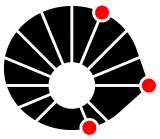
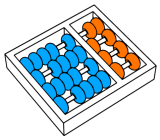




Request scheduling policies for 360° video applications using QUIC

N. Nattis *N. Fonseca* *C. Melo*

Relatório Técnico - IC-PFG-23-50
Projeto Final de Graduação
2024 - Setembro

UNIVERSIDADE ESTADUAL DE CAMPINAS
INSTITUTO DE COMPUTAÇÃO

Request scheduling policies for 360° video applications using QUIC

Nicolas Bissoli Nattis* Nelson Luis Saldanha da Fonseca†
César Augusto Viana Melo‡

30/09/2024

Abstract

With the rising demand for high-quality 360° video driven by increasing adoption of virtual reality technology, providing high visual quality for the user is crucial for a better end-user experience. This paper investigates the use of server-side request scheduling in order to mitigate visual degradation, especially tile missing ratio within the user's field of view. We evaluate three scheduling policies – first-in first-out (FIFO), strict priority (SP), and weighted fair queuing (WFQ) – through simulations and measure their effect, comparing benefits and drawbacks. These findings aim to provide insights for the selection of scheduling policies in real-world QUIC applications, contributing to ongoing development of immersive environment solutions.

1 Introduction

Recent advances in the market for virtual reality (VR) technologies are anticipated to further popularize 360° video, used for creating immersive environments. As these applications gain popularity, exploring techniques to enhance their efficiency in delivering a high-quality user experience becomes crucial.

This study analyzes the use of server-side prioritization in the context of mitigating visual degradation, particularly within the field of view, where the user is more likely to notice artifacts. We specifically compare three scheduling policies: first-in first-out (FIFO), strict priority (SP), and weighted fair queuing (WFQ), and assess the impacts of each on service quality to identify drawbacks and benefits of each technique.

Our scope is an implementation utilizing the QUIC protocol, a modern and standardized protocol for client-server communication. The results can then be interpreted in the context of what benefits a real-world QUIC application might get. The techniques are also able transferable to HTTP/3, which is built on top of QUIC.

*Instituto de Computação, Universidade Estadual de Campinas, 13083-852 Campinas, SP.
†Instituto de Computação, Universidade Estadual de Campinas, 13083-852 Campinas, SP.
‡Instituto de Computação, Universidade Federal do Amazonas, 69080-900 Manaus, AM.

The remainder of this paper is organized as follows: Section 2 provides a theoretical background on 360° video and immersive environments; Section 3 details the scheduling policies; Section 5 presents the numerical results; and Section 6 concludes the study.

2 Theoretical background

This section provides an overview of immersive environments and the QUIC protocol, both fundamental to the premise of this paper.

2.1 Immersive environments

In the context of this document, immersive environments are achieved by displaying 360° video using virtual reality (VR) hardware, where users experience a 3D space through stereoscopic displays. Users are able to rotate the field of view (or camera) to view different parts of the environment, typically by rotating the head while using a head-mounted display [1].

Multiple application domains currently exist for 360° video. For example, van der Hooft et al. [1] cites healthcare (specifically for mental health therapies and training), education, and entertainment. LaValle [2] cites immersive cinema, telepresence (live panoramic view of locations), and the use of virtual reality for empathy focused projects, where the user perceives the struggles of different people from their own perspective. As the technology evolves and becomes more affordable and widespread, it is likely that new applications across various domains will emerge.

The scope of this document focuses on the software aspects of delivering 360° video. Specifically, the software can be fine-tuned to match hardware characteristics, but is not necessarily constrained by them. This means that new or different technologies can benefit as well from software optimizations in the 360° video delivery.

To deliver multiple spatial parts of the 360° video, the video is divided both temporally (into segments of fixed duration) and spatially (into tiles of a certain dimension) that can be assembled incrementally into a single contiguous 360° video [1]. An important aspect of virtual reality playback is that, at any given time, the screens only show a portion of the segment – the field of view. While the field of view can be rotated, large rotations are assumed to occur infrequently in typical usage. Therefore, the quality of the video in the field of view has a larger impact on the overall user experience compared to the quality of the video outside it.

This paper addresses the challenge of optimizing the order in which tiles are transmitted to enhance perceived quality. In our study, we assume a fixed bitrate, meaning that if a tile arrives too late to be displayed, it is considered missing during the playback. Quality is measured by the missing ratio of tiles.

Some clients could pause the playback to wait for the tiles (increasing the buffering), but this can be undesirable: buffering larger durations in 360° video increases memory consumption significantly due to the larger data volume when compared to traditional video. This can become prohibitive for memory-constrained devices or web content. Buffering also

introduces delays that are unacceptable in scenarios such as real-time interaction or live streaming.

By measuring the missing ratio of tiles in high priority regions (inside the field of view) and low priority regions (outside the field of view) it is possible to compare the different techniques used for tile prioritization. Distinguishing between missing tiles inside and outside the field of view allows for a more accurate measurement of perceived video quality, as missing tiles outside the field of view may go unnoticed by the user, whereas missing tiles inside the field of view have a direct impact on the experience.

2.2 QUIC

The 360° video applications discussed in this document are evaluated within the context of using QUIC as the transport protocol. QUIC (Quick UDP Internet Connections) is a standardized protocol built on top of UDP, described in the RFC 9000 [3]. The adoption of QUIC for delivery of 360° video has been suggested by van der Hooft et al. [1] as an approach to reduce latency seen in traditional HTTP/2 or previous and TCP approaches.

The QUIC protocol is designed to offer speed, security, and reliability while addressing some of the limitations of TCP. QUIC uses fewer messages for handshaking compared to TCP with TLS, reducing connection establishment latency, and its ability to multiplex streams mitigates head-of-line blocking.

In the context of web applications, QUIC is notable for its use in HTTP/3. Although HTTP/3 is the primary use case for QUIC in web applications, this paper focuses on applications that directly use QUIC, independent of the HTTP/3 protocol. The techniques presented can also be adapted for use in an HTTP/3 context since they operate at the application layer.

