Enabling Adaptive and Reliable Video Delivery Over Hybrid Unicast/Broadcast Networks

Casper Haems
Jeroen van der Hooft
Hannes Mareen
casper.haems@ugent.be
Ghent University - imec
Ghent, Belgium

Peter Steenkiste
Carnegie Mellon University
Pittsburg, PA, USA

Glenn Van Wallendael
Tim Wauters
Filip De Turck
Ghent University - imec
Ghent, Belgium

ABSTRACT

The increasing demand for high-quality video streaming, coupled with the necessity for low-latency delivery, presents significant challenges in today's multimedia landscape. In response to these challenges, this research explores the optimization of adaptive video streaming by integrating 5G terrestrial broadcasting with over-the-top (OTT) streaming methods. A comprehensive integration of forward error correction (FEC), temporal layer injection (TLI), and broadcast techniques enhance the robustness and efficiency of content delivery over broadcast networks and reduce unicast bandwidth to zero in low loss environments. Multiple strategies are compared through an extensive emulation setup for reducing latency in the end-to-end video delivery chain to sub 3-second live latency, demonstrating the effectiveness of a hybrid unicast-broadcast approach in achieving low-latency while maintaining high-quality video streaming performance with significantly reduced bandwidth. For 62.99% of viewers, unicast bandwidth can be reduced to as low as zero when broadcasting the top 3 TV channels.

CCS CONCEPTS

• **Networks** → **Network experimentation**; • **Information systems** → **Multimedia streaming**.

KEYWORDS

Video delivery, Multimedia streaming, 5G terrestrial broadcast, Low latency

ACM Reference Format:
Casper Haems, Jeroen van der Hooft, Hannes Mareen, Peter Steenkiste, Glenn Van Wallendael, Tim Wauters, and Filip De Turck. 2024. Enabling Adaptive and Reliable Video Delivery Over Hybrid Unicast/Broadcast Networks. In *The 34th edition of the Workshop on Network and Operating System Support for Digital Audio and Video (NOSSDAV '24), April 15–18, 2024, Bari, Italy.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3651863.3651880

1 INTRODUCTION

Live multimedia content delivery is transitioning from traditional broadcast television to the dominance of mobile over-the-top (OTT) streaming, straining existing internet infrastructure, and necessitating the development of novel adaptive video streaming approaches. The current evolution of telecommunications technologies, particularly the advent of 5G and 6G, has enabled new possibilities. Promising advances facilitated by 5G are the utilization of High-Power High-Tower (HPHT) networks [13] and 5G-enabled satellites [11] for efficient video streaming to a large audience across vast geographical areas. By integrating with existing public mobile networks, intelligent automatic offloading of high-demand video content to 5G broadcast networks promises enhanced spectrum utilization and alleviated congestion on public mobile networks.

This paper addresses the critical challenge of enabling broadcast networks for adaptive and reliable video delivery, with a particular focus on mitigating packet loss and reducing latency. Traditional approaches to video streaming, relying on hypertext transfer protocol (HTTP) [15]/transmission control protocol (TCP) [14], face limitations, especially during high-demand events such as major sports broadcasts where millions of viewers are concurrently engaged [27]. In such scenarios, leveraging broadcast technologies can significantly reduce network bandwidth consumption while ensuring seamless video delivery to a massive audience. Challenges persist in ensuring reliability and minimizing latency, particularly for interactive use cases, where low-latency data transmission is essential. To address these challenges, a hybrid adaptive streaming approach is introduced that combines the efficiency of broadcast networks with the flexibility of traditional unicast connections. By offloading popular video streams to broadcast networks and enabling the recovery of lost packets using HTTP protocols, the approach aims to optimize network resources and enhance the overall quality of video delivery. To achieve low-latency, innovative techniques are used, such as the common media application format (CMAF) [18] with chunked transfer encoding (CTE). Moreover, the use of temporal layer injection (TLI) [24] as a new adaptivity technique further enhances the delivered video quality. This approach improves upon other works and contributes by resolving loss problems with a combined forward error correction (FEC) [31] and unicast repair, while aiming for a low-latency solution and still enhancing the adaptivity.

The paper discusses related research in Section 2, explores the system architecture in Section 3, presents experiment overviews and results in Section 4, and concludes the findings and discussions in Section 5.

2 BACKGROUND AND RELATED WORK

Today, most content providers employ an HTTP/TCP-based video streaming approach, with HTTP adaptive streaming (HAS) as the standard method. Segment durations typically range from two to

NOSSDAV '24, April 15–18, 2024, Bari, Italy
2024. ACM ISBN 979-8-4007-0613-4/24/04
https://doi.org/10.1145/3651863.3651880

ten seconds [10], resulting in end-to-end delays of several seconds. While existing literature explores optimizing parameters in both HTTP Live Streaming (HLS) and Dynamic Adaptive Streaming over HTTP (DASH) to mitigate latency [19, 29, 36], the focus often neglects a hybrid streaming approach inclusive of recovery mechanisms [6]. Alternatives such as HTTP/2 push-based methods aim to optimize network utilization and decrease latency but may not achieve bandwidth efficiency comparable to traditional broadcasting methods [21]. Additionally, leveraging features of HTTP/3 can sometimes lead to compromises in the Quality of Experience (QoE) [7, 8]. Arunruangsirilert et al. demonstrated the feasibility of delivering low-latency video segments over 5G networks [5]. Shurdi et al. [33] continues by carrying out a broad, but not in-depth, overview of a system for realizing linear content using 5G broadcast. Several studies based on the 5G-MAG reference tools provide insights into seamless switching between broadband and broadcast [16, 34, 35, 37]. However, these works lack detailed bandwidth and latency metrics that may be necessary for a comprehensive evaluation. Papers testing the feasibility of low-latency implementations [22, 23] do not demonstrate the potential effects on bandwidth. Others focus on broad use case testing [25] but lack bandwidth and latency metrics from live stream experiments. Despite the lack of research that includes bandwidth and latency measurements from hybrid livestream experiments, frameworks for measuring the QoE and Quality of Service (QoS) do exist [38, 39]. The approach provided in this paper bridges the gap to give a comprehensive understanding of how these solutions work together and show the effects on bandwidth, latency, and adaptability. OTT streaming, characterized by the delivery of video content directly over the internet to users, bypassing traditional distribution channels like cable or satellite, is central to the modern multimedia content delivery landscape and enables flexibility and on-demand access.

