Additional Metrics in DTP and QUIC Performance Evaluation

1st Jiayi You
M.S. in Computer Science
Georgetown University
Washington, D.C., USA
jy797@georgetown.edu

2nd Xinyi Huang
M.S. in Computer Science
Georgetown University
Washington, D.C., USA
xh230@georgetown.edu

***Abstract*—In the context of multiplayer games or online video conferencing, timely data transmission is crucial - any delay can severely impact real-time interaction. The UDP protocol is widely used in these scenarios, but it is not ideal due to unreliable data transmission. The QUIC protocol has been extended over UDP to provide more accurate RTT estimates, packet loss detection as well as retransmission, however, QUIC lacks awareness of packet deadlines and still requires applications to be forced to build custom and complex time-sensitive data transfer mechanisms. DTP is a new time-aware transport protocol that enhances QUIC by ensuring that packets meet application-specific deadlines. Its unique active-drop-at-sender scheduling and adaptive redundancy mechanisms enables as many packets as possible to reach the receiver on time and reduces time consumption due to retransmission. However, the evaluations of DTP mainly focus on testing the success rate of small data arriving within the deadline under varying bandwidth and packet loss rates and does not mention the transfer of large data, varying packet error rates and deadline or priority biased cases. Therefore, this report further evaluates DTP's performance in maintaining high integrity and timeliness of valid packet transmission by comparing DTP and QUIC's packet deadline success rates in these areas.**

***Index Terms*—real-time data transmission, QUIC, network performance evaluation**

I. Introductions

With the advancement of the internet, real-time applications such as live video conferencing and multiplayer gaming are increasingly gaining attention. However, these applications have strict requirements for their data to arrive before a specific time (i.e., a deadline) [1]. Failure to meet these deadlines can have different impacts to varying degrees. For instance, automotive engine control systems typically need to respond in milliseconds to control fuel injection, ignition, and emissions systems etc., and delayed signals can cause engine failure or damage [2]. For online video conferencing, the end-to-end latency needs to be maintained below a certain level, which is about 100 millisecondsto guarantee participants to interact smoothly [1]. Given the high risk to human life involved with pacemakers, they require continuous, precise and timely responses in less than a second to maintain the normal rhythm of the heartbeat [2]. In general, packets that fail to meet the deadline are discarded as obsolete and quickly overwritten by later packets, rendering the system unable to run smoothly. Thus, protocols that are sensitive to deadline data transfer are particularly important.

Currently, most real-time applications adopted the User Datagram Protocol (UDP) to transmit data because of its fast speed and low latency. Particularly, UDP does not have reliability mechanisms such as retransmission and acknowledgement (which are instead implemented by Transmission Control Protocol (TCP)) [3]. While these mechanisms ensure reliable delivery, they may introduce latency that is not suitable for real-time applications [3]. Taking real-time games as an example, it is particularly beneficial because these games require a lot of immediate responses, such as the movement of multiple players in the game and the synchronization of these moves with other players [3]. However, the network environment is considerably complex, and precisely because UDP does not guarantee the integrity and order of data, when packet loss, packet error, network congestion and other conditions occur, the experience of the game players is degraded due to not receive information well.

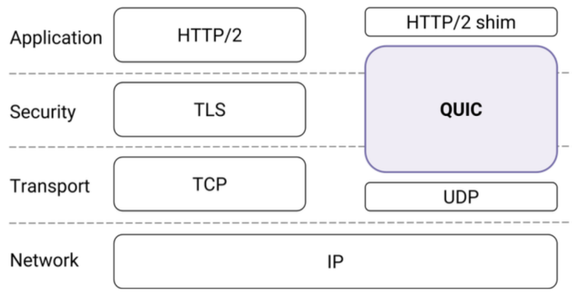

Fig. 1. QUIC within the HTTPS protocol stack. [4]

Indeed, it is significantly challenging to improve the UDP in transport layer as it involves modifying the operating system kernel and deploying it to today's computers widely [4]. Consequently, QUIC (Quick UDP Internet Connections) protocol has been developed by Google [4]. QUIC is a brand-new transport protocol developed based on UDP, which crosses the application layer and the transport layer (as shown in Fig. 1) and combines the advantages of TCP reliable transmissions and UDP low latency [4]. This is mainly reflected in the following aspects:

- **Low connection establishment latency.** In TCP, the purpose of establishing a network connection is to create a stable communication channel between multiple devices. QUIC also provides such connection establishment, however, its client can cache information about the server without subsequent round trips, and the data can be transmitted immediately after the client handshake packet is sent without waiting for a reply from the server, thus greatly reducing connection latency [4].

