Enabling eBPF-based packet duplication for robust volumetric video streaming

Peng Qian*, Ning Wang*, Foh Chuan Heng*, Jia Zhang†, Carl Udora*, Rahim Tafazolli*
*5GIC&6GIC, Institute for Communication Systems (ICS), University of Surrey, Guildford, Surrey, U.K.
† Department of Computer Science and Technology, Tsinghua University, Zhongguancun Laboratory, Beijing, China
Email: {peng.qian, n.wang, c.foh, cu00029, r.tafazolli}@surrey.ac.uk, joycezhangjia@gmail.com

***Abstract*—Volumetric video streaming, an innovative media application, facilitates the real-time and immersive teleportation of individuals or objects into the virtual environment of the audience. Unlike conventional video streaming applications, volumetric content is particularly vulnerable to network fluctuations, which can lead to performance degradation such as reduced FPS, delayed frame delivery. In this research, we introduce an eBPF-based network function that duplicates packets along the pathways between network nodes, ensuring timely packet delivery amid network instability. Furthermore, we propose a path elimination algorithm to discard paths incapable of delivering frames within the target latency. Our implementation and evaluation validate the rapid and robust performance achieved across various resolution levels.**

***Index Terms*—Volumetric video streaming, extended reality (XR), eBPF**

I. Introduction

Volumetric video streaming is a type of extended reality (XR) media application that can enable real-time content presentation in the form of 6 degrees of freedom, offering users enhanced visual freedom and even diverse interaction involving multiple sensory perceptions. Different from the conventional media types, the rapid increment of capturable points number beyond the million level [1] and the booming evolution of the end user device bring unprecedented challenges for assuring satisfactory Quality of Experience (QoE) to volumetric streaming users.

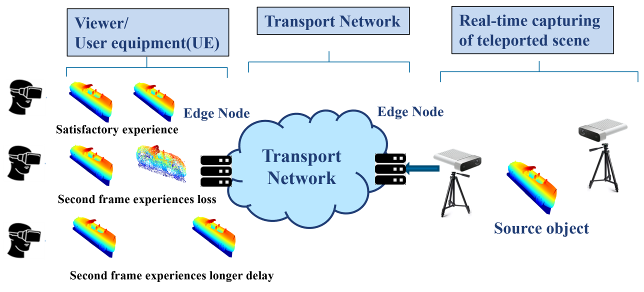


Fig. 1: Remote volumetric video streaming

Fig. 1 depicts a volumetric video streaming application, with a volumetric camera capturing the image of a real-time object and transferring it to a remote user through the transport network. The required transmission and computation capabilities have been overwhelmed by the unaffordable point scale, which can reach up to multiple GB/s levels [1]. Consequently, ensuring satisfactory user experience of volumetric streaming flow when traversing a transport network connecting two sites faces unprecedented challenges, due to its sensitivity to latency and packet loss. Recent reports on commercial transport network measurements covering intra-continental and intercontinental scenarios reveal a noticeable observation [2], [3], [4]. Despite the boosted bandwidth and latency which can gradually support worldwide volumetric streaming [27], network uncertainties like random packet loss and variant latency persist, exhibiting either periodic or stochastic behaviour, leading to increased difficulties in obtaining fine-grained and accurate measurement results [2], [3], [4]. More importantly, the impact of these network fluctuations on volumetric video streaming is noteworthy. Any packet loss or delay causing interrupted or significant delays between frames can result in interrupted point cloud rendering and then fragmented volumetric frame displays. This can further lead to spatial architectural distortions and irregular physical movements in the perceived teleported scenario, disrupting the immersive experience and diminishing the sense of presence and realism for the user.

In this work, towards a robust delivery performance of volumetric video streaming under these inevitable network uncertainties, we propose an intelligent extended Berkeley Packet Filter (eBPF) [5] function pair that duplicates packets across multiple paths in a transport network. There are three key novel components in this solution: The first component is packet duplication. This technique has been primarily applied to small packets [6], [7], with its efficacy has been approved in modern cloud/edge networks' multiple available paths that can ensure low and robust delivery latency without retransmission. This inspires us to implement it to volumetric streaming which demands equally stringent latency. Secondly, eBPF represents a potent kernel-based packet processing technology, with proven superior speed over conventional network packet forwarding techniques [8]. Given the majority of recent research efforts have been paid to its kernel function extension and performance evaluation [9], [10], [11], it is novel and valuable to utilise its diverse in-kernel fast packet processing interfaces to design tailored application improvement. As a multimedia frame with packet duplication may temporarily lead to bandwidth redundancy, the third component entails a successive path elimination algorithm aimed at rapidly trimming unnecessary paths, significantly reducing the total bandwidth cost.