For broadcasting, File Delivery over Unidirectional Transport (FLUTE) and Real-Time Object Delivery over Unidirectional Transport (ROUTE), are the primary protocols for efficient and reliable file delivery over broadcast networks [28, 43]. These protocols, along with standards such as DVB-I over 5G [17], enable a hybrid approach. Moreover, error correction techniques such as FEC using Raptor codes [32] are essential for enhancing error resilience, despite introducing complexity and potential latency [20]. This paper studies broadcast with FEC, unicast repair, and low-latency techniques such as CMAF-CTE in detail.

3 SYSTEM ARCHITECTURE

The architecture, shown in Figure 1, consists of three main components. A server, one or more proxies and one or more clients. The server is able to serve files over unicast (UC), but also transmit these over broadcast/multicast (MC). The system incorporates a file-caching proxy at the receiver end to enhance robustness and bandwidth efficiency. This proxy can be situated within a 5G-capable home gateway, the client device, or similar infrastructure and acts as a buffer for received files. In scenarios where a file fails to arrive at the proxy over broadcast/MC, the file can still be retrieved through traditional UC methods, thereby ensuring uninterrupted content delivery. The client functions as a DASH viewer, responsible for playing back multimedia content provided by the server.

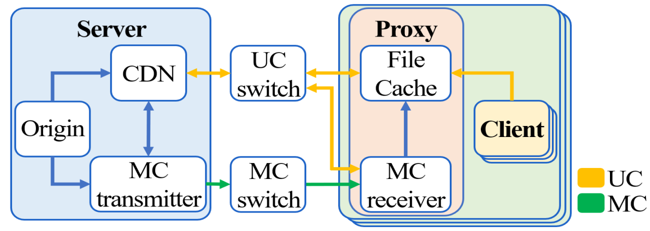

Figure 1: The system architecture. Multicast is used for emulating a broadcast network.

3.1 Streaming and file caching workflow

The proxy manages content delivery by combining caching and HTTP communication. When a client requests a live stream, it obtains the Media Presentation Description (MPD) [42], which holds the stream information, from the server. Upon receiving the MPD, the client initiates playback. As the client streams and sequentially requests new content, the proxy checks if a requested segment is in the cache; if not, it fetches the segment file from the server, caches it, and delivers it to the client. If the file is already in the cache, then it can be delivered immediately to the client.

3.2 Broadcast workflow

The broadcast workflow, displayed in Figure 2 outlines the process of broadcasting for a single file, such as a segment of a live stream using the FLUTE protocol. Initially, the file undergoes some preparations, including loading into memory and potentially applying FEC encoding. The file is segmented into blocks, with optionally Raptor FEC encoding and decoding performed at the block level, allowing for partial decoding before the complete file is received. These blocks are further divided into symbols, each sized to fit within a maximum transmission unit (MTU) [30] along with accompanying identification data. The server then sends a file delivery table (FDT) over broadcast to the proxy, containing vital metadata about the file. The reception of this metadata is required to inform the receiver on how to handle later incoming packets. Following the transmission of the FDT, the server sends the actual file by transmitting symbols over broadcast, encapsulated within asynchronous layered coding (ALC) [41] packets. On the receiving end, the proxy creates the necessary file objects in memory upon receipt of the FDT and awaits the arrival of symbols for each file.

A novel recovery protocol is introduced in this paper, which is that if some symbols are lost during transmission, a repair mechanism is activated that requests missing symbols over UC from the server. This mechanism, called packet recovery (PR), is triggered by a deadline, which is based on when the file should have been transmitted. Once all symbols are retrieved, the file is deemed complete, and FEC decoding is attempted before storing it in the file cache of the proxy.

3.3 Extensions to the FLUTE workflow

The FLUTE workflow has been extended to encompass various enhancements aimed at optimizing file delivery and addressing potential challenges. Firstly, to increase the chance of arrival of

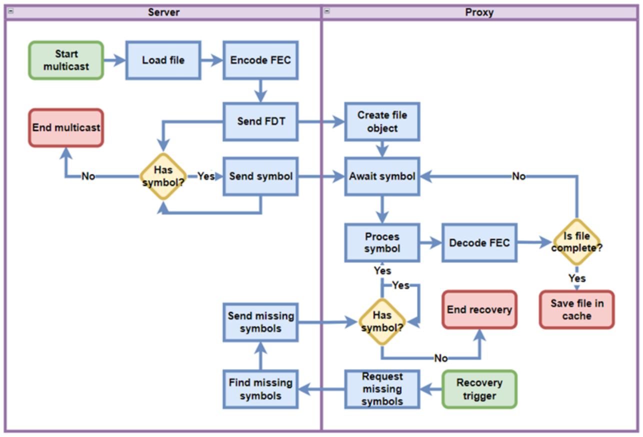

Figure 2: Workflow for broadcasting a file.

