FEC Killed The Cut-Through Switch

Omer S. Sella
University of Cambridge
Omer.Sella@cl.cam.ac.uk

Andrew W. Moore
University of Cambridge
Andrew.Moore@cl.cam.ac.uk

Noa Zilberman
University of Cambridge
Noa.Zilberman@cl.cam.ac.uk

ABSTRACT

Latency penalty in Ethernet links beyond 10Gb/s is due to forward error correction (FEC) blocks. In the worst case a single-hop penalty approaches the latency of an entire cut-through switch. Latency jitter is also introduced, making latency prediction harder, with large peak to peak variance. These factors stretch the tail of latency distribution in Rack-scale systems and Data Centers, which in turn degrades performance of distributed applications. We analyse the underlying mechanisms, calculate lower bounds and propose a different approach that would reduce the penalty, allow control over latency and feedback for application level optimisation.

1 INTRODUCTION

Latency has been long known to have an adverse effect on systems, from the annoyance users feel when a website is slow to load, to application performance degradation [27]. Patterson *et al.* [20] observed over a decade ago that bandwidth improvements are made at the expense of latency, and in particular that the rate of network latency improvement stagnates next to the rate of bandwidth improvement.

Over the last decade, network bandwidth has improved from 10Gb/s to 400Gb/s per port [5]. Switch traversal latency has also improved, going down from 10-30μs [24] to 300ns [16]. The introduction of new link speeds is, unexpectedly, threatening the continued decline in end-to-end latency. Forward Error Correction (FEC), used to reduce the bit error rate on a link, has lead to an increase in latency that will affect all network devices.

As figure 1 shows, at 25Gb/s the additional latency contributed by FEC is at the order of a packet traversal through a commodity cut-through switch [25], and twice the latency through a state-of-the-art switch [9]. At 100Gb/s, FEC latency is at the order of a read transaction from a DRAM. Beyond 100Gb/s, a decoding time that is not link-speed dependent becomes the dominant latency contributor (as shown in Figure 2), and the FEC block has to be further buffered. These numbers are no longer negligible, especially with the increasing popularity of scale out systems and in in-network applications, requiring remote access.

To quote Cheshire [6]: “Once you have bad latency you’re stuck with it”. Therefore it is important to understand why we are stuck with FEC-induced latency, and what is the scale of the latency penalty.

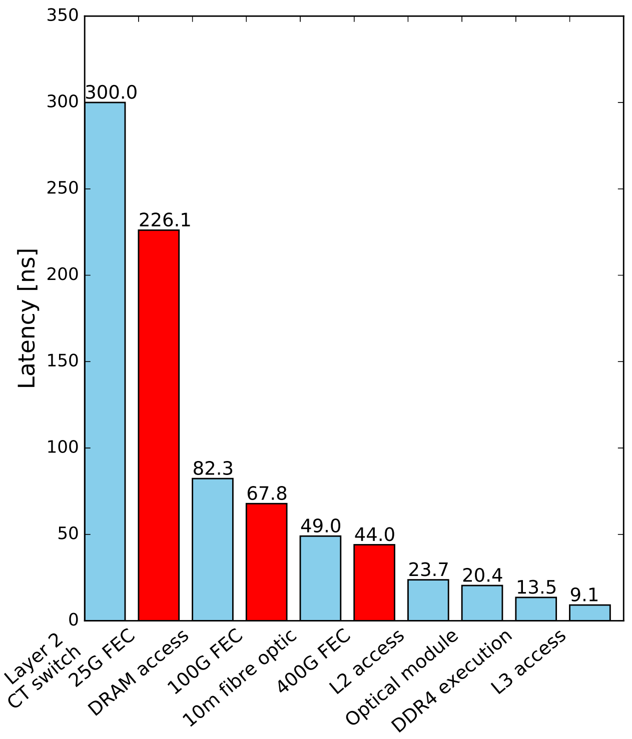

Figure 1: Comparing the scale of latency in components of networked-systems. FEC induced latency is marked in red.