Through the implemented functionalities and test cases conducted on a practical volumetric streaming platform, we demonstrate that this innovative eBPF network functionality consistently achieves stable performance in terms of Frames per Second (FPS) and frame delay, even amidst inevitable fluctuations in the transport network.

This paper is organised as follows: In section II, we provide a literature review of recent efforts on point cloud streaming optimisation and the eBPF evolution. In section III, we provide a detailed illustration of the automatic eBPF packet duplication function deployed in a node pair. Then, we explain the implementation and test platform, with extensive evaluations followed by a solid discussion and conclusion.

II. Related Works

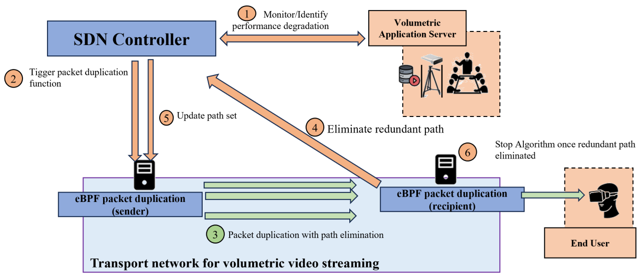
A. *Volumetric video streaming and optimisations*

In recent years, significant research has been focused on minimising the transmission volume costs associated with volumetric frames. In [12], the author examined the combined impact of computational cost and network communication cost, and introduced an innovative resource allocation framework aimed at enhancing QoE by selecting tiles at various resolution levels. The QoE problem is formulated as a non-linear integer optimisation problem, and the evaluation shows user QoE can be improved under different distances to the volumetric object. In [13], the author introduced a framework that generates a 2-Dimensional (2D) view from volumetric video on a cloud server and streams this 2D video to the client. Additionally, to improve the motion-photon latency caused by delayed user head motion, an auto regression model has been designed to predict user behaviours in a 5-window sequence. In [14], the author investigates the potential of machine learning solutions to facilitate immersive experiences for next-generation video services. A novel point cloud streaming method is proposed with a Reinforcement Learning (RL) algorithm, extracting key semantic features for delivery and rendering. Given that these approaches mainly focus on reducing transmitted traffic volume by offline generating the learnable video features, our approach is orthogonal since it solves the problem caused by network uncertainties in real-time streaming.

B. *Transport Layer Multipath adaptation*

Another conventional research direction to enhance content delivery robustness is multipath transport layer solutions to split traffic into different paths by sensing real-time path conditions and carefully deciding the splitting ratio [15], [16]. Similarly as other approach focusing on apply parallel connections on single path (e.g., rate synchronisation [28] and protocol switch [29]), these approaches also face challenges in managing sub-flow on different sockets, with the eBPF solution can natively bounded to physical interfaces with a simple one-off triggering signal. Regarding the redundant steering mode which also generates packet replica on multiple sub-flows, it has been investigated to enhance service reliability and latency, with Load balancing mode [18] or prioritisation replication with Spearman's correlation coefficient [19]. These strategies show potential gains in various aspects, like latency and throughput. Additionally, the emerging UDP-based transport layer Quick UDP Internet Connections (QUIC) is being armed with these multipath abilities [17], [18], with pluggable components of conventional network coding techniques [19]. However, conventional multipath-based solutions have several disadvantages. On the one hand, the transport layer-based Multi-Path TCP (MPTCP) solution requires multiple sockets and connections to be established and maintained, then packets on different connections need to traverse the network stack from kernel space to user space with an additional copy. On the other hand, once a packet is affected by a delay spike, not only unnecessary retransmission and congestion control will be conducted on each path individually, but also the rendering of the frame will be blocked until the data arrives with the risk of out-of-order delivery on multiple paths. Therefore these shortages of the aforementioned work inspire us to apply eBPF-based packet duplication on volumetric streaming flow, which only involves layer 2/3 processing without redundant functionalities.

