IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 28 February 2023, accepted 8 April 2023, date of publication 17 April 2023, date of current version 21 April 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3268008

RESEARCH ARTICLE

Adaptive Partially Reliable Delivery of Immersive Media Over QUIC-HTTP/3

HEMANTH KUMAR RAVURI[1], **(Student Member, IEEE), MARIA TORRES VEGA**[1,2], **(Senior Member, IEEE), JEROEN VAN DER HOOFT**[1], **(Member, IEEE), TIM WAUTERS**[1], **(Member, IEEE), AND FILIP DE TURCK**[1], **(Fellow, IEEE)**
[1]IDLab, Department of Information Technology, Ghent University, 9052 Ghent, Belgium
[2]Dynamical Systems, Signal Processing and Data Analytics (STADIUS), Group T Leuven Campus, Department of Electrical Engineering, KU Leuven, 3000 Leuven, Belgium

Corresponding author: Hemanth Kumar Ravuri (hemanthkumar.ravuri@ugent.be)

This work was supported by ICON Project INTERACT, realized in collaboration with imec, with project support from Flanders Innovation and Entrepreneurship [Vlaanderen Agentschap Innoveren & Ondernemen (VLAIO)]; and in part by FWO WaveVR Project under Grant G034322N. The work of Jeroen van der Hooft was supported by the Research Foundation–Flanders (FWO), under Grant 1281021N.

ABSTRACT The increasing popularity of head-mounted displays (HMD) and depth cameras has encouraged content providers to offer interactive immersive media content over the internet. Traditionally, dynamic adaptive streaming over HTTP (DASH) is the go-to standard for video streaming. However, HTTP is built on top of protocols such as the transmission control protocol (TCP), which prioritize reliability over latency, thereby, inducing additional delay due to acknowledgments and retransmissions, especially on lossy networks. In addition, such reliable protocols suffer from head-of-line (HOL) blocking problem at various levels, leading to playout interruptions of the video streaming application. The third generation of HTTP, i.e., HTTP/3 was recently introduced to deal with the issues posed by TCP. As a major change, HTTP/3 replaces TCP with QUIC at the transport layer, which solves the HOL problem at the transport layer. Moreover, the datagram extension of QUIC allows for unreliable data delivery, just like UDP, which could substantially reduce the latency. Combining this feature of QUIC with the quality adaptation capabilities of DASH-based streaming could bring interactive immersive media delivery to the next level. This work proposes the integration of DASH with the concept of partial reliability of QUIC to reduce playout interruptions and increase the quality of the delivered immersive content on lossy networks. Here, the DASH scheme takes quality and prioritization decisions based on the changing network conditions and the user's viewport, respectively. Then, the part of the content with the highest priority, i.e., within the viewport, is delivered reliably and the rest unreliably. To the best of our knowledge, this is the first work to combine adaptive streaming with partial reliability. Herein, we provide an implementation of a headless player which supports HTTP/3 over partially reliable QUIC as well as state-of-the-art protocols like HTTP/3 over reliable QUIC and HTTP/2 over TCP. We performed an extensive evaluation of our proposed solution using real-world 5G throughput traces and bursty packet loss conditions, using point cloud streaming as the use case. Firstly, our evaluation shows that HTTP/2 is highly intolerant to loss and not suitable for streaming immersive media. Furthermore, even at a loss as high as 5%, the partially reliable framework achieves 46% higher throughput and delivers the content with 33% fewer playout interruptions compared to the reliable counterpart. Since current point cloud decoders are sensitive to loss, we applied the forward error correction mechanism to the data sent unreliably to ensure that the client decodes the content at a probability of 99.9%. Applying this overhead to our solution provides a significant gain of 25% in the throughput compared to the state of the art.

INDEX TERMS QUIC, TCP, HTTP/3, point cloud, video streaming.

The associate editor coordinating the review of this manuscript and approving it for publication was Feng Lin.

I. INTRODUCTION

Recent advances in capture and display technology have led to immersive media, such as 360-degree video, Virtual

Reality and holographic content [1]. Immersive media is expected to enable a plethora of opportunities to support interactive application domains, such as immersive training, immersive surgery, or multi-user interactive gaming [1]. For the sake of interactivity and quality, these application domains impose stringent requirements on the network in terms of high bandwidth (between 100 Gbps and 1 Tbps) and ultra-low latency (down to a Motion-To-Photon latency of 20 milliseconds [2]).

Networking architectures are undergoing a transformation to cope with such requirements, where efforts are being made to offer solutions on several layers of the network stack. There are a range of solutions offered on the network layer, such as software-defined networking (SDN) [3], service chaining [4], and network slicing [5]. However, while such solutions offer resources on the network layer, they are not optimized to deal with the dynamic packet loss ratios experienced by future wireless networks. For instance, millimeter wave (mmWave) communications, which are considered as one of the cornerstones to next-generation wireless communication architectures [2], suffer from packet loss due to their low penetration power. This loss presents a challenge to the transport layer protocols and applications running on top of them [6].

Nowadays, video streaming applications using dynamic adaptive streaming over hypertext transfer protocol (HTTP) (DASH) rely on the transmission control protocol (TCP) for the sake of reliability. However, under fluctuating lossy conditions, TCP suffers from issues like head-of-line (HOL) blocking, slow connection setup times, and delay due to retransmissions [7]. In the context of video streaming, this translates to startup delay, service delay, lower throughput, and playout interruptions. However, delivering content using unreliable protocols such as user datagram protocol (UDP) is not desirable on lossy networks, as the quality of the content is sacrificed at the cost of latency. The third generation of HTTP, i.e., HTTP/3, has recently been introduced to deal with the problems posed by TCP [8]. As a major change, HTTP/3 replaces TCP with QUIC at the transport layer. QUIC, recently standardized, is a multiplexed protocol built on top of UDP. While QUIC is expected to solve several problems of TCP, like HOL blocking at the transport layer, it still has HOL blocking at the stream level [9]. This becomes pertinent for video streaming when different streams carry interdependent data. Very few works have addressed this issue by means of deadline-based scheduling techniques [10], [11]. However, their reactive approach is not suitable for interactive immersive media streaming, as the action is only taken after the deadline has passed, leading to delays. The aim of this work is to deliver immersive media content to the user at the highest possible quality and the lowest number of playout freezes on networks impaired with packet loss.

In our previous work, we presented a proof of concept of a partially reliable variant of QUIC (QPR) [7]. This work offered preliminary insights into the functioning of the protocol using a static scenario, both in terms of content and networking conditions. Inspired and encouraged by these early results, herein, we propose a framework that integrates dynamic adaptive streaming over HTTP (DASH) with the concept of partial reliability on QUIC. This brings the best of both worlds, i.e. reliability offered by TCP and real-timeliness of UDP. Moreover, our framework is able to adapt to the changing network conditions and user's field of view (FoV) and take quality and priority decisions. Since the human perceived vision is restricted peripherally to 120 degrees, we argue that it is sufficient to send the data in the FoV reliably and the rest unreliably. Thus, based on the priority, the content of the chosen quality can be delivered reliably within the FoV or unreliably outside of it.

The following are the main contributions of this paper:

- First, we present our envisioned DASH-enabled partially reliable QUIC framework for immersive media streaming. Then, we provide insights into the functioning of the framework like connection establishment, data transfer etc. Furthermore, provide a means to map the semantics of DASH with QPR. In the envisioned framework, DASH supports bitrate adaptation and is complemented with viewport prediction mechanisms. To the best of our knowledge, this is the first work that integrates the concept of partial reliability with adaptive streaming.
- Next, we evaluate the proposed framework using a client-server-based implementation, where the client is a headless player capable of requesting and playing out the volumetric video (point clouds) adaptively. Here, we provide algorithms for the functioning of the client and the server. Along with the proposed mechanism, the implementation supports HTTP/2 (H2) and HTTP/3 (H3) which rely on reliable QUIC. To the best of our knowledge, this is the first work to evaluate the performance of H3 and DASH for volumetric media streaming. To this end, we perform an extensive experimental evaluation using real-world 5G throughput traces. Furthermore, to evaluate the limits of the mechanism, we offer a means to vary the degree of reliability in the protocol.
- Moreover, the evaluation involves emulation of packet loss similar to a wireless network scenario, using the Gilbert-Elliot model. We evaluate the performance against state-of-the-art H2 and H3 which rely on reliable QUIC.
- Currently, volumetric media such as point clouds use MPEG's video point cloud compression (V-PCC) [12] to achieve a higher compression ratio. However, it is not loss-resilient and crashes while decoding content under packet loss. To mitigate this issue, we have added forward error correction (FEC) to the data being sent unreliably to ensure the client decodes the content at a probability of 99.9%.

Our evaluation shows that, even at a loss ratio of 5%, as observed in extreme cases of 5G wireless networks [13], the proposed framework achieves 46% higher throughput, is capable of delivering the content with 33% fewer playout interruptions and 50% less freeze duration compared to

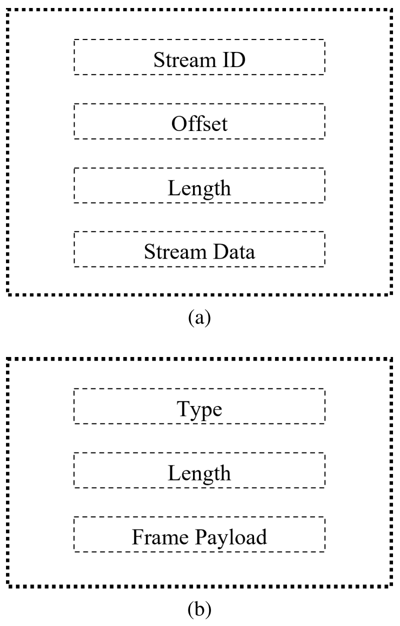


FIGURE 1. Frame structures, (a) QUIC STREAM frame [14], (b) HTTP/3 STREAM frame [8].

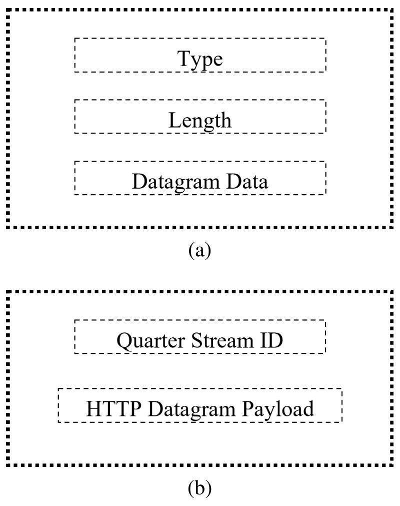


FIGURE 2. Frame structures, (a) QUIC DATAGRAM frame [15], (b) HTTP/3 DATAGRAM frame [16].

reliable QUIC. Furthermore, even with an additional overhead due to the FEC, the proposed framework achieved more than 20% higher throughput over reliable QUIC. In the considered scenarios, the results imply that the proposed partially reliable framework is a better candidate for handling the network loss while adaptively streaming immersive media at a much higher quality compared to the state-of-the-art transport layer protocols. By delivering the content with fewer playout interruptions, it is expected to improve the user's Quality of Experience (QoE). Furthermore, the framework does not limit itself to immersive media delivery, but could support other application domains with the requirement of prioritized data delivery on lossy networks.

The remainder of this paper is organized as follows. A brief background about QUIC and an overview of the related work are presented in Section II. Our envisioned DASH-enabled partially reliable QUIC framework and its functioning, are presented in Section III, followed by the experimental setup and results in Section IV. Finally, this paper is concluded in Section V.

II. BACKGROUND

This section provides a brief background of QUIC followed by insights regarding HOL blocking problem and immersive media streaming using DASH over HTTP/3. Furthermore, it describes the related work and positions this paper with respect to the state-of-the-art.