It is important to note that, while it is recommended that QUIC implementations provide means to control transport-level stream prioritization (described in the Section 2.3 of the RFC 9000 [3]), not all implementation do so. This may also not be available if a higher level protocol like HTTP/3 is being used. The techniques for prioritization provided in this document work at the application-level, which enables them to function independently of transport-level prioritization. In more sophisticated applications, these application-level approaches could be layered on top of lower layer transport-level priority policies to further optimize content delivery.

The client and server used by this document are implemented using the Go programming language and use the quic-go [4] library for the implementation of the QUIC protocol.

2.2.1 Congestion control and reliable delivery

QUIC implements congestion control and loss detection mechanisms as detailed in the RFC 9002 [5]. Loss detection is based on acknowledgements, where a packet is considered lost if its acknowledgment is not received within a specific timeframe. Reliable delivery is achieved by retransmitting lost packets. The congestion control adjusts the transmission rate based on the observed loss ratio. Similar to TCP, after an initial slow start period, the

transmission rate stabilizes around the detected channel capacity. Therefore, lost packets cause both retransmissions and reduction in transmission rate.

Although the congestion mechanism relates packet loss to network congestion, packet loss can also happen due to factors such as signal interference in wireless networks, which the congestion control mechanism misinterprets as network congestion, resulting in reduced transmission rates. However, reliable delivery is still achieved through retransmissions.

When packet loss occurs, retransmissions are implicitly prioritized within QUIC's reliable delivery mechanisms, although this is not explicitly stated in the specifications. This prioritization is essential to ensuring timely recovery of lost packets and maintaining reliability guarantees.

3 Scheduling policies

This section has the goal to describe the three scheduling policies that will be explored: First In, First Out (FIFO), Strict Priority (SP) and Weighted Fair Queuing (WFQ). These are classic network scheduling policies with various applications. In order to contextualize the policies within the application, traits that can be advantageous or disadvantageous for the use-case of 360° video are presented for the policies.

3.1 First In, First Out (FIFO)

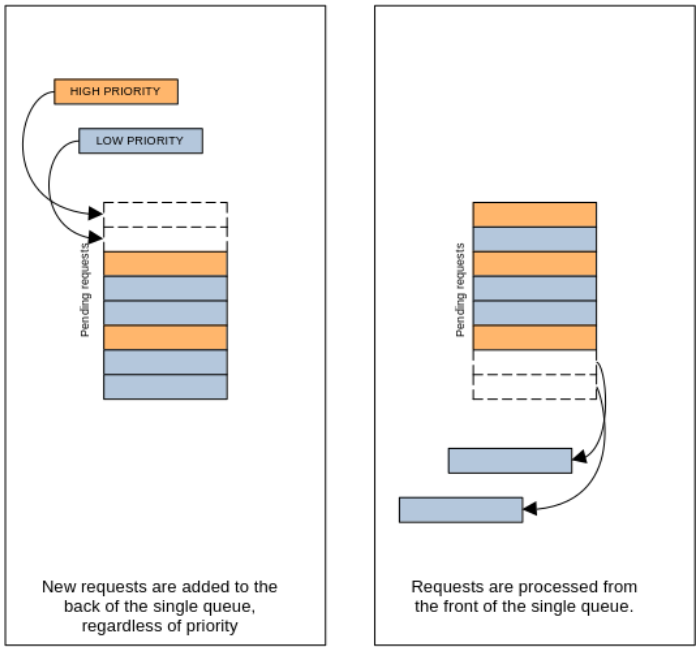


Figure 1: Example of the FIFO scheduling policy

In the FIFO scheduling policy, all received requests are inserted into a single queue and are dequeued in order of arrival to produce responses. As a result, the first request that

is received is the first one to be processed, and so on. This policy is also known as "first come, first served". Figure 1 provides an illustration of the FIFO policy as it receives and processes requests.

Being the simplest scheduling policy, FIFO is a reference point for other policies, but it does not provide prioritization; all requests are treated equally. In the context of immersive 360° video, all regions of the visual field will be equally affected as a result of this lack of prioritization. When there's not enough time to receive all tiles, tiles will be missing from both regions currently inside the field of view and those outside it, and at roughly equal ratios.

3.2 Strict Priority (SP)

In the SP scheduling policy, received requests are placed into queues – referred to as priority groups – corresponding to their priority levels, and elements are dequeued from the priority group with the highest priority value first. Within each priority group, all requests have the same priority and therefore a first-in first-out algorithm is used. The Figure 2 provides an illustration of the SP policy receiving and processing requests.