In this paper we examine how the FEC chosen by recent IEEE Ethernet standards [12–14] adds latency and the jitter contributed by the FEC. We compare the effect on cut-through and store-and-forward switches and calculate the jitter envelope. Finally, we propose a different design that has lower latency and utilizes FEC to monitor *link health* and provide *latency prediction*. This proposal fits latency-sensitive environments, such as intra-data center connections and Rack-scale systems.

The rest of this paper is organized as follows: In section 2 we explain why FEC is used and its inherent latency. Section 3 explains how mapping of Ethernet frames to FEC blocks leads to high latency jitter. A roadmap to latency-sensitive design paradigm is presented in 4, whereas 5 discusses related work. We conclude our analysis in section 6.

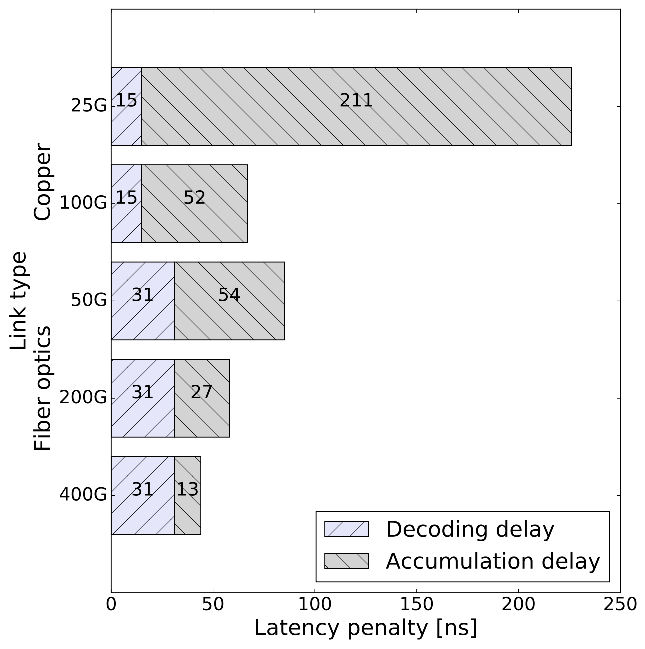

Figure 2: Accumulation time and decoding time for different types of links. A device clock of 1GHz is assumed for the decoding delay.

2 MOTIVATION

The bare minimum requirement of a networked system is that frames are going through. This is quantified by the Frame Loss Ratio (FLR). Simply put, a physical link is required to pass Ethernet frames from one port to another without loosing too many of them. This is the role of the physical layer, which cuts the frames into bits, and moves them across media, e.g., copper, fiber optics. Many techniques exist in order to move these bits faster, but all have the effect of higher Bit Error Ratio (BER) as data rate increases, which in turn increases the FLR. In order to avoid high FLR, an FEC was added in recent interconnect standards. The goal of the FEC is to achieve FLR lower than a given target, e.g. 6.2×10^{-10} [12].

FEC codes used by the recent interconnect standards [12] are Reed-Solomon (RS) block codes [23]. As a new block of data arrives at the receiver, it is first checked for errors. This requires accumulating all bits of the block and holding them in a buffer as illustrated in figure 3. As a consequence, the first bit of the block is experiencing the maximal delay, as it has to wait for the rest of the $N-1$ bits of the FEC block to arrive. We refer to this delay as the *accumulation delay* (T_{acc}). The accumulation delay depends on the block size and the link bandwidth, but not on the clock frequency of the device (switch or network interface card (NIC)).