A. QUIC

In essence, QUIC, recently standardized, can be seen as multiplexed TCP with integrated TLS built on top of UDP. Thus, it brings the best of both worlds together, i.e., it provides the same guarantees as TCP such as reliable delivery and flow control, while also integrating security and stream multiplexing into its design. The basic unit of communication between endpoints is the QUIC packet. These packets contain QUIC frames, which carry control information and application data between endpoints. Furthermore, QUIC packets are carried in UDP datagrams to enable hassle-free integration into present networking systems. Application protocols running on top of QUIC exchange information over QUIC streams, which are ordered sequences of bytes. QUIC offers several benefits over TCP. It reduces the connection establishment latency, and it resolves the HOL blocking problem at the transport level. This is detailed in the next subsection. Unlike TCP, which runs in kernel space, QUIC runs in the user-space. This provides developers with an opportunity to experiment and configure the protocol without waiting for kernel upgrades. Furthermore, QUIC does not come with default congestion and flow control algorithms. Instead, it lets the user plugin their own congestion and flow control strategies based on the application. Currently, most QUIC implementations use TCP's congestion and control flow mechanisms by default. QUIC's loss recovery builds on the TCP's experience and uses most of its recovery mechanisms. It further simplifies the recovery mechanism by introducing monotonically increasing packet numbers to differentiate between new and retransmitted packets [14].

As already introduced, QUIC incorporates the concept of streams at the transport layer. Frames are the fundamental unit of communication, which contain ordered stream data and are encapsulated in QUIC packets. In the context of this work, the most relevant frame types are the STREAM frame and the DATAGRAM frame. The STREAM frame is responsible for creating and carrying the data stream [14]. The frame's stream ID field (Figure 1a) allocates a unique identifier to the stream. Furthermore, it has an offset field, which specifies the byte offset in the stream for the data in that particular frame. The length field specifies the length of the data carried by the frame, and the stream data field carries the data that should be delivered. To enhance the transmission, HTTP/3 also facilitates streams at the application layer and frames form a fundamental unit of communication.

Figure 1b presents the general structure of the HTTP/3 frame [8]. Even though HTTP/3 has several frame types, the most important are HEADERS and DATA frames, as they form the basis for HTTP requests and responses. Each request-response transaction forms a bidirectional HTTP/3 stream, which is abstracted to a unique QUIC stream and is identified using its stream ID. Streams are independent of each other, so if one stream is blocked or suffers packet loss, this does not prevent progress on other streams. When HTTP fields and data are sent over QUIC, the transport layer handles most of the stream management. The transport layer buffers and orders received stream data, exposing a reliable byte stream to the application.

An application can communicate unreliably by sending a datagram over a QUIC connection. To this end, it generates a DATAGRAM frame (Figure 2a [15]) and sends it in the first available packet. Like STREAM frames, DATAGRAM frames contain application data, but they lack an identification parameter analogous to the stream ID. Hence, it is important for the application to differentiate between specific DATAGRAM frames by allotting an identifier for the logical flow of datagrams. Accordingly, HTTP/3 introduces its own DATAGRAM frame at the application layer, which has an identifier field called the quarter stream ID [16]. All HTTP/3 datagrams associated with a request can be allotted to the specific stream and managed by the transport layer. It should be noted that the DATAGRAM frames are neither retransmitted upon loss nor does the transport provide any explicit flow control signaling. If the receiver is unable to provide the necessary resources to process the frames, it may simply drop them. However, it is important for the application to decide the type of data that should be sent unreliably without impacting its overall performance.

B. HTTP AND HOL BLOCKING

Since its inception in 1997, HTTP/1 has become the de-facto standard for web communication, as it provided significant performance optimizations compared to its predecessors and transformed the manner in which requests and responses were exchanged between clients and servers [17]. HTTP/1 works on top of TCP at the transport layer for reliable transmission of data. With time, it has, however, proven to be suboptimal for web communication, due to the exponential growth in internet users and the appearance of new application domains. Among others, HTTP/1 suffers from slow response time due to HOL blocking, or lack of request pipelining. HOL blocking refers to the scenario when a packet belonging to a particular request is lost, and all other packets of the same or different requests in the queue are blocked till the lost packet is retransmitted. This leads to high service delay, which is highly detrimental to the application's performance, especially, for those with a low latency requirement.

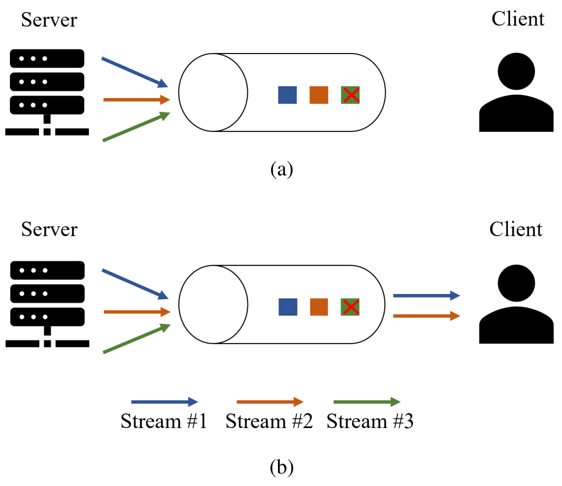


FIGURE 3. HOL blocking as encountered by different protocols, (a) TCP, (b) QUIC.

As a step-up, HTTP/2 introduced the concept of a stream, i.e., a logical flow of byte-data that comes with a unique identifier belonging to a particular request. Furthermore, the multiplexing feature of HTTP/2 enables parallel transmission of requests under one connection. This feature is expected to resolve HOL blocking as the frames belonging to different streams are multiplexed. Thereby, it lets an application process new requests without waiting for previous ones to complete. However, this approach did not yield the expected results. As the underlying transport layer (TCP) lacks knowledge of upper-layer protocols, it assumes that all the data belongs to a single opaque stream. Moreover, since HTTP/2 uses a single TCP connection if a packet belonging to one of the streams is lost, the subsequent packets of the whole connection cannot be processed till the lost packet is retransmitted [18].

HTTP/3, the latest version of HTTP maintains the semantics as its predecessors, but comes with a major change on the transport layer by replacing TCP with QUIC. Unlike TCP, QUIC introduces streams on the transport layer, thereby making it aware of the streams being multiplexed on the application layer. The streams on the transport layer can be mapped to those on the application layer, which lets the transport layer process streams individually [7].

Figure 3 illustrates HOL blocking as encountered by TCP (Figure 3a) and QUIC (Figure 3b). A client-server pair communicates using three streams over TCP and a packet belonging to stream 3 is lost. Then, ensuing packets belonging to all the streams are blocked until the lost packet is retransmitted. This leaves the client idle for a certain time interval without any data to process, leading to disruptions to the application service running on top. In the context of video streaming, such interruptions are called freezing events, and they are highly detrimental to a user's quality of experience (QoE). In the case of QUIC, on the other hand, streams 1 and 2 can still be processed while stream 3 is blocked till the lost packet is retransmitted. Thus, while QUIC solves HOL blocking problem to a certain extent, the issue persists at the stream level. This can be challenging when the data being carried by different streams is interdependent.

So far, research has come up with solutions based on data schedulers at various levels of the networking stack to solve HOL blocking problem in TCP [18]. However, QUIC has been envisioned as a solution at the protocol level. Thus far, works have mainly focussed on utilizing the multiplexing feature offered by QUIC to transmit unrelated data on independent streams to avoid the HOL blocking problem [19]. As suggested earlier, the problem still persists while carrying interdependent data. Not many works have addressed this issue so far. Cui et al. have proposed DASH+, a modified DASH algorithm that supports the multiplexing transmission of QUIC. DASH+ allows clients to request multiple video segments on separate streams using the same QUIC connection. Since the segments are interdependent, the authors have defined a deadline for each segment, after which the playback may be affected. Whenever a stream experiences congestion and the deadline of the segment being carried by the stream is passed, all the resources of the connection are provided to that stream to avoid playback interruptions. However, the solution may not be scalable as it only deals with one congested stream [10]. Recently, in their work, Sander et al. have evaluated the impact of resource prioritization on HOL blocking problem in H3. Resource prioritization has been used to schedule the data transmission on various streams. They found that parallel strategies are only helpful with respect to HOL blocking for high random loss and low BDP scenarios, such that enough streams are active to bridge intra-stream HOL blocking [11]. Moreover, such scheduling techniques are not suitable for immersive media delivery due to the need for interactivity. Our work takes a different approach by incorporating prioritization with the concept of partially reliable data delivery. the following subsection provides an overview of related work in the domain of partially reliable delivery and positions our work with respect to the state-of-the-art.

C. QUIC AND PARTIALLY RELIABLE DELIVERY

As mentioned previously, QUIC still has HOL blocking at the stream level, and this can be challenging when the data on different streams is interdependent. Moreover, the presence of reliability at the transport layer proves to be a performance bottleneck, especially for applications with low-latency requirements. In our previous work, we have shown that not all applications require complete reliability at the transport layer [7]. This subsection provides an overview of related work in the domain of partially reliable data delivery.

So far, only a few works have appeared on deploying partially reliable data delivery for video streaming. The concept of different frames in a video having distinct priorities was exploited by Palmer et al. [20]. Their extension to QUIC, named ClipStream, sends I-frames and end-of-stream markers reliably, while P-frames and B-frames are sent on unreliable streams. These unreliable streams perform opportunistic transmission, sending new data instead of retransmitting the lost data. Recently, they have integrated their system with VOXEL, an adaptive algorithm that trades off frame losses for optimizing a user's QoE. Depending upon the network conditions, VOXEL drops different frames with minimal impact on QoE. However, their approach adds a lot of overhead at the application layer, as it uses an additional step to mark each frame as reliable and unreliable, and another step at the receiver to deal with the out-of-order delivery of frames. Such overhead is not desirable for interactive applications with low-latency and high-bandwidth requirements.

Vivian et al. proposed QUICsilver, a partially reliable QUIC implementation that performs selective retransmissions based on deadline awareness [21]. QUICsilver allows a client to skip lost packets if the playback deadline is approaching for previously received packets with a higher packet number. However, this approach is not scalable, as the sender should keep track of playback deadlines for all the clients requesting the content. Moreover, their approach has proved to be useful only for links with latency lower than 66 ms. Beyond this, QUICsilver behaves as a completely unreliable delivery mechanism.

Recently, Michel et al. [22] took a different approach in realizing a partially reliable data delivery mechanism using QUIC. They designed a variant of FEC called flexible erasure correction and integrated it with QUIC. This enables an application to select the level of reliability, ranging from completely reliable (using retransmissions) and completely unreliable (using various FEC mechanisms to recover the lost data). Through experimental analysis, they showed that under lossy conditions, their approach outperformed the reliable variant of QUIC in terms of download time. However, while transferring files with size higher than 1 GB, their approach added substantial overhead, leading to drastic decrease in the throughput and very high CPU utilization.

As mentioned in Section I, in [7] we have introduced the partially reliable transport layer framework. The datagram extension to the QUIC protocol was deployed to transfer data in parallel, using reliable streams and unreliable datagram flows on a single connection. We evaluated the performance of the proposed approach using two use cases, namely tiled 360-degree video streaming and point-cloud-based volumetric video streaming. Since it is preliminary work in this direction, the framework was mostly left static, i.e. the quality of the content was fixed along with the user's viewport. Moreover, as part of the evaluation, the network parameters were left static and evaluation was limited to the download time of the segments. The results have shown that the proposed partially reliable QUIC protocol (QPR), had lower download times compared to the state-of-the-art HTTP/2 and reliable QUIC under static networking conditions. Compared to the state-of-the-art, our work takes a different approach by considering the user's FoV to decide which part of the video needs to be delivered reliably. Our approach does not add any additional overhead, since we leverage QUIC's datagram extension to realize unreliable data delivery. The outcome of this work has inspired us to apply QPR for dynamic networking scenarios, and in the context of video streaming, this led us to integrate QPR with DASH. To the best of our knowledge, this is the first work to provide a means to integrate the concept of partial reliability with adaptive streaming. The following subsection provides related work in the context of DASH using HTTP/3 and positions our work with respect to the state-of-the-art.

D. DASH-BASED IMMERSIVE MEDIA STREAMING USING HTTP/3

With the advent of content delivery networks, DASH has been a de-facto standard for continuous video streaming with dynamic networking conditions and heterogeneous devices [23]. Primarily, the DASH framework involves encoding the content at different bitrates and delivering it adaptively to the client based on the available bandwidth. To this end, DASH relies on HTTP at the application layer. Thus, it is heavily impacted by the problems described previously, i.e., HOL blocking.

