

Point Cloud Streaming with Latency-Driven Implicit Adaptation using MoQ

Andrew C. Freeman

Baylor University

Waco, Texas, USA

andrew_freeman@baylor.edu

Michael Rudolph

Leibniz University Hannover

Hannover, Germany

michael.rudolph@ikt.uni-hannover.de

Amr Rizk

Leibniz University Hannover

Hannover, Germany

amr.rizk@ikt.uni-hannover.de

Abstract—Point clouds are a promising video representation for next-generation multimedia experiences in virtual and augmented reality. Point clouds are notoriously high-bitrate, however, which limits the feasibility of live streaming systems. Prior methods have adopted traditional HTTP-based protocols for point cloud streaming, but they rely on explicit client-side adaptation to maintain low latency under congestion. In this work, we leverage the delivery timeout feature within the Media Over QUIC protocol to perform implicit server-side adaptation based on an application’s latency target. Through experimentation with several publisher and network configurations, we demonstrate that our system unlocks a unique trade-off on a per-client basis: applications with lower latency requirements will receive lower-quality video, while applications with more relaxed latency requirements will receive higher-quality video.

Index Terms—point cloud, streaming, QUIC, MoQ, MDC

I. INTRODUCTION

As virtual reality (VR) and augmented reality (AR) systems gain traction for consumers, the demand for compelling new multimedia experiences is increasing. For example, users can leverage point cloud cameras to easily capture themselves and their environments in 3D space. These point clouds may be streamed to other users to view with head-mounted displays. However, enormous data rates and encoding complexity make point-cloud streaming difficult.

In this paper, we first examine the existing streaming methods relying on pull-based HTTP protocols and client-side adaptation. We then discuss the in-progress Media Over QUIC (MoQ) protocol specification and its relevant features for server-side adaptation. Finally, we introduce a streaming system with multiple description coding, MoQ transport, and implicit adaptation based on an application’s target latency.

Experiments on a real-world dataset demonstrate that our system unlocks a novel trade-off between latency and quality. Real-time applications such as teleconferencing can maintain ultra low latency by receiving a small subset of the original points, while one-to-many live stream viewers with relaxed latency targets can receive higher quality streams. The underlying adaptation is implicit, as the relay server sends only the data that can be delivered within the application’s latency target. Under a reasonable publisher configuration and moderate network congestion, we observe an average 90.8% increase in throughput and a 0.0024-point improvement in the

PCQM metric when increasing the latency allowance from 50 ms to 500 ms.

II. RELATED WORK

A. Point Cloud Streaming

Inspired by the widespread adoption of HTTP Adaptive Streaming (HAS) for video distribution, early work in point cloud streaming proposed extensions of Dynamic Adaptive Streaming over HTTP (DASH) [1] to point cloud content [2]. Extensions of this framework allow for viewport-dependent [3] or orientation-dependent [4], [5] quality adaptation under a bandwidth constraint to increase user quality based on their relative position to the content. Using QUIC for transport in HTTP/3 based DASH systems for immersive multimedia [6] allows increased throughput and resilience against packet loss. However, these systems were designed for on-demand content consumption, but do not meet the low latency requirements for immersive conferencing.

Promoting a paradigm shift, recent work proposed a Multiple Description Coding (MDC) framework for immersive teleconferencing [7]. Point clouds are divided into non-overlapping partitions, which are independently coded for transmission. In contrast to the quality representations offered in HAS-based systems [2], [3], bandwidth adaptation is carried out by varying the number of representations transmitted. This has similarities to the scalable coding paradigm introduced for traditional video coding [8], which can increase the quality of a base layer representation through a number of enhancement layers. However, scalable coding requires a fixed decoding order, while MDC representations are independently decodable. As a result, MDC representations can be freely combined, yielding a combinatorial number of possible qualities.

Further, point clouds offer a natural fit for deriving non-overlapping segmentation. Related work uses Grid-partitioning [9] to derive a fixed number of representations, which is not perfectly suited for sparse geometries in point clouds. Random Uniform Sampling of points to receive a subgroup [7] allows the system to freely adjust the number of partitions and the number of points per partition.

In this work, we explore the MDC paradigm but seek to use delivery timeouts in QUIC to perform implicit, latency-based quality adaptation.