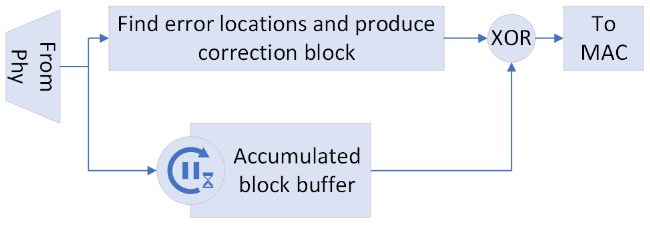

Figure 3: Generic RS decoder architecture.

The second type of delay caused by this FEC is the *decoding delay* (T_{dec}), which is the time it takes the device to find the location of errors within a block and correct them. The decoding delay depends on the chosen code, decoder and the device clock, but not on the link-speed. We assume a reference decoder, efficient at least as reported in [26]. The literature depicts decoders that take more clock cycles, for example [4, 19]. We illustrate in figure 2 the accumulation and decoding delay for different types of copper and fiber optic links.

3 THE FEC EFFECT DECODED

Figure 2 shows two trends. The first is that the accumulation delay drops with link speed. This is because when using the same number of bits in an FEC block, the faster they are transmitted the faster they are accumulated. The second trend is that the decoding time remains constant per FEC type. Since there are only two FEC types demonstrated in the figure, the decoding delay is shown to be either $15ns$ or $31ns$.

Figure 4 shows three representative examples of how a frame could be mapped into an FEC block. A frame consists of a destination field, a source field, the payload and the Frame Checksum (FCS). An FEC block consists of data and redundancy bits. A frame can thus be mapped to the data bits of an FEC block only, as the redundancy bits are generated by an encoder. As the offset between a frame's header and the beginning of an FEC block increases, the latency due to accumulation reduces. When the frame overflows to another FEC block, additional latency is incurred. This is depicted in figure 5, for a frame of 64Bytes, which is less then a tenth of an FEC block of size of 5140 data, on a 25Gb/s link.

3.1 FEC's effect on switches

The effect of FEC on latency differs between store-and-forward (SF) and cut-through (CT) switches. Store and forward switches wait for the entire frame to arrive, and for the FCS field to be checked, before processing the frame. Cut-through switches will start processing the frame as soon as the header has arrived [8]. Consider three cases, illustrated in figure 4:

- A frame is completely contained within an FEC block.

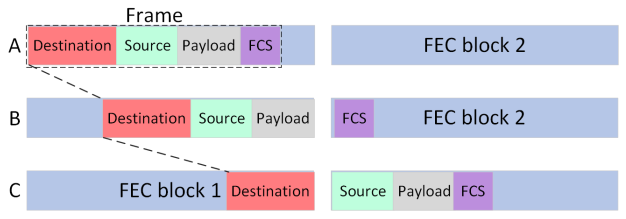

Figure 4: The three cases of a frame offset within an FEC block: (A) the entire frame is contained within the FEC block, (B) part of the data and the FCS of a frame is mapped to a second FEC block, and (C) part of the header is mapped to a second FEC block.

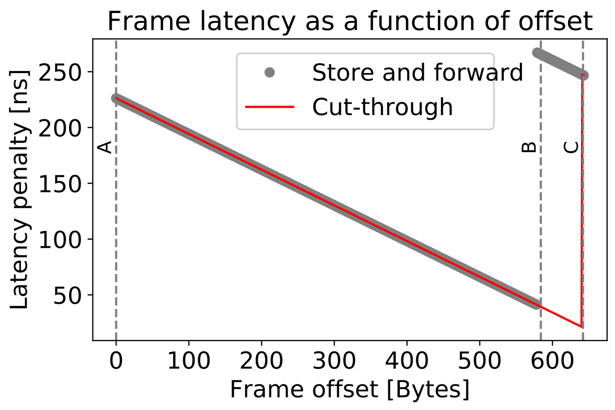

Figure 5: Latency incurred on a 64B frame as a function of header offset within an FEC block. Three important domains are marked A,B,C corresponding to figure 4

- A frame header of is contained within one FEC block, but part of the data and FCS is in the next FEC block.
- Part of the header (e.g., Ethernet MAC destination address) is contained in the first FEC block, while the rest of the header and the payload are in a second FEC block.

