Selective Enablement of L4S Transport for Latency-Sensitive Multimedia Delivery

Dhananjay Lal and Christopher Phillips
Adeia Inc.
3025 Orchard Parkway
San Jose, California 95134
Email: {dj.lal, chris.phillips}@adeia.com

***Abstract*—The L4S standard has been gaining ground in the recent past, bringing together application and network service providers for building a more responsive internet. While application providers are working towards enabling L4S in their interactive applications, service providers are building dual queue architectures in their access networks. However, the state-of-the-art applies L4S to entire flows and may raise fairness considerations, esp. if there is deliberate abuse of the standard. While queue protection policing is currently also being developed, we propose novel methods for selectively enabling L4S, premised on judicious use of the low latency queue resource. In our proposal, we consider both ultra-low latency and low latency interactive video applications, wherein L4S is invoked based on PES picture size and CMAF chunk/fragment size, respectively. Further, we also contemplate transport layer handling with independent L4S and non-L4S QUIC streams and separate send/receive buffers when both pathways are used in conjunction for delivery. We conclude with recommendations of future work for further evaluation and development of our proposed methods.**

I. Introduction

According to "Low Latency, Low Loss, and Scalable Throughput (L4S) Internet Service: Architecture," RFC 9330 [1], "queuing remains a major, albeit intermittent, component of latency. For instance, spikes of hundreds of milliseconds are not uncommon, even with state-of-the-art Active Queue Management (AQM). ... It has been demonstrated that, once access network bit rates reach levels now common in the developed world, increasing link capacity offers diminishing returns if latency (delay) is not addressed." [1] further states that "Queuing delay degrades performance intermittently. ... It occurs 1) when a large enough capacity-seeking (e.g., TCP) flow is running alongside the user's traffic in the bottleneck link, which is typically in the access network, or 2) when the low latency application is itself a large capacity-seeking or adaptive rate flow (e.g., interactive video)."

The L4S standard has been introduced to help address these issues. As stated in [1], "L4S is based on the insight that the root cause of queuing delay is in the capacity-seeking congestion controllers of senders, not in the queue itself. With the L4S architecture, all Internet applications could (but do not have to) transition away from congestion control algorithms that cause substantial queuing delay and instead adopt a new class of congestion controls that can seek capacity with very little queuing. These are aided by a modified form of Explicit Congestion Notification (ECN) from the network. With this new architecture, applications can have both low latency and high throughput."

With L4S, network service providers have introduced dual queuing in their network (especially the access network, that may have limited capacity). While most queue-building traffic passes through a Classic/Default queue, some traffic in need of low latency may use a low latency queue. This "priority lane" is typically expected to be used by ultra-low latency, non-queue-building traffic. However, the authors hypothesize that other applications (including queue-building applications) may also benefit from this queue if they use it selectively.

As stated in [1], "The Dual-Queue Coupled AQM ... acts like a 'semi-permeable' membrane that partitions latency but not bandwidth. As such, the two queues (shown in Figure 1) are for transitioning from Classic to L4S behavior, not bandwidth prioritization."

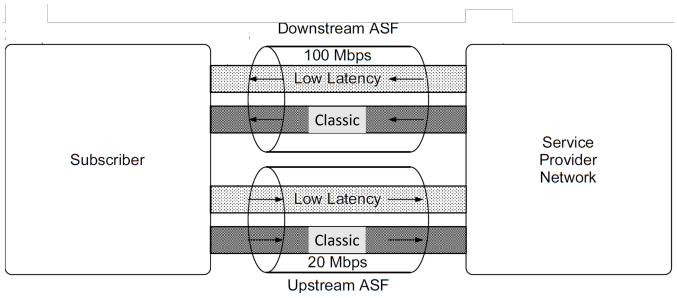

Fig. 1. Low Latency DOCSIS Service Flow [2]

Further, it states that "the scheduler can serve the L4S queue with priority (denoted by the '1' on the higher priority input), because the L4S traffic isn't offering up enough traffic to use all the priority that it is given. Therefore, for latency isolation on short timescales (sub-round-trip), the prioritization of the L4S queue protects its low latency by allowing bursts to dissipate quickly; but for bandwidth pooling on longer timescales (round-trip and longer), the Classic queue creates an equal and opposite pressure against the L4S traffic to ensure that neither has priority when it comes to bandwidth – the tension between prioritizing L4S and coupling the marking from the Classic AQM results in approximate per-flow fairness."