Research suggests that TCP-based solutions prioritize quality over latency, thereby, inducing additional delay due to acknowledgements and retransmissions [5]. Even though quality is a crucial factor for immersive media, latency is of the utmost importance to accomplish the Motion-to-Photon limit, therefore ensuring interactive transmission and avoiding motion sickness. However, in the context of deploying DASH for immersive media delivery, most of the recent works employ either HTTP/1 or HTTP/2 [24]. So far, DASH-based immersive media delivery over HTTP/3 has remained largely unexplored. Cao et al. [25] evaluated the performance of several protocols for mobile augmented reality applications. As part of their work, they compared TCP and QUIC and showed that QUIC with a lower client-to-server latency, is suitable for 5G scenarios while streaming 4K videos. However, the work is limited to the transport layer and does not involve integrating DASH with HTTP/3. Nguyen et al. [26] compared the performance of adaptive bit rate algorithms while deploying HTTP/3 and HTTP/2. They experimentally showed that HTTP/3 performs better than HTTP/2 in lossy networks. However, their work is largely limited to regular videos with large buffer sizes. Guillen et al. [27] proposed SAND/3, an SDN-based QoE control method for DASH over HTTP/3. Using Google QUIC, an early implementation of the protocol, they were able to reduce the quality shifts by 40% and minimize the video interruptions compared to the state-of-the-art mechanisms. However, their work did not involve an implementation of HTTP/3 but used an emulation of the protocol because it was unavailable. Moreover, the work involves streaming a 2D video as well. To the best of our knowledge, this is the first work which involves DASH-based immersive media streaming using HTTP/3.

In our previous work [7], we provided an early take on the concept of QPR with evaluation only in a static space. The content quality was fixed and the underlying network had high enough bandwidth. Furthermore, the chosen loss pattern was not close to real-world wireless network conditions, which are more bursty. However, the results showed promising. Thus, encouraged by those early results, this work presents a fully integrated QPR framework enhanced with DASH. By providing bitrate adaptation and viewport traces, this work makes the scenario more dynamic and closer to the real-world. Furthermore, the evaluation is more complete, and it involves networking conditions closer to wireless networks in the form of 5G throughput traces and bursty loss using the Gilbert-Elliot model. Finally, this work explores the effects of a varying degree of reliability on the performance of the protocol and offers a theoretical solution to retrieve the content using FEC.

III. METHODOLOGY

This section describes the functionality of the proposed adaptive partially reliable framework for immersive media delivery, which consists of the partially reliable QUIC and the DASH integration. Figure 4 presents our envisioned end-to-end architecture with point cloud content delivery as a use case. To this end, the server stores multiple point cloud objects encoded at different qualities and segmented temporally into segments. The client employs a viewport mechanism to predict the user's FoV in the next segment. This enables the client to assign different priorities to the point cloud objects. Furthermore, the client's adaptive streaming mechanism assigns qualities to the point cloud objects based on their priority and over all bandwidth consumed by the previous segment. Once the qualities are fixed, the client can request the content from the server. Since the client deploys partially reliable QUIC (QPR), it is capable of requesting a certain part of the content to be delivered unreliably. Since the objects are already assigned a certain priority, the ones in the FoV (with the highest priority) will be requested reliably and the rest unreliably. In the presence of loss, reliable streams experience HOL blocking and add to the delay. Since the proposed mechanism delivers only the most important content reliably, the experienced delay is considerably reduced. Furthermore, the content delivered unreliably does not experience any HOL blocking, as it is indifferent to loss. Thus, the overall time taken to download and reconstruct the scene is lower compared to that of reliable delivery, thereby avoiding potential playout freezes.

The client deploys HTTP/3 on the application layer, where its semantics are mapped to QPR on the transport layer (Figure 5). HTTP/3 relies on QPR to provide confidentiality and integrity, protection of data, peer authentication, and reliable/unreliable, orderly, per-stream delivery. The following subsections describe each of these functionalities of the framework in detail.

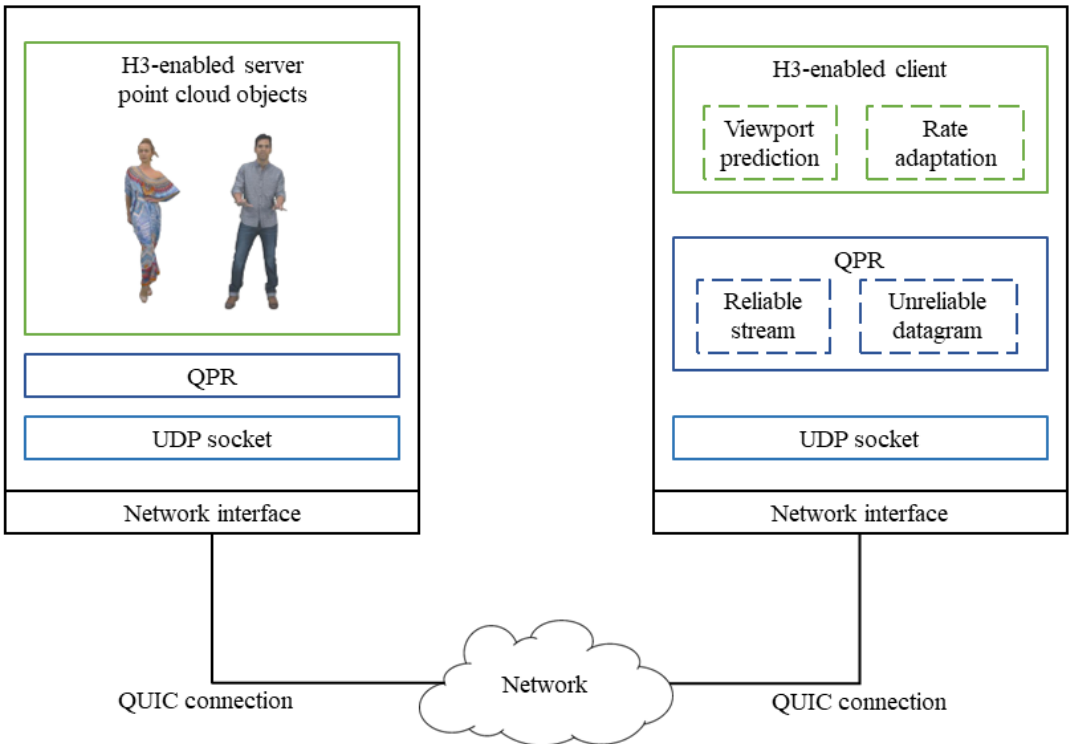


FIGURE 4. Adaptive partially reliable immersive media delivery.

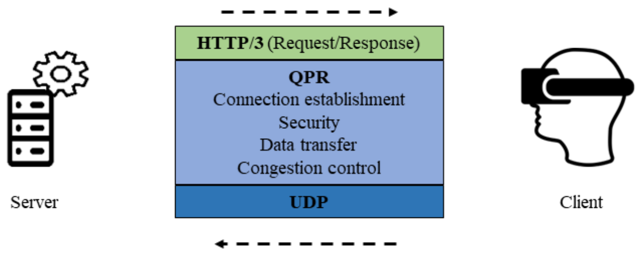


FIGURE 5. Stack diagram of the proposed framework along with functionalities of each layer. In the figure, H3 denotes HTTP/3.

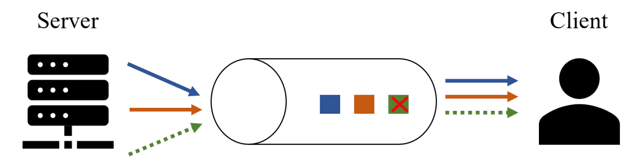


FIGURE 6. QPR handling the HOL blocking problem.

A. MAPPING SEMANTICS

The frame types introduced in II form the basis for QPR which lets the client decide the priority of the data and provides an application programming interface (API) to send reliable or unreliable requests accordingly. The reliable requests are handled using the STREAM frames and the unreliable requests are handled using the DATAGRAM frames. As observed in regular HTTP semantics, methods involved in handling a request-response transaction, such as GET, POST, RESPONSE, BODY etc. [17], are mapped onto the DATA frame on the application layer. For the sake of unreliable delivery, two methods called REQUEST and RESPONSE are defined and mapped onto the DATAGRAM frame on the HTTP/3 layer. As the name suggests, the REQUEST method lets an end-point send unreliable requests and the RESPONSE method lets the other respond unreliably. As mentioned earlier, the request-response pair uses a unique quarter stream ID and can co-exist with other reliable/unreliable streams. Since the data sent on an unreliable stream is not retransmitted, such streams will never experience HOL blocking. As illustrated in Figure 6, a packet loss event on the unreliable stream will never interrupt the transmission process.

B. CONNECTION ESTABLISHMENT AND SECURITY

As in the case of regular HTTP/3, the establishment of a connection between the client and the server starts at the transport layer. As illustrated in Figure 7, the client sends the Initial packet, which includes a TLS1.3 *ClientHello* message. The server responds by sending a TLS1.3 *ServerHello* message. This step negotiates the connection IDs for the new connection along with the exchange of keys as part of the TLS1.3 handshake. This establishes a shared secret to protect the confidentiality and authenticity of future packets. Next, the server sends a Handshake packet with transport layer parameters including the rest of the TLS server messages

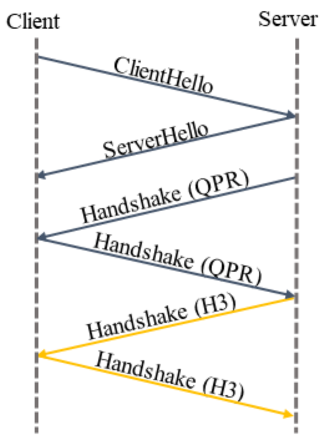


FIGURE 7. Connection establishment. In the figure, H3 denotes HTTP/3.

(in particular the messages related to server authentication). The handshake ends with a reply message that contains the client's settings. In this step, the TLS cipher group (e.g., X25519) is negotiated. Furthermore, this step also involves the negotiation of the application layer protocol negotiation (ALPN), which is HTTP/3 in this case. Table 1 presents the list of the transport layer parameters exchanged.

Unlike the regular QUIC handshake, QPR's handshake involves the exchange of an additional transport layer parameter called max_datagram_frame_size. This parameter represents the maximum size of an unreliable DATAGRAM frame (including the frame type, length, and payload) the endpoint is willing to receive, in bytes. The default value for this parameter is 0, which indicates that the endpoint does not support DATAGRAM frames. A value greater than 0 indicates that the endpoint supports the unreliable delivery and is willing to receive such frames on this connection. This ends the handshake process on the transport layer.

Next, the server initiates the handshake at the application layer by sending the HTTP/3 settings frame on the control stream. While connection level options related to the core QUIC protocol are set in the initial handshake, the HTTP/3-specific settings are conveyed in the SETTINGS frame. It should be noted that each endpoint must send a SETTINGS frame as the initial frame of their respective HTTP/3 control stream. In the current case, the *HTTP/3_dgram_enabled* setting is set to true to let the application know that both endpoints support unreliable data delivery. Additionally, settings such as the length of the datagram queue are also exchanged, to let both endpoints know the amount of data that can be sent unreliably. This concludes the connection establishment process, allowing the endpoints to communicate with each other.

C. DATA TRANSFER AND CONGESTION CONTROL

The client initiates communication by sending an HTTP/3 GET request to the server. To realize partial reliability, we envision the co-existence of reliable streams and unreliable datagrams in one connection. Based on the negotiated parameters and settings, the client can send either a reliable or unreliable request using the same connection.

Figure 8, presents a way for the client to integrate QPR with DASH. Firstly, the client verifies its playout queue for content. Since there will be no content initially, it subsequently checks the buffer. Next, it evaluates the status of the connection, i.e., whether there is a connection established with any server. If not, the client initiates the connection to the server. Once the connection is established, it checks the download queue to generate requests. For the first segment, the download queue is empty, and hence the client requests the content at the lowest available quality for all the point cloud objects. Here, the client's viewport prediction module predicts the viewer's FoV and prioritizes objects based on several conditions (e.g. visible area of the object, and the distance of an object from the viewer). Based on the priority, the client can populate the download queue with a combination of reliable and unreliable requests. As the download progresses, the buffer gets filled, after which the content is sent to the queue to enable play out. At this point, the client's rate adaptation module uses the observed throughput for the previous segment and the predicted viewport information for the next segment to allot different qualities to the objects using a bitrate adaptation algorithm. Next, priority-based ranking is used to make a decision on the reliability aspect. Finally, the player populates the download queue to request the content for the next segment.