In the first and last cases, store and forward and cut-through switches will experience the same latency:

$$T^{A} = T_{acc} - T_{offset} + T_{dec}$$

and

$$T^{C} = (T_{acc} - T_{offset}) + T_{acc} + 2 \times T_{dec}$$

correspondingly. The middle case is different: a cut through switch will only need to wait for the header, and therefore will experience a latency of

$$T^{B}_{Cut-through} = T_{acc} - T_{offset} + T_{dec}$$

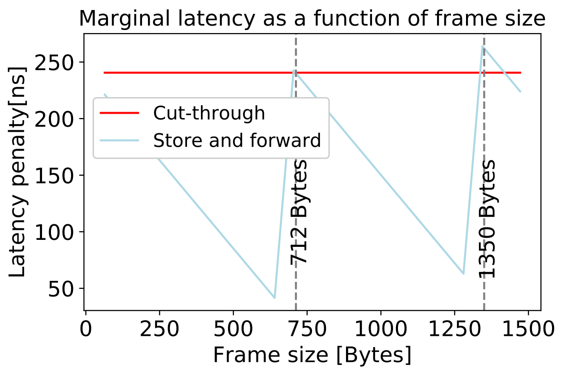

Figure 6: Marginal latency added to store and forward and Cut-through switch over a single hop, using 25Gb/s with FEC.

while the store and forward switch will wait for *both* FEC blocks to arrive, i.e.:

$$T^{B}_{Store\ and\ forward} = (T_{acc} - T_{offset}) + t_{acc} + 2 \times T_{dec}$$

. In the common worst case for 25Gb/s up to

$$246ns = (T_{acc} - T_{offset}) + T_{acc} + 2 \times T_{dec}$$

are added.

This calculation, however, does not take into account the accumulation that was bound to happen anyway for a store and forward switch. While a cut-through switch only needs 16 bits of header, a store and forward switch would have had to accumulate the entire frame anyway. Once reducing the frame accumulation time, we obtain a more complete picture of FEC's effect on the two types of switches, especially on a cut through switch. This is depicted in figure 6 as the marginal latency. Clearly the impact of the marginal latency on a cut-through switch is devastating: the cut through switch not only becomes akin to a store and forward switch, processing packets of (close to) FEC block size, but also suffers from a latency penalty in the order of traversing an entire cut-through switch.

3.2 Latency jitter

FEC does not only add latency: the variation of frame offset within an FEC block is accumulated over each hop and results in jitter. This is best demonstrated by examining the latency added by FEC when traversing a Fat-Tree [1] topology as in figure 8. Figure 7 demonstrates the effect of FEC alone, for a given frame size, traversing five hops through the network. The latency envelope is contained by the difference between the highest latency line and the lowest latency line. As the figure shows, traversing multiple hops through a network introduces significant jitter, as in every hop the header's offset within an FEC block varies.

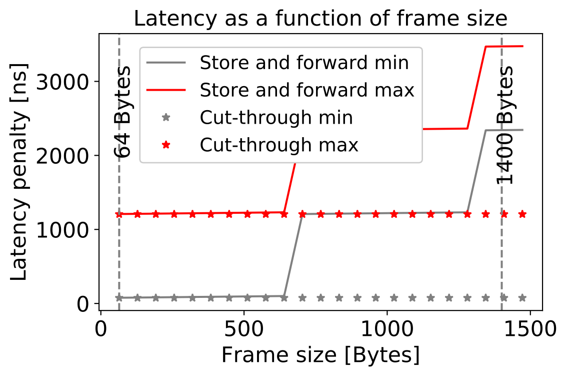

Figure 7: The latency envelope incurred by FEC for traversing a fat-tree topology.

