Priority-aware Forward Error Correction for HTTP

Nooshin Eghbal
University of Alberta
Edmonton, Canada
eghbal@ualberta.ca

Paul Lu
University of Alberta
Edmonton, Canada
paullu@ualberta.ca

ABSTRACT

TCP has been replaced by QUIC in the latest version of HTTP (i.e., HTTP/3) to reduce the Head-Of-Line (HOL) blocking problem. Also, there have been improvements to HTTP's priority mechanism for Web resources, beyond the existing tree-based one, to improve page-load times. We propose that Forward Error Correction (FEC) for selected, high-priority resources, can be (re-)introduced to HTTP to further reduce HOL blocking effects and improve page-load times, while maintaining reasonable overheads. A prototype and experiment are discussed.

CCS CONCEPTS

• **Networks → Network protocol design**.

KEYWORDS

HTTP, QUIC, Forward Error Correction (FEC), HTTP

ACM Reference Format:
Nooshin Eghbal and Paul Lu. 2022. Priority-aware Forward Error Correction for HTTP. In *Applied Networking Research Workshop (ANRW '22), July 25–29, 2022, PHILADELPHIA, PA, USA.* ACM, New York, NY, USA, 3 pages. https://doi.org/10.1145/3547115.3547195

1 INTRODUCTION

The Transmission Control Protocol (TCP) guarantees the in-order delivery of a stream of bytes. Therefore, all of the data after a lost packet are blocked until the retransmission of the lost packet is received. This is the known as the Head-Of-Line (HOL) blocking problem.

HTTP/1.1 and HTTP/2 both use TCP. As a result, they suffer from TCP's HOL blocking problem, which contributes to increased page-load times in the case of packet loss. The latest version of HTTP (i.e., HTTP/3) was developed to solve this problem by running over a reliable UDP-based multiplexed protocol called QUIC [8].

QUIC supports multi-streaming which means the data can be sent over independent streams so a packet loss in any of the streams would not block the data on other streams. This is useful for HTTP resources that can be processed incrementally such as images. However, QUIC still suffers from intra-stream HOL blocking. In other words, QUIC delivers data within each stream with total ordering but there is no ordering between the data sent over different streams.

Loading Web pages requires downloading different types of resources, such as HTML, JavaScript (JS), Cascading Style Sheet (CSS), and image files. Whereas image files can be rendered incrementally, JS and CSS need to be fully downloaded to be useful [9]. Therefore, HTTP prioritization and resource scheduling between the client and the server can minimize the page-load time.

In HTTP/2, the server builds a dependency tree to know the relative priority between requests [15]. However, this tree-based approach is complicated and difficult to use for both browsers and servers. Recently, the more practical Extensible Prioritization Scheme [11] has been proposed. In this approach, the priority of each HTTP response is set by the client to an absolute value between 0 and 7, in descending order of priority. Also, there is an incremental parameter for each request, that can be set to let the server know that the resource data can be sent incrementally using multi-streaming.

Knowing the priority of the requested resources at the server, we propose that Forward Error Correction (FEC) can be used **only** for the resources with higher priority, especially the non-incremental ones (e.g., JS and CSS). Without FEC, a lost packet will still cause extra latency in page loading, as packets are retransmitted. Other low-priority resources that can be applied/rendered incrementally, like image files, can still take advantage of multiplexing/multi-streaming over QUIC. And only high-priority resources incur the overheads of FEC, while improving latency.

Notably, we are proposing a selective, priority-aware use of FEC for HTTP. In the first version of QUIC by Google [13], FEC was a built-in option. However, FEC was dropped from QUIC due to the overhead and negative experimental results [7]. Other studies considered FEC over QUIC, but for all resources [10], [4], [12]. However, with the Extensible Prioritization Scheme, it may be time to reconsider FEC.