Figure 9 presents the request handling on the server side. The server runs in two states, that is, reading and writing. Once the connection is established, the server runs in the reading state, waiting for requests. Upon receiving a request, the server filters it according to the type of request and handles it accordingly. Firstly, a reliable request is processed as follows:

- Upon receiving the GET request, the server responds by sending a response message followed by the body message on a reliable bidirectional stream. These can be identified by the client using the stream ID.
- Next, the server's out_buffer is filled and made available for QPR to generate corresponding QUIC packets that will be sent to the UDP socket for delivery.
- QPR deploys congestion control to determine the rate at which the data is sent to the client by the transport layer.
- Once all data are transferred, the server sends an HTTP/3 *FIN* message to the client to indicate the end of the transfer.

Second, an unreliable request is dealt with in the following manner:

- Upon receiving an unreliable request carried out by a datagram frame, the server starts populating the datagram queue with data chunks of a size determined by the negotiated max_datagram_frame_size parameter.
- Next, the datagram queue is made available for QPR to generate QUIC packets that will be sent to the UDP socket.

TABLE 1. Transport layer parameters exchanged during the handshake (* additional parameter for QPR).

Parameter	Description
max_idle_timeout	Maximum time a connection can be idle
max_udp_payload_size	Maximum size of UDP payload size
initial_max_data	Maximum data that can be allowed for a connection
initial_max_stream_data_bidi_local	Maximum data that can be allowed for a local bidirectional stream
initial_max_stream_data_bidi_remote	Maximum data that can be allowed for a remote bidirectional stream
initial_max_stream_data_uni	Maximum data that can be allowed for a unidirectional stream
initial_max_streams_bidi	Maximum number of bidirectional streams per connection
initial_max_streams_uni	Maximum number of unidirectional streams per connection
max_ack_delay	Maximum allowed delay for acknowledgments
max_datagram_frame_size*	Maximum size of an unreliable datagram frame
cc_algorithm	Congestion control algorithm

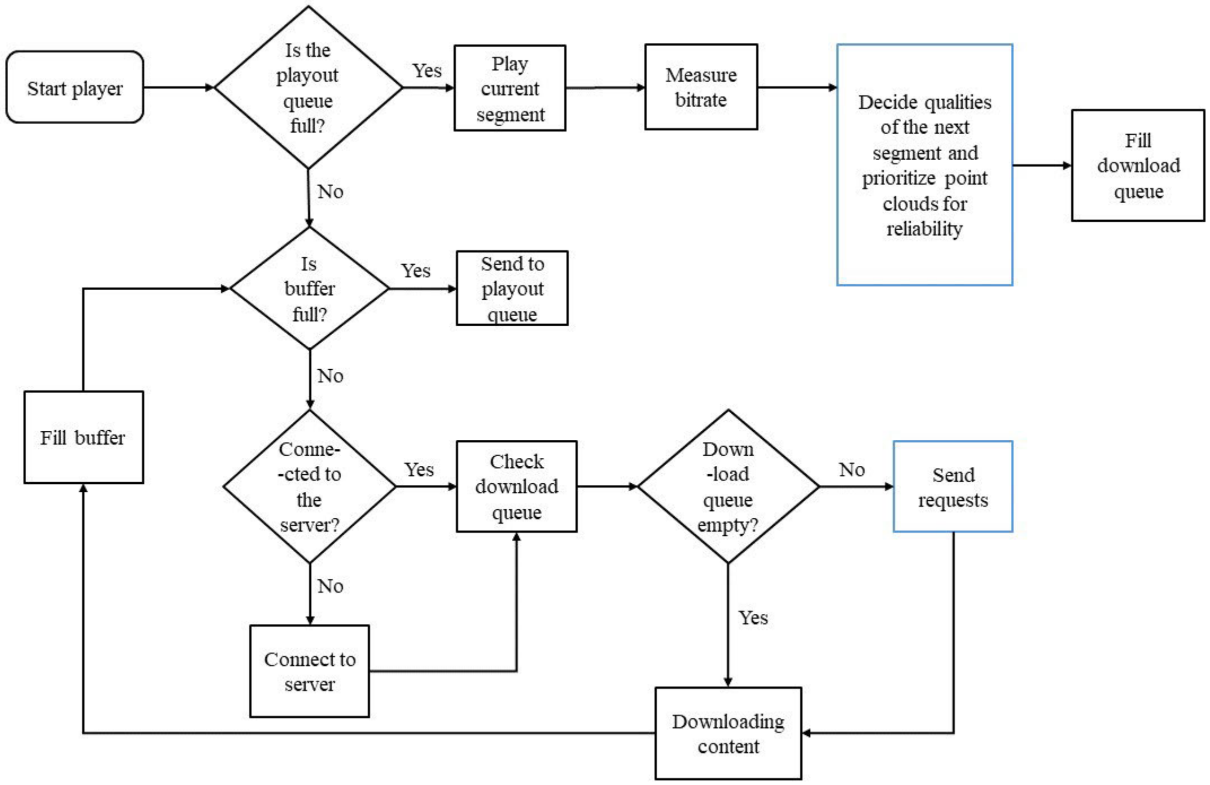


FIGURE 8. Flow chart describing the player algorithm.

- In addition, QPR deploys congestion control on unreliable datagrams, similar to that of reliable streams. However, the acknowledgment handling is slightly different for unreliable datagram frames. While the data, which is sent unreliably, will be transmitted at the rate defined by the congestion window (which is fixed), acknowledgments are only handled after data delivery. This is justified because retransmissions are not necessary in this case. In the case of reliable delivery, the sender waits for acknowledgments to send the next set of packets.
- Since the datagram extension is still a work in progress, there is currently no provision to let the client know when the end of the message is reached. One of the options is to send a standard message on the last datagram. However, there is no guarantee that the message will reach the client. We argue that it is necessary for the client to know the end of the message, even when all the data is lost. This is to move on to the next request, either reliable or unreliable. To this end, the second option is to use the control stream on the HTTP layer to reliably send an end-of-the-data message to the client.

IV. EXPERIMENTAL EVALUATION

To evaluate the proposed approach, we have used an immersive point cloud video streaming use case. Below, we first discuss the evaluation scenario, the implementation, and the evaluation setup, before presenting the obtained results.

A. EVALUATION SCENARIO

Point cloud-based volumetric video streaming allows evaluating the performance of the proposed approach for applications with high-bandwidth and low-latency requirements. Here, the content is delivered on demand, i.e., the content

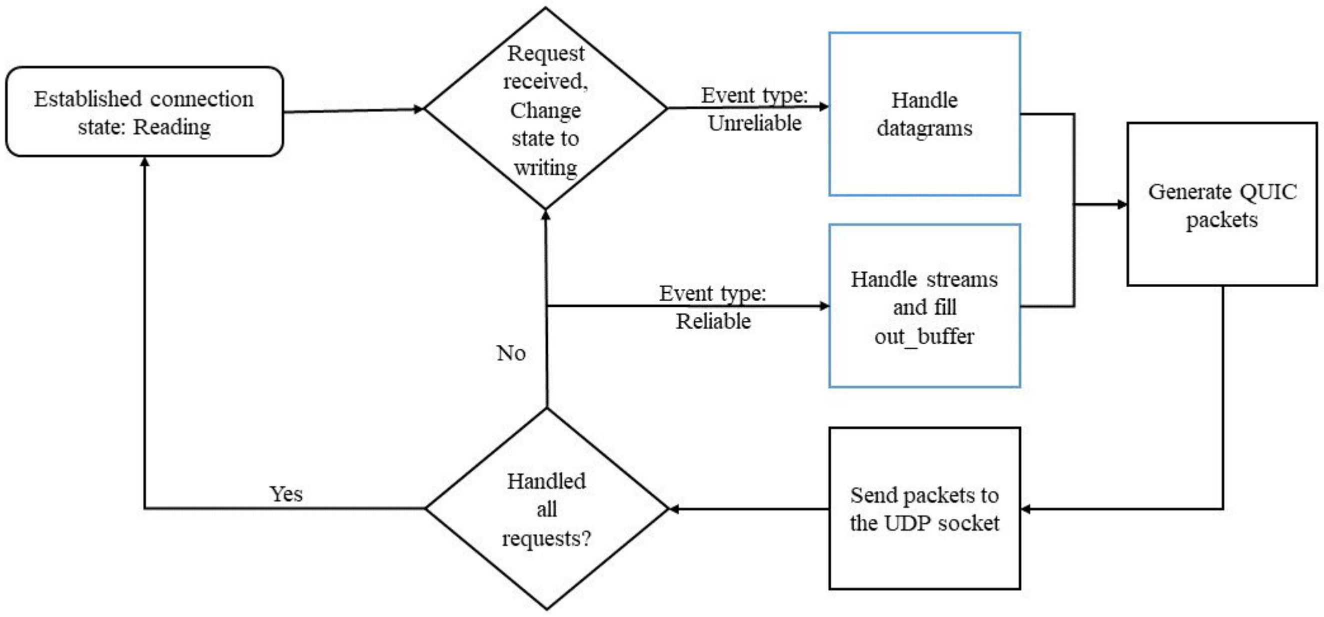

Established connection state: Reading
Request received, Change state to writing
Event type: Unreliable
Handle datagrams
Event type: Reliable
Handle streams and fill out_buffer
Generate QUIC packets
Send packets to the UDP socket
Handled all requests?
No
Yes

FIGURE 9. Flow chart describing the request handling on the server side.

TABLE 2. Observed bitrates (Mb/s) of four point cloud objects.

Quality	#1	#2	#3	#4
1	2.40 ± 0.01	3.50 ± 0.03	4.48 ± 0.01	5.00 ± 0.03
2	3.62 ± 0.01	5.03 ± 0.04	7.14 ± 0.01	8.65 ± 0.05
3	5.81 ± 0.03	7.83 ± 0.05	12.08 ± 0.03	15.86 ± 0.11
4	10.00 ± 0.07	13.63 ± 0.09	21.95 ± 0.07	30.16 ± 0.21
5	18.00 ± 0.13	24.72 ± 0.24	40.35 ± 0.14	53.51 ± 0.37

is pre-captured, pre-encoded, and delivered when the client demands it.

First, the 8i data set was used to generate a volumetric video scene using point clouds [28]. The four objects are first compressed using the MPEG reference V-PCC encoder [12]. Each object is encoded to five qualities resulting in the bit rates presented in Table 2. Since the original dynamic objects only possess 300 frames (or ten seconds of video) the resulting footage is played out 12 times, alternating between forward and backward movement as to smoothly change the location of the objects. This results in a more realistic video length of 120 seconds, or 2 minutes. Furthermore, the sequences are temporally split into one-second-long segments and made them available in the server for the client to request on demand. As suggested in [29], to generate the scene, the four point cloud objects are placed in a circle, looking outwards and the user moves around these objects, always focusing on the center of the circle.

B. IMPLEMENTATION

For the sake of evaluation, we have implemented a headless player, which can request immersive video content from the server and simulate playback for the user, as described in Figure 8. In order to emulate the viewport prediction module, we have provided the player with a trace of the user's FoV. By default, the player requests the objects with top two priority reliably and the other two unreliably. The priority of the objects is determined based on the visible area of the point cloud to the viewer. The distance of the object has been used as a tie-breaker. In addition, the degree of reliability can be configured from completely reliable to completely unreliable. To enable interactivity, the buffer size is fixed to one second, which is equal to the segment duration. After each segment, the player logs different parameters related to the download content, such as throughput, download time, quality, playout interruptions (freezes), and total freeze duration. The client is implemented as such that, should the buffer be empty, playout is stalled; in this case, the user would see the objects standing still rather than moving.