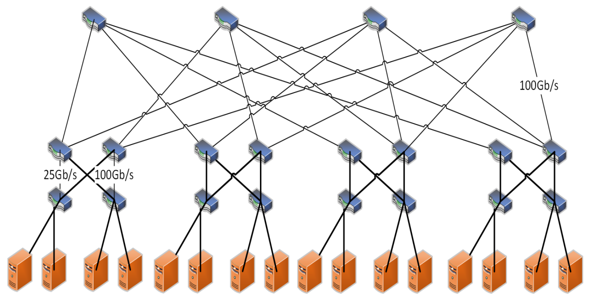

Figure 8: A heterogeneous system where several link types may cause latency jitter.

3.2.1 Jitter on multi-lane links using FEC. The case of interconnects made of multiple lanes, say four (e.g., 100Gb/s), is different: by stripping the FEC blocks over four physical lanes on the transmit side and marshalling them on the receive side, we get the same appearance as if the accumulation time was reduced by a factor of 1/4. However, there is a hidden pitfall to note here: marshalling requires alignment and de-skew, as one physical lane may be longer than another. The contributors to a lane's length are many: from the trace on the circuit board and a fiber's length to the delay within an optical transceiver. The standard allows for up to 180ns latency due to this reason, but in practice this value is typically considerably lower. While figure 2 shows latency components as induced by FEC, it does not account for any de-skew latency, which is deployment specific and not induced by the FEC.

3.2.2 Link Heterogeneous Systems. Another potential contributor to latency jitter is the use of different links speeds over paths. For example, as data centers gradually grow over time and add new equipment, higher link speeds may be used in newly deployed switches. Figure 8 demonstrates a case where all links are 100Gb/s, except for the dashed link (left most aggregation switch) which is 25Gb/s. Using numbers from 2, a frame traversing that link would experience 159ns more latency than any other path. If multi path routing is allowed, changing a single link would inject a jitter of 159ns.

4 POSSIBLE SOLUTIONS

In this section we describe potential solutions that naturally emerge when looking at an end to end system, and in particular - a controlled one.

4.1 How did we get here?

The best place to look for solutions is by examining the (multiple) FEC codes that were not chosen by the standards [7]. In the context of the FEC, latency was perceived as secondary to bandwidth. Stronger FEC with lower FLR have been traded off for higher bandwidth. The distinction that this is a *no-return* decision was already made in [6], twenty years ago. Once the code parameters were chosen for 100Gb/s, they were propagated to later standards such as for 50Gb/s and 400Gb/s.

4.2 What's next?

IEEE standards focus on existing types of networks. For new technologies, and emerging solutions, different considerations may apply. As an example, for rack-scale systems that require shorter links and substantially lower latency we need to re-think the latency-bandwidth trade off. In particular, a way is needed to control latency, not to mention predict it. Any solution that presumes to aid in lowering latency and providing an accurate prediction of latency should consider:

- Codes with sufficiently short accumulation time.
- A decoder that has an adjustable decoding time.
- Tracking of error types and characteristics rather than just BER.
- A mechanism for feeding back metrics to upper layers.

In addition, we assert that for reliability purposes, there should be an option to trade almost all bandwidth for coding gain. This is to ensure data transmission until the problem is diagnosed and solved. This case applies especially for temperature fluctuations and time dependant degradation.

4.3 Other Reed-Solomon codes

The Reed-Solomon codes picked by the standards were not the only potential coders. A different candidate that would reduce accumulation time would be an RS(N=255,K=241,m=8) code that requires accumulating just 2,040 bits. This code requires slightly more bandwidth as its rate is 2410/2550 =