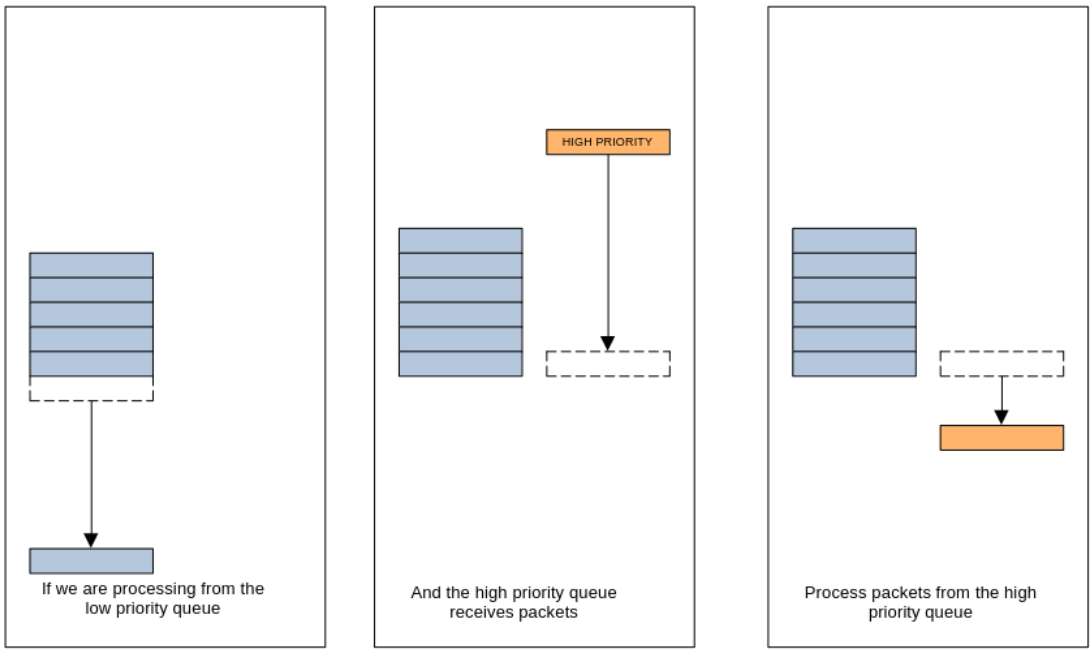


Figure 2: Example of the SP scheduling policy

The SP policy explored in this document is a packet-level (or request-level) preemptive variant, meaning the processing of a priority can be partially completed and interrupted in order to respond to a higher priority packet. If the server receives a higher priority request, the processing of low priority group is preempted (i.e., interrupted) in order for the higher priority group to be processed.

While SP provides a relatively simple implementation and its behaviour is independent

of dynamic run-time characteristics, it has a significant drawback: it can easily starve the low priority group, as it has no fairness guarantees.

In the context of immersive 360° video, this means that the regions within the field of view will usually not suffer from missing tiles unless network resources are very scarce. This comes at the cost of high missing ratio outside the field of view. Although prioritizing the field of view is a desired trait, the inherent strictness of the policy prevents finer adjustments. For example, it may be desirable to maintain a lower missing ratio in the field of vision without penalizing regions outside it too severely, to provide a smoother experience when the user rotates the field of view.

3.3 Weighted Fair Queuing (WFQ)

In the WFQ scheduling policy, received requests are arranged into priority groups corresponding to each priority level. Dequeuing is performed so that each priority group receives a share of resources (i.e., network time) that is ideally proportional to the weight assigned to it. The Figure 3 provides a graphical representation of the WFQ policy, with weight distributions set at 2:1.

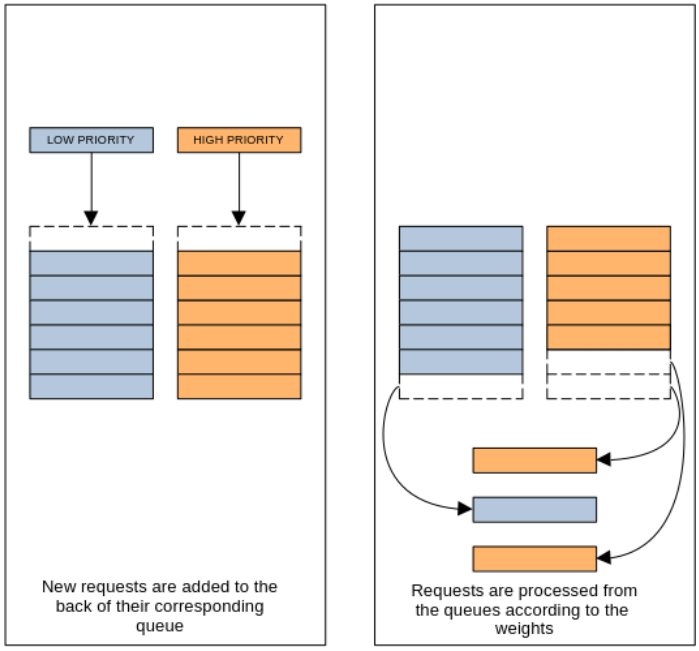


Figure 3: Example of the preemptive WFQ scheduling policy

WFQ can guarantee fairness if correctly configured, unlike SP. Also, the weights can be customised to achieve greater or lesser differentiation.

However, there are drawbacks to WFQ as well. Its implementation is not as straightforward as FIFO or SP, usually involving the usage of virtual time, where each processed request corresponds to one share of virtual time. More critically, it requires proper weight configuration. Ideal weights may vary depending on the packet distribution over priority

groups, which can be either static or a dynamic runtime characteristic, depending on the application. In the latter case, it is more challenging to find optimal weights, and it may be necessary to implement dynamic weight adjustments. For our application, for simplicity, it is assumed that the packet distribution is static.

4 Experimentation setup

The goal of this section is to give an overview of the tools and the implementation used for the experiments. Section 4.1 describes the Mininet tool used for simulating the network environment, and Section 4.2 describes the implementation of the client and server.

4.1 Mininet

For the experimental setup, simulations were performed using the Mininet tool. Mininet allows the creation of a network topology with multiple hosts connected together with specified link parameters [6].

As the tests include a single client and a single server, the resulting network can be very simple: two hosts connected via a switch, illustrated in Figure 4. An actual network would likely be more complex, but the overhead added by routers and other equipment can be effectively modeled simply with link delay and background traffic.

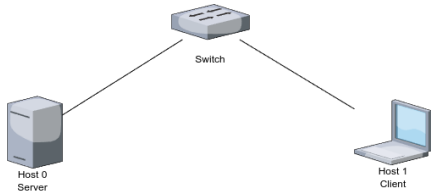


Figure 4: Network topology

To simulate network usage by other applications, background traffic is generated using the iPerf tool [7], which generates UDP traffic that occupies a provided bandwidth (in bits/second) with constant packet rate (CPR).

The test scripts that start the simulation automatically invoke iPerf, which is done via the Python API for the Mininet tool. Thus, the background traffic is consistently started and stopped for each simulation without manual setup.

4.2 Go client and server

For this study, both the server and the test client were implemented using the Go programming language and the quic-go library. This section focuses mostly on the server, where the scheduling logic resides.