The player is built using Cloudflare's "QUIC, HTTP, ETC" (QUICHE) [30], which is a savory implementation of QUIC and provides support for HTTP/3. Furthermore, we have implemented a simple server as described in Figure 9, using QUICHE. For the sake of comparison, we also added HTTP/2 support for both the player and the server.

Unlike TCP, which runs in the kernel space, each QUIC message, including control messages, triggers a context switch between the kernel and user spaces, leading to additional computational overhead. However, recent efforts like generic segmentation offloading (GSO) for UDP on Linux have enabled applications to bundle and transfer multiple UDP segments between the user space and the kernel at the cost of one. Hence, we have implemented the system after optimizing both QR and QPR using the GSO mechanism. Furthermore, we have increased the receiver-side UDP buffer sizes to 25 MB, to ensure that no data is lost because of the unavailability of the buffer.

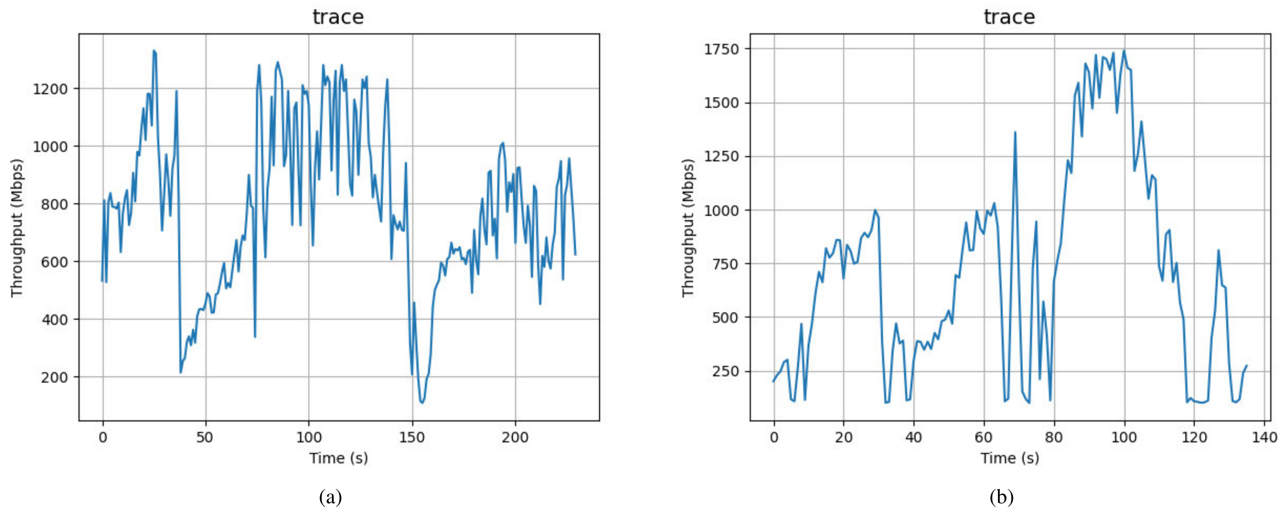


FIGURE 10. **5G throughput traces collected when a user is (a) walking, (b) driving.**

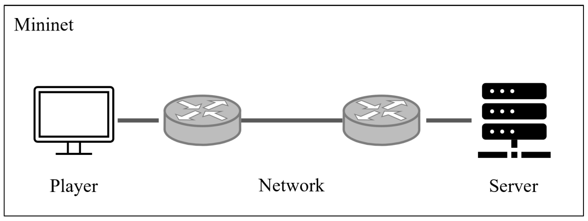


FIGURE 11. **Experimental setup, using mininet to host a virtual network within a node on the virtual wall infrastructure.**

As reported by earlier studies, QUIC's congestion control selects a suboptimal congestion window in the networks with a high bandwidth-delay product (BDP) [31]. This performance can be attributed to QUIC's early exit from the slow-start phase and sluggish congestion-avoidance phase. Since QUIC does not come with a specialized congestion algorithm, most of the current implementations use TCP's congestion control algorithms like Cubic and New Reno [32]. These may not be optimal for QUIC's design when handling high BDP networks. We observed in early experiments that TCP reaches the optimal congestion window for data delivery in tens of milliseconds. Since congestion control mechanisms are out of scope for this work, we have fixed the initial congestion window value for QUIC as BDP where bandwidth is the lowest observed value in the respective throughput trace.

C. EVALUATION SETUP

To evaluate the impact of network impairments on the performance of the proposed approach for streaming immersive media, a network setup is emulated using MiniNet6 [33], where the player and server are implemented on individual virtual hosts (Figure 11). We have deployed traffic control (tc) to vary the link parameters, namely available bandwidth, latency and loss. To vary the available bandwidth, we have used two 5G traffic traces provided by [34]. The first trace (Figure 10a) corresponds to a video streaming session in which the user is walking (trace 1), while the second trace (Figure 10b) corresponds to a session in which the user is driving (trace 2). Next, one-way link delay is fixed at 10 ms, which is a reference value for the observed delay in 5G networks [13]. Finally, we have varied the (one-way) packet loss from 0%-5% using the Gilbert-Elliott model, which causes bursty loss close to that observed in wireless networks [11], [35]. We have chosen loss ratio to be 0%-5% based on [20] and [26]. The player is provided with a user trace in such a way that the viewport is predicted in a clairvoyant manner [29]. We have implemented the Mininet-based setup on a node (Hexacore Intel E5645 (2.4 GHz) CPU and 24 GB of RAM) hosted on imec's Virtual Wall infrastructure [36] as seen in Figure 11.

As mentioned previously, for every experiment, we have logged the throughput, the quality of each object per segment, the number of freezes, and the total freeze duration. Each experiment for a combination of use case, delivery mechanism, and network configuration has been run for ten iterations, which is enough for statistical significance. Even though we have mentioned a few works which deploy the concept of partial reliability in Section II, the comparison with this approach is not possible for several reasons. Firstly, each work has deployed a different flavor of QUIC making it impractical to compare them. For instance, Michel et al. [22] deploy a version called PQUIC which is a Python-based implementation and their work has been designed as a plugin for PQUIC. We have deployed QUICHE, a RUST/C++ based implementation, which is more optimized for performance and supports all the latest draft versions. Next, none of the works are tailored for immersive media streaming. Moreover, only one solution [20] has been developed for video streaming, which is based on QUIC-GO and is not open source. Furthermore, comparable datasets, content, or implementations are not available at places like the DASH industry forum either [37]. Hence, we have compared the performance of the proposed framework against the reliable flavor of QUIC and TCP. To this end, we have integrated DASH with QUIC using HTTP/3 and TCP using HTTP/2. Henceforth, QPR refers to partially reliable QUIC, where half the data (top two ranked point cloud objects for each segment) are sent reliably and the rest unreliably.

D. EVALUATION RESULTS

To provide the state-of-the-art benchmark, we first present the results for the performance of HTTP/2 under varying loss. Figure 12a and Figure 12b show the CDF of the observed throughput of HTTP/2 for varying loss. Sensitivity of HTTP/2 towards loss is evident: the performance deteriorates by an average of 40% when the packet loss ratio is as low as 0.05%. Furthermore, the performance of the system breaks (an average of 10 Mbps) at a loss of 0.5% in both cases, as the observed throughput is almost consistent for any higher loss ratio. Thus, the presence of HOL blocking is detrimental to the performance of systems delivering immersive video. The key takeaway from this result is that HTTP/2 is intolerant towards loss, and it becomes a performance bottleneck while adaptively delivering immersive video.

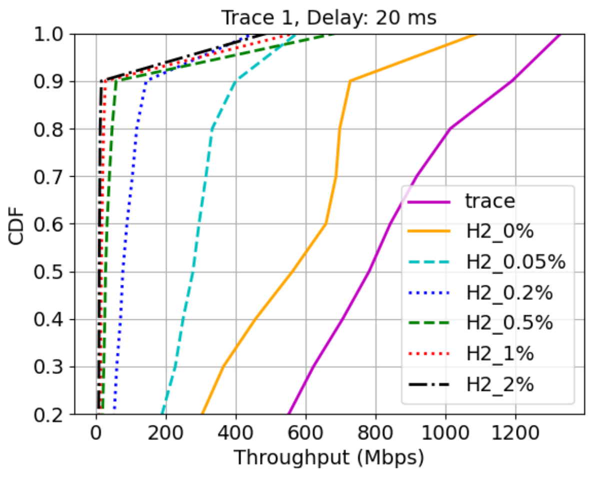
For the following, the performance of reliable QUIC and QPR will be compared with the HTTP/2 case of an ideal loss scenario (0%) and an average packet loss of 2%. Figure 13 presents the performance of HTTP/2, reliable QUIC and QPR when the loss is varied. When the loss is 0%, HTTP/2 performs better than reliable QUIC and QPR in terms of throughput for both trace 1 and trace 2. This can be attributed to the user space implementation of QUIC, which adds additional computational overhead. This is evident when the available bandwidth is high, as the system has to process a higher number of QUIC packets. Moreover, QPR performs better than reliable QUIC at 0% as the acknowledgment handling is slightly different for unreliable datagram frames. While the data, which is sent unreliably, is transmitted at the rate defined by the congestion window, the acknowledgments are only handled after complete data delivery.

In the presence of loss, HTTP/2's throughput performance degrades badly. For instance, the average throughput decreases by a factor of 50 as the loss increases from 0% to 2%. As discussed earlier, this performance shows the impact of HOL blocking, and it only gets worse due to TCP's strict congestion control. Furthermore, in the presence of loss, QPR outperforms reliable QUIC in all cases. When the loss is 2%, for instance, the average throughput of QPR is 50% higher than that of reliable QUIC. Despite being a reliable delivery mechanism, reliable QUIC proved to be more loss tolerant than HTTP/2, since HOL blocking is limited to stream level unlike at the connection level in HTTP/2. In terms of the observed qualities, the deterioration in throughput performance due to loss directly results in lower qualities in HTTP/2. As the loss is increased from 0% to 2%, all four point cloud objects are downloaded at the lowest quality. Furthermore, compared to reliable QUIC, in the case of QPR, four more segments experience higher quality for the object with the highest priority and three more segments for the second-highest priority. This shows QPR delivers better experience to the users than reliable QUIC while delivering immersive video content adaptively. A similar performance has been observed in the case of trace 1 as well.

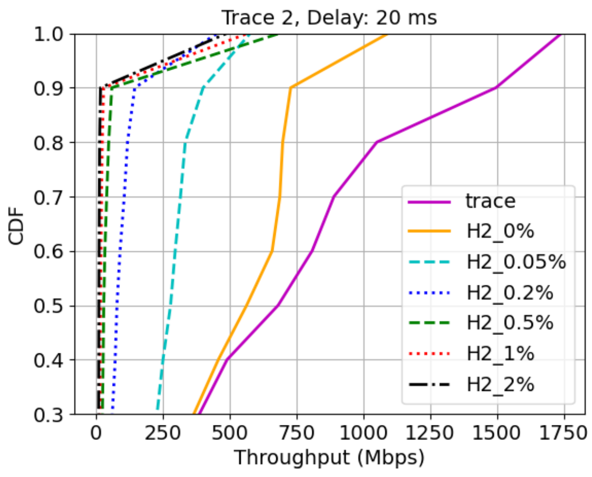
Even though QPR delivers data at higher quality, a certain amount of data is lost. As mentioned in Section II, current immersive media decoders are sensitive to loss and cannot completely decode the content if packets are lost. Hence, it is not yet possible to reconstruct the scene and evaluate the quality using metrics such as PSNR. However, point cloud encoding is still a work in progress and out of scope of this work.

In order to evaluate the performance of the system at higher packet loss, we have increased the loss ratio to 5%. As presented in Figure 14, even at such high loss, QPR has 46% higher average throughput than reliable QUIC. Furthermore, even at 100 times higher loss, QPR has double the throughput compared to HTTP/2. This shows the robustness of the proposed mechanism to packet loss. Furthermore, the two objects in the viewport are delivered at higher quality for 10% more segments in the case of QPR compared to reliable QUIC. However, at a loss higher than 5%, the player deploying reliable QUIC started crashing when there is congestion. This is more pronounced in the case of trace 2, where the congestion is relatively high. This behavior can be attributed to the bursty loss during the congestion period. We have observed that the time-out event occurs by the time lost packets are retransmitted, leading the player to crash. Hence, we have limited our study to at most 5% packet loss.