ANRW '22, July 25–29, 2022, PHILADELPHIA, PA, USA

ACM ISBN 978-1-4503-9444-4/22/07…$15.00
https://doi.org/10.1145/3547115.3547195

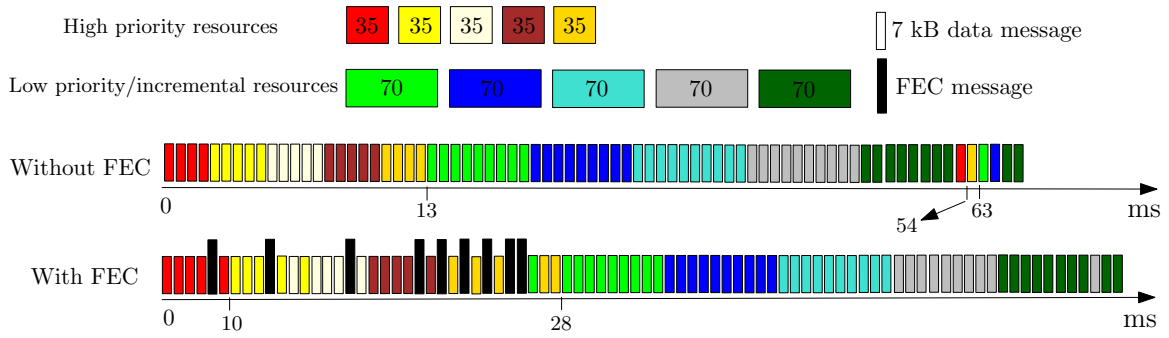

Figure 1: The arrival order of messages w/o FEC over UDT.

2 UDT AND 2D XOR FEC

As a proof of concept, we have implemented FEC over UDP-based Data Transfer (UDT) [5] with a two-dimensional XOR-based (2D XOR) FEC. Both UDT and QUIC are UDP-based, and FEC over QUIC would be the next step in our work.

UDT is a user-level, reliable protocol designed for large data transfers over Wide-area Networks (WANs). Multiple Congestion Control Algorithms (CCA) are supported by UDT, including a non-loss-based CCA called UDPBlast. With UDPBlast, there is a fixed sending rate that will not change, even with packet loss. Evaluating our priority-aware FEC for HTTP with a loss-based CCA remains as a future work.

UDT has two modes: 1) stream mode and 2) message mode. The stream mode is like TCP and supports total ordering only but in the message mode we can set an order flag for each message to be delivered in total order or arrival order. For this work, we ran the experiments in message mode with arrival order delivery to avoid HOL blocking problem in the case of packet loss to get close to QUIC.

The original UDT source code does not include any FEC implementation. We have implemented an XOR-based FEC called 2D XOR inside UDT [1] and used it for our experiments in this paper. In 2D XOR FEC method, we build 2D matrices of the original packets and send the XOR of all rows and all columns as redundant packets to the destination.

The interesting thing about 2D XOR FEC is that we can recover burst packet loss patterns with the help of column recovery while in 1D XOR FEC we can recover only one packet loss in each row [2][3]. The other advantage of 2D XOR FEC is that it is still simple and would have low computational overhead. In our implementation of 2D XOR over UDT, the user needs to set the number of rows and columns.

3 PRELIMINARY RESULTS

We evaluate the idea of using FEC only for high-priority resources through a simple workload using our 2D XOR FEC implementation over UDT in an emulated testbed. We use the Netem-tc Linux tool [6] to set the loss rate to 2% and RTT to 50 ms between two nodes in Emulab testbed [14]. Each node has one 3 GHz 64-bit Xeon processor and are networked with 1 Gbps Ethernet.

In our synthetic workload, we have 5 high-priority resources each 35 kB (e.g., JS and CSS) and 5 low priority resources each 70 kB (e.g., images) that need to be sent to the client. In Figure 1, the UDT message size is 7 kB. Therefore, each resource is fragmented into multiple messages (e.g., each high-priority resource is sent using 5 messages and each low priority resource is sent using 10 messages).

We consider two scenarios: 1) Not using FEC at all, and 2) using 2D XOR FEC only for the 5 high-priority resources. We set the size of the 2D matrix for the second scenario to 5*5 to include the messages of all 5 high-priority resources (i.e., 25 messages in total). In Figure 1, we show the order in which we receive messages for both scenarios. Note that "With FEC", the red high-priority resource arrives and is recovered (using FEC) by time 10 ms. "Without FEC", the red resource arrives much later at 54 ms, after a retransmission. Furthermore, "With FEC" all 5 high-priority messages arrive or are recovered after 28 ms. "Without FEC", we need to wait for an RTT (i.e., 50 ms) to receive the retransmission of a lost message of the golden resource, and it takes 63 ms to receive all 5 high-priority resources.