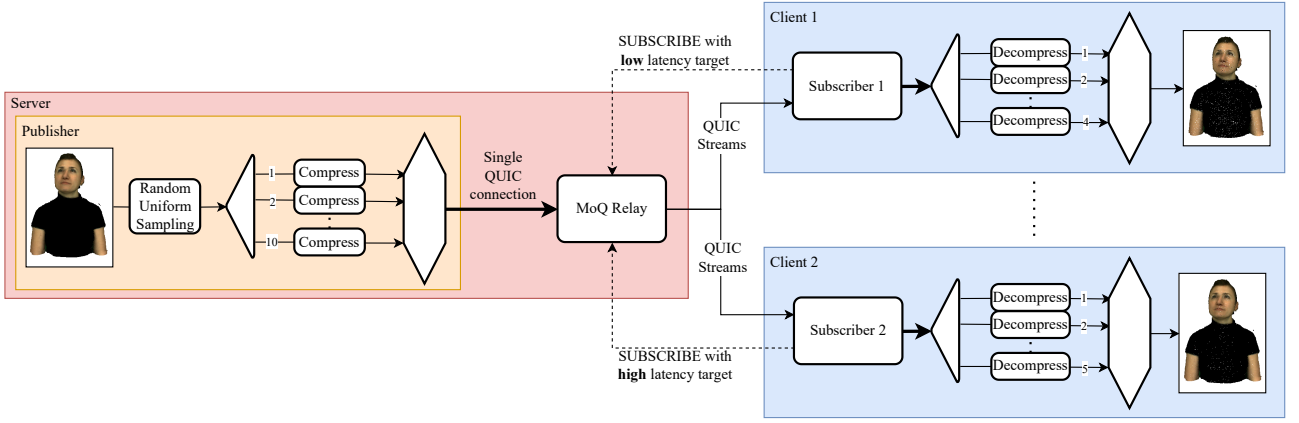


Fig. 1: Overall system diagram with an example from our testing. A client that subscribes with a higher latency target will generally receive more representations than a client with a lower latency target, resulting in a higher-quality frame reconstruction. For a given frame, each sampled representation is sent across a unique QUIC stream.

B. QUIC/Media Over QUIC

In recent years, QUIC has come to constitute much of the everyday Internet traffic that was previously dominated by TCP. The protocol itself is built on UDP, but it is focused on reliable delivery. Compared to TCP, QUIC offers in-built authentication and encryption, enables faster connection initiation, avoids “retransmission ambiguity,” and prevents head-of-line blocking [10]. Although the protocol resides in user-space, this allows QUIC to be updated more frequently and safely than one implemented in the OS kernel. Indeed, much of QUIC’s early development was made possible through large-scale experiments by Google in their Chrome browser [10].

Early on, Google realized the benefits to QUIC-based video transmission in their YouTube product, dramatically reducing the latency for users to begin video playback. Building on these efforts, the Internet Engineering Task Force (IETF) has recently been developing a bespoke multimedia streaming protocol built on QUIC, termed Media Over QUIC (MoQ) [11], [12]. Whereas traditional video streaming protocols such as DASH and HLS rely on HTTP mechanisms for pull-based media delivery, MoQ uses push-based delivery to minimize the end-to-end latency [13].

The core data abstractions within MoQ are Objects, Groups, Subgroups, and Tracks. For most purposes, an Object will map to a single encoded video frame. Multiple Objects constitute a Group, which is meant to be independently decodable, such as a Group of Pictures (GOP) produced through standard video encoding. Groups can be subdivided into Subgroups, which represent related but separately-decodable collections of Objects (e.g., temporal layers in a video). A Track carries all the Groups/Subgroups of a semantic representation, such as a certain-quality video or audio stream. The client can dynamically subscribe or unsubscribe to Tracks as needed to receive the desired content or perform rate adaptation. Each Track publisher will create a new QUIC stream for each Group or Subgroup that it transmits, which helps mitigate head-of-

line blocking issues during congestion [12].

III. METHOD

We propose a low-latency scalable point cloud streaming system built on MoQ. Whereas prior streaming methods perform explicit client-side rate adaptation in the application layer, our approach handles adaptation through the coordination of server-side and QUIC signaling in the transport layer. This allows us to support both low- and high-latency applications without the overhead of explicit bandwidth estimation or application-specific adaptation algorithms. Applications that can tolerate higher latencies will automatically receive more data. Fig. 1 provides an overview of our system.