The server uses a single stream handler thread for every connection, which handles the requests which are received asynchronously. The main thread continually listens for incoming connections, spawning a new stream acceptor thread and a stream handler thread

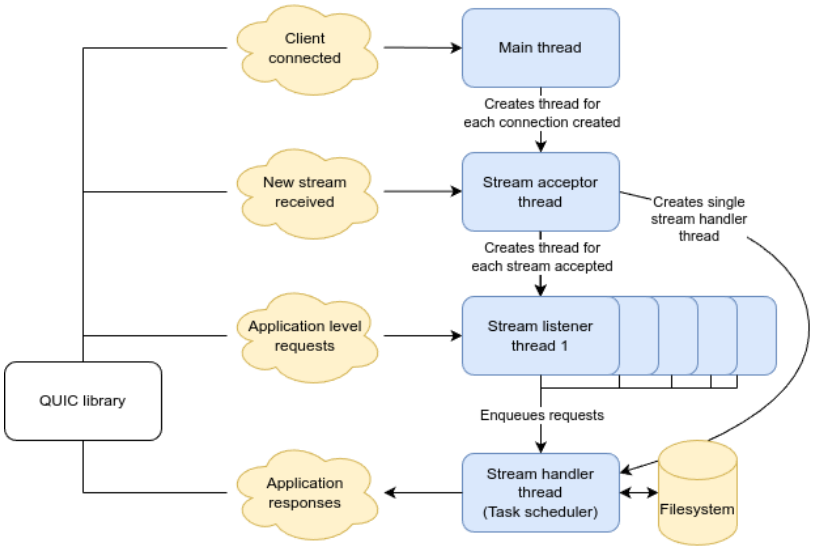


Figure 5: Server architecture overview

for every connection. The stream acceptor listens for incoming streams, spawning a stream listener thread for each. Each stream listener enqueues the requests into the stream handler thread. Figure 5 shows a diagram of the server architecture.

A sequence diagram shown in Figure 6 exemplifies the interaction between the server components. The server receives a connection, followed by a stream, and then two requests on that stream. The requests are then enqueued for handling and processed later, producing responses which are sent to the client. After this, the stream and connections are closed by the client.

The test client is designed with the goal of testing the scheduling policies. It simulates playback by generating requests in intervals of one second, which correspond to the time the playback of a single segment takes, and collects statistics to allow plotting.

Each request contains information for locating which tile to serve – the segment number and tile number –, the priority group to use – high or low –, and the request timeout.

The request timeout is provided by the client to prevent the server from responding to a request which is no longer valid: if the playback of a segment has already began, the client no longer accepts tiles for the segment, and transmitting these would be unnecessary. In our implementation, the server optimistically assumes that the time between the client sending a request and the server receiving it is significantly smaller than the request timeout. This allows the server to implement the timeout by simply comparing it to the time elapsed since the request was received in the server side. A more sophisticated approach could estimate the client-server latency to compensate for this time.

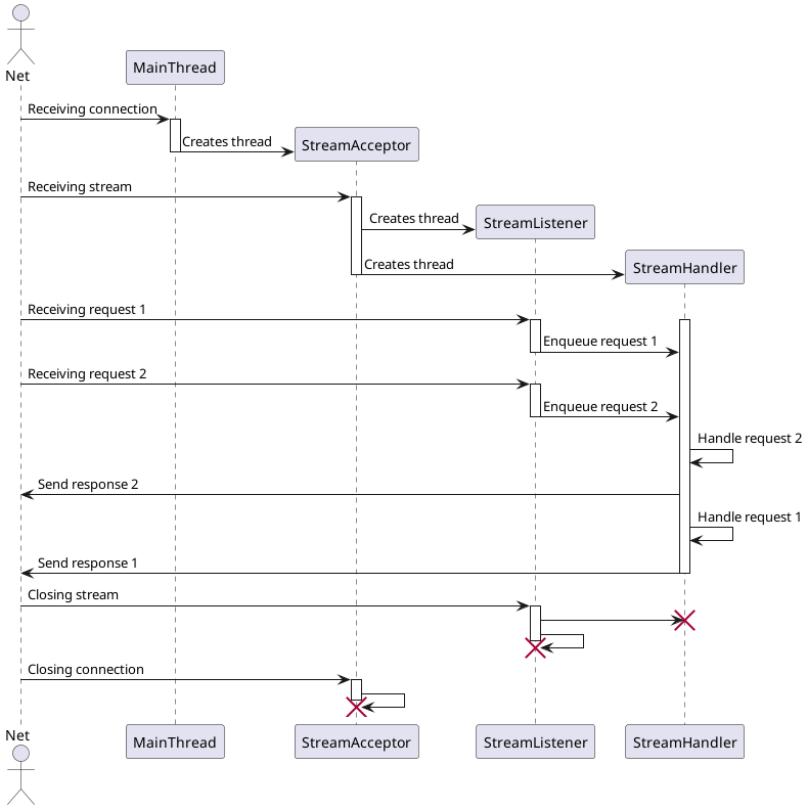


Figure 6: Sequence diagram for sending responses

5 Numerical results

This section presents the numerical study carried out to assess scheduling policies deployed to manage queues fed by immersive video content.

We emulated an immersive video application using the client-server architecture. The client-side emulates a video player that deals with tile-based immersive content, i.e., it requests, stores and stitches tiles of video segments and buffers them in a finite queue. Stitched segments are dequeued accordingly and their completeness is assessed, i.e., the missing ratio of each segment is recorded.

The server-side emulates a video server that manages all files related to the content. This server enqueues incoming requests and processes them accordingly. Each request carries a playback deadline that turns them disposable at the server-side.

Two hosts are created and connected through a dedicated link that receives the application and background traffic. The background traffic emulates other applications' traffic, generated on both ends. In the following, we present the networking factors and their setups to establish the resources used to move requests and data in both directions.