- **Stream Multiplexing.** QUIC also supports streams multiplexing within the same connection, which greatly reduces latency for sending packet queues and sending blocking [4]. Even if there is a packet loss, this only affects the stream carrying that packet, packets from other streams can still be combined and delivered to the application, which improves the reliability of data transmission [4].
- **Loss Recovery.** QUIC uses a retransmission mechanism to improve reliability. Unlike TCP, it mitigates 'repass' ambiguity and header blocking by using monotonically increased packet counts and an improved acknowledgement mechanism [4]. These techniques indirectly reduce the latency associated with retransmission.
- **Flow control & Congestion Control.** To limit the data buffer size on each stream and the entire connection, QUIC utilizes a connection-level and stream-level flow control strategy, which effectively prevents slow flows from occupying too much buffer and causing queue header congestion. At the same time, this refined flow control allows for faster response and data processing, as it reduces the footprint of a single stream on the entire connection resource, thus speeding up the transfer of data across all streams.

Additionally, QUIC streams have properties specific to real-time applications, including timestamp rendering, priority setting to strike a balance between reliability and latency [5]. It can be found that QUIC could be potentially beneficial to real-time applications. In fact, owing to these outstanding features, QUIC has been actively researched and developed for applications such as live streaming and cloud multiplayer gaming [5].

Nonetheless, Zhang et al. [1] point out that QUIC, while effective in providing accurate RTT, bandwidth estimations, efficient loss detection as well as priority considerations, lacks certain features that are critical to meeting application deadlines. Meanwhile, some applications use complex methods of block compression to reduce bandwidth requirements, which often introduce dependencies between blocks. If a packet dependent data is delayed, its delivery may be meaningless [1]. QUIC does not specifically respect these relationships [1]. Hence, Zhang and his team proposed a Deadline-aware Transport Protocol (DTP), a transmission protocol sensitive to data deadline. DTP extends QUIC by incorporating an 'Active Dropping Sender Scheduler' (Section II), which balances packet deadlines and priorities. It strives to timely transmit high-priority data blocks, selectively discarding less critical ones when necessary [1]. Moreover, DTP builds a deadline-aware adaptive redundancy module based on the forward error correction (FEC) algorithm, aimed at reducing retransmission delays resulting from packet loss by providing redundancy [1]. In the next section, we will provide a detailed overview of these two mechanisms.

II. Motivation

In this section, we first introduce the architecture and the unique transmission mechanisms of DTP. Following this, we discuss the limitations of its evaluation methods and explain our motivation for proposing new evaluation metrics.

A. Architecture

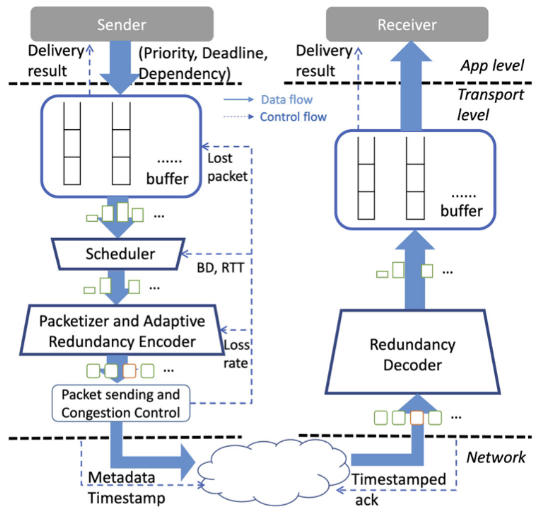

Fig. 2. The architecture of DTP. [1]

The architectrue of DTP is shown in Fig. 2. Initially, at the sending end, the data block and its metadata are stored in the sending queue [1]. Its scheduler module then selects the data blocks to send and discards the outdated ones [1].

Next, the encapsulation and redundancy modules break up the application's data blocks into packet streams [1]. Redundant modules will generate redundant packets to reduce retransmission delays when conditions are met [1].

