutorials*, Nov. 2014. [Online]. Available: http://dx.doi.org/10.1109/COMST.2014.2375213.

[2] S. Podlipnig and L Böszörmenyi, "A survey of web cache replacement strategies," *ACM Computing Surveys (CSUR)*, vol. 35, no. 4, pp. 374–398, 2003.

[3] Y. Cheng, J. Chu, S. Radhakrishnan, and A. Jain, *TCP Fast Open*, RFC 7413 (Experimental), Internet Engineering Task Force, Dec. 2014.

[4] T. Flach, N. Dukkipati, A. Terzis, B. Raghavan, N. Cardwell, Y. Cheng, A. Jain, S. Hao, E. Katz-Bassett, and R. Govindan, "Reducing web latency: the virtue of gentle aggression," in *Proc. of ACM SIGCOMM*, (Hong Kong, China), 2013, pp. 159–170.

[5] J. Nagle, *Congestion Control in IP/TCP Internetworks*, RFC 896, Internet Engineering Task Force, Jan. 1984.

[6] R. Braden, *Requirements for Internet Hosts - Communication Layers*, RFC 1122 (INTERNET STANDARD), Updated by RFCs 1349, 4379, 5884, 6093, 6298, 6633, 6864, Internet Engineering Task Force, Oct. 1989.

[7] S. Cheshire, *TCP performance problems caused by interaction between Nagle's algorithm and delayed ACK*, Self-published online: http://www.stuartcheshire.org/papers/NagleDelayedAck/, May 2005.

[8] G. Minshall, "A proposed modification to Nagle's algorithm," Internet Draft draft-minshall-nagle, Jun. 1999, Work in progress. [Online]. Available: http://tools.ietf.org/html/draft-minshall-nagle.

[9] J. Gettys, "Bufferbloat: dark buffers in the Internet," *IEEE Internet Comput.*, vol. 15, no. 3, pp. 96–96, 2011.

[10] R. Pan, P. Natarajan, C. Piglione, M. Prabhu, V. Subramanian, F. Baker, and B. VerSteeg, "PIE: a lightweight control scheme to address the bufferbloat problem," in *Proc. of the IEEE International Conference on High Performance Switching and Routing (HPSR)*, Jul. 2013.

[11] K. Nichols and V. Jacobson, "Controlling queue delay," *ACM Queue*, vol. 10, no. 5, May 2012.

[12] M. Alizadeh, A. Greenberg, D. Maltz, J. Padhye, P. Patel, B. Prabhakar, S. Sengupta, and M. Sridharan, "Data Center TCP (DCTCP)," in *Proc. of ACM SIGCOMM*, (New Delhi, India), Sep. 2010.

[13] C. Raiciu, M. Handley, and D. Wischik, *Coupled Congestion Control for Multipath Transport Protocols*, RFC 6356 (Experimental), Internet Engineering Task Force, Oct. 2011.

[14] M. Belshe and R. Peon, "SPDY protocol," Internet Draft draft-mbelshe-httpbis-spdy, Feb. 2012, Work in progress.

[15] D. A. Patterson and J. L. Hennessy, *Computer Organization and Design, Fourth Edition, Fourth Edition: The Hardware/Software Interface (The Morgan Kaufmann Series in Computer Architecture and Design)*, 4th. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2008.

[16] U. Cummings and M. Zeile, "Focalpoint II, a low-latency, high bandwidth switch/router chip," in *Proc. of the Symposium on High Performance Chips (Hot Chips)*, (Stanford, Califonia), Aug. 2007. [Online]. Available: http://www.hotchips.org/wp-content/uploads/hc_archives/hc19/3_Tues/HC19.07/HC19.07.03.pdf.