FDTs in lossy environments, periodic FDT retransmissions are implemented, increasing the likelihood of reception. This is important because without FDTs, the receiver can not recognise incoming symbols. In cases where a proxy receives unrecognised symbols, these symbols are stored in a separate buffer until the arrival of the FDT that holds the necessary metadata. Upon FDT arrival, the symbols in this buffer are processed. Next, a simple FDT UC recovery mechanism has been implemented, allowing the receiver (proxy) to request the most recent FDT over UC in case the buffer grows too large. Figure 3 illustrates the core concept of using broadcast/multicast, highlighting a specific window for broadcasting and receiving a video segment, termed the broadcast time frame. Within this time frame, the server broadcasts the segment, while the proxy receives and processes symbols for timely playback. Factors such as FEC and PR reduce the transmission window. The recovery deadline marks the trigger for PR, typically set as the difference between itself and the start of the buffering and playback (BP) phase. In cases of insufficient bandwidth, extending the time frame becomes necessary to prevent delays in PR, ensuring that symbol processing concludes within the designated window without interfering with the broadcast transmission time.

4 EVALUATION

This section starts with a brief initial theoretical evaluation, then an explanation of the setup, followed by bandwidth-centric use cases, and concludes with latency experiments.

4.1 Theoretical evaluation

4.1.1 Content distribution. A Zipf-like distribution [9] is employed based on a dataset on viewing behavior [1], modeling 25 TV channels with clients uniformly clustered, each cluster featuring a single proxy. The probability corresponds closely to $f(x) = 1/(x^{\alpha} \sum_{i=1}^{n} i^{-\alpha})$, $x = 1, 2, , n$, with $\alpha = 1.265$ and n TV channels.

4.1.2 Impact of number of clients, proxies, and broadcasted TV channels. In the case of hybrid broadcast/unicast, the primary interest is in reducing unicast traffic between the server and the proxy.

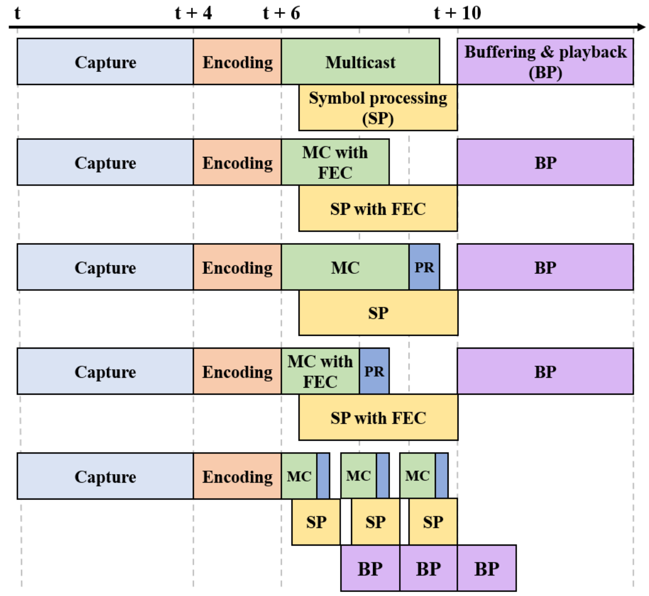

Figure 3: Deadlines for the broadcast approach with DASH. The last line uses LL-DASH through CMAF-CTE. Each CMAF chunk is sent individually.

When no TV channels are available through broadcast, the relative number of unicast streams decreases with an increased number of clients through file caching only. When the m most popular channels are provided through broadcast, an additional reduction can be achieved. This reduction is most significant for a lower number of clients: as the number of clients increases, the number of unicast streams per client converges to 0. Providing $m = 3$ TV channels through a broadcast results in a relative reduction of 63%, 57%, 52% and 42% for 1, 3, 5 and 10 clients, respectively. Increasing this number further to $m = 8$ results in respective reductions of 83%, 79%, 76% and 71%. Naturally, broadcasting all considered channels without packet loss results in no unicast traffic. Thus, broadcasting TV channels can significantly reduce the need for unicast traffic, with more significant gains for a lower number of clients clustered together behind a proxy at the home gateway. Reductions are most significant when a single client is considered, corresponding to the case where every consumer has access to a dedicated 5G broadcast receiver. In this case, broadcasting 8 channels can reduce unicast traffic by 83%.

4.2 Experimental setup

A setup was made in an on-premise large-scale generic networking environment [2]. The setup is shown in Figure 4. Each blue box represents a node, a bare-metal machine. There is one server node, two nodes acting as switches, and $0 < n_{proxies} \leq 5$ proxy-client nodes. The switches enable unique bandwidth traces and loss settings on a per-proxy basis.

Each node in this setup is equipped with 24 GB of RAM, a 1 GBit NIC (Intel Corporation 82576 Gigabit Network Connection (rev 01)), a 250 GB HDD, and an Intel® Xeon® Processor E5645 (2.4 GHz). Note that this processor has no support for AVX2, which is used by the FEC library for hardware acceleration.

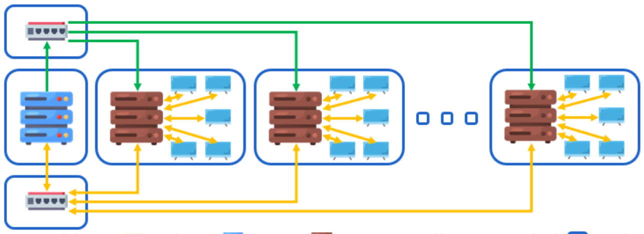

Figure 4: Experimental setup, hosted on a large-scale generic networking environment.

Table 1: Bitrate ladder for the selected video.