These packets will be sent to the Packet Sending and Congestion Control module, which sends the packets, collects confirmation information and detects loss, similar to the QUIC/TCP protocol [1]. Lost packets have a higher priority and are placed in front of the corresponding send queue [1].

Finally, when the data stream reaches the receiving end, the transport layer recombines the data into chunks and delivers them to the upper layer [1].

B. Mechanisms I: Mapping from Blocks to Streams

Most applications generate and process data in blocks, and real-time applications require these blocks to meet strict deadlines. QUIC is primarily a stream-based protocol [1]. To adapt QUIC for block transmission, DTP extends its capabilities. DTP maps blocks directly to QUIC streams, allowing block cancellation through QUIC's standard stream cancellation process. For mapping block and stream identifiers, DTP utilizes Equation (1) [1].

$$stream\_id = (block\_id \ll 2) \,|\, stream\_type\_bits \quad (1)$$

C. Mechanisms II: Active-drop-at-sender (ADS) scheduler

The function of the Active-drop-at-sender (ADS) scheduler is to schedule the sending queue to ensure that as many critical data blocks as possible are transmitted before deadlines [1]. It needs to resolve two kinds of conflicts: conflicts between high-priority data blocks that are far from the cut-off time and low-priority data blocks that are near the cut-off time, and conflicts between data blocks that are being transferred and more important data blocks that are waiting to be transferred [1]. For the former, the scheduler chooses to send the low-priority data block, but also guarantees that the high-priority data block reaches the receiver before the deadline [1]. For the latter, the scheduler implements a preemptive scheduling policy. However, if this strategy is too aggressive, frequent switching could cause data sending to crash [1]. So, the scheduler uses the following equations to calculate priority p (the smaller, the higher) for each data block and sends it:

$$\begin{aligned} remain\_time = deadline - \frac{remain\_size}{estimated\_bandwidth} \\ - passed\_time - one\_way\_delay \quad (2) \end{aligned}$$

$$f(remain\_time) = \begin{cases} \frac{remain\_time}{deadline}, & \text{if } remain\_time > 0 \\ \min\left(-\frac{remain\_time}{deadline}, 1\right) + \beta, & \text{otherwise} \end{cases} \quad (3)$$

$$\begin{aligned} Weighted\_p = ((1-\alpha) \times f(remain\_time) \\ + \alpha \times \frac{priority}{p_{max}} \times unsent\_ratio \quad (4) \end{aligned}$$

The remain_time refers to the time remaining before the deadline for sending a packet [1]. The unsent_ratio indicates the proportion of unsent data in a packet [1]. A β value is a weighted priority value added to a block of data that is about to expire. The greater β value, the less likely it is that the data block will be considered [1]. The α value is used to adjust whether DTP prefer to consider block priority or deadline. The smaller the α, the more important the deadline [1].

D. Mechanisms III: Deadline-aware adaptive redundancy

As we known, a packet loss requires four RTTs to be successfully received (3 ACKs and a retransmission), which is time-consuming. Therefore, DTP implements adaptive redundancy for packets with less than two RTTs remaining before the cut-off time [1]. To be specific, this redundant module uses FEC to add n packets for the next m consecutive packets. This $m+n$ packet is a redundant group, and any m packets in it can reconstruct the original m packets. When a receiver accepts a redundant packet, it detects if the packet is lost. If so, the receiver decodes the redundant group and extracts the original packet [1].

E. Limitations on evaluation on DTP and our motivation

In original DTP experiments, to evaluate ADS scheduler, DTP's performance was tested by sending three 200KB blocks with distinct priorities every 100 ms under different bandwidths. To more accurately simulate real-world scenarios such as high-definition video conferencing or large files transfers with higher data loads, we increased the size of the data blocks beyond 200KB (SectionIII-A). Such testing can provide deeper insights into DTP's efficiency and reliability in handling large data blocks under various bandwidth conditions. In addition, the suitability of large and small data blocks differs. For instance, large data blocks may perform better in environments with high bandwidth and low latency, while small data blocks could be more effective in limited bandwidth or unstable network conditions. Adjusting the testing approach in this way can offer significant data support for further optimization and application scenario selection of DTP.

III. Implementation and Evaluation

To align with the test environment described in the DTP paper closely, we rented two virtual machines. One serves as the server and the other as the client. Each virtual machine is equipped with Intel 4vCPUs, 8GB memory, and runs on Ubuntu 20.04.