III. EBPF BASED PACKET DUPLICATION FRAMEWORK



1 Monitor/Identify performance degradation
SDN Controller
Volumetric Application Server
2 Tigger packet duplication function
5 Update path set
4 Eliminate redundant path
6 Stop Algorithm once redundant path eliminated
eBPF packet duplication (sender)
eBPF packet duplication (recipient)
3 Packet duplication with path elimination
End User
Transport network for volumetric video streaming

Fig. 2: The proposed eBPF packet duplication function pair

In this section, we provide an overview of the proposed network function, which is deployed across ingress and egress node pairs in transport networks. This is followed by a detailed component breakdown and a path elimination algorithm designed to reduce unnecessary paths and minimise bandwidth costs

A. *System Overview*

As shown in Fig. 2 the proposed eBPF-based packet duplication relies on two separate network nodes as a sender-receiver pair, coordinating with a centralised Software Defined Networking (SDN) controller to exchange signalling for packet duplication. These network functions nodes are running as daemons, responsible for duplicating packets for volumetric applications along the paths between them in the transport network. Meanwhile, the SDN controller monitors and manages the video streaming flow and instructs nodes to start, stop, or adjust the duplication policy, once it detects the performance of a given volumetric flow is degrading. These operations are automated, with each step detailed as follows:

Step 1, 2: Monitoring the performance of the deployed volumetric application in the transport network and triggering packet duplication

The centralised SDN controller has a global visibility into real-time topology and flow [2], allowing it to detect performance changes in volumetric streaming. Using methods for real-time video QoE estimation [20], the controller

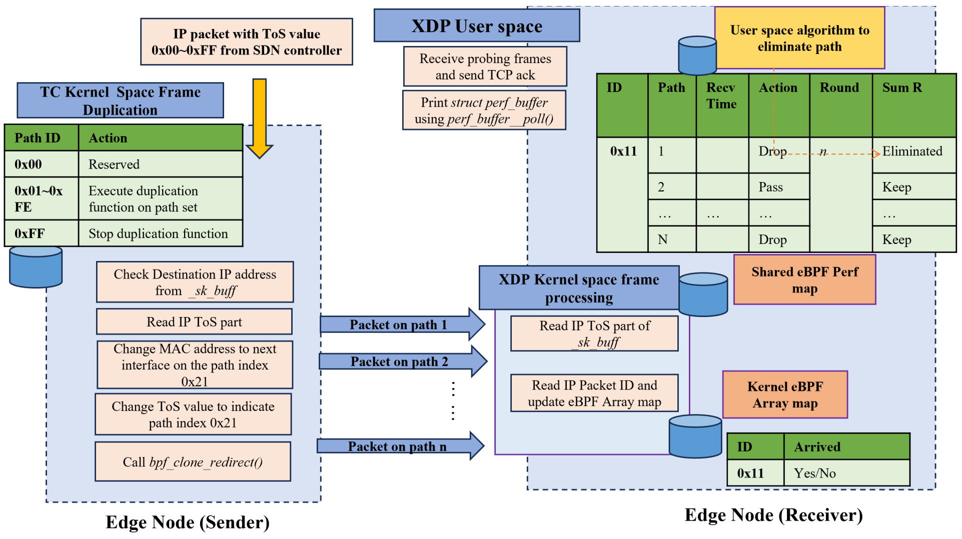

IP packet with ToS value 0x00~0xFF from SDN controller
TC Kernel Space Frame Duplication
Path ID
Action
0x00
Reserved
0x01~0x FE
Execute duplication function on path set
0xFF
Stop duplication function
Check Destination IP address from _sk_buff
Read IP ToS part
Change MAC address to next interface on the path index 0x21
Change ToS value to indicate path index 0x21
Call bpf_clone_redirect()
Packet on path 1
Packet on path 2
Packet on path n
Edge Node (Sender)
XDP User space
Receive probing frames and send TCP ack
Print struct perf_buffer using perf_buffer__poll()
User space algorithm to eliminate path
ID
Path
Recv Time
Action
Round
Sum R
0x11
1
Drop
n
Eliminated
2
Pass
Keep
…
N
Drop
Keep
Shared eBPF Perf map
XDP Kernel space frame processing
Read IP ToS part of _sk_buff
Read IP Packet ID and update eBPF Array map
Kernel eBPF Array map
ID
Arrived
0x11
Yes/No
Edge Node (Receiver)

Fig. 3: The design of the proposed eBPF based packet duplication function pair

adopts a simple policy to monitor the video to identify performance drops. Lacking direct access to transport and application layer data, the SDN controller will mainly focus on the uplink frame request packet, including timestamps and packet sizes, to sufficiently infer consecutive application layer round-time-trip time and therefore real-time video QoE [20]. In practice, our policy triggers the eBPF function if the interval between requests exceeds a pre-defined threshold, for example, 150ms, for three consecutive instances. Upon detection of such performance degradation, the SDN controller will send an IPv4 packet as a signalling message, to the sender side node to initiate packet duplication function. In the IPv4 packet, there is a special indication bit set in the Type-of-Service (ToS) filed in the IPv4 packet, indicating to start of the duplication function.