5.1 Scenarios

This subsection describes the numeric results obtained when running the simulations using the scenarios presented in the Table 1. These scenarios were defined with the goal of approximating last mile content delivery and are referenced in later subsections. In each scenario, the channel has a constant 100 Mbps bandwidth, which is shared between the 360° video application communication and an emulated background load. Each scenario defines a background load, either 10% or 30% of the total bandwidth.

The link delay, which is defined as either 10 ms, 16 ms, or 24 ms, models a near-instant communication scenario, an intermediate scenario, and a scenario where the equipment is busy and therefore the communication has greater latency. Link loss rate is 2% for the results presented in this document unless noted otherwise.

Table 1: Experimentation scenarios

Scenario	Background load	Delay
#1	10%	24 ms
#2	30%	24 ms
#3	10%	16 ms
#4	30%	16 ms
#5	10%	10 ms
#6	30%	10 ms

Another parameter that affects the policies' behavior is the distribution of priorities across the packets. In Figure 7 the tile missing ratios for high and low priorities are shown, where 50% of packets are low priority and 50% of packets are high priority.

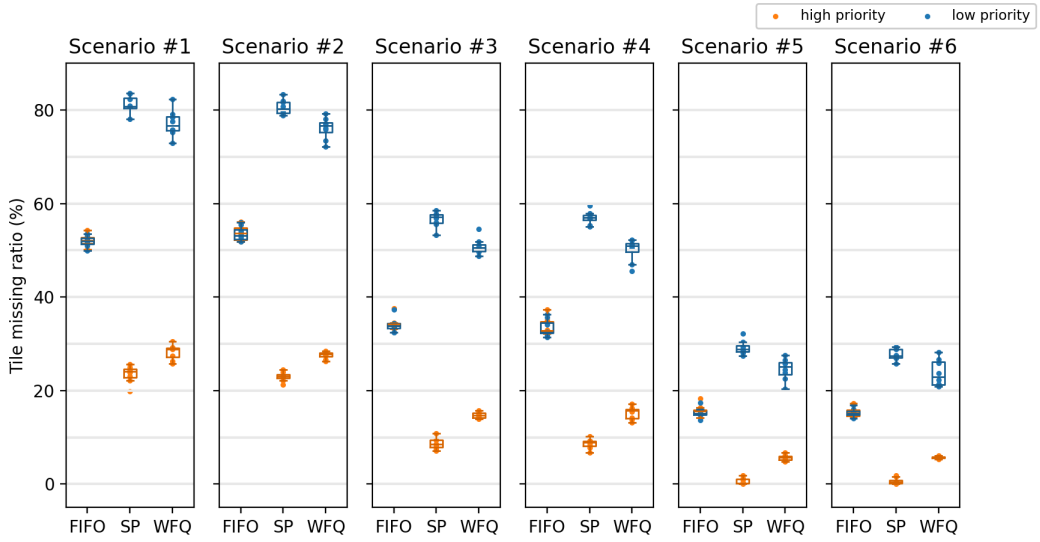


Figure 7: Tile missing ratios for each scenario when 50% of packets are high priority

In Figure 8 the tile missing ratios are shown when 70% of packets are low priority and 30% of packets are high priority. The WFQ weights have been compensated due to the difference in the packet distribution: this is explained further in Section 5.5.

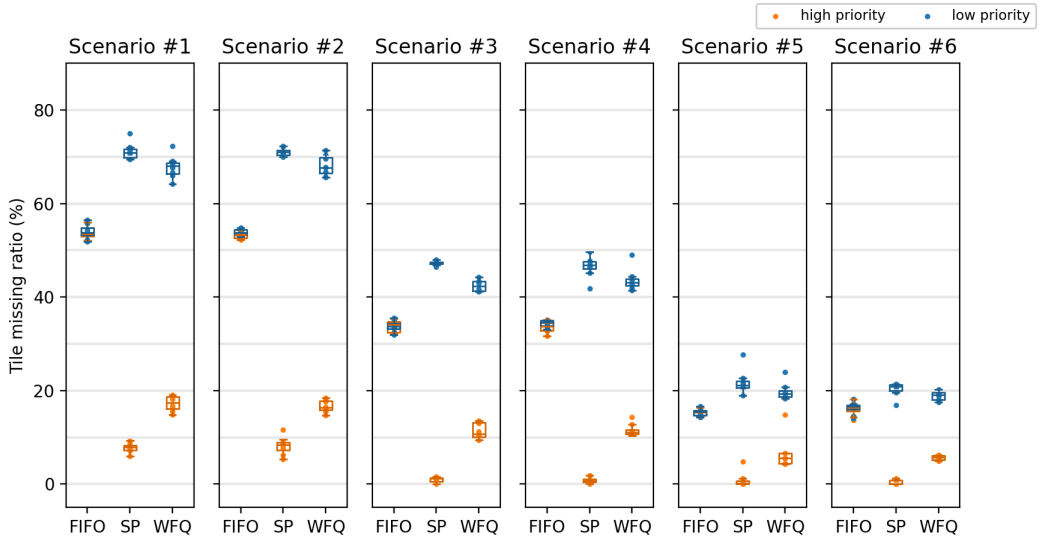


Figure 8: Tile missing ratios for each scenario when 30% of packets are high priority

It is important to note that in the given scenarios, there's a strong correlation between the link delay and the tile missing ratio. On the order hand, the increase of background load from 10% to 30% does not result in a measurable increase in the tile missing ratio. From this, it is possible to conclude that the link delay is the bottleneck for the tile missing ratio in the given scenarios.

The Table 2 shows the combined tile missing ratio, which takes into account all packets regardless of priority. Neither the distribution of packet priority nor the applied policy has a measurable impact on the combined tile missing ratio; although intuitive, this result is important for demonstrating that the connection quality does not degrade from the implemented policies.