In the DTP GitHub repository, an example written in *Rust* is provided. In this example, the client sends a request to the server, and upon receiving the request, the server sends a file from a specified path back to the client. The client then receives the file and displays its content. Based on the provided example, we modified the file reading and writing methods to accommodate large file operations [1]. Utilizing this revised example, we then conducted tests to evaluate performance. The following sections introduce our methodology for conducting further tests on the DTP.

A. Send 1Mb blocks with distinct priorities

We divide one 6.4Mb video into six 1Mb blocks and send blocks with distinct priorities under varying bandwidth conditions. The priorities of the first three blocks are 0, 1, 2 separately (0 means highest priority). The server send these three blocks based on their priorities. The priorities of the last three blocks are also 0, 1, 2. These three blocks are sent based on their priorities either. To evaluate the performance of DTP, we also let server send 6.4Mb video to the client by QUIC. Fig. 3 compares DTP and QUIC. Data arrival size records the size of video which the client can receive in the same limited time.

In Fig. 3, we can see that when the bandwidth is larger than 4 Mbps, the performances of QUIC and DTP are the same. The videos can be received by the client completely. When the bandwidth is smaller than 4 Mbps, the performance of DTP is better than that of QUIC. The client can receive more data by DTP than by QUIC. The results of comparison show that in scenarios involving the transfer of large files,

[1]The project can be accessed at GitHub

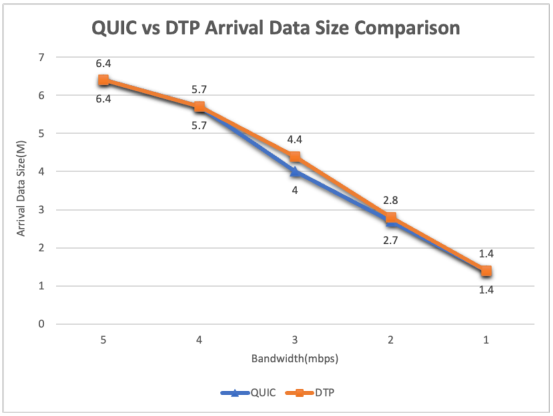

Fig. 3. Various static bandwidth.

where the files are segmented into sizable data blocks, DTP demonstrates superior performance under conditions of limited network resources by holistically considering both deadlines and priorities in its mechanism, outperforming QUIC.

B. Set a range of α

In DTP, α adjusts application specified priority and deadline implicated priority. We set α as 0.2, 0.4, 0.6, 0.8. The server sends one 6Mb video with the same setting of deadline and priority each time. TABLE I shows the performance of DTP under different bandwidths for each α value. The data in the table represents the amount of data received by the client within a specified timeframe.

TABLE I
VARIOUS STATIC α

α\Bandwidth	1 Mbps	3 Mbps	5 Mbps	7 Mbps
0.2	1.2 M	3.3 M	5.2 M	6.4 M
0.4	1.2 M	3.2 M	5.2 M	6.4 M
0.6	1.2 M	3.1 M	5.3 M	6.4 M
0.8	1.1 M	3.2 M	5.2 M	6.4 M

In the TABLE I, we can see that under the same bandwidth and the same setting of deadline and priority, the value of α does not change the performance of DTP significantly. It can show that ADS scheduler has strong robustness, which means the trade-off between priority and deadline has a minimal impact on performance. The result maybe caused by the following 2 reasons:

(1) When we are testing DTP, other factors, such as network latency, bandwidth utilization, or packet loss rate, may have a more significant impact on performance than the scheduling strategy controlled by the α value.

(2) The ADS scheduler's formula does not effectively capture the impact of varying weight allocations between priority and deadline.

C. Modify different package loss rate

To see the performance of adaptive redundancy mechanism of DTP when transferring big file, we compare DTP with QUIC. The server send one 1.4Gb video to the client. We use tc tool to modify different package loss rate. Fig. 4 shows how QUIC performs under different package loss rates. And Fig. 5 shows the performance of DTP. The completion rate is calculated by received data by client / sent data by server.