«««< HEAD

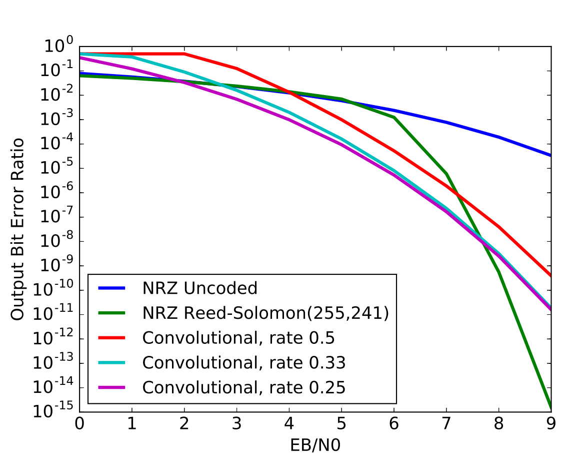

Figure 9: Comparison of codes' performance using NRZ signalling. Output BER as a function of signal to noise ratio.

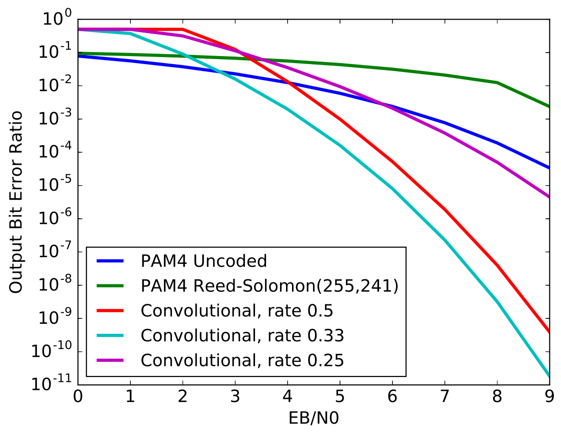

Figure 10: Comparison of codes' performance using PAM-4 signalling. Output BER as a function of signal to noise ratio.

0.945 vs. 5140/5280 = 0.973 used for 25Gb/s links. The meaning is either reducing the data bandwidth by 2.8%, or increasing the signalling rate, sometimes referred to as "overclocking". Trading bandwidth for lower latency is not a new practice, also adopted by high-frequency trading and proposed in, e.g., [2]. This solution also has the benefit of reducing the buffer size required as in figure 3.

4.4 Non Block Codes, and programmable decoding time

Reed-Solomon codes are not the only family of codes that can be used. A family of convolutional codes with various rates could be used to trade bandwidth for a stronger code, while sharing the same Viterbi decoder [10]. This coupling allows "programming" the latency in advance by changing the code used while maintaining the same hardware. This is because the latency incurred by the Viterbi algorithm could be bounded by the decoding window, which is effectively the number of bits the decoder needs to accumulate bits before they are decoded. In order to avoid a long decoding window, a stronger code can be chosen, which trades bandwidth for coding gain, rather than latency. Figures 9 and 10 compare such codes with an RS(N=255,K=241,m=8) code. The figures show the performance of each code at different regions of signal to noise ratio.

4.5 Putting it all together

Figure 11 proposes an architecture for a controlled system that enables optimizing latency and bandwidth given link's conditions. The architecture includes the receiver's ability to request an FEC change. A feedback to the control plane would include the effective bandwidth on the link, FLR, latency statistics, and would be used for load balancing and routing. In this sense we can assure not only programming the characteristics of a link, but also maintaining real time statistics on it with the option to act upon it in the application layer.

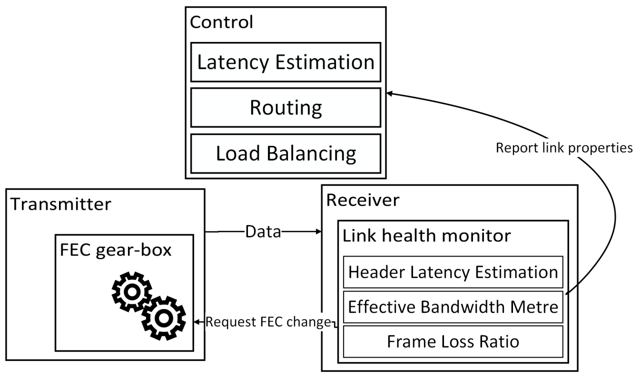

Figure 11: A fully aware system with feedback.