Table 2: Combined tile missing ratio summary

	50% high priority packets			**30% high priority packets**		
Scenario	FIFO	SP	WFQ	FIFO	SP	WFQ
#1	52%	52%	53%	54%	52%	53%
#2	54%	52%	52%	54%	52%	53%
#3	34%	33%	33%	34%	33%	33%
#4	34%	33%	33%	34%	33%	33%
#5	16%	15%	15%	15%	16%	16%
#6	15%	14%	15%	16%	14%	15%

5.2 Impact of initial playback latency

When simulating a playback, the time between the first request and the start of the playback is defined by a parameter. This section evaluates the impact of this parameter, called initial playback latency, over the experimental results.

A small initial playback latency corresponds to an increased tile missing ratio of the first segment: a lower latency provides a shorter time window for the communication before the segment is played back – less time to buffer segment –, therefore allowing less tiles to be received. Figure 9 shows the relation between initial playback latency and tile missing ratio when a single segment is played.

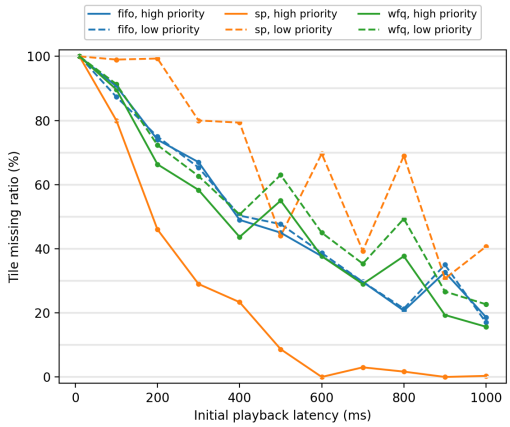


Figure 9: Impact of initial playback delay, single segment

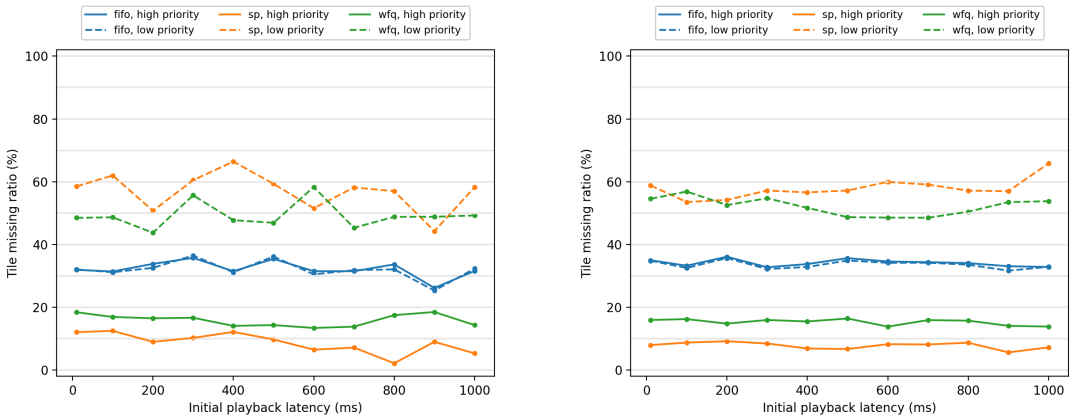
All simulations in this section use the Scenario 3 defined in Section 5.1. 50% of tiles are low priority, and 50% high priority.

However, this only affects the first segment. The time to receive other segments is determined by the time that the playback of the previous segment takes, which in the case of the simulations is fixed at 1000 ms, as the buffering always includes only a single segment due to drawbacks caused by buffering of large durations, as discussed in the Section 2.1.

Because the total tile missing ratio is an average of the tile missing ratios of the segments, the impact of the initial playback latency on the total tile missing ratio diminishes as the number of segments increases.

Figure 10 displays the relationship between initial playback latency and tile missing ratio for playback with multiple segments. With 20 segments, the tile missing ratio is less correlated with the initial playback latency than seen with one segment, although there is still some correlation visible for SP and WFQ using high priority, where the tile missing ratio slightly drops as the initial playback latency increases. With 76 segments, there is no apparent correlation.

Therefore, when a test spans 76 segments, the initial playback latency can be chosen arbitrarily without significantly affecting the overall experimental results. Unless otherwise noted, the simulations on this document will use an arbitrary fixed value for initial playback latency.



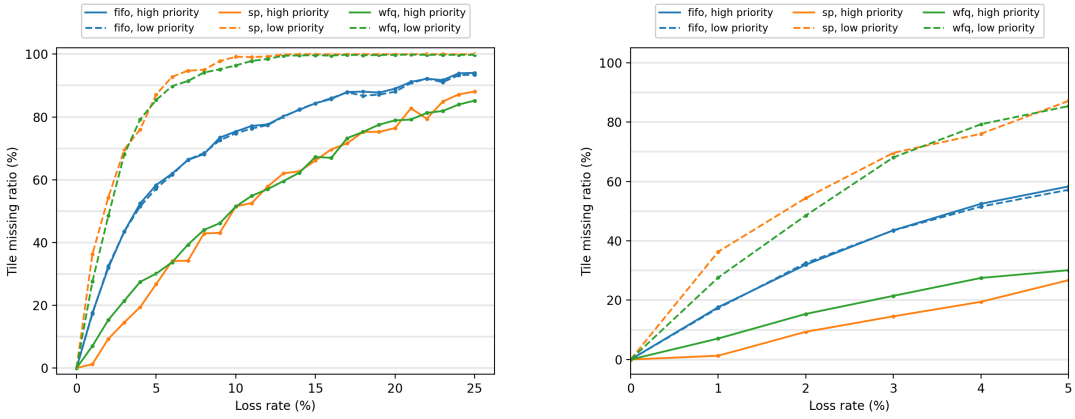
(a) 20 segments (b) 76 segments

Figure 10: Impact of initial playback delay, multiple segments

5.3 Impact of loss rate

This section evaluates the impact of the link loss rate over the tile packet missing ratio. It is important to note that, because of QUIC's reliable delivery mechanism (see Section 2.2.1), a lost IP packet does not immediately correspond to a missing application packet. Instead, the correlation is caused by an increase in latency, as the packet is only received when a retransmission is, at a later time, received.