White et al. [2] further elaborate on the mechanism for

avoiding starvation: "To enable the Low Latency Queue to rapidly dequeue an arrived burst of traffic, the Inter-Service-Flow scheduler gives a higher weight to the Low Latency Queue than it does to the Classic Queue. The coupling to the Low Latency AQM counterbalances the weighted scheduler by making low-latency applications leave space for Classic traffic. This ensures that the weighted scheduler does not give priority over bandwidth, as a traditional weighted scheduler would." From RFC 9332 [3]: "The scheduling weight of the Classic queue should be small (e.g., 1/16) ... if L4S traffic is over-aggressive or unresponsive, the scheduler weight for Classic traffic will at least be large enough to ensure it does not starve in the short term" and "The scheduler draining the two queues MUST give L4S packets priority over Classic, although priority MUST be bounded in order not to starve Classic traffic". The dual queuing paradigm has also been proposed for other access networks, such as in wireless cellular communication [4].

Binary Codepoint	Codepoint Name	Meaning
00	Non-ECT	Not ECN-Capable Transport
01	ECT(1)	L4S-Capable Transport
10	ECT(0)	ECN-Capable Transport
11	CE	Congestion Experienced

TABLE I
ECN MARKINGS AND THEIR MEANINGS

L4S enablement on a packet occurs by marking the ECN bits in the packet IP header. As shown in Table I, ECT(1) marking indicates that the sender is capable of L4S transport. If a network element experiences congestion, it converts the 2-bit ECN marking from ECT(1) to CE. The markings are echoed back to the sender in acknowledgements from the receiver. The sender is then required to reduce throughput in a scalable manner.

II. MOTIVATION

It is acknowledged by the community of practice that interactive applications with dynamic content such as conversational audio/video, Cloud gaming, Virtual Reality (VR) Streaming, video conferencing, etc. benefit from L4S delivery. Thus, on the one hand internet service providers like Comcast are implementing dual queuing in their access network [5], on the other hand application developers, like Nvidia, and developers of application environments, like Apple, have expressed or released support for L4S. For example, [6] declares that "Apple's QUIC implementation fully supports L4S, whereas Apple's TCP implementation supports only the receiver-side L4S." L4S delivery almost always involves video transmission. Further, it is difficult to police how application service providers may use or abuse ECN packet markings to minimize latency in their application traffic. Thus, ISPs are looking to implement queue protection mechanisms that transfer packets back to the Classic queue from the low latency queue if they detect abuse of the low latency queue [7]. The authors hypothesize that some sender-side controls that enable use of a low latency service flow on a selective basis would minimize situations in which such abuse would occur.

In this paper, we explore 2 different types of video transmissions: ultra low-latency video such as Cloud Gaming that requires RTT latency of 10–100 milliseconds, and low-latency OTT video streaming such as live sports that typically requires RTT latency of 1–10 seconds. Currently, it is understood that L4S delivery would be enabled for an entire flow belonging to the former traffic class, while the latter class of traffic, being queue-building, does not need L4S. In our proposal, both classes of traffic would be able to leverage L4S delivery, albeit selectively. This derives from the insight that while video encoders deliver an aggregate bit rate based on the target bit rate input, the instantaneous bit rate can vary drastically and may need assistance from the network for delivery when it surges. Hence, the determination to selectively enable L4S is made based on the size of a picture/frame or Packet Elementary Stream (PES) packet for ultra low-latency RTP delivery or CMAF fragment size for low-latency delivery.