Id	TLI	CRF	Bitrate 1s (Mb/s)	Bitrate 4s (Mb/s)
5	No	20	7.1	6.6
4	Yes	Mixed	4.7	4.2
3	No	30	2.2	2.1
2	Yes	Mixed	1.4	1.2
1	No	51	0.4	0.4

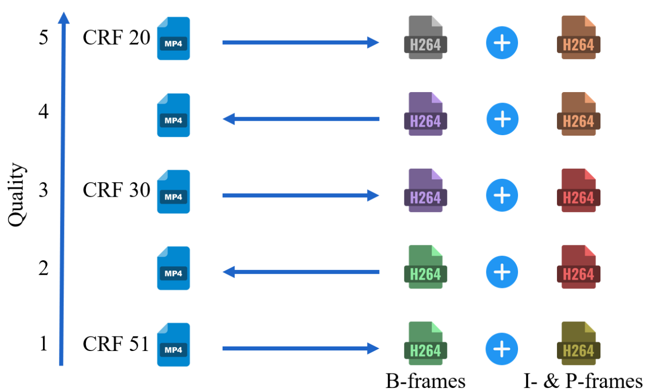

Figure 5: The principle of temporal layer injection.

A single video serves as the focal point, with the flexibility to duplicate it when multiple videos are necessary. The chosen video is "Avatar: The Way of Water | Official Teaser Trailer" [4], capped at a length of 96 seconds for the bandwidth experiments. From this video, five quality layers are encoded, and each layer is available in either 1 or 4-second segments. An overview of the bitrate ladder is given in Table 1. Quality representations 2 and 4, encoded using TLI, utilize frames from other representations, as shown in Figure 5. Both representations use the following rule: B-frames come from one id lower, while I- and P-frames come from one id higher. For instance, if the proxy cache contains quality representation 1, it only need to fetch I- and P-frames from representation 3 or 2 to generate representation 2. Similarly, if representation 4 is cached, the proxy only fetches B-frames from quality representation 5 to obtain representation 5.

4.3 Bandwidth-reducing use cases

In all experiments, the bandwidth is measured on the interfaces rather than within the application. This approach aims to provide a more accurate representation of real-life bandwidth utilization.

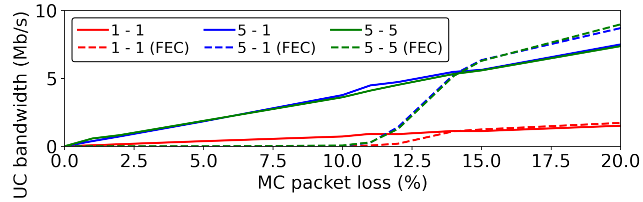

Figure 6: Impact of packet loss on the required unicast bandwidth. The first number in the legend stands for $n_{proxies}$, while the second number stands for $n_{clientsperproxy}$.

4.3.1 Use case 1: OTT unicast only vs 5G broadcast + PR. The first use case compares a unicast-only approach with a broadcast-only approach. Only the highest quality level of 4-second segments is available in this scenario. Next, losses are introduced on the multicast link to display the system's ability to recover using PR over unicast while maintaining optimal video quality. Generally the unicast traffic is expected to adhere to the following linear function: $Bandwidth_{UC} = Bandwidth_{MC} \times Loss \times n_{proxies}$. When multiple clients watch the same video behind a proxy, no major increase of bandwidth is to be expected. Minor deviations are possible due to the measuring on the interfaces, but Figure 6 shows that the results from the experiments align with the linear function as expected.

4.3.2 Use case 2: 5G broadcast + PR + FEC. The effect of applying 15% Raptor FEC overhead is displayed in Figure 6 as well. Under 10% loss, FEC can recover almost all cases. Between 10% and 15%, the chances of decoding failure increase drastically. Above 15%, not enough symbols will arrive to be able to decode content of the buffer, thus all the missing symbols need to be requested. Consequently, the unicast bandwidth trajectory now mirrors the linear function observed previously, with 15% higher multicast and unicast bandwidth. Future work could implement better FEC schemes, such as RaptorQ [26], to improve coding efficiency and drastically reduce the UC bandwidth overhead. With RaptorQ, one only needs to recover $K_{max} + 2$ symbols for a 99.9999% chance of decode success [3], with K_{max} the maximum number of symbols per block.

4.3.3 Use case 3: 5G broadcast + PR + FEC + OTT unicast offloading. This use case advances from the second scenario and delves into the dynamic integration of hybrid 5G terrestrial broadcasting with existing mobile OTT streaming. It considers ten generated popularity distributions of multiple channels based on the Zipf-like probabilities from before, offering insights for creating a dynamic, adaptive streaming platform that balances bandwidth usage based on stream popularity and availability. Here, a proxy only handles the channels that are being watched by its clients. It will not introduce PR bandwidth for channels that are not of interest to its clients. The choice of proxy placement at the home gateway simulates a common real-world scenario and enhances the view of how a situation with a greater number of proxies with fewer clients per proxy would work. For these evaluations, a theoretical multicast bandwidth cap of 25 Mb/s is established, and 3 distinct strategies are tested.

First, the 3 most popular channels are selected and broadcast at the highest quality (representation 5). The remaining 22 channels are made available only through unicast, maintaining a restriction

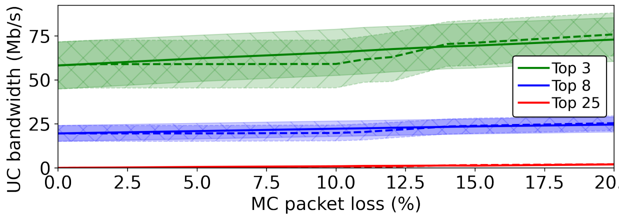

Figure 7: Required unicast bandwidth as a function of packet loss, for different broadcasting strategies with 5 proxies and 5 clients per proxy. Dashed lines are with FEC enabled.

to the highest quality as well. This approach aims to serve the highest quality to 62.99% of all viewers.

