Multiplexed Stream Delivery in HTTP/1.1, HTTP/2, and HTTP/3

This document will discuss the different methods by which hypertext transfer protocols operate. HTTP in short is based on the client-server model, where clients will request information about a webpage, and the web server will respond with the required resources. This interaction is scrutinized in every version of HTTP, and for good reason. The speed of data acquisition can make or break a service; each fleeting second increases the probability that a user will click off and go to a different site. APIs also make up a large percentage of all internet traffic. As shown in a report made by Akamai Technologies, "A recent analysis of Akamai's ESSL network revealed an 83% to 17% split between API and HTML traffic on our secure content delivery network. This is a significant increase since the same survey was performed in 2014[1]". API traffic can also be limited by the delivery methods of HTTP, and the demand for faster speeds is rising with each year.

HTTP/1.1

HTTP/1.1, established in RFC 2616, was the first major revision introduced to the web. It brought about the first implementation of multiplexing. RFC 2616 states, "In HTTP/1.0, most implementations used a new connection for each request/response exchange. In HTTP/1.1, a connection may be used for one or more request/response exchanges, although connections may be closed for a variety of reasons[2]". HTTP/1.0's shortcoming was its inability to reuse and repurpose connections, which added further complications and time for loading web pages. 1.1 didn't need to manage a multitude of request/response connections, which saved CPU processing time and memory for routers and hosts alike. Stream multiplexing also reduced network congestion, improved latency, and allowed for servers to support a sliding window approach(Which allowed clients to make multiple requests without waiting for each subsequent response). These changes improved HTTP, and provided quicker access to web objects for users.

HTTP/2

HTTP/2, established in RFC 7540, also scrutinized it's predecessor's method of content delivery. HTTP/1.1 implementations at the time of 2's installment were still suffering from redundant TCP connections being established, and head-of-line blocking(When a large resource was being pipelined in the first queue position, and held up other frames from being processed). RFC 7540 states, "[HTTP/2] allows interleaving of request and response messages on the same connection and uses an efficient coding for HTTP header fields. It also allows prioritization of requests, letting more important requests complete more quickly [3]". HTTP/2 leveraged stream multiplexing to fragment responses, allowing for out of order delivery. This removed the need for blocked or erroneous requests or responses from preventing the processing of other requests

on stream. On top of the multiplexing, flow control was also enhanced by prioritization streams. Setting stream priority provided clients and servers with the ability to have stream dependence(Where one stream can't be processed until X amount of other streams have already been processed), as well as selective sending of frames, so that the most important packets are sent first. These improvements allowed for a more modular approach to delivery, and each stream was being processed at the maximum efficiency(For the time).

HTTP/3

HTTP/3, established in RFC 9114, was the first revision to move away from TCP. HTTP/2's biggest downfall was due to it being built off of TCP. TCP made successful multiplexing of requests and responses difficult, as if any were lost, all of them would have to be retransmitted. To not diverge too much from prior implementations, the initial request/response transaction is done over TCP, though all subsequent streams are done over UDP to provide a more granular service. QUIC, or Quick UDP Internet Connections, was made prematurely to the release of HTTP/3, and revolutionized multiplexing. In RFC 9114's overview, it states, "QUIC provides protocol negotiation, stream-based multiplexing, and flow control…Each request-response pair consumes a single QUIC stream. Streams are independent of each other, so one stream that is blocked or suffers packet loss does not prevent progress on other streams [4]". QUIC took many inspirations from HTTP/2's multiplexing implementation, though it didn't suffer from the same TCP flaws. Just as HTTP/2 had priority streams, HTTP/3 had priority parameters. These priority parameters were denoted in the HTTP header field, and came in two flavors.

Alongside RFC 9114, HTTP/3's priority parameters were defined in RFC 9218, and are described as the "Extensible Prioritization Scheme". The first priority parameter introduced in the urgency (u) parameter. When talking about response urgency, RFC 9218 states, "The chosen value of urgency can be based on the expectation that servers might use this information to transmit HTTP responses in the order of their urgency[5]". HTTP/3's Urgency fully expanded on HTTP/2's implementation, while still providing the same function. The Incremental (i) parameter was the other priority flag. This flag was a boolean statement, which allowed for servers to control at what times the client would receive responses. RFC 9218 states, "Incremental delivery is most useful where multiple partial responses might provide some value to clients ahead of a complete response being available[5]". This portioned delivery method gave clients a more adaptive approach to requesting data, and allowed for responses to be partially delivered to save time and network costs. Additionally, HTTP/3 brough with it reprioritization of requests. Due to the aforementioned urgency parameter, there were defined certain types of resources that would be given a low priority(not in values, as 0 means highest priority) by default, and reprioritization allowed for those resources' urgency values to be changed at a moment's notice. This gave users further moderation of what resources were loaded, when; decreasing transaction times, while still establishing client-side control.

These improvements, the QUIC handshake, and being built off of UDP bolstered HTTP/3's adoption due to the major changes to end-user experience. HTTP/3 stands as the

current latest revision, and for good reason. It brought with it the largest performance improvements of all revisions, and didn't diverge from its roots.

Sources:

[1] Akamai Technologies, "[state of the internet] / security Retail Attacks and API Traffic Report: Volume 5, Issue 2" 02/19 Available: amakai.com, https://www.akamai.com/site/it/documents/state-of-the-internet/state-of-the-internet-security-retail-attacks-and-api-traffic-report-2019.pdf [Accessed: Dec 16, 2022]

[2] Fielding, R., Gettys, J., and, J. Mogul, Ed., "Hypertext Transfer Protocol (HTTP/1.1)", RFC 2616, June 1999, https://www.ietf.org/rfc/rfc2616.txt [Accessed: Dec 16, 2022]

[3] Belshe, M., Peon, R., and M. Thomson, Ed., "Hypertext Transfer Protocol Version 2 (HTTP/2)", RFC 7540, DOI 10.17487/RFC7540, May 2015, https://www.rfc-editor.org/rfc/rfc7540 [Accessed: Dec 16, 2022]

[4] Akamai Technologies, M. Bishop, Ed., "Hypertext Transfer Protocol Version 3 (HTTP/3)", RFC 9114, June 2022, https://datatracker.ietf.org/doc/rfc9114/ [Accessed: Dec 16, 2022]

[5] Fastly, Cloudflare, and L. Pardue, Ed., "Extensible Prioritization Scheme for HTTP", RFC 9218, ISSN 2070-1721, June 2022, https://www.rfc-editor.org/rfc/rfc9218.html [Accessed: Dec 16, 2022]