III. PROPOSED METHOD FOR ULTRA LOW LATENCY INTERACTIVE APPLICATIONS

When video is encoded at a set bitrate, the encoder encodes the video to average out to a bitrate over time, achieved by a defined buffer model on a client device. This allows the encoder to encode I (Intra), P (Predicted) and B (Bidirectional) pictures which all have varied sizes where an I picture is typically very large as compared to P and B pictures. P pictures are typically larger than B pictures. The difference between one frame and the next also impacts the picture size, particularly in Cloud Gaming. Due to the extreme low latency requirement, an encoder may be configured to encode an I picture at the beginning while every picture after the I picture is a P picture. B pictures are not encoded in Cloud Gaming due to the increased latency. Further, there is typically no need to generate an IDR frame often since no other client devices will need to join the video stream. In these cases, an IDR picture may be created at the start of the video stream followed by P pictures. In other cases, the GOP size parameter may enforce the generation of an I frame periodically.

An example is a racing game where the difference in size from one frame to the next can be significant. Some racing games allow a user playing the game to switch views from the front windshield to a left, right or rear view. The difference from one frame to the next will cause the picture sizes to increase significantly. The series of frames in Figure 2 is an example of two scene changes causing very large P picture sizes to be generated on the encoded first frame of the scene change.

The graph of picture sizes in Figure 3 shows very large P pictures being generated in a cloud gaming environment, where, depending on the difference from one picture to the next, a very large amount of data may need to be encoded. Notice there are several frames that are $\sim$600 KB, typical at a scene change as described above. The encoded video in this example is AVC (h.264) 4K 60Hz at 85 Mbps. Since this is

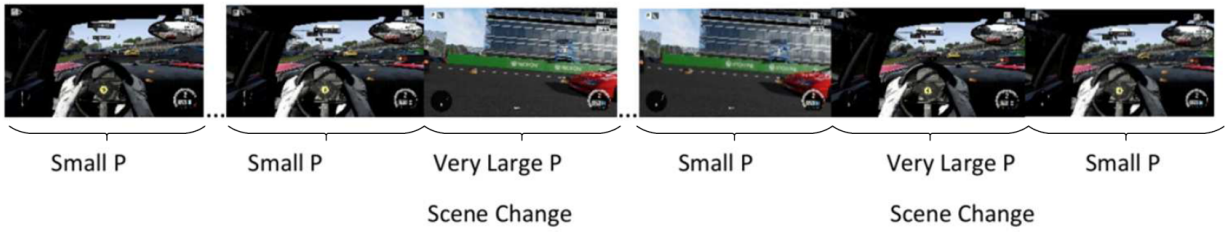

Fig. 2. Scene Changes in Cloud Gaming that lead to large P frames

extremely low latency delivery with a minimal buffer size, to deliver 600 KB in 16.67 ms will require a spike in bandwidth requirement to ∼288 Mbps. The video will average over time to 85 Mbps and depending on the buffer model of the client device, this would not pose a problem since many frames will be much smaller in size allowing the buffer to not drain completely and rebuild on smaller size frames. Further, our empirical results indicate that when a GOP size is enforced, say every 2 seconds, then the I frames get much larger than the P frames, creating a similar surge in required delivery throughput.

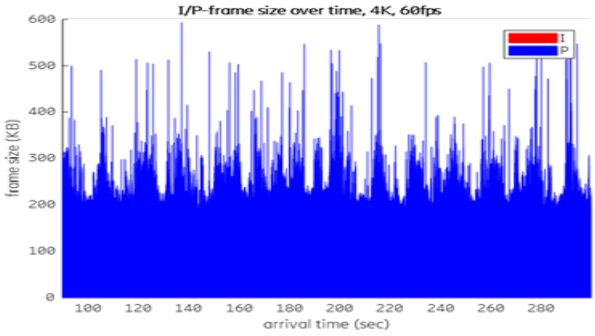

Fig. 3. Predicted pictures in a cloud gaming environment with an IP GOP structure with all P pictures following the I picture

The proposed scheme aims to selectively mark transport packets which encapsulate a PES (Packetized Elementary Stream) as high priority based on a frame size. If the number of transport packets to encapsulate and transport a picture or PES packet are beyond a threshold, all the transport packets of the large, encoded picture will be L4S-enabled.