4 CONCLUDING REMARKS

Page-load delays hurt the user experience on the Web, and the consequences of packet loss for HOL blocking continue to plague HTTP such that HTTP/3 has switched to using QUIC instead of TCP. Without a priority mechanism in HTTP, FEC did not necessarily make sense due to the overheads of using FEC for all resources. However, since HTTP now has a priority mechanism, a priority-aware FEC for HTTP can make reasonable trade-offs between overheads and page-load latency. Our experiments with FEC over UDT provide evidence for these benefits. Implementing FEC over QUIC remains as future work.

REFERENCES

[1] Nooshin Eghbal and Paul Lu. 2021. Low-Variance Latency Through Forward Error Correction on Wide-Area Networks. In *2021 IEEE 46th Conference on Local Computer Networks (LCN)*. IEEE, 90–98.

[2] Simone Ferlin, Stepan Kucera, Holger Claussen, and Özgü Alay. 2018. Mptcp meets fec: Supporting latency-sensitive applications over heterogeneous networks. *IEEE/ACM Transactions on Networking* 26, 5 (2018), 2005–2018.

[3] Tobias Flach, N Dukkipati, Y Cheng, and B Raghavan. 2013. Tcp instant recovery: Incorporating forward error correction in tcp. *Working Draft, IETF Secretariat, Internet-Draft draft-flach-tcpm-fec-00, July* (2013).

[4] Pablo Garrido, Isabel Sanchez, Simone Ferlin, Ramon Aguero, and Ozgu Alay. 2019. rQUIC: Integrating FEC with QUIC for robust wireless communications. In *2019 IEEE Global Communications Conference (GLOBECOM)*. IEEE, 1–7.

[5] Yunhong Gu and Robert L Grossman. 2007. UDT: UDP-based data transfer for high-speed wide area networks. *Computer Networks* 51, 7 (2007), 1777–1799.

[6] Stephen Hemminger et al. 2005. Network emulation with NetEm. In *Linux conf au*, Vol. 5. Citeseer, 2005.

[7] Arash Molavi Kakhki, Samuel Jero, David Choffnes, Cristina Nita-Rotaru, and Alan Mislove. 2017. Taking a long look at QUIC: an approach for rigorous evaluation of rapidly evolving transport protocols. In *Proceedings of the 2017 Internet Measurement Conference*. 290–303.

[8] Adam Langley, Alistair Riddoch, Alyssa Wilk, Antonio Vicente, Charles Krasic, Dan Zhang, Fan Yang, Fedor Kouranov, Ian Swett, Janardhan Iyengar, et al. 2017. The quic transport protocol: Design and internet-scale deployment. In *Proceedings of the conference of the ACM special interest group on data communication*. 183–196.

[9] Robin Marx, Tom De Decker, Peter Quax, and Wim Lamotte. 2019. Of the Utmost Importance: Resource Prioritization in HTTP/3 over QUIC.. In *WEBIST*. 130–143.

[10] François Michel, Quentin De Coninck, and Olivier Bonaventure. 2019. QUIC-FEC: Bringing the benefits of Forward Erasure Correction to QUIC. In *2019 IFIP Networking Conference (IFIP Networking)*. IEEE, 1–9.

[11] Kazuho Oku and Lucas Pardue. 2020. Extensible prioritization scheme for http. *Work in Progress, Internet-Draft, draft-ietfhttpbis-priority-02* 1 (2020).

[12] V. Roca, F. Michel, I. Swett, and M. Montpetit. 2019. Sliding Window Random Linear Code (RLC) Forward Erasure Correction (FEC) Schemes for QUIC. *draft-roca-nwcrg-rlc-fec-scheme-for-quic-02* (2019).

[13] Ian Swett Ryan Hamilton, Janardhan Iyengar and Alyssa Wilk. 2016. QUIC: A UDP-Based Secure and Reliable Transport for HTTP/2. *Internet-Draft draft-hamilton-early-deployment-quic-00* (2016).

[14] Brian White, Jay Lepreau, Leigh Stoller, Robert Ricci, Shashi Guruprasad, Mac Newbold, Mike Hibler, Chad Barb, and Abhijeet Joglekar. 2002. An integrated experimental environment for distributed systems and networks. *ACM SIGOPS Operating Systems Review* 36, SI (2002), 255–270.

[15] Maarten Wijnants, Robin Marx, Peter Quax, and Wim Lamotte. 2018. HTTP/2 prioritization and its impact on web performance. In *Proceedings of the 2018 World Wide Web Conference*. 1755–1764.