As mentioned previously, the degree of reliability can be configured on the player. In order to test the limits of the proposed approach, we have increased the unreliability to 100%. Here, we present the results for a case when the unreliability is varied from 0% (Q0) to 100% (Q100). When the unreliability is 0%, all four objects are delivered reliably, and when it is 100%, all four objects are sent unreliably. As shown in Figure 15, the throughput improves with an increase in unreliability. This is true for both the cases in which there is no loss and in which there is 5% loss. For example, at 5% loss, QUIC with 100% unreliability has 125% more throughput than QUIC with 0% unreliability. Figure 16 presents the observed qualities of point cloud objects in the case of trace 2 and for 5% loss. The gain in terms of throughput translates into quality gain. For example, in the case of QUIC with 100% unreliability, the top three prioritized objects are delivered at the highest quality for almost all the segments. Interestingly, the top two prioritized objects are delivered at the highest quality even when the unreliability is 75%. From QUIC with 50% unreliability to QUIC with 75% unreliability, four more segments are delivered at higher quality for the object with priority three. The key takeaway

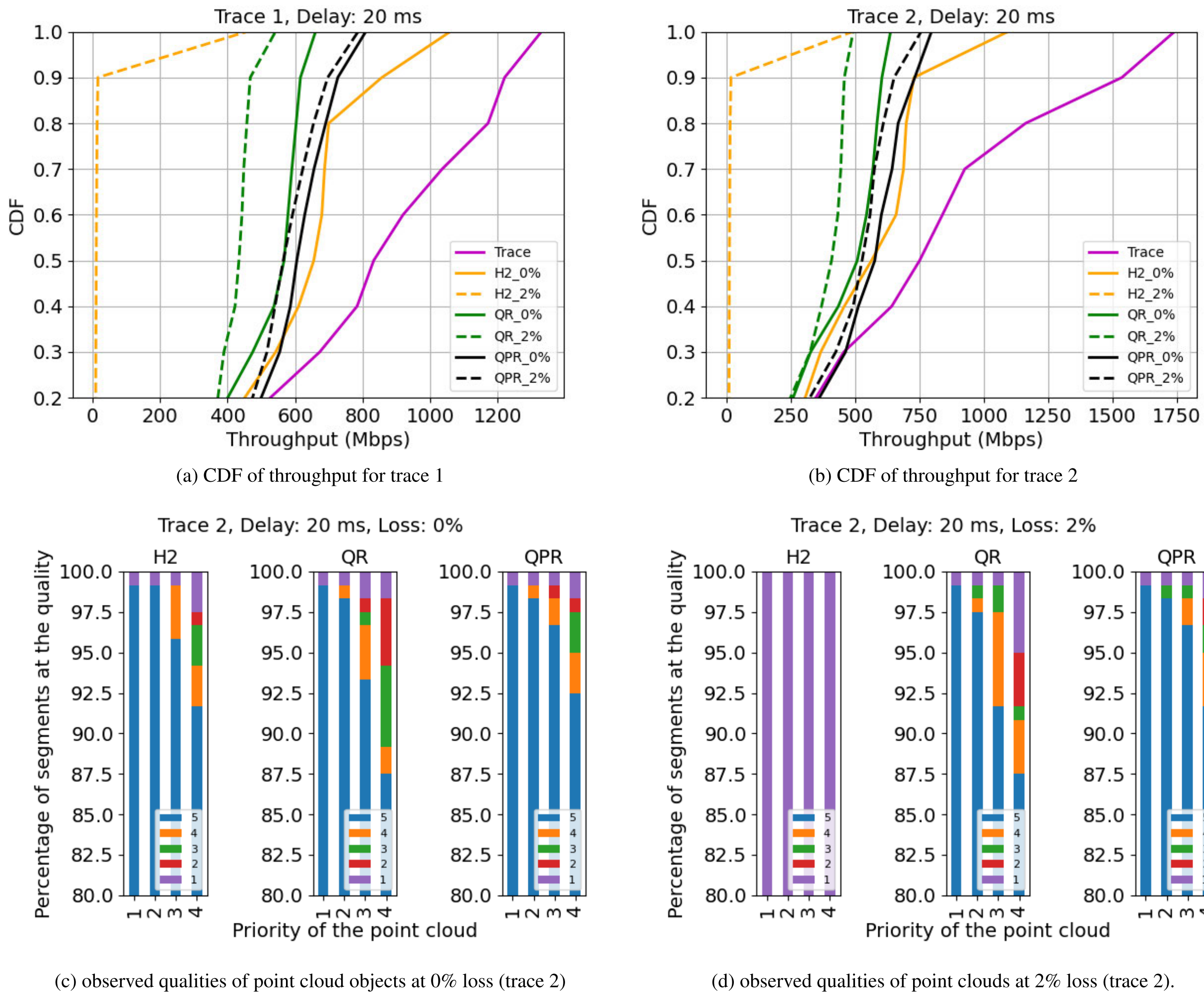


(a) Trace 1



(b) Trace 2

FIGURE 12. **CDF of throughput performance of HTTP/2 for different traces. In the figure, H2 denotes HTTP/2, while the percentage behind it denotes the packet loss ratio.**



(a) CDF of throughput for trace 1

(b) CDF of throughput for trace 2

(c) observed qualities of point cloud objects at 0% loss (trace 2)

(d) observed qualities of point clouds at 2% loss (trace 2).

FIGURE 13. **Performance of HTTP/2 (H2), reliable QUIC (QR), QPR for varying loss.**

from this result is that the increase in unreliability improves the performance in terms of throughput and observed quality. However, as mentioned earlier, the objects that experience loss cannot be reconstructed yet. So, the impact of loss on the observed quality cannot be determined in this work.

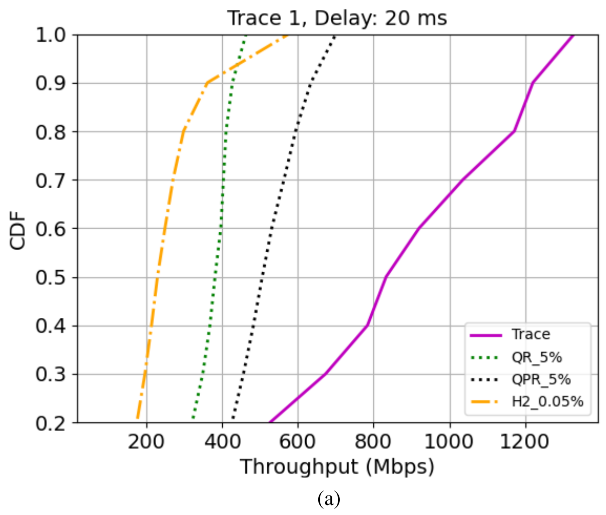


FIGURE 14. CDF of throughput of HTTP/2 (H2), reliable QUIC (QR) and QPR at 5% loss, (a) trace 1, (b) trace 2.

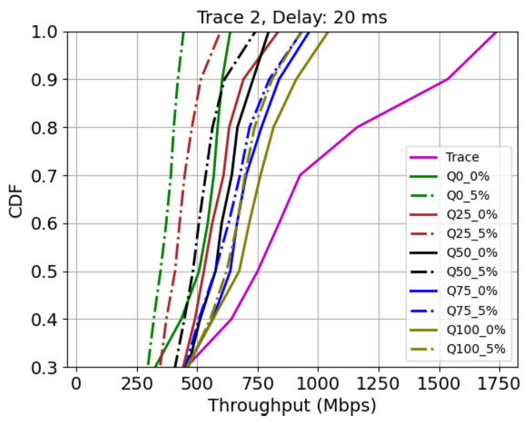


FIGURE 15. Results for variable reliability when trace 2 is applied, CDF of throughput at 0% and 5% loss.

TABLE 3. Observed freezes and total freeze duration (TFD) for trace 2 for HTTP/2 and sub-variants of QPR when unreliability is varied from 0% (Q0) to 100% (Q100).

Protocol	Loss	#Freezes(avg)	TFD (s)
HTTP/2	0%	8±0	2.9±0.2
HTTP/2	2%	105±2.2	60±4.8
Q0	0%	9±0.5	3±0.1
Q0	5%	12±0.2	3.3±0.4
Q50	0%	8±0	2±0.1
Q50	5%	9±0.7	2.2±0.2
Q100	0%	8±0	1.8±0.1
Q100	5%	8±0	1.8±0.1

In addition, important parameters to consider while discussing the quality of a video streaming session are the number of freezes and the total freeze duration. Studies have shown that freezing events in a VR video severely degrade the QoE compared to artifacts in the rendered image (during loss). Furthermore, freezing events affect the visual attention of a user [38]. Hence, we present the observed freezes and freeze duration (Table 3) during the immersive video delivery for various transport layer cases. It should be noted that QUIC does not experience any freezes in the case of trace 1 as the minimum available bandwidth is higher than the required bandwidth. Furthermore, the performance of HTTP/2 is similar to that of trace 2, and so we present only results for trace 2.

In the case of HTTP/2, as the loss increased from 0% to 2%, almost 87% of the segments experience freezes with a total freeze duration of 60 seconds. Given that almost all objects are delivered at the lowest quality, the performance of HTTP/2 is highly undesirable and detrimental to the user's QoE. The performance of Q0 deteriorated with an increase in loss. HOL blocking impacts both the reliable delivery mechanisms, adds to the delay, and becomes a performance bottleneck. However, the impact is lower in reliable QUIC as it is limited to stream level. Furthermore, as the unreliability is increased from 0 to 100%, the number of freezes and total freeze duration reduced considerably. For example, at 5% loss, there are 33% fewer freezes and 50% less freeze duration for QUIC with 100% unreliability compared to QUIC with 0% unreliability. Also, when unreliability is 100% the framework does not experience any additional freezes when the loss is increased from 0% to 5%.

To test the limits of the system, we have increased the loss percentage gradually to 10%. For trace 2 when the loss is 10%, the three variants of QUIC from the unreliability of 0% to 50% have broken down at the first congestion point in the trace (observed at around 6 seconds) as the default timeout of 10 seconds is crossed. QUIC with 75% unreliability recovered after the first congestion point and completed the streaming session with 13 freezes and a total freeze duration of 9 seconds. However, no deterioration is seen in the case of the variant with 100% unreliability, where all the data is sent unreliably.

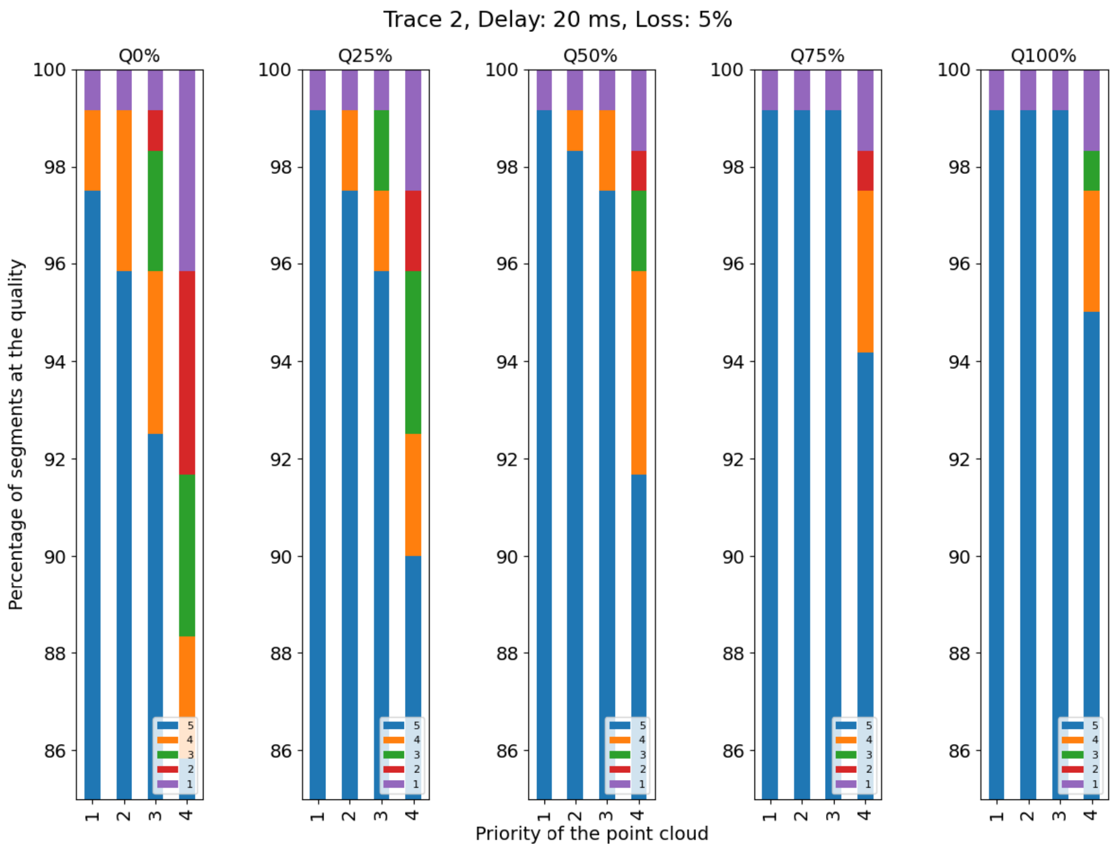


FIGURE 16. Results for variable reliability when trace 2 is applied, observed qualities of point cloud objects at 5% packet loss.