5 RELATED WORK

The fight against latency is held across the board. Be it in hardware, code design, scheduling, time-slotting, protocol handshaking and overhead. It is being characterized, controlled where possible, reduced when achievable. In this section we touch on just a few examples. Optimization of latency via packet time slot allocation and path assignment is demonstrated in [21]. Analysis of events and overheads in the micro-second order are portrayed in [3], which also explains the difficulty in porting solutions from High Performance Computing to data centers and even proposes re-examining layering and abstraction. Latency measurement such as [22, 27] are important for characterisation and understanding of latency components on applications' performance. At the same time work similar to [2] provides insights into the trade-off between latency and bandwidth. In [11] redesign of the network stack and introduction of a new transport protocol guarantee low latency completion for short flows. Adjustable latency at the expense of reliability, and adaptive FEC is presented in [15].

6 CONCLUSIONS

In this paper we presented the “why” and “how” latency is introduced by FEC in high speed links. We demonstrated that frame offset can cause significant latency jitter, and that link heterogeneous systems may accumulate it. In practice, FEC turns cut-through switches into store-and-forward switches handling FEC block size frames.

It is of paramount importance that we understand that whatever latency injected to the system could never be taken off. It is also suggested that instead of a bottom up design (first link, then application) we should strive to a top down design. We proposed a programmable FEC gearbox that allows trading between latency and bandwidth to achieve performance. Feedback from this FEC gearbox is ported to the control plane, for latency to be controlled and prescribed. Future work will have to advance in two fronts: 1. understanding how latency impacts applications and 2. finding efficient FEC schemes that obey these observations. This vision of co-designed application and hardware is in the core of rack-scale systems.

REFERENCES

[1] Mohammad Al-Fares, Alexander Loukissas, and Amin Vahdat. 2008. A scalable, commodity data center network architecture. In *ACM SIGCOMM Computer Communication Review*, Vol. 38. ACM, 63–74.

[2] Mohammad Alizadeh, Abdul Kabbani, Tom Edsall, Balaji Prabhakar, Amin Vahdat, and Masato Yasuda. 2012. Less is more: trading a little bandwidth for ultra-low latency in the data center. In *Proceedings of the 9th USENIX conference on Networked Systems Design and Implementation*. USENIX Association, 19–19.

[3] Luiz Barroso, Mike Marty, David Patterson, and Parthasarathy Ranganathan. 2017. Attack of the killer microseconds. *Commun. ACM* 60, 4 (2017), 48–54.

[4] Richard E Blahut. 2003. *Algebraic codes for data transmission*. Cambridge university press.

[5] Broadcom. 2017. Broadcom Tomahawk 3. (2017). https://www.broadcom.com/blog/broadcom-s-tomahawk-3-ethernet\-switch-chip-delivers-12-8-tbps-of-speed-in-a-single-16-nm-device

[6] Stuart Cheshire. 1996. It's the Latency, Stupid. (1996). http://www.stuartcheshire.org/rants/latency.html

[7] Roy Cideciyan and John Ewen. 2011. Transcoding/FEC Options and Trade-offs for 100 Gb/s Backplane and Copper Cable. (2011). http://www.ieee802.org/3/bj/public/nov11/cideciyan_01a_1111.pdf

[8] Paul T Congdon, Prasant Mohapatra, Matthew Farrens, and Venkatesh Akella. 2014. Simultaneously reducing latency and power consumption in openflow switches. *IEEE/ACM Transactions on Networking (TON)* 22, 3 (2014), 1007–1020.

[9] Exablaze. 2017. *Exalink Fusion*. https://exablaze.com/downloads/pdf/ExaLINK_Fusion_Brochure.pdf.

[10] G David Forney. 1973. The viterbi algorithm. *Proc. IEEE* 61, 3 (1973), 268–278.