Second, extend the broadcasting coverage to include the 8 most popular channels, delivering them all at medium quality (representation 3). The 17 other channels are accessible solely through unicast, with all channels restricted to medium quality. This approach aims to serve the medium quality to 82.51% of all viewers.

Third, broadcast all 25 streams at the lowest quality. All 25 channels are broadcast, and users can access them at the lowest quality level (representation 1). This approach aims to serve the lowest quality to all viewers.

Figure 7 illustrates the results of implementing these strategies based on the generated distributions, with a 95%-confidence interval shaped by the number of proxies, clients, and distributions. Despite the small number of proxies used, the findings underscore the validity of each strategy. Adopting a strategy of broadcasting the top 3 channels proves practical in scenarios where most users opt for the highest quality, providing viewers with uninterrupted access to high-quality content. Here, the unicast bandwidth went from 78.8 Mb/s to 58.2 Mb/s. Similarly, streaming the top 8 channels at medium quality results in notable efficiency gains. In the absence of multicast, the unicast bandwidth requirement is measured at 24.6 Mb/s. Introducing broadcast reduces the bandwidth to 19.6 Mb/s. Broadcasting all 25 channels at their lowest quality ensures basic quality accessibility for all users, particularly beneficial in environments with limited unicast bandwidth.

Combining multicast strategies can provide a nuanced solution. By allocating specific multicast channels to top-performing video streams at higher quality levels, followed by the next set of streams at medium quality, and filling the remaining multicast bandwidth with less popular streams at the lowest quality, a well-rounded approach is achieved. This strategy ensures that a significant portion of viewers, approximately 62.99%, receive their stream at the highest quality, while 24.56% receive medium quality streams, and 12.45% receive streams at the lowest quality.

4.3.4 Use Case 4: 5G broadcast + PR + FEC + TLI. Building on use case 2, this scenario introduces TLI to enhance the adaptability of the system. TLI enables the broadcasting of a certain quality and allows clients/proxies to improve the quality by requesting better frames over unicast if sufficient bandwidth is available. Initially, the livestream is distributed to five proxies, each with five clients watching at quality level 4, while multicasting is conducted at quality level 3. The results in Figure 8 illustrate that with multicast enabled, proxies only need to request I- and P-frames of quality

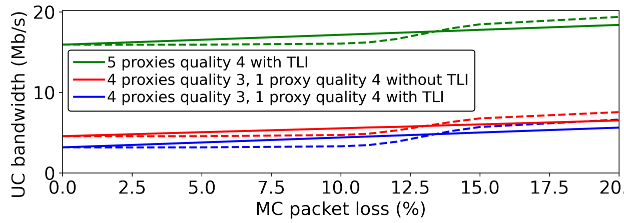

Figure 8: Required unicast bandwidth as a function of packet loss, with or without TLI. Dashed lines are with FEC enabled.

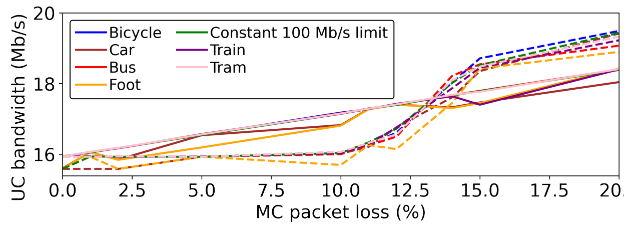

Figure 9: Required unicast bandwidth as a function of packet loss, using 4G/LTE bandwidth traces. Dashed lines are with FEC enabled.

level 5 to obtain quality level 4, as B-frames can be obtained from segments of the mutlicasted quality level 3. Consequently, the unicast bandwidth is reduced from 18.4 Mb/s to 15.9 Mb/s, when no multicast loss occurs. Another test focuses on only a single proxy serving clients at a higher quality, reducing the unicast bandwidth from 4.556 Mb/s to 3.19 Mb/s, a significant 30% decrease, showcasing the efficiency gains achieved with TLI. Finally, real-world bandwidth traces [40] are applied to the interfaces of the unicast switch with 5 proxies. Here, the clients can choose to watch at quality 3 or 4. The results in Figure 9 demonstrate a consistent trend despite variations in bandwidth. All clients attempt to watch at quality 4, proving the feasibility of TLI.

4.4 Latency-reducing experiments

4.4.1 Impact of video parameters. The impact of video resolution, segment duration, and CMAF-CTE on resulting video bitrates is investigated using the ffmpeg command line [12] with the x264 codec. The bitrates of the avatar trailer at 1920x816 resolution compressed at CRF 18, for a 4-second segment without CMAF-CTE, 1-second segment without CMAF-CTE, and a 4-second segment with CMAF-CTE are 6.66 Mb/s, 6.99 Mb/s and 6.66 Mb/s respectively. Additionally, Table 2 shows the bitrates for different chunk durations. These results show that lower segment durations lead to higher bitrates due to the inclusion of additional IDR-frames. However, when CMAF-CTE is used, the increase in bitrate is not significant, Additionally, lowering the chunk durations results in a limited increase in bitrate.

First, a live stream with 4-second segments was multicast, aiming for minimal live latency. The average achieved live latency was 5.6 seconds. A deadline for package recovery is sought in this use case as well. When properly configured, FEC does not increase live latency for 4-second segments. Figure 10 shows the time required to complete multicast reception handling after requesting missing

Table 2: Observed bitrates for CMAF-CTE encoded segments at a resolution of 1920x816 and a CRF of 18, for different chunk durations.