A. Point Cloud Sampling

Point cloud sampling is the process of selecting a subset of points from a larger point cloud to reduce the data rate, ideally while retaining important visual information. In this early work, we simply use Random Uniform Sampling as proposed by [7] to derive non-overlapping partitions of the point cloud, using a pre-defined number of partitions. Each partition is separately compressed with the Draco encoder [14] and placed into a MoQ Object with a corresponding frame ID. By default, Draco quantizes the coordinates of the points, reducing the positional accuracy to achieve higher lossy compression ratios. However, we found that this quantization yields severe artifacts when our partitions are recombined, as there are regular gaps between groups of points. As such, we disable quantization for our work. Although this lowers the compression rate of each partition, our system can react to limited network bandwidth by sending a subset of these sampled representations. All Objects produced from a given source frame have the same ID, but are published across different MoQ tracks as described below.

B. Session Management

When a publisher begins serving data, it first signals to the MoQ relay the number of Tracks it intends to publish. For

our system, the Track count matches the number of Draco encoders used, such that each sampled representation level is sent across its own Track. These Tracks are assigned QUIC flow control priorities based on their Track IDs, such that lower-ID tracks are more likely to have their data sent during periods of congestion.

When a subscriber joins the session, it issues a `SUBSCRIBE` message for *all* available tracks. Within this message is the `DELIVERY_TIMEOUT` parameter. This conveys the receiver’s maximum tolerable latency for any received Object, given in milliseconds. Our key hypothesis was that an increase in delivery timeout will allow more data to propagate to the receiver, especially under periods of high congestion.

C. Transport Control

The relay forwards the received Objects from the publisher to the subscriber. When the Publisher first creates a new Object, we insert the current system timestamp in the Object’s metadata. Since Objects are served in chunks, we check at each chunk boundary if the delivery timeout has elapsed since the Object was first received at the relay. If the timeout is exceeded, we *close* the underlying QUIC stream associated with that Object’s Group. This is a slight departure from the MoQ draft specification, which dictates that streams be “reset” when delivery timeouts are exceeded [12]. A stream reset issues an error code to the stream receiver. When using WebTransport as our QUIC API wrapper in practice, we found that many stream cancellations or resets in rapid succession would cause the entire connection to be closed.

In our system, however, we do not want the QUIC connection to close during periods of high loss: we simply want to receive as much data as possible within the target latency. When the relay closes a stream due to a delivery timeout, then, we can still detect this at the application layer without a stream error code. When the receiver attempts to read an Object from a closed stream, it encounters a read error. The receiver interprets this error to mean that any remaining Objects within the Group cannot be delivered. By maintaining a consistent Object ID numbering and Group size, we use this as an explicit signal for Object drops.

D. Reconstruction

When an Object is either received or its Group is dropped for some Track, the receiver notifies a dedicated reconstruction thread. Once this thread has received an Object message for every track, it concatenates the Objects’ decoded point arrays. The result is a point cloud with some subset of the original points, which may be saved or displayed as needed. In total, the end-to-end latency, L , of a frame is defined by

$$L = \max_{i \in I} (E_i) + \min(\max_{i \in I} (D_i), T)$$

where I is the set of Objects (sampled representations) for the frame, E_i is the encoding latency for an Object, D_i is the transmission and decoding latency for an Object, and T is the delivery timeout.

IV. EVALUATION

A. Evaluation Setup

To test our system, we select the point cloud sequences from the Microsoft Voxelized Upper Bodies (MVUB) dataset [15]. Each sequence consists of 7-8 seconds of video recorded at 30 frames per second (FPS) with a spatial resolution of 512^3 voxels.

We utilized a test bench consisting of a server with an AMD Ryzen 3700X CPU and a client with an Intel Core Ultra 7 165H CPU. We found that the server could not consistently encode the source point clouds at the 30 FPS rate, so we emulated a point cloud production rate at 20 FPS.

We utilized the `moq-rs` open-source repository as the starting point for our implementation. This repository technically implements the simplified MoQ-Lite draft specification [16], but the mechanisms we employed are held in common between MoQ and MoQ-Lite. We implemented the delivery timeout mechanism and its associated control messages as described in Secs. III-B and III-C. We ran both the publisher and the MoQ relay on our server machine, connected via a network switch on a 1 Gbps link with the client machine.