Figure 4 is an example system architecture for the proposed ultra low latency Real Time Protocol (RTP) delivery system, with the key path in selectively enabling L4S delivery shown in blue. The RTP sender contains a video parser to parse the PES stream for frame, slice and tile data. The video parser sends the RTP multiplexer the parsed PES bit stream data for identifying the video specifics (frames, slices, tiles) contained in the PES packet based on offsets within the PES packet. Typically, an entire frame is contained in a single PES packet. This allows the RTP multiplexer to generate data structures within which there is a priority marker that is set based on a threshold number of 1500 byte RTP packets required to transmit the frame, slice or tile data. The Transmission Scheduler retrieves the first packet in the transmission priority queue, extracts the RTP packet data, header + payload data, and parses the data structure for the priority flag. If the priority flag is set to 1, the UDP socket is set to ”L4S-enabled” and the corresponding RTP packet will have the ECN bits set to ECT(1) when transmitted over the UDP port to the client device. This setting will allow the packet to receive low latency flow treatment in the network, provided L4S is supported. When the client device receives the L4S-enabled RTP packet, the RTP receiver on the client device will echo the ECN bits in the response sent from the client device to the server. If the priority flag is set to 0, the UDP socket is set to ”L4S-disabled” and the RTP packet will have the ECN bits cleared when it is transmitted over the UDP port to the client device.

Figure 5 is an example priority queue made of the new data structure which identifies what data is included in the RTP multiplexed packet. In RTP transport, it is assumed there is a source identifier (SSRC) for video and audio. This data structure includes the RTP Sequence Number which is also included in the RTP packet header, a Frame Number for the video or audio, a PES Packet Number which in most cases will be the same for a frame, the Slice number which will always be 0 if the video is encoded with 1 slice per frame, the Tile which will always be null when tiles are not in use, an ”Is Audio” flag for identifying whether the packet is transporting audio data, a Priority flag which indicates if the packet should be ”L4S-enabled” or not, and the 1500 byte RTP Multiplexed Packet (headers + payload Data). Since this is an example of 1 slice per frame and the tile data is null, it is handled as prioritizing the RTP packets at the frame level. The system will set the Priority flag for all video and audio packets from the first RTP packet for the video frame data to the last packet for the video frame data and all interleaved audio packets for that video frame.

The latency distribution of packets with selective L4S enablement is expected to be bimodal. While L4S packets pass through a channel with lower latency and lower loss probability, non-L4S packets encounter channel conditions with higher latency and higher loss. To maintain the sequence of packets at the receiver even as packets may arrive out-of-order, we propose that separate L4S and non-L4S transport buffers are maintained. As packets arrive in the buffers, they are sequenced back to the original RTP stream before sending to the demultiplexer. This is discussed in more detail later and illustrated in Figure 7.