Segment (s)	Chunk (s)			
	0.5	1 s	2 s	4 s
1	6.99	6.99		
2	6.76	6.76	6.76	
4	6.66	6.65	6.65	6.65

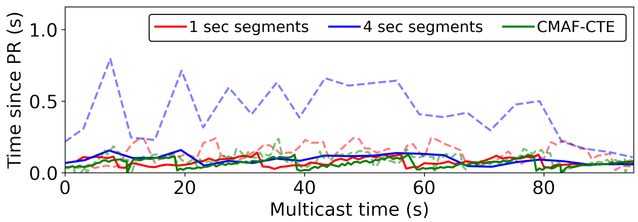

Figure 10: Time measured to complete multicast reception after PR triggers with 20 % loss. Bigger file sizes require more time. Enabling FEC (dashed lines) significantly increases the required time.

symbols over unicast. This metric includes server preparation time for recovery. To reduce preparation time, retaining processed files briefly in memory after multicast transmission could be beneficial, particularly with FEC, as it would skip the encoding step.

Figure 11 illustrates that encoding and decoding time increases with file size. Specifically, it shows the time taken for segment recovery for 4-second durations under varying loss scenarios. Recovery time escalates with increased loss but is mostly influenced by total file size rather than loss extent.

Based on observed results, it is recommended, for this specific scenario, to set the recovery trigger 200 ms before the end of the time frame for 4-second segments without FEC, and 900 ms before with FEC, to optimize the recovery process and ensure timely playback under up to 20% loss. For higher loss rates, adjusting the triggers accordingly is advised, such as setting them to 400 ms without FEC and 1 second with FEC for 80% loss.

In the next experiment the system streams segments with a duration of 1 second, resulting in increased key frames and higher bandwidth requirements. Consequently, more throughput is demanded from the server within a smaller time frame. An average live latency of 2.5 seconds is achieved with this approach. The recovery trigger is ideally set at least 400 ms before the end of the time frame, as depicted in Figure 7. However, this setup leaves only 600 ms for multicast completion, which may be insufficient in some cases. Increasing live latency could mitigate potential issues, as discussed in Section 4.4.1. Additionally, the measured buffer length indicates a significant reduction compared to the previous test due to lower latency. However, a smaller buffer makes the livestream less stable, with instances where the buffer reaches zero, causing brief pauses in playback.

For the last experiment, CMAF-CTE is implemented on 4-second segments, each containing 8 chunks of 0.5 seconds each. Using this configuration as the multicast time frame, a live latency of 2.52

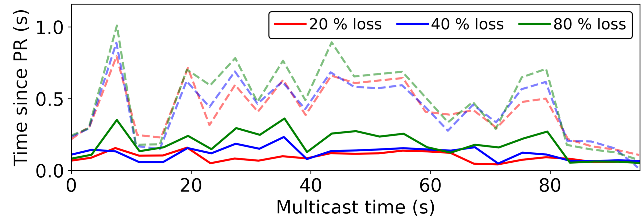

Figure 11: Time measured to complete multicast reception after recovery for different network loss rates. Dashed lines are with FEC enabled.

seconds is achieved, similar to the previous test. However, a notable difference can be observed: the buffer never runs empty, and therefore, the playback rate remains consistent. This finding suggests that reducing the buffer to achieve a lower live latency, would introduce the issues encountered in the previous test. A recovery trigger set 250 ms before the end of the time frame is effective for this test. However, this setting leaves only half of the time frame for the multicasting process, and thus requires more bandwidth.

5 CONCLUSIONS

The hybrid adaptive streaming approach, which integrates 5G terrestrial broadcasting with FEC and package recovery, presents a robust and scalable solution for reducing unicast traffic while optimizing resource utilization. Seamlessly incorporating package recovery mechanisms ensures continuous streaming in lossy environments, while reducing unicast bandwidth to effectively 0 Mb/s in lossless scenarios without FEC and even 0 Mb/s in low loss scenarios with FEC. Real-world emulations demonstrate the effectiveness of prioritizing popular content, balancing quality and bandwidth efficiency, and ensuring basic availability for all viewers. The introduction of temporal layer injection further enhances resource utilization, offering dynamic adjustments to streaming quality with less significant bandwidth increases. Next, this approach also achieves sub 3-second live latency, meeting the evolving demands of multimedia content delivery with improved user satisfaction and efficiency in streaming services. Future work could expand on this approach by integrating the ROUTE protocol. Additionaly, exploring FEC schemes such as RaptorQ could further improve the bandwidth and latency aspect. And finally, adapting the system for volumetric media would extend the applicability to emerging immersive content formats.

ACKNOWLEDGMENTS

This work has been funded by Flanders Innovation & Entrepeneurship (5G BROADCAST, Grant Agreement HBC.2021.0674, https://www.imec-int.com/en/research-portfolio/5g-broadcast)

REFERENCES

[1] [n. d.]. TV | CIM — cim.be. https://www.cim.be/televisie?type=yearly_shares&year=2022&region=north. [Accessed 01-02-2024].

[2] [n. d.]. Virtual Wall. https://doc.ilabt.imec.be/ilabt/virtualwall/. [Accessed 01-02-2024].