We tested a variety of network conditions and publisher configurations to evaluate the efficacy of our system. We varied the outgoing bandwidth from the server using the `tc` and `netem` utilities, examining the rate values 300, 600, and 900 Mbps. We ran our publisher with both 5 and 10 encoders and 5 and 30 frames per MoQ Group. We initiated our client with delivery timeout values of 50, 100, 500, and 1500 milliseconds. Offline, we computed the Point Cloud Quality Metric (PCQM) [17] scores for the reconstructed images. Below, we report 1 minus the PCQM score (1-PCQM), and assigned a score of 0 if no representations were received for a given frame. Note that “acceptable” 1-PCQM scores should be > 0.98 , with 1.0 indicating perfect reconstruction [17].

B. Results

We provide the average change in throughput across our various test conditions in Tab. II. A key result is that *throughput generally correlates with the latency allowance*. This is evident at a low bandwidth (300 Mbps) and moderate timeout (500 ms). Importantly, these throughput gains correspond to improvements in the PCQM quality, as shown in Tab. I. In Fig. 2, we illustrate the rate of “stall” scenarios, which we define as frames for which we receive zero representations within the delivery timeout. Although the use of only 5 Tracks yields higher average visual quality under low bandwidth, it also induces severe stall rates at low timeouts. We emphasize that the quality impact of a playback stall is subjective and should be an area of future investigation.

Under the 10-Track configuration, we observe an improvement when shifting from 5 FPG to 30 FPG. Specifically, we see fewer stall events, higher throughput, and better visual quality. For example, with 600 Mbps bandwidth, we observe an average 90.8% increase in throughput and a 0.0024-point improvement in the PCQM metric when increasing the latency

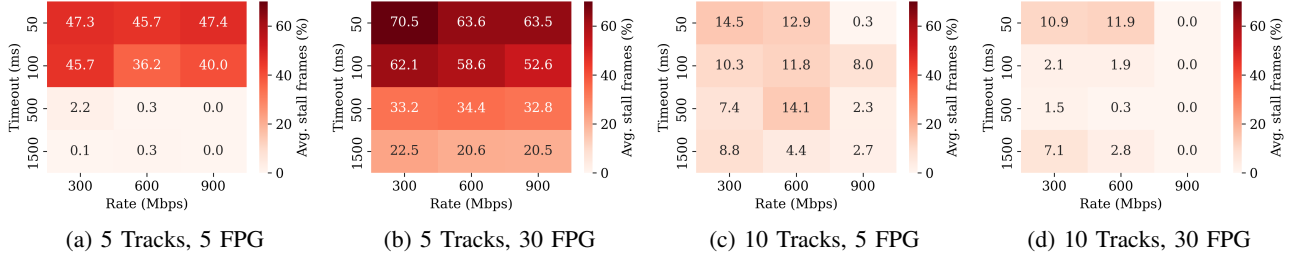


Fig. 2: Frequency of stall scenarios across all test configurations. A frame is counted as stalling if not a single representation is received. “FPG” denotes Frames Per Group.

# Tracks	FPG	300 Mbps				600 Mbps				900 Mbps			
		Timeout (ms)				Timeout (ms)				Timeout (ms)			
		50	100	500	1500	50	100	500	1500	50	100	500	1500
5	5	0.9922	-0.0001	+0.0008	+0.0009	0.9941	-0.0006	+0.0008	+0.0011	0.9945	+0.0004	+0.0038	+0.0046
5	30	0.9925	-0.0001	+0.0002	+0.0009	0.9946	-0.0002	+0.0001	+0.0007	0.9941	+0.0002	+0.0020	+0.0025
10	5	0.9916	+0.0004	+0.0005	+0.0007	0.9953	-0.0009	+0.0007	+0.0009	0.9990	-0.0017	-0.0007	-0.0008
10	30	0.9911	+0.0005	+0.0017	+0.0018	0.9940	+0.0007	+0.0024	+0.0014	0.9981	+0.0003	+0.0009	+0.0008

TABLE I: Average change in 1-PCQM score compared to the baseline 50 ms timeout. Positive changes indicate a quality improvement. Here, we examine only the frames for which we receive at least one representation (i.e., no “stall” frames).