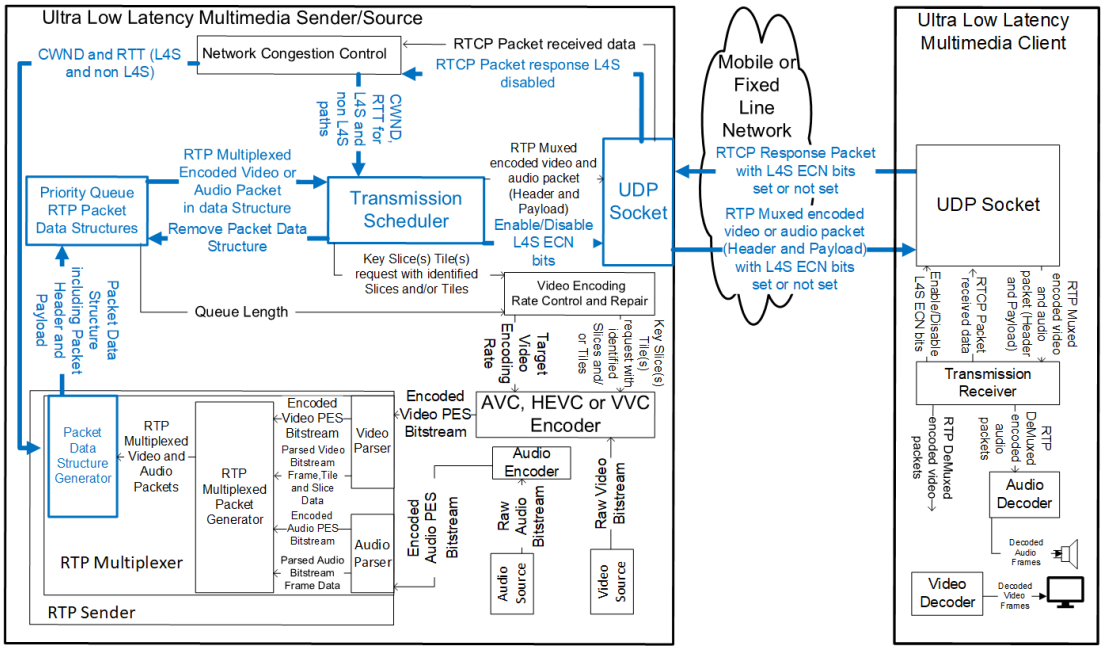

Fig. 4. Example system architecture for prioritized frame, slice or tile delivery optimization

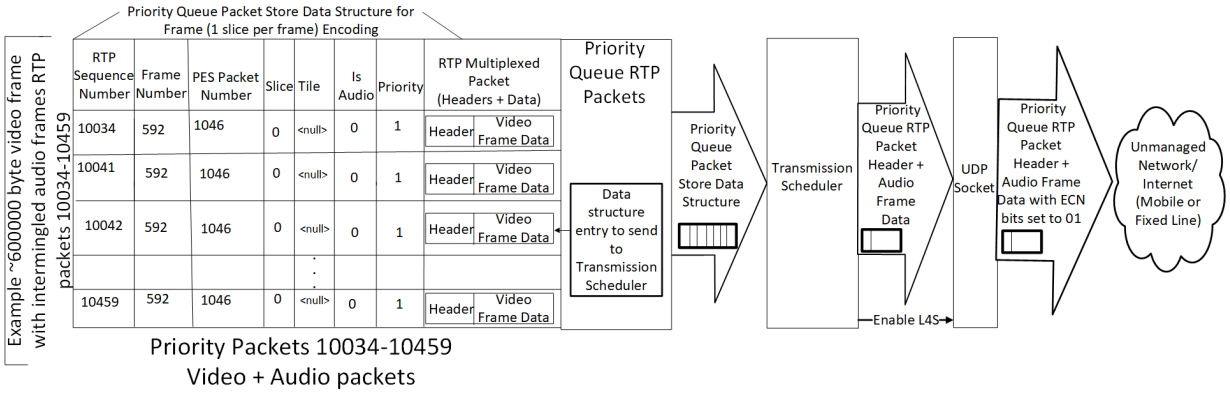

Fig. 5. An example queue with data structure entries for the RTP packets within a frame when there is 1 slice per frame

Congestion control in RTP-RTCP requires elements of proprietary implementation based on application, while using generally understood principles for adapting the target encoding rate in the "video encoding rate control and repair module" at the RTP Sender, shown in Figure 4. [8], which explains the design of Stadia, a currently obsolete Cloud Gaming service from Google, describes these well understood principles.

To account for L4S-enabled packets and non-L4S enabled packets, which experience different RTTs and loss probabilities, we further propose that the RTP sender estimates delay and loss separately for L4S and non-L4S channels. These estimations are performed and periodically updated by the sender as RTCP retransmission requests and receiver reports are received. To provide a single target bitrate to the video encoder's rate controller, a weighted average of the bitrates estimated for L4S and non-L4S paths is calculated.