To summarize, our experimental evaluation illustrates that a completely reliable protocol is not suitable to adaptively deliver immersive video on lossy networks. While HTTP/2 is not recommended, reliable QUIC has shown a clear improvement in the presence of loss. However, the proposed partially reliable transport layer for immersive media delivery has proven to be a better candidate. Furthermore, as the percentage of unreliable data delivered increased, the performance of the system improved further. In the context of adaptive delivery, the number of quality switches reduced considerably, along with freezes. However, the current point cloud decoders are sensitive to loss and the content delivered unreliably in lossy conditions cannot be reconstructed, yet.

E. QPR WITH FEC

As hinted previously, one of the shortcomings of the proposed approach is decoding the delivered content under lossy conditions. This becomes important while streaming immersive multimedia like tiled 360-degree videos and volumetric media like point clouds. 360-degree videos are known to use high-efficiency video coding (HEVC) for encoding and tiling purposes. The HEVC decoder is not robust against packet losses, as it crashes if packets are missing [39]. Volumetric media like point clouds currently use MPEG's video point cloud compression (V-PCC) for achieving a higher compression ratio up to 1000 [40]. Even this is not known to be loss-resilient and crashes under packet loss. In the case of HEVC, loss-resilient mechanisms have already been developed using techniques such as FEC [39]. V-PCC is still a work in progress and error-resilient point cloud video encoding, source/channel coding or even error recovery at the user end are still open research issues.

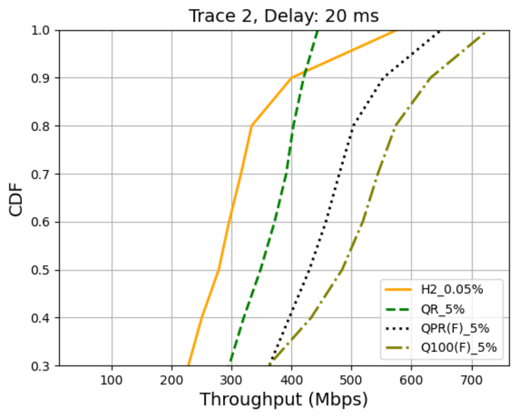


FIGURE 17. CDF of throughput when FEC is applied for unreliable data delivery.

One possible way to mitigate the issues posed by the point cloud decoders would be by integrating the partially reliable data delivery framework with FEC. This offers error protection to the data being sent unreliably over lossy networks and helps the decoders to recover the content at a probability higher than 99.9%. To this end, Zverev et al. [41] have introduced rQUIC, where the QUIC protocol is integrated with a coding module. Using extensive evaluation based on varying network parameters, they have determined the additional overhead needed to achieve an optimal performance in terms of latency while ensuring guaranteed delivery.

This inspired us to theoretically evaluate the performance of QPR while using FEC. Based on their evaluation of additional overhead needed for transferring a file over a link with 25 ms delay and different loss rate, we have calculated the approximate overhead needed for QPR. By integrating this overhead into our evaluation results, we present (Figure 17) a new set of results to compare the performance against HTTP/2 and reliable QUIC. Here, QPR(F) refers to a case where the top two point clouds with the highest priority are sent reliably, and the last two are sent unreliably after encoding using FEC. This process adds an average overhead of 10% per segment. Furthermore, Q100(F) refers to the case where all four objects are sent unreliably after encoding them using FEC which adds an average overhead of 22% per segment. Despite having an additional overhead, the results are in line with the previous cases where the partially reliable framework outperforms the state-of-the-art reliable delivery mechanisms, even at a loss as high as 5%. Moreover, QUIC with 100% unreliability performs the best with almost 20% gain in terms of throughput compared to QUIC with 50% unreliability. In this case, despite being unreliable, the padding given by FEC would let the client recover the data at a probability higher than 99.9% and the additional gain in throughput makes it a viable candidate for future use cases involving immersive media delivery.

However, there is a limitation to this approach. Beyond a loss ratio of 6.5%, the overhead increases beyond 50% which can reduce the throughput gain considerably, thereby reducing the quality of content. In such a scenario, part of the data being sent unreliably may not be protected and hence cannot be recovered, since the decoders are sensitive to packet loss. This leads to a situation of trade-off between the quality and recovery of the content. For instance, as reported previously, the performance of Q100 does not deteriorate even at 10% packet loss. However, taking the current FEC-based recovery approach adds almost 83% overhead, which reduces the quality of the content being sent. Dealing with this tradeoff is out of the scope of the current work, but will be investigated in the future.

V. CONCLUSION

In this paper, we have presented an adaptive partially reliable data framework based on QUIC-hypertext transfer protocol (HTTP)/3 for delivering immersive video. Our work enables the co-existence of reliable streams and unreliable datagrams on the same connection, with the integration of DASH to provide quality adaptation to the immersive multimedia. To the best of our knowledge, this is the first work to integrate adaptive multimedia delivery with the concept of partially reliable delivery. We have evaluated the performance of the proposed approach extensively, using a point-cloud-based volumetric video streaming use case. To this end, we have implemented a headless player which supports different flavors of the proposed framework at the transport layer. Using information on the user's viewport as a means to prioritize the data, we have deployed the proposed framework to deliver content in the viewport at the highest possible quality (based on the available bandwidth) reliably, and the rest unreliably. We have used 5G throughput traces and emulated bursty loss behavior using the Gilbert-Elliot model to bring the evaluation as close as possible to a real-world scenario. Furthermore, we have compared the performance of the proposed approach against two state-of-the-art reliable delivery mechanisms, namely HTTP/2 and reliable QUIC. Results show that under perfect network conditions, HTTP/2 is preferable over the others. However, reliable delivery mechanisms are sensitive to packet loss: while HTTP/2 is highly intolerant to loss, reliable QUIC has shown certain improvement but is still not robust. In the considered scenarios, the proposed framework (QPR) shows better performance than the state-of-the-art approaches in terms of throughput, observed video quality, and freezes. Furthermore, to evaluate the limits of the mechanism, when the degree of reliability is reduced, the performance has shown further improvement, with a completely unreliable system outperforming the others. Finally, we have evaluated the performance of the system when unreliable data is protected using forward error correction (FEC). This allows the client to recover the content delivered unreliably despite the packet loss with a probability of 99.9%. To this end, we have used earlier work to estimate the overhead and applied it theoretically to our results. Despite the overhead, QPR outperformed alternative delivery schemes. Hence, the proposed framework is preferable under impaired network conditions, and the choice of reliability has to be made based on application-specific conditions such as content, priority, and the user's viewport.

The proposed mechanism can be highly useful in future networking scenarios, such as millimeter wave (mmWave) or terahertz communications where scenarios with ultra-high throughput are combined with bursty losses (e.g., when in presence of an obstacle or loss of line of sight) [2]. Our mechanism has proven to tackle such bursty loss in a better way compared to state-of-the-art solutions. In addition, since mmWave communications allow multi-connectivity in the form of multiple links, the proposed mechanism can be deployed on each link with variable reliability based on the network conditions [42]. Furthermore, the framework does not limit itself to immersive media delivery but can support application domains such as the internet of things (IoT), smart cities, and industry 4.0, which come with the requirement of prioritized data delivery, especially on lossy networks [43],

[44]. In addition, developments at IETF to enable deploying an HTTP/3 proxy using the MASQUE protocol can complement the proposed mechanism to reduce the end-to-end delays through content caching. Also, since the mechanism allows for 100% unreliable data delivery which has shown the best performance under high packet loss, selective data protection using FEC can be deployed to deliver the data at lower latency.

As part of future work, we will evaluate the performance of the proposed framework on future wireless networks and optimize it for the scenario. In addition, we will design an application layer mechanism to make a decision on both quality and percentage of reliability and adapt it based on the network conditions. To this end, we will analyze the problem mathematically and develop an optimal model to solve the problem. Furthermore, we will conduct subjective studies once the V-PCC is able to decode the content under loss.

ACKNOWLEDGMENT

Project partners are imec, Rhinox, Pharrowtech, Dekimo, and TEO.

REFERENCES

[1] L. Muñoz-Saavedra, L. Miró-Amarante, and M. Domínguez-Morales, “Augmented and virtual reality evolution and future tendency,” *Appl. Sci.*, vol. 10, no. 1, p. 322, Jan. 2020.

[2] M. T. Vega, C. Liaskos, S. Abadal, E. Papapetrou, A. Jain, B. Mouhouche, and G. Kalem, “Immersive interconnected virtual and augmented reality: A 5G and IoT perspective,” *J. Netw. Syst. Manage.*, vol. 28, no. 4, pp. 796–826, 2020.

[3] H. K. Ravuri, M. T. Vega, J. van der Hooft, T. Wauters, and F. De Turck, “A scalable hierarchically distributed architecture for next-generation applications,” *J. Netw. Syst. Manage.*, vol. 30, no. 1, pp. 1–32, Jan. 2022.

[4] J. Santos, J. van der Hooft, M. T. Vega, T. Wauters, B. Volckaert, and F. De Turck, “Efficient orchestration of service chains in fog computing for immersive media,” in *Proc. 17th Int. Conf. Netw. Service Manage. (CNSM)*, Oct. 2021, pp. 139–145.

[5] A. Clemm, M. T. Vega, H. K. Ravuri, T. Wauters, and F. De Turck, “Toward truly immersive holographic-type communication: Challenges and solutions,” *IEEE Commun. Mag.*, vol. 58, no. 1, pp. 93–99, Jan. 2020.

[6] Y. Ren, W. Yang, X. Zhou, H. Chen, and B. Liu, “A survey on TCP over mmWave,” *Comput. Commun.*, vol. 171, pp. 80–88, Apr. 2021.

[7] H. K. Ravuri, M. T. Vega, J. van der Hooft, T. Wauters, and F. De Turck, “Partially reliable transport layer for quicker interactive immersive media delivery,” in *Proc. 1st Workshop Interact. Extended Reality*, Oct. 2022, pp. 41–49.

[8] M. Bishop. (Feb. 2021). *Hypertext Transfer Protocol Version 3 (HTTP/3)*. [Online]. Available: https://datatracker.ietf.org/doc/html/draft-ietf-quic-http-34

[9] R. Marx, T. De Decker, P. Quax, and W. Lamotte, “Of the utmost importance: Resource prioritization in HTTP/3 over QUIC,” in *Proc. 15th Int. Conf. Web Inf. Syst. Technol.*, 2019, pp. 130–143.

[10] C. Cui, Y. Lu, S. Li, J. Li, and Z. Ruan, “DASH+: Download multiple video segments with stream multiplexing of QUIC,” in *Proc. 10th Int. Conf. Adv. Cloud Big Data (CBD)*, Nov. 2022, pp. 66–72.