# Tracks	FPG	300 Mbps			600 Mbps			900 Mbps		
		Timeout (ms)			Timeout (ms)			Timeout (ms)		
		100	500	1500	100	500	1500	100	500	1500
5	5	-0.6	105.4	116.1	-1.8	6889.1	6694.4	2.8	23458.0	26226.2
5	30	55.8	360.2	712.2	65.7	258.2	644.5	65.1	936.9	3649.1
10	5	19.7	27.9	36.7	16.5	47.4	52.0	-20.5	-8.3	-8.1
10	30	25.6	74.1	74.8	52.7	90.8	75.2	3.2	9.8	7.9

TABLE II: Average change in throughput (%) compared to the baseline 50 ms timeout.

allowance from 50 ms to 500 ms. This result is somewhat surprising, as we expected that the faster recovery with smaller Groups would yield higher throughput under network congestion. In practice, the larger Group size functions to better mitigate congestion as a Track waits longer to begin sending again once a Group is dropped. That is, we better avoid overwhelming the QUIC congestion controller by not sending less frequent bursts of data.

Whereas low-latency HAS adaptation simply aims to minimize the end-to-end latency, our system unlocks a unique **trade-off between latency and quality**. If users can tolerate higher latency, they will receive more sampled representations and experience higher visual quality.

V. CONCLUSION AND FUTURE WORK

We present a point cloud streaming system with implicit server-side adaptation through the use of MDC and MoQ. The system dynamically selects how many point cloud sample representations to send based on the receiver’s target latency, thereby unlocking a novel trade-off between latency and visual quality. We illustrated that applications with stringent low-latency requirements, such as immersive teleconferencing, can maintain minimal delays by receiving a smaller subset of point cloud representations. Conversely, applications with more relaxed latency targets, like one-to-many live streaming, can benefit from higher-quality streams as more data

propagates to the receiver within the allowable delay. This implicit adaptation, integrated with our point cloud sampling and QUIC-based transport, enhances the viability of diverse VR/AR applications. A tight coupling of point cloud sampling, compression, and streaming with MoQ can unlock more adaptive and high-quality experiences in extended reality systems.

While Random Uniform Sampling offers the low latency necessary for immersive conferencing, we believe that content-adaptive partitioning schemes are a promising direction to derive representations of varying importance. This can be elegantly paired with prioritization mechanisms in MoQ, allowing us to give more precedence to representations deemed more important during transmission. For example, human faces should have a higher priority than their clothing for teleconferencing applications.

Finally, there is significant room to explore the impact of implicit adaptation on the user’s quality of experience (QoE). Our PCQM measurements indicate a predictable latency-distortion trade-off for individual frames, but we do not yet quantify the impact of potentially drastic changes in the received quality or stall scenarios when no representations are received for a given frame. Subjective user studies will be necessary to evaluate and mitigate these factors. From there, we can better determine the optimal encoder and sampling configurations to maximize QoE.