IV. Proposed Method for Low Latency Live OTT ABR Streaming

In contrast to ultra-low latency use cases, OTT ABR live-streaming is a low latency use case (typically 1-10 seconds). OTT ABR is a pull model where the client device requests segments for download over HTTP. Since the MP4 container format was created as the format for file-based content, it has needed improvements for use in ABR streaming. The MP4 multiplexer had to complete the multiplexing of the segment before it could be placed on the CDN origin, distributed to the client devices, and made available for play out. This caused an increase in latency for initial play out of video. Thus, while changing channels in an OTT environment, the rendering time of the video was a poor user experience. Multiplexing changes for MP4 containers have been included in the MPEG standard and have resulted in the CMAF format being added to the

MPEG-4 Part 14 specification. This allows the multiplexer to include a new movie fragment (MOOF) box to be included into the multiplexed stream. Consequently, the segment is subdivided into fragments/chunks. Each fragment or chunk is separated by a MOOF box. Further, the demultiplexer can demultiplex at the fragment level, rather than having to download the complete segment to begin play out.

A novel approach for optimizing OTT CDN delivery of CMAF fragments or chunks is proposed in Figure 6. The system speeds up initial rendering time for a live stream on a channel change. It also speeds up the initial rendering when a user input time-shifts live TV beyond the current playing live segment. Additionally, this system architecture can provide an optimization for delivery of large chunks and prevent the ABR client device from dropping the bitrate when it is close to its threshold of calculated bitrate for moving to a lower quality in its bitrate ladder. Through research investigating how ABR clients behave, it has been observed that these client devices will adjust bitrate lower when the calculated bitrate is $\sim$85% of the represented bitrate in the manifest bitrate ladder.

In our proposal, a sender may decide to use the low latency queuing pathway for delivery of a CMAF fragment/chunk to a client via selective L4S enablement based on its size being above a threshold.

As shown in Figure 6, a live stream received from a broadcaster is converted by an ABR transcoder into ABR encoded streams based on a series of bitrates. The ABR encoded streams are fed to a live ABR CMAF-compatible packager. The packager is configured to generate a series of MP4 CMAF segments. Each CMAF segment is fragmented/chunked based on a defined number of frames per chunk for all encoded ABR streams in the bitrate ladder and all audio encodings based on audio languages, formats, etc. The Live CMAF Packager generates and updates a live ABR manifest based on the new live segment chunks being generated.

While the manifest may not usually be distributed to the CDN edge, in this architecture the manifest is also sent to the CDN edge nodes and updated at all edge nodes along with the segments. When a client requests a segment from an edge node, the MP4 CMAF parser parses the multiplexed file to determine the size for each chunk in the segment to be delivered. It sends the requested CMAF segment fragment size and the segment requested bitrate to a Threshold Calculation system, which determines a threshold size for L4S-enablement based on the requested segment bitrate and the size of the CMAF segment's fragment/chunk. The MP4 CMAF Container Chunk Size Parser, based on the received calculated threshold size, enables or disables L4S using the ECN markings shown in Table 1.

To explain the impacts of our proposed scheme, we reference the Mathis' model for the TCP throughput as bounded by its congestion control algorithms, which takes into account link characteristics such as Maximum Segment Size (MSS), Round-Trip Time (RTT) and packet loss probability p [9].

$$T = \frac{(MSS*C)}{(RTT*\sqrt{p})}$$

Constant C lumps together several terms that are typically invariant for a given combination of TCP implementation, ACK strategy, and loss mechanism. As is evident from this model, the throughput is inversely proportional to the RTT and (square root of) loss probability p. Since RTT and p reduce for L4S channel pathways, the average throughput increases. Note that depending on the chunk/fragment size, a few to several hundreds of transport packets may be L4S-enabled. Low latency CMAF generally uses smaller fragments – this reduces the potential "queue-building" load on the low latency network buffers, albeit the specific large fragment being accelerated may require a deeper buffer relative to the average fragment. Thus, even though the average throughput is increased for transport of a chunk/fragment, the increase in throughput from a segment standpoint is instantaneous.

If the network buffers experience congestion at the time that the fragment is being transmitted over the L4S pathway, then the CE bit markings indicate to the sender that they must scale back their throughput. The sender may reduce throughput by using logic that is similar to a packet loss response, such as re-entering the slow start phase of congestion control (Ex., for TCP Tahoe, reducing $cwnd$ to a small multiple of MSS, and halving slow start threshold). One of the artifacts of using L4S and non-L4S pathways with a Head-of-line (HOL) blocking protocol such as TCP/QUIC (HTTP/2 or HTTP/3) is that RTT for L4S chunks may be lower than others. An L4S-enabled chunk may arrive sooner, putting chunks out of order at the receiver. This creates a duplicate-ACK interpretation problem in the congestion control algorithms, with the client inferring that a non-L4S chunk was lost, reducing its congestion window, and throughput.