[11] Mark Handley, Costin Raiciu, Alexandru Agache, Andrei Voinescu, Andrew W Moore, Gianni Antichi, and Marcin Wójcik. 2017. Re-architecting datacenter networks and stacks for low latency and high performance. In *Proceedings of the Conference of the ACM Special Interest Group on Data Communication*. ACM, 29–42.

[12] IEEE. 2014. IEEE Standard for Ethernet Amendment 2: Physical Layer Specifications and Management Parameters for 100 Gb/s Operation Over Backplanes and Copper Cables. (2014).

[13] IEEE. 2016. 802.3by-2016 - IEEE Standard for Ethernet - Amendment 2: Media Access Control Parameters, Physical Layers, and Management Parameters for 25 Gb/s Operation. (2016). http://ieeexplore.ieee.org/servlet/opac?punumber=7526269

[14] IEEE. 2018. P802.3cd/D3.1, Feb 2018 - IEEE Draft Standard for Ethernet Amendment: Media Access Control Parameters for 50 Gb/s and Physical Layers and Management Parameters for 50 Gb/s, 100 Gb/s, and 200 Gb/s Operation. (2018). http://ieeexplore.ieee.org/servlet/opac?punumber=8288802

[15] Ryan J Marcotte and Edwin Olson. 2016. Adaptive forward error correction with adjustable-latency QoS for robotic networks. In *Robotics and Automation (ICRA), 2016 IEEE International Conference on*. IEEE, 5283–5288.

[16] Mellanox. 2017. Spectrum2. (2017). http://www.mellanox.com/page/products_dyn?product_family=277&mtag=spectrum2_ic

[17] Micron. 2017. Why CAS latency isn't an accurate measure of memory performance. (2017). http://uk.crucial.com/gbr/en/memory-performance-speed-latency

[18] Daniel Molka, Daniel Hackenberg, and Robert Schöne. 2014. Main memory and cache performance of Intel Sandy Bridge and AMD Bulldozer. In *Proceedings of the workshop on Memory Systems Performance and Correctness*. ACM, 4.

[19] Todd K Moon. 2005. Error correction coding. *Mathematical Methods and Algorithms. Jhon Wiley and Son* (2005).

[20] David A Patterson. 2005. Latency Lags Bandwidth.. In *ICCD*. 3–6.

[21] Jonathan Perry, Amy Ousterhout, Hari Balakrishnan, Devavrat Shah, and Hans Fugal. 2015. Fastpass: A centralized zero-queue datacenter network. *ACM SIGCOMM Computer Communication Review* 44, 4 (2015), 307–318.

[22] Diana Andreea Popescu, Noa Zilberman, and Andrew William Moore. 2017. Characterizing the impact of network latency on cloud-based applications‘ performance. (2017).

[23] Irving S Reed and Gustave Solomon. 1960. Polynomial codes over certain finite fields. *Journal of the society for industrial and applied mathematics* 8, 2 (1960), 300–304.

[24] Stephen M Rumble, Diego Ongaro, Ryan Stutsman, Mendel Rosenblum, and John K Ousterhout. 2011. It's Time for Low Latency.. In *HotOS*, Vol. 13. 11–11.

[25] Mellanox Technologies. 2017. *Mellanox Spectrum-2 Ethernet Switch*. http://www.mellanox.com/related-docs/prod_silicon/PB_Spectrum-2.pdf.

[26] Zhongfeng Wang and Chung-Jue Chen. 2011. Feasibility Of 100G-KR FEC. (2011). http://www.ieee802.org/3/100GCU/public/may11/wang_01_0511.pdf

[27] Noa Zilberman, Matthew Grosvenor, Diana Andreea Popescu, Neelakandan Manihatty-Bojan, Gianni Antichi, Marcin Wójcik, and Andrew W Moore. 2017. Where has my time gone?. In *International Conference on Passive and Active Network Measurement*. Springer, 201–214.