The correlation between link loss rate and tile missing ratio is shown in the Figure 11. The Scenario 3 of Section 5.1 is used. 50% of tiles are low priority, and 50% high priority. The figure is shown zoomed out, with the loss rate in the interval 0-25%, and zoomed in, with the loss rate in the interval 0-5%.



(a) Zoomed out: 0-25% (b) Zoomed in: 0-5%

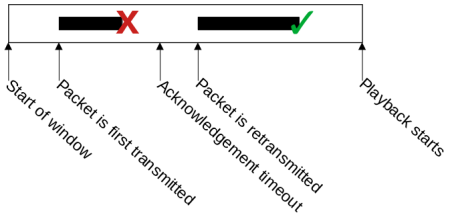
Figure 11: Impact of loss rate

It is possible to see that the correlation between link loss rate and tile missing ratio is not

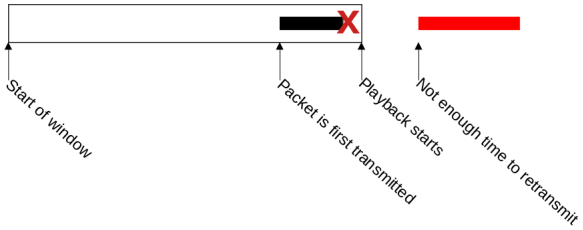
linear. The low priority tile missing ratio initially has a much steeper increase for the SP and WFQ policies: as a result, an increase in loss rate exacerbates the priority differentiation. However, after a certain point – about 5% –, the low priority tile missing ratio starts to increase more slowly as it approaches 100%. From this point on, an increase in the missing ratio actually reduces the priority differentiation (assuming SP or WFQ policies are being used).

The phenomenon that happens with lower link loss rates, where priority differentiation is greater, can be explained by the order in which packets are sent over the wire: SP and WFQ transmit high priority packets early (near the start of the transmission) – in the case of SP all high priority packets are sent early, and in the case of WFQ there are proportionally more early high priority packets –, and early packets have a larger time window for being eventually retransmitted. If a QUIC packet is submitted near the start of the reception time window, it is more likely for there to be enough time for a retransmission to succeeded in the future than if it were lost near the end of the reception time window. Figure 12 is a visual representation of this phenomenon.

In this case, we consider that the attempt to transmit the packet always occurs earlier than the start of the playback. This means that, in the case where there is "not enough time to retransmit", it actually means that the retransmission still causes a missing ratio: in fact, the retransmission happens anyways – because once a submission is made, QUIC will retransmit until everything is acknowledged. If a packet is so late that its transmission is scheduled after its playback instant, however, the packet will not be transmitted, as the server automatically detects the timeout and prevents the tile from unnecessarily using network resources.



(a) Early packet



(b) Later packet

Figure 12: Visualization of differentiation exacerbated by loss rate

With a higher loss ratio this breaks down: while at first only low priority packets are too

late, eventually the higher latency caused by the early packets being retransmitted makes it so that packets are initially submitted later, and even high priority packets are way too late to have successful transmissions.

5.4 Impact of choice of WFQ weights

Choosing appropriate WFQ weights is important to the performance of that policy. As the weight values get close to each other – close to 1:1 –, the policy approaches FIFO behaviour, and as the weight values get farther apart, the policy approaches SP behaviour.

With the goal of measuring the impact of the weight selection, simulations were executed with varying values of high priority weight. The low priority weight is fixed at 100. In each simulation, the link has 100 Mbps of bandwidth, 10% of background load and 14 ms of link delay. 50% of tiles are low priority, and 50% high priority.

The Figure 13 shows the results of the simulations. In one extreme, the 100:100 ratio the policy performs as a FIFO, where there is no difference in tile missing ratio across the priorities. As the difference increases, the tile missing ratio raises for the low priority and lowers for the high priority.

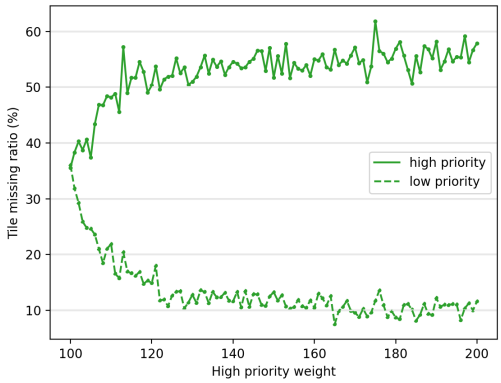


Figure 13: Impact of WFQ weights

The weights 120:100 (i.e., 6:5), unless otherwise noted, were chosen for the results in the rest of the document: under the circumstances of this simulation, these are not too close to either extremes, and therefore produce results distinct from both FIFO and SP policies.

It is necessary to note that some changes in circumstances may require adjusting WFQ weights to keep the same relative fairness. In particular, Section 5.5 shows that fixed WFQ weights vary their behaviour as the distribution of requests across priorities changes, requiring compensation.

5.5 Impact of request distribution across priorities

When requesting a segment, some tiles are allocated low priority, and some are allocated high priority. This distribution affects both SP and WFQ.

In SP, a lesser amount of high priority tiles allows the high priority tiles to be quickly received at the start of the reception window. Any retransmissions then have a large window to be received, reducing loss rate (a phenomenon similar to seen in what's seen on Section 5.3).

In WFQ, the behaviour is more complicated. Each priority group is treated as a single entity which is served by the WFQ scheduler. As such, if a group has less packets than another, the actual time allocated per packet of the first priority group is increased.

For example, let's consider a simplified scenario, where the scheduler splits up a determined duration of time between tiles, ignoring retransmissions. Consider the weights are 2:1, and there are 10 tiles of low priority, and 10 tiles of high priority. Also consider that the total transmission time to be scheduled between the tiles is 30 ms.