[3] 2010. *RaptorQ™ Technical Overview.* Technical Report. QUALCOMM Incorporated. https://www.qualcomm.com/content/dam/qcomm-martech/dm-assets/documents/RaptorQ_Technical_Overview.pdf

[4] 20th Century Studios. 2022. Avatar: The Way of Water | Official Teaser Trailer. www.youtube.com/watch?v=a8Gx8wiNbs8

[5] Kasidis Arunruangsirilert, Bo Wei, Hang Song, and Jiro Katto. 2022. Performance Evaluation of Low-Latency Live Streaming of MPEG-DASH UHD video over Commercial 5G NSA/SA Network. In *2022 International Conference on Computer Communications and Networks (ICCCN).* 1–6. https://doi.org/10.1109/ICCCN54977.2022.9868877

[6] Abdelhak Bentaleb, May Lim, Mehmet N. Akcay, Ali C. Begen, Sarra Hammoudi, and Roger Zimmermann. 2023. Toward One-Second Latency: Evolution of Live Media Streaming. arXiv:2310.03256 [cs.NI]

[7] Divyashri Bhat, Amr Rizk, and Michael Zink. 2017. Not so QUIC: A Performance Study of DASH over QUIC. In *Proceedings of the 27th Workshop on Network and Operating Systems Support for Digital Audio and Video* (Taipei, Taiwan) *(NOSSDAV'17).* Association for Computing Machinery, New York, NY, USA, 13–18. https://doi.org/10.1145/3083165.3083175

[8] Mike Bishop. 2022. HTTP/3. RFC 9114. https://doi.org/10.17487/RFC9114

[9] Lee Breslau, Pei Cao, Li Fan, Graham Phillips, and Scott Shenker. 1999. Web caching and Zipf-like distributions: evidence and implications. In *IEEE INFOCOM '99. Conference on Computer Communications. Proceedings. Eighteenth Annual Joint Conference of the IEEE Computer and Communications Societies. The Future is Now (Cat. No.99CH36320)*, Vol. 1. 126–134 vol.1. https://doi.org/10.1109/INFCOM.1999.749260

[10] Inc. Cloudflare. [n. d.]. What is MPEG-DASH? | HLS vs. DASH. https://www.cloudflare.com/learning/video/what-is-mpeg-dash/. [Accessed 01-02-2024].

[11] Marius Corici, Konstantinos Liolis, Christos Politis, Alexander Geurtz, Joe Cahill, Shane Bunyan, Thomas Schlichter, Florian Völk, and Adam Kapovits. 2020. Satellite is 5G. (nov 2020).

[12] FFmpeg Developers. 2023. ffmpeg tool. https://ffmpeg.org

[13] Jeta Dobruna, Enjesa Spahiu, Matevž Pogačnik, and Mojca Volk. 2022. 5G Streaming: IP-based vs. High-Power High-Tower broadcast. In *2022 45th Jubilee International Convention on Information, Communication and Electronic Technology (MIPRO).* 1507–1511. https://doi.org/10.23919/MIPRO55190.2022.9803623

[14] Wesley Eddy. 2022. Transmission Control Protocol (TCP). RFC 9293. https://doi.org/10.17487/RFC9293

[15] Roy T. Fielding and Julian Reschke. 2014. Hypertext Transfer Protocol (HTTP/1.1): Message Syntax and Routing. RFC 7230. https://doi.org/10.17487/RFC7230

[16] David Gomez-Barquero, Jordi Joan Gimenez, and Roland Beutler. 2020. 3GPP enhancements for television services: LTE-based 5G terrestrial broadcast. *Wiley Encyclopedia of Electrical and Electronics Engineering* (2020).

[17] Europeon Telecommunications Standards Institute. 2023. DVB-I service delivery over 5G Systems; Deployment Guidelines. https://www.etsi.org/deliver/etsi_tr/103900_103999/103972/01.01.01_60/tr_103972v010101p.pdf. [Accessed 01-02-2024].

[18] International Organization for Standardization (ISO). 2020. *Information technology - Multimedia application format (MPEG-A) - Part 19: Common media application format (CMAF) for segmented media.* Technical Report. https://www.iso.org/standard/79106.html

[19] International Organization for Standardization (ISO). 2022. *Information technology - Dynamic adaptive streaming over HTTP (DASH) - Part 1: Media presentation description and segment formats.* Technical Report. https://www.iso.org/standard/83314.html

[20] Dongyub Ko, Kijong Koo, and Do Young Kim. 2016. Reducing the decoding complexity of RaptorQ codes for delay sensitive applications using a simplified and scaled-down matrix. *AEU - International Journal of Electronics and Communications* 70, 9 (2016), 1356–1360. https://doi.org/10.1016/j.aeue.2016.05.006

[21] Hung T. Le, Thoa Nguyen, Nam Pham Ngoc, Anh T. Pham, and Truong Cong Thang. 2018. HTTP/2 Push-Based Low-Delay Live Streaming Over Mobile Networks With Stream Termination. *IEEE Transactions on Circuits and Systems for Video Technology* 28, 9 (2018), 2423–2427. https://doi.org/10.1109/TCSVT.2018.2850740

[22] Jae-young Lee, Sung-Ik Park, Hyun-Jeong Yim, Bo-Mi Lim, Sunhyoung Kwon, Sungjun Ahn, and Namho Hur. 2020. IP-Based Cooperative Services Using ATSC 3.0 Broadcast and Broadband. *IEEE Transactions on Broadcasting* 66, 2 (2020), 440–448. https://doi.org/10.1109/TBC.2020.2983301

[23] Carlos M. Lentisco, Luis Bellido, Andrés Cárdenas, Ricardo Flores Moyano, and David Fernández. 2023. Design of a 5G Multimedia Broadcast Application Function Supporting Adaptive Error Recovery. *IEEE Transactions on Multimedia* 25 (2023), 378–388. https://doi.org/10.1109/TMM.2021.3126458

[24] Hannes Mareen, Peter Lambert, and Glenn Van Wallendael. 2023. Temporal Layer Injection for Fast Bitrate Ladder Creation in Live Video Streaming. *IEE International Symposium on Multimedia* (2023).