Step 3: Sending duplicated packet

The sender-side eBPF function will first filter the incoming packets. Once it matches the target volumetric application's destination address (e.g. user device), the packet will be cloned and sent on multiple disjoint paths simultaneously. Accordingly, for the arrived packets with the same packet ID, the receiver node deployed closer to the user device side will only keep the packet that arrive first and drop other replicas, to ensure for each packet ID only one copy will be loaded by the upper layer. The benefit of packet duplication is that despite network fluctuation may change any path to have unpredictable link degradation, the packet can be delivered by the best network resources among them without a complex real-time path quality estimation. Consequently, with the guaranteed delivery performance, both the application server and the user won't perceive any negative effect caused by unpredictable random loss or delay spike.

Step 4,5: Eliminate unnecessary path and update to sender function through SDN controller

To prevent unnecessary frames sent along paths where delivered packet replicas are consistently dropped due to performance constraints, the receiver node will execute a path elimination algorithm to remove such paths. The algorithm is performed per packet ID in a sliding window manner; the input information will be a timestamp of each packet replica, and the output is the path index to be eliminated. Then the decision will be sent to SDN control via Restful API through the signalling path, and SDN will immediately update its local record and forward this updated path set to the sender side function to instruct it to trim the unused path from its packet duplication path set.

Step 6: Stop packet duplication function and restart At the receiver node, once there is only one path in the path set, the algorithm will stop. At the same time, the duplication function is implicitly stopped once the sender side has no more path to be duplicated. If the network instability diminish and multiple paths resume exhibiting comparable performance, the algorithm will automatically detect this and restrict packet forwarding to a single path, thereby ensuring reliable delivery. The condition that can trigger the restart of the packet duplication function is the SDN controller identifies a performance drop of the application again, then it will issue a signalling message to the sender side function to enable all path duplication.

B. eBPF function design in edge node pair

We elaborate on the detailed eBPF program design for both sender and receiver edge nodes (refer to Fig. 3). For the sender part, we develop an eBPF-TC based program in the kernel space to monitor all outgoing packets to the remote content server. By extracting the IP destination from the `_sk_buff` part, the matched packets are internally directed to the duplication program function. The TC kernel program adopts an eBPF hash array map to save the current duplication policy with a value length of 8 bits and a unique key index (e.g. 32-bit destination IP address). The eBPF hash array map is a special data structure designed to support efficient storage and retrieval of key-value data between user and kernel spaces, commonly used for tasks like network analysis and performance monitoring. Unlike the usual hash map data struct in the programming language, the eBPF hash array map can operate within the kernel, enabling shared data access between kernel modules and user programs with ignorable processing delay. Each bit in the 8-bit length represents one of the paths that can be selected. In detail, a default value 0x00 means packet duplication is disabled. Code bits 0x01 to 0xfe are reserved for each path combination. For instance, 0x03 indicates path index two and path index one will be added in the path set, and 0x07 indicates path one, two, three will be enabled. Code 0xff is another indication code to disable the path duplication function, and once the sender side node receives this, it will set current path duplication strategy to be 0x00.

Once the incoming volumetric packets are loaded into the duplication function, the current packet duplication policy will be extracted from the hash map, and the packet will be duplicated to the corresponding network interfaces with `bpf_clone_redirection()` function. To enable the recipient to identify the packet and its route, the function overwrites the ToS value in the IP header with a predefined value indicating each path index. It also updates the destination Media Access Control (MAC) address to the next adjacent interface on the selected path according to the local Address Resolution Protocol (ARP) table. Communication from the sender node to the SDN controller relies on standard IPv4 packets with customised ToS values to manage the duplication policy.