QUIC has been steadily replacing TCP as a protocol of choice in HTTP adaptive streaming. QUIC may use independent logical transports between a source and a destination, called streams, to reduce Head-of-Line (HoL) blocking that may occur when all the data is sent as a single byte-stream. Streams are individually flow-controlled, allowing an endpoint to limit memory commitment and to apply back pressure. An arbitrary number of streams may operate concurrently, albeit each stream will be logically ordered as a sequence of bytes received in the order they are sent. Separate streams, however, are not necessarily delivered in the original order. In the IETF, a new working group, Media over QUIC (MOQ), was formed in 2022 to study the possible enhancements that QUIC may bring for low-latency live-streaming [10].

In this work, we propose that the transmission scheduler at the sender separates L4S and non-L4S packets into separate QUIC streams. Thus, flow control and congestion control for these streams is performed separately, and even though out-of-order arrival may occur when considering the 2 streams together, HoL blocking occurs for each stream separately (Figure 7). The solution is therefore HTTP/3 ready.

V. Conclusion and Future Work

In this paper, we proposed novel methods to selectively enable L4S packet markings for ultra-low and low latency

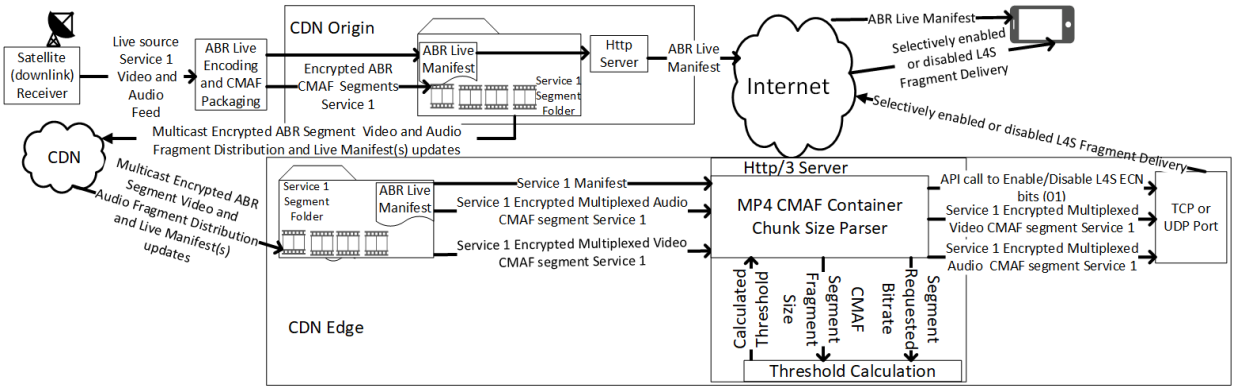

Fig. 6. System Architecture for OTT ABR Delivery

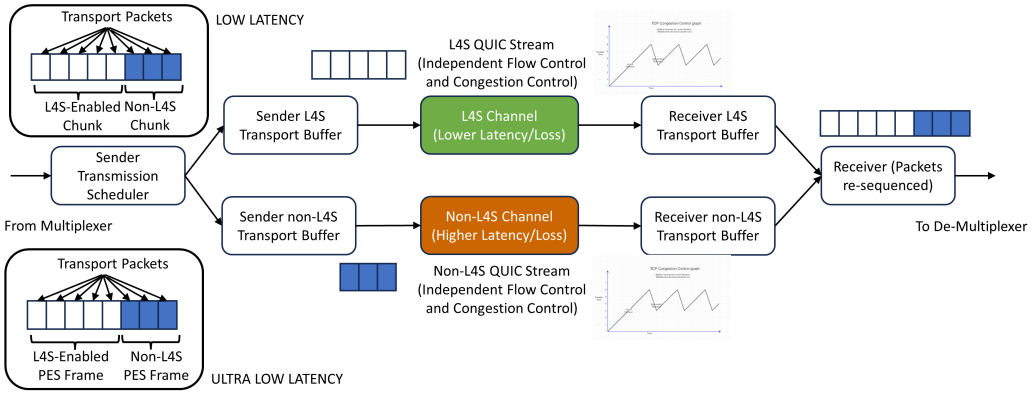

Fig. 7. Handling Out of Order Packet Delivery for Ultra-Low Latency and Low Latency Use Cases

video applications. Our methods intuitively model the low latency queue resource as a buffer that handles the overflow from the classic queue resource during sudden bandwidth increases. We discussed how the PES frame/tile/slice size may be used as a determinant for enabling L4S in ultra-low latency applications. For low latency video, we discussed how the CMAF chunk/fragment size may be used similarly for enabling L4S. Finally, we proposed novel transport layer methods for handling out-of-order packet delivery when selective L4S enablement creates 2 different pathways for a flow.