[25] De Mi, Joe Eyles, Tero Jokela, Swen Petersen, Roman Odarchenko, Ece Öztürk, Duy-Kha Chau, Tuan Tran, Rory Turnbull, Heikki Kokkinen, et al. 2020. Demonstrating immersive media delivery on 5G broadcast and multicast testing networks. *IEEE Transactions on Broadcasting* 66, 2 (2020), 555–570.

[26] Lorenz Minder, Amin Shokrollahi, Mark Watson, Michael Luby, and Thomas Stockhammer. 2011. RaptorQ Forward Error Correction Scheme for Object Delivery. RFC 6330. https://doi.org/10.17487/RFC6330

[27] Utilities One. 2023. Analyzing the Impact of Network Congestion on Wireless Video Streaming — utilitiesone.com. https://utilitiesone.com/analyzing-the-impact-of-network-congestion-on-wireless-video-streaming#anchor-1. [Accessed 01-02-2024].

[28] Toni Paila, Rod Walsh, Michael Luby, Vincent Roca, and Rami Lehtonen. 2012. FLUTE - File Delivery over Unidirectional Transport. RFC 6726. https://doi.org/10.17487/RFC6726

[29] Roger Pantos and William May. 2017. HTTP Live Streaming. RFC 8216. https://doi.org/10.17487/RFC8216

[30] Jon Postel. 1981. Internet Protocol. RFC 791. https://doi.org/10.17487/RFC0791

[31] Vincent Roca, Mark Watson, and Ali C. Begen. 2011. Forward Error Correction (FEC) Framework. RFC 6363. https://doi.org/10.17487/RFC6363

[32] Amin Shokrollahi, Thomas Stockhammer, Michael Luby, and Mark Watson. 2007. Raptor Forward Error Correction Scheme for Object Delivery. RFC 5053. https://doi.org/10.17487/RFC5053

[33] Olimpjon Shurdi and Algenti Lala. 2023. Towards Broadcasting Linear Content over 5G Network. *ESI Preprints* 18 (2023), 513–513.

[34] Daniel Silhavy, Klaus Kühnhammer, Johann Mika, Thomas Stockhammer, and Jordi J Gimenez. 2023. 5G-MAG (Media Action Group) reference tools: Putting 5G in action for media. *SMPTE Motion Imaging Journal* 132, 1 (2023), 18–24.

[35] Daniel Silhavy, David Waring, Dev Audsin, Richard Bradbury, Johann Mika, Klaus Kuehnhammer, Kurt Krauss, and Jordi J. Gimenez. 2023. 3GPP Rel-17 5G Media Streaming and 5G Broadcast powered by 5G-MAG Reference Tools. In *Proceedings of the 2nd Mile-High Video Conference* (Denver, CO, USA) *(MHV '23).* Association for Computing Machinery, New York, NY, USA, 85–90. https://doi.org/10.1145/3588444.3591009

[36] Iraj Sodagar. 2011. The MPEG-DASH Standard for Multimedia Streaming Over the Internet. *IEEE MultiMedia* 18, 4 (2011), 62–67. https://doi.org/10.1109/MMUL.2011.71

[37] Thomas Stockhammer, Daniel Silhavy, Jordi Gimenez, Richard Bradbury, Johann Mika, David Waring, Yiqing Cao, Shilin Ding, and Kurt Krauss. 2023. Media over 5G in Action – Target 2023 for 5G-MAG Reference Tools. In *2023 IEEE International Symposium on Broadband Multimedia Systems and Broadcasting (BMSB).* 1–6. https://doi.org/10.1109/BMSB58369.2023.10211147

[38] Raza Ul Mustafa, Md Tariqul Islam, Christian Rothenberg, Simone Ferlin, Darijo Raca, and Jason J. Quinlan. 2020. DASH QoE Performance Evaluation Framework with 5G Datasets. In *2020 16th International Conference on Network and Service Management (CNSM).* 1–6. https://doi.org/10.23919/CNSM50824.2020.9269111

[39] Raza Ul Mustafa, David Moura, and Christian Esteve Rothenberg. 2021. Machine Learning Approach to Estimate Video QoE of Encrypted DASH Traffic in 5G Networks. In *2021 IEEE Statistical Signal Processing Workshop (SSP).* 586–589. https://doi.org/10.1109/SSP49050.2021.9513804

[40] Jeroen van der Hooft, Stefano Petrangeli, Tim Wauters, Rafael Huysegems, Patrice Rondao Alface, Tom Bostoen, and Filip De Turck. 2016. HTTP/2-Based Adaptive Streaming of HEVC Video Over 4G/LTE Networks. *IEEE Communications Letters* 20, 11 (2016), 2177–2180. https://doi.org/10.1109/LCOMM.2016.2601087

[41] Lorenzo Vicisano, Mark Watson, and Michael Luby. 2010. Asynchronous Layered Coding (ALC) Protocol Instantiation. RFC 5775. https://doi.org/10.17487/RFC5775

[42] Cedric Westphal, Stefan Lederer, Christopher Mueller, Andrea Detti, Daniel Corujo, Jianping Wang, Marie-Jose Montpetit, Niall Murray, Christian Timmerer, Daniel Posch, Aytac Azgin, and Will (Shucheng) LIU. 2016. Adaptive Video Streaming over Information-Centric Networking (ICN). RFC 7933. https://doi.org/10.17487/RFC7933

[43] Waqar Zia, Thomas Stockhammer, Lena Chaponniere, Giridhar Mandyam, and Michael Luby. 2022. Real-Time Transport Object Delivery over Unidirectional Transport (ROUTE). RFC 9223. https://doi.org/10.17487/RFC9223