REFERENCES

- [1] I. Sodagar, "The mpeg-dash standard for multimedia streaming over the internet," *IEEE Multimedia*, vol. 18, no. 4, pp. 62–67, 2011.
- [2] M. Hosseini and C. Timmerer, "Dynamic Adaptive Point Cloud Streaming," in *Proceedings of the 23rd Packet Video Workshop*. Amsterdam, Netherlands: ACM, Jun. 2018, pp. 25–30. [Online]. Available: <https://dl.acm.org/doi/10.1145/3210424.3210429>
- [3] J. Van Der Hooft, T. Wauters, F. De Turck, C. Timmerer, and H. Hellwagner, "Towards 6dof http adaptive streaming through point cloud compression," in *Proceedings of the 27th ACM International Conference on Multimedia*, 2019, pp. 2405–2413.
- [4] M. Rudolph and A. Rizk, "View-adaptive streaming of point cloud scenes through combined decomposition and video-based coding," in *Proceedings of the 1st International Workshop on Advances in Point Cloud Compression, Processing and Analysis*, 2022, pp. 41–49.
- [5] S. Subramanyam, I. Viola, A. Hanjalic, and P. Cesar, "User centered adaptive streaming of dynamic point clouds with low complexity tiling," in *Proceedings of the 28th ACM International Conference on Multimedia*, 2020, pp. 3669–3677.
- [6] H. K. Ravuri, M. T. Vega, J. Der Van Hooft, T. Wauters, and F. De Turck, "Adaptive partially reliable delivery of immersive media over quic-http/3," *Ieee Access*, vol. 11, pp. 38 094–38 111, 2023.
- [7] M. De Fré, J. van der Hooft, T. Wauters, and F. De Turck, "Scalable mdc-based volumetric video delivery for real-time one-to-many web rtc conferencing," in *Proceedings of the 15th ACM Multimedia Systems Conference*, ser. MMSys '24. Association for Computing Machinery, 2024, p. 121–131.
- [8] H. Schwarz, D. Marpe, and T. Wiegand, "Overview of the Scalable Video Coding Extension of the H.264/AVC Standard," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 17, no. 9, pp. 1103–1120, Sep. 2007, conference Name: IEEE Transactions on Circuits and Systems for Video Technology. [Online]. Available: <https://ieeexplore.ieee.org/document/4317636>
- [9] A. Chen, S. Mao, Z. Li, M. Xu, H. Zhang, D. Niyato, and Z. Han, "Multiple description coding for point cloud," in *ICC 2024 - IEEE International Conference on Communications*, 2024, pp. 3408–3413.
- [10] A. Langley, A. Riddoch, A. Wilk, A. Vicente, C. Krasic, D. Zhang, F. Yang, F. Kouranov, I. Swett, J. Iyengar, J. Bailey, J. Dorfman, J. Roskind, J. Kulik, P. Westin, R. Tenneti, R. Shade, R. Hamilton, V. Vasiliev, W.-T. Chang, and Z. Shi, "The QUIC Transport Protocol: Design and Internet-Scale Deployment," in *Proceedings of the Conference of the ACM Special Interest Group on Data Communication*, ser. SIGCOMM '17. New York, NY, USA: Association for Computing Machinery, Aug. 2017, pp. 183–196. [Online]. Available: <https://dl.acm.org/doi/10.1145/3098822.3098842>
- [11] Z. Gurel, T. Erkilic Civelek, A. Bodur, S. Bilgin, D. Yenicieri, and A. C. Begen, "Media over QUIC: Initial Testing, Findings and Results," in *Proceedings of the 14th ACM Multimedia Systems Conference*, ser. MMSys '23. New York, NY, USA: Association for Computing Machinery, Jun. 2023, pp. 301–306. [Online]. Available: <https://doi.org/10.1145/3587819.3593937>
- [12] S. Nandakumar, V. Vasiliev, I. Swett, and A. Frindell, "Media over QUIC Transport," Internet Engineering Task Force, Internet Draft draft-ietf-moq-transport-13, Jul. 2025, num Pages: 93. [Online]. Available: <https://datatracker.ietf.org/doc/draft-ietf-moq-transport>
- [13] A. Bentaleb, M. Lim, M. N. Akcay, A. C. Begen, S. Hammoudi, and R. Zimmermann, "Toward One-Second Latency: Evolution of Live Media Streaming," *IEEE Communications Surveys & Tutorials*, pp. 1–1, 2025. [Online]. Available: <https://ieeexplore.ieee.org/document/10943205/>
- [14] "google/draco," May 2025, original-date: 2016-12-05T20:14:02Z. [Online]. Available: <https://github.com/google/draco>
- [15] C. Loop, Q. Cai, S. Escolano, and P. Chou, "Microsoft voxelized upper bodies—a voxelized point cloud dataset, document iso," *IEC JTC1/SC29 Joint WG11/WG1 (MPEG/JPEG)*, m38673 M, vol. 72012, p. 2016, 2016.
- [16] L. Curley, "Media over QUIC - Transfork," Internet Engineering Task Force, Internet Draft draft-lcurley-moq-transfork-03, Jan. 2025, num Pages: 19. [Online]. Available: <https://datatracker.ietf.org/doc/draft-lcurley-moq-transfork>
- [17] G. Meynet, Y. Nehmé, J. Digne, and G. Lavoué, "PCQM: A full-reference quality metric for colored 3D point clouds," in *Twelfth International Conference on Quality of Multimedia Experience (QoMEX)*. IEEE, 2020, pp. 1–6.