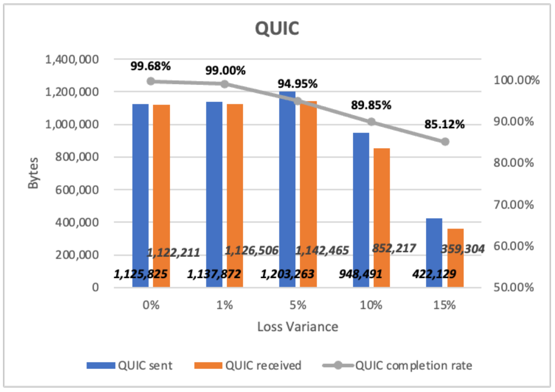

Fig. 4. Various package loss rate of QUIC.

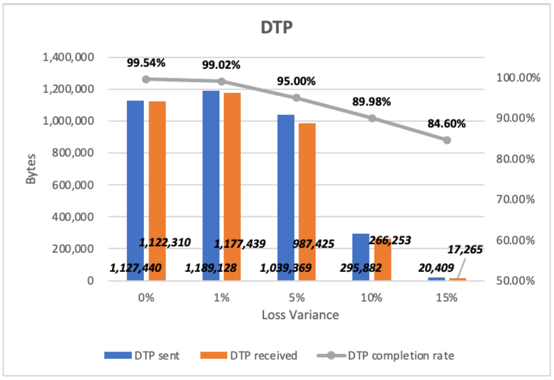

Fig. 5. Various package loss rate of DTP.

In Fig. 4 and Fig. 5, we can see that when package loss rate is less than 10 percent, DTP performs better than QUIC. The adaptive redundancy mechanism of DTP reduces the retransmission data and improves the completion rate. However, when loss variance is 1 percent, we can clearly see that server send mush more data by DTP than by QUIC. As DTP uses FEC to add n packets for the next m consecutive packets, the server sends many redundancy packages. When the package loss rate increases, the transmission performance decreases for the abundance of redundant data packets. We can see that when the package loss rate is 15 percent, DTP performs worse than QUIC.

Therefore, although DTP's redundancy mechanism can reduce retransmission, when transferring big file data, increased data packages may reduce the efficiency of transmission. It is important to consider the trade off when using DTP.

D. Modify different error package rate

To better understand how DTP performs in the real world with dynamic network conditions, we let the server send one 1.4Gb video to client under different error package rate.

Besides, we compare DTP with QUIC to see how redundancy mechanism performs. Fig. 6 and Fig. 7 shows the performances of QUIC and DTP.

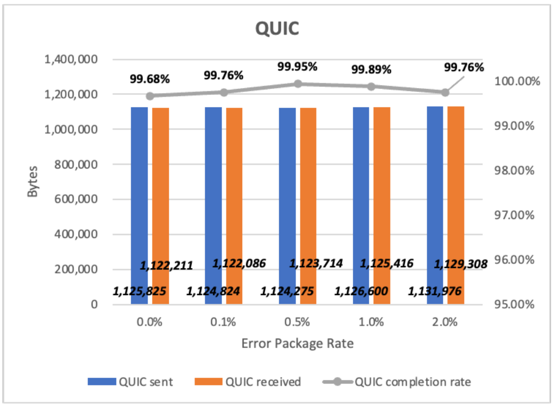

Fig. 6. Various static error package rate of QUIC.

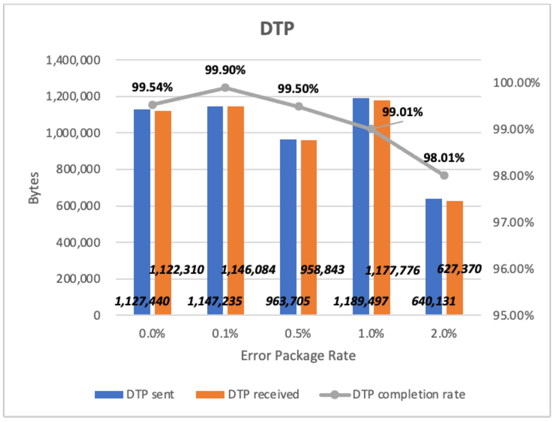

Fig. 7. Various static error package rate of DTP.

In Fig. 6 and Fig. 7, we can see that only the error package rate is less than 0.1 percent, DTP performs better than QUIC. Meanwhile, server sends more data by DTP than by QUIC. Therefore, when transferring big file data under high error package rate, QUIC performs better. However, according to a study on the Hindawi, IPv6 networks have an average packet loss rate of 0.25 percent, while IPv4 networks have an average packet loss rate of 0.33 percent. This suggests that the packet loss rate in the network may indeed be in the range of 0.1 percent to 0.2 percent [6]. In this case, DTP performs well in big file data transmission and other applications with deadline requirements, like gaming and online conferences.