[11] C. Sander, I. Kunze, and K. Wehrle, “Analyzing the influence of resource prioritization on HTTP/3 HOL blocking and performance,” in *Proc. Netw. Traffic Meas. Anal. Conf.*, 2022, pp. 1–10.

[12] S. Schwarz, M. Preda, V. Baroncini, M. Budagavi, P. Cesar, P. A. Chou, and R. A. Cohen, “Emerging MPEG standards for point cloud compression,” *IEEE J. Emerg. Sel. Topics Circuits Syst.*, vol. 9, no. 1, pp. 133–148, Mar. 2018.

[13] D. Xu, A. Zhou, X. Zhang, G. Wang, X. Liu, C. An, Y. Shi, L. Liu, and H. Ma, “Understanding operational 5G: A first measurement study on its coverage, performance and energy consumption,” in *Proc. Annu. Conf. ACM Special Interest Group Data Commun. Appl., Technol., Archit., Protocols Comput. Commun.*, Jul. 2020, pp. 479–494.

[14] J. Iyengar and M. Thomson, “RFC 9000: QUIC: A UDP-based multiplexed and secure transport,” Internet Eng. Task Force (IETF), Wilmington, DE, USA, Tech. Rep. ISSN: 2070-1721, 2021.

[15] T. Pauly, E. Kinnear, and D. Schinazi. (Mar. 2022). *An Unreliable Datagram Extension to QUIC*. RFC. [Online]. Available: https://www.rfc-editor.org/info/rfc9221

[16] D. Schinazi and L. Pardue. (Mar. 2022). *HTTP Datagrams and the Capsule Protocol*. [Online]. Available: https://datatracker.ietf.org/doc/html/draft-ietf-masque-h3-datagram-08

[17] R. Corbel, E. Stephan, and N. Omnes, “HTTP/1.1 pipelining vs HTTP2 in-the-clear: Performance comparison,” in *Proc. 13th Int. Conf. New Technol. Distrib. Syst. (NOTERE)*, Jul. 2016, pp. 1–6.

[18] S. Ahmad and M. J. Arshad, “Enhancing fast TCP’s performance using single TCP connection for parallel traffic flows to prevent head-of-line blocking,” *IEEE Access*, vol. 7, pp. 148152–148162, 2019.

[19] H. Zhao, B. He, H. Zhou, J. Zhou, Q. Qi, J. Wang, H. Sun, and J. Liao, “QUIC-enabled data aggregation for short packet communication in mMTC,” in *Proc. IEEE Conf. Comput. Commun. Workshops*, May 2022, pp. 1–2.

[20] M. Palmer, T. Krüger, B. Chandrasekaran, and A. Feldmann, “The QUIC fix for optimal video streaming,” in *Proc. Workshop Evol., Perform., Interoperability QUIC*, Dec. 2018, pp. 43–49.

[21] V. Band, “QUICsilver: Optimising QUIC for use with real-time multimedia traffic,” M.S. thesis, School Comput. Sci., Univ. Glasgow, Glasgow, Scotland, 2019.

[22] F. Michel, A. Cohen, D. Malak, Q. D. Coninck, M. Medard, and O. Bonaventure, “FlEC: Enhancing QUIC with application-tailored reliability mechanisms,” *IEEE/ACM Trans. Netw.*, early access, Aug. 18, 2022, doi: 10.1109/TNET.2022.3195611.

[23] J. Kua, G. Armitage, and P. Branch, “A survey of rate adaptation techniques for dynamic adaptive streaming over HTTP,” *IEEE Commun. Surveys Tuts.*, vol. 19, no. 3, pp. 1842–1866, 3rd Quart., 2017.

[24] A. Yaqoob, T. Bi, and G.-M. Muntean, “A survey on adaptive 360° video streaming: Solutions, challenges and opportunities,” *IEEE Commun. Surveys Tuts.*, vol. 22, no. 4, pp. 2801–2838, 4th Quart., 2020.

[25] J. Cao, X. Su, B. Finley, A. Pauanne, M. Ammar, and P. Hui, “Evaluating multimedia protocols on 5G edge for mobile augmented reality,” in *Proc. 17th Int. Conf. Mobility, Sens. Netw. (MSN)*, Dec. 2021, pp. 199–206.

[26] M. Nguyen, H. Amirpour, C. Timmerer, and H. Hellwagner, “Scalable high efficiency video coding based HTTP adaptive streaming over QUIC,” in *Proc. Workshop Evol., Perform., Interoperability QUIC*, Aug. 2020, pp. 28–34.

[27] L. Guillen, S. Izumi, T. Abe, and T. Suganuma, “SAND/3: SDN-assisted novel QoE control method for dynamic adaptive streaming over HTTP/3,” *Electronics*, vol. 8, no. 8, p. 864, Aug. 2019.

[28] E. d’Eon, B. Harrison, T. Myers, and P. A. Chou, *8I Voxelized Full Bodies—A Voxelized Point Cloud Dataset*, document WG11M40059, 2017.

[29] J. van der Hooft, T. Wauters, F. De Turck, C. Timmerer, and H. Hellwagner, “Towards 6DoF HTTP adaptive streaming through point cloud compression,” in *Proc. 27th ACM Int. Conf. Multimedia*, Oct. 2019, pp. 2405–2413.

[30] R. Marx, J. Herbots, W. Lamotte, and P. Quax, “Same standards, different decisions: A study of QUIC and HTTP/3 implementation diversity,” in *Proc. Workshop Evol., Perform., Interoperability QUIC*, Aug. 2020, pp. 14–20.

[31] J. Manzoor, L. Cerdà-Alabern, R. Sadre, and I. Drago, “On the performance of QUIC over wireless mesh networks,” *J. Netw. Syst. Manage.*, vol. 28, no. 4, pp. 1872–1901, Oct. 2020.

[32] I. Abdeljaouad, H. Rachidi, S. Fernandes, and A. Karmouch, “Performance analysis of modern TCP variants: A comparison of cubic, compound and new Reno,” in *Proc. 25th Biennial Symp. Commun.*, 2010, pp. 80–83.

[33] K. Kaur, J. Singh, and N. S. Ghumman, “Mininet as software defined networking testing platform,” in *Proc. Int. Conf. Commun., Comput. Syst. (ICCCS)*, 2014, pp. 42–139.

[34] A. Narayanan, X. Zhang, R. Zhu, A. Hassan, S. Jin, X. Zhu, X. Zhang, D. Rybkin, Z. Yang, Z. M. Mao, F. Qian, and Z.-L. Zhang, “A variegated look at 5G in the wild: Performance, power, and QoE implications,” in *Proc. ACM SIGCOMM Conf.*, Aug. 2021, pp. 610–625.

[35] G. Haßlinger and O. Hohlfeld, "The Gilbert-Elliott model for packet loss in real time services on the internet," in *Proc. 14th GI/ITG Conf.-Meas., Modeling Evalutation Comput. Commun. Syst.*, 2008, pp. 1–15.

[36] (2022). *IMEC IDLabt ilab.t*. [Online]. Available: https://doc.ilabt.imec.be/ilabt/virtualwall/

[37] *Dash Industry Forum: Catalyzing the Adoption of MPEG*. Accessed: Feb. 27, 2023. [Online]. Available: https://dashif.org/

[38] A. van Kasteren, K. Brunnström, J. Hedlund, and C. Snijders, "Quality of experience of 360 video—Subjective and eye-tracking assessment of encoding and freezing distortions," *Multimedia Tools Appl.*, vol. 81, no. 7, pp. 9771–9802, Mar. 2022.

[39] P. Piñol, M. Martínez-Rach, O. López, and M. P. Malumbres, "Protection of HEVC video delivery in vehicular networks with RaptorQ codes," *Sci. World J.*, vol. 2014, pp. 1–9, Jan. 2014.

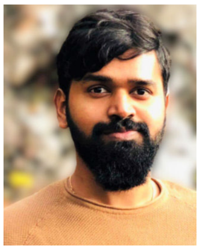
[40] J. Li, C. Zhang, Z. Liu, R. Hong, and H. Hu, "Optimal volumetric video streaming with hybrid saliency based tiling," *IEEE Trans. Multimedia*, early access, Feb. 23, 2022, doi: 10.1109/TMM.2022.3153208.

[41] M. Zverev, P. Garrido, F. Fernandez, J. Bilbao, O. Alay, S. Ferlin, A. Brunstrom, and R. Aguero, "Robust QUIC: Integrating practical coding in a low latency transport protocol," *IEEE Access*, vol. 9, pp. 138225–138244, 2021.

[42] Y. Liu, J. Liu, A. Argyriou, and S. Ci, "MEC-assisted panoramic VR video streaming over millimeter wave mobile networks," *IEEE Trans. Multimedia*, vol. 21, no. 5, pp. 1302–1316, May 2019.

[43] T. Qiu, K. Zheng, M. Han, C. L. P. Chen, and M. Xu, "A data-emergency-aware scheduling scheme for Internet of Things in smart cities," *IEEE Trans. Ind. Informat.*, vol. 14, no. 5, pp. 2042–2051, May 2018.

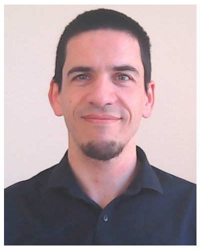
[44] V. Reddy and K. Venkatesh, "Role of software-defined network in industry 4.0," in *Internet of Things for Industry 4.0*. Berlin, Germany: Springer, 2020, pp. 197–218.



HEMANTH KUMAR RAVURI (Student Member, IEEE) received the B.Sc. degree in electronics and communication engineering from Jawaharlal Nehru Technological University, Hyderabad, India, in 2014, and the M.Sc. degree in electrical engineering, with an emphasis on telecommunication systems from BTH, Karlskrona, Sweden, in 2016. He is currently pursuing the Ph.D. degree with the Department of Information Technology, Ghent University—imec. His research interests include networks and service architectures, high precision networks for next generation multimedia delivery, quality of service, and QoE in multimedia systems.

MARIA TORRES VEGA (Senior Member, IEEE) received the M.Sc. degree in telecommunication engineering from the Polytechnic University of Madrid, Spain, in 2009, and the Ph.D. degree from the Eindhoven University of Technology, The Netherlands, in 2017. From 2017 to 2022, she was a Postdoctoral Researcher in multimedia delivery with the Department of Information Technology, Ghent University—imec, Belgium. Since October 2022, she has been a tenure track Assistant Professor of human centric control of immersive systems with KU Leuven, Belgium. Her research interests include quality of experience in immersive multimedia systems and autonomous management of future networks.

JEROEN VAN DER HOOFT (Member, IEEE) received the M.Sc. and Ph.D. degrees in computer science engineering from Ghent University, Belgium, in 2014 and 2019, respectively. He is currently a Postdoctoral Fellow with Ghent University. His studies were supported by a grant provided by FWO. His main research interests include the end-to-end quality of experience optimization in adaptive video streaming and low-latency delivery of immersive video content.



TIM WAUTERS (Member, IEEE) received the M.Sc. and Ph.D. degrees in electro-technical engineering from Ghent University, in 2001 and 2007, respectively. He has been a Postdoctoral Fellow of the FWO-V with the Department of Information Technology (INTEC), Ghent University, where he is currently a Senior Researcher with imec. His work has published in more than 150 scientific publications. His main research interests include design and management of networked services, covering multimedia distribution, cybersecurity, big data, and smart cities.

FILIP DE TURCK (Fellow, IEEE) leads the Network and Service Management Research Group, Ghent University, Belgium, and imec. He has involved in several research projects with industry and academia. He has coauthored over 700 peer-reviewed articles. His research interests include the design of secure and efficient softwarized networks and cloud systems. He was elevated as an IEEE Fellow for technical contributions to adaptive networks and service management. He has served as the Chair for the IEEE Technical Committee on Network Operations and Management (CNOM); and a Steering Committee Member for the IFIP/IEEE IM, IEEE/IFIP NOMS, IEEE/IFIP CNSM, and IEEE NetSoft Conferences. In addition, he served as the Editor-in-Chief for IEEE TRANSACTIONS ON NETWORK AND SERVICE MANAGEMENT (TNSM), for four years.