In this example, 20 ms will be scheduled for the high priority group, and 10 ms for the low priority group. This means that each high-priority tile is allocated 2 ms, and each low-priority tile is allocated 1 ms.

However, if we change the distribution from 10 tiles of low priority and 10 tiles of high priority (a 0.5 high priority ratio) to 15 tiles of low priority and 5 tiles of high priority (a 0.25 high priority ratio), these figures change. 20 ms are still allocated for the high priority group, and 10 ms are still allocated for the low priority group. However, now each high priority tile is allocated around 2.3 ms, and each low priority tile is allocated 4 ms.

By changing the distribution of priorities, now each low priority tile is being allocated more time than a high priority tile. This effectively means a priority inversion happens.

Although the example is a very simplified model of reality, this behavior is still seen in practice. Figure 14 shows how the WFQ behaviour changes drastically as the priority distribution changes. On the simulations presented in this section, the Scenario 3 of Section 5.1 is used.

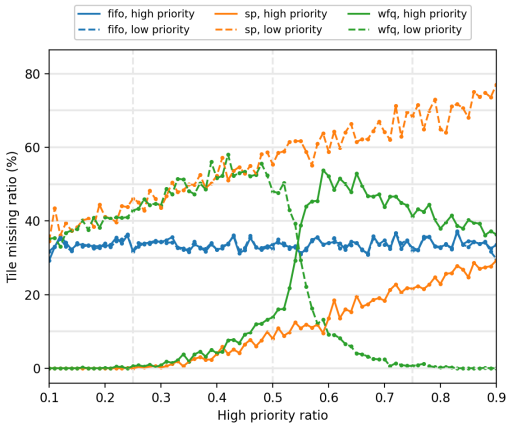


Figure 14: Impact of priority distribution ratio

In order to counteract the influence of the priority distribution, it is possible to compensate the WFQ weights. This keeps the actual "time per packet" ratios fixed, however this means it is necessary to coordinate WFQ weight selection with priority distribution. The

compensation works by adjusting the WFQ high priority weight, $W_1(x)$, according to the ratio of high priority tiles over all tiles, x, while the WFQ low priority weight, W_0, remains constant.

The ratio of $W_1(x)$ over $(W_0 + W_1(x))$ is called the weight ratio, $\varphi_1(x)$, as shown in Equation 1. By starting from a known $\varphi_1(0.5)$, which is the weight ratio of high priority when 50% of tiles are high priority, it is possible to calculate $\varphi_1(x)$ for any given x, as shown in Equation 2.

$$\varphi_1(x) = \frac{W_1(x)}{W_0 + W_1(x)} \tag{1}$$

$$\varphi_1(x) = 2x \cdot \varphi_1(0.5) \tag{2}$$

From $\varphi_1(x)$, it is then possible to get $W_1(x)$ for any x by solving Equation 3, derived from Equation 1. By doing that, we adjust the weight ratio in proportion to the ratio of high priority packets.

$$\cdot W_1(x) = \frac{\varphi_1(x) \cdot W_0}{1 - \varphi_1(x)} \tag{3}$$

For this section, $W_0 = 100$ and $\varphi_1(0.5) = \frac{6}{11}$, which produces $W_1(0.5) = 120$. We can derive $W_1(x)$ from these values, arriving in Equations 4 and 5.

$$\varphi_1(x) = \frac{12x}{11} \tag{4}$$

$$W_1(x) = \frac{100 \cdot \varphi_1(x)}{1 - \varphi_1(x)} = \frac{1200x}{11 - 12x} \tag{5}$$

The Figure 15 shows the behaviour with the WFQ compensation. The curves for other priorities are the same for Figures 14 and 15.

Figure 15: Impact of priority distribution ratio with compensated WFQ weights

It is possible to see that the compensated WFQ behaves much better, without priority inversion. The cost for this compensation comes as either the requirement for a fixed priority distribution known by the server beforehand – as done in our case –, or additional complexity by implementing some form of dynamic weight calculation.

6 Conclusion

The results presented in this paper allow a comparison of advantages and drawbacks of each of the three scheduling policies – FIFO, SP, and WFQ – in the context of 360° video applications. These results show that both SP and WFQ outperform FIFO when delivery time of regions inside the field of view is preferred: they better utilize the channel to deliver content of higher significance to the user.

When comparing SP and WFQ, the greater flexibility of WFQ can be seen as the weights are adjusted, but the need to tune these weights introduces complexity. While SP does not require further configuration, WFQ needs to either know the ratio of the priorities across request beforehand to use static weights, or to be extended with a more complex dynamic weight adjustment.

References

[1] J. van der Hooft, H. Amirpour, M. T. Vega, *et al.*, "A tutorial on immersive video delivery: From omnidirectional video to holography," *IEEE Communications Surveys & Tutorials*, vol. 25, no. 2, pp. 1336–1375, 2023. DOI: 10.1109/COMST.2023.3263252.

[2] S. LaValle, *Virtual Reality*. Cambridge University Press, 2019. [Online]. Available: https://msl.cs.uiuc.edu/vr/web.html.

[3] J. Iyengar and M. Thomson, *QUIC: A UDP-Based Multiplexed and Secure Transport*, RFC 9000, May 2021. DOI: 10.17487/RFC9000. [Online]. Available: https://www.rfc-editor.org/info/rfc9000.

[4] M. Seemann, *The quic-go protocol suite*. [Online]. Available: https://quic-go.net/docs/.

[5] J. Iyengar and I. Swett, *QUIC Loss Detection and Congestion Control*, RFC 9002, May 2021. DOI: 10.17487/RFC9002. [Online]. Available: https://www.rfc-editor.org/info/rfc9002.

[6] M. P. Contributors, *Mininet overview*. [Online]. Available: http://mininet.org/overview/.

[7] iPerf Contributors, *iPerf - The ultimate speed test tool for TCP, UDP and SCTP*. [Online]. Available: https://iperf.fr/.