IV. Discussion

This section will discuss the possible limitations of our three comparative evaluations of DTP and QUIC regarding data arrival sizes at different bandwidths, as well as packet success rates in fluctuating packet error rates and alpha value adjustments favoring either deadlines or priorities. This is primarily related to network conditions, topology and timing.

Poor network conditions. Due to the inability to consistently ensure a favorable network testing environment, which often related to the internal network of the virtual machine provider, we frequently encountered transmission timeouts and interruptions caused by traffic fluctuations. As a result, we had to conduct tests repeatedly. It is possible that the evaluation results may vary considerably at different times. In this way, we must also select the more stable, or average results from these repeated tests to make sure the reliability of our experiments.

Topology limitation. The original DTP experiments used three virtual machines to simulate network communication, one as a client, one as a server, and one as a separate router running tc to simulate required network conditions. In contrast, we entered tc commands on just two endpoints. This difference could affect the accuracy of our evaluation results. On the one hand, the DTP only used a single virtual machine as the central point for all network traffic, instead, our network traffic is spread across both ends, which may not always keep conditions consistent on both ends, even though our commands are the same. On the other hand, simulating tc on every virtual machine is not realistic, which may affect the reliability of our experimental conclusions.

Timing limitation. What we call large files were merely 6.6Mb and 1.4Gb of video files in our evaluations. Actually, our planned traffic model is a 10Gb and a 20Gb large video file. The key reason for the shift in planning is the lack of time. Because under the condition of large packet loss rate and error rate and small bandwidth, the transmission of 1.4Gb can even reach 5 to 10 minutes, which is really time-consuming.

Generally speaking, if conditions permit, we will more rigorously rent 3 virtual machines like the DTP tests and do the experiment under the premise of good network conditions. Furthermore, if timeframe permits, we will use larger video files as our transmission models to make the evaluation results more convincing.

V. Conclusion

In this report, we propose additional metrics for evaluations on DTP and QUIC, including big data transfers, packet arrival rates with different bandwidth, packet success rates with varying packet error rates, and transfer cases with biased deadlines and priorities. The results show that DTP outperforms QUIC in large files transimission under normal weak network conditions because DTP considers priority and application specified deadline. However, under the conditions of extreme resource stress, the performance of DTP gradually decreased, while QUIC showed stable performance. Therefore, weighing the use of DTP and QUIC according to different big data transmission and network environments is something we need to consider.

We believe that our experiment is useful. This is because the deployment of real-time applications based on QUIC is already accelerating, and the emergence of DTP will make the user experience of real-time applications even better. Our evaluation of DTP on large file transfer will also promote the optimization and improvement of DTP, so that it can be more widely used in the future possible real-time applications of big data transfer. At the same time, the DTP inspection and improvement we mentioned for various weak network environments can allow users in more regions to join the real-time application. In the future, we look forward to seeing more areas where DTP combines with emerging technologies for real-time data synchronization.

REFERENCES

[1] J. Zhang, H. Shi, Y. Cui, F. Qian, W. Wang, K. Zheng, and J. Wu, "To punctuality and beyond: Meeting application deadlines with dtp," pp. 1–11, Oct 2022.

[2] Wikipedia, "Real-time computing — Wikipedia, the free encyclopedia," 2023, [Online; accessed 10-December-2023]. [Online]. Available: https://en.wikipedia.org/wiki/Real-time-computing

[3] Robots.net, "What internet protocol does online gaming use?" https://robots.net/tech/what-internet-protocol-does-online-gaming-use/, 2023, accessed: 2023-12-10.

[4] A. Langley *et al.*, "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, 2017, p. 183–196. [Online]. Available: https://doi.org/10.1145/3098822.3098842

[5] Z. Gurel, T. E. Civelek, A. C. Begen, and A. Giladi, "A fresh look at live sports streaming with prioritized media-over-quic transport," *Comm-Con'23*, 2023.

[6] F. Li, X. Wang, T. Pan, and J. Yang, "A case study of ipv6 network performance: Packet delay, loss, and reordering," *Mathematical Problems in Engineering*, vol. 2017, p. Article ID 3056475, 2017.