While the proposed methods have intuitive appeal, further research is needed to validate them. For example, the relative sizing of buffers for default and low latency queuing needs further evaluation. Future work may also investigate the threshold sizes for PES packets or CMAF fragments at which L4S is enabled. Proper adjustment of these controls shall ensure that both the low latency queue and the classic queue can be drained even as instantaneous bandwidth surges occur sporadically.

REFERENCES

[1] B. Briscoe, K. D. Schepper, M. Bagnulo, and G. White, "Low Latency, Low Loss, and Scalable Throughput (L4S) Internet Service: Architecture," RFC 9330, Jan. 2023. [Online]. Available: https://www.rfc-editor.org/info/rfc9330

[2] G. White, K. Sundaresan, and B. Briscoe, "Low Latency DOCSIS: Technology Overview," Feb 2019.

[3] K. D. Schepper, B. Briscoe, and G. White, "Dual-Queue Coupled Active Queue Management (AQM) for Low Latency, Low Loss, and Scalable Throughput (L4S)," RFC 9332, Jan. 2023. [Online]. Available: https://www.rfc-editor.org/info/rfc9332

[4] P. Willars, E. Wittenmark, H. Ronkainen, C. Östberg, I. Johansson, J. Strand, P. Lédl, and D. Schnieders, "Enabling time-critical applications over 5g with rate adaptation," Ericsson-Deutsche Telekom Whitepaper, May 2021.

[5] J. Livingood, "Comcast Kicks Off Industry's First Low Latency DOCSIS Field Trials," Jun. 2023. [Online]. Available: https://corporate.comcast.com/stories/comcast-kicks-off-industrys-first-low-latency-docsis-field-trials

[6] "Testing and Debugging L4S in Your App," Apple Developer Documentation. [Online]. Available: https://developer.apple.com/documentation/network/testing_and_debugging_l4s_in_your_app

[7] B. Briscoe and G. White, "The DOCSIS(r) Queue Protection Algorithm to Preserve Low Latency," Internet Engineering Task Force, Internet-Draft draft-briscoe-docsis-q-protection-07, Nov. 2023, work in Progress. [Online]. Available: https://datatracker.ietf.org/doc/draft-briscoe-docsis-q-protection/07/

[8] M. Carrascosa and B. Bellalta, "Cloud-gaming: Analysis of Google Stadia traffic," *Computer Communications*, vol. 188, pp. 99–116, 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0140366422000810

[9] J. Antunes, "Validating A Very Simple Model for TCP Throughput," Jul. 2013. [Online]. Available: https://www.thousandeyes.com/blog/a-very-simple-model-for-tcp-throughput

[10] L. Curley, K. Pugin, S. Nandakumar, V. Vasiliev, and I. Swett, "Media over QUIC Transport," Internet Engineering Task Force, Internet-Draft draft-ietf-moq-transport-03, Mar. 2024, work in Progress. [Online]. Available: https://datatracker.ietf.org/doc/draft-ietf-moq-transport/03/