When the duplicated packets are sent, at the receiver side a combination of XDP kernel space and user space program will be enabled to receive and process the packet replicas (see the right block in Fig. 3). In the kernel space program, the packet ID at the IP layer will be extracted from the `sk_buff` structure, and the first packet replica that arrives will be inserted into an eBPF hash array map. The following arrival of the same packet will be dropped by finding out the existence of the same packet ID in the hash map. Additionally, as the packet ID in IPv4 header only has a 16bit length, two packets containing different content but having the same ID may arrive at different stages of a video stream, due to looping index (e.g. 0x0001 to 0xFFFF then circles back to 0x0001 again). Then we set a adjustable time threshold (e.g. 1 second), that if the incoming packet ID is already in the map and the time difference between the stored packet and current packet is less than this threshold, we regard them as belonging to the same packet, and drop the later. Otherwise, the deprecated packet will be removed from the HashMap with the update of the latest packet arrival timestamp. For saving and reporting each packet's arrival time, ID, and other header values to user space for analysis purposes, the XDP kernel program will utilise an eBPF perf output map to store such values. The key in this map is the packet ID, but the value is designed to be a customised structure that can store arrival time, and other useful information from the header (e.g. header length, payload length). Different from the eBPF hash array map, the eBPF perf output map is specially designed to transmit data from the kernel to user space, commonly employed for real-time performance monitoring and tracing purposes without incurring any noticeable delay for the large-scale data processing procedure. At the user space program, it will load the compiled kernel program object and find the corresponding perf output map. The `perf_buffer__poll` function will be called to periodically print the real-time information to a local file, which will be used as input for the path elimination algorithm. For the communication interface to the SDN controller, different from the sender side, the communication relies on a separated user space program to read the output of packet statistics, calculate the plugged path elimination, and trigger a Restful message to the SDN controller.

C. successive path elimination algorithm

Algorithm 1: Sliding Window Based Successive Path Elimination Algorithm

```
Input: $W$: sliding window size, $n$: packet ID, $\alpha$: tolerated packet
       arrival time gap
Input: $P_i \in P$: each path in pathset $P$, $i$: path index
Input: $N_i(n)$: the count of packet arrives first on this path for
       packets $[n, n-W+1]$
Input: $R_i(n)$: the reward of path $i$ for delivering packet $n$
Input: $T_i(n)$: the packet arrival time through path $i$ for packet $n$
Output: trimmed pathset $P$
1  for $n$ in all incoming packets' ID do
2      Observe all packet replicas have arrived, extract $T_i(n)$ from
       eBPF map and $T_{\min}(n)$ denotes the earliest timestamp;
3      for $i$ in $1 \ldots I$ do
4          if $T_i(n) - T_{\min}(n) < \alpha$ then
5              $R_i(n) = 1$;
6              $N_i(n) += 1$;
7          if $n > W$ then
8              $N_i(n) -= R_i(n-W)$;
9              if $N_i(n) == 0$ then
10                 Remove path $i$ from $P$;
11     if $n > W$ then
12         for $i$ in $1 \ldots I$ do
13             if $N_i(n) == W$ then
14                 return $\{i\}$;
15     Save trimmed pathset $P$;
```

Considering the presence of lower-performing paths in parallel routes, which are almost incapable of delivering packets within the desired time, it is imperative to efficiently identify and remove these paths from the set eligible for packet duplication to prevent unnecessary bandwidth wastage. To address this, we propose a sliding window-based path elimination algorithm to remove under-performing paths failing to meet predefined delivery criteria. The design principle is that, instead of tracing detailed information from the transport layer congestion control algorithm, the path reward for delivering each packet is set to be a simple 0-1 based value. In detail, for each packet ID n the receiver node can save the first arrival timestamp as $T_{\min}(n)$, and if other packet replicas arrive within the range of $[T_{\min}(n), T_{\min}(n)+\alpha]$, it will be awarded by one, the path index i reward of the current packet $R_i(n)$ is 0.

As described in Algorithm 1, at the beginning the cumulative reward sum in a sliding window $N_i(n)$ and an instant reward at $R_i(n)$ will be initialised as 0, while $T_i(n)$ stores the packet arrival time of ID t on path i. For every packet received, once the XDP user side code identifies it receives all packet replicas from all the paths in the current path set, it will first set the packet that arrived first as $T_{\min}(n)$, and calculate each replica's arrival time to determine whether it is delivered within a threshold. A packet that arrives within this threshold can be regarded as valuable for the current volumetric flow, and its current reward $R_i(n)$ can be set as one and then increase $N_i(n)$ accordingly.

After updating each packet arrival information of all paths, the path with zero cumulative rewards will be removed if the number of packets that have arrived so far is larger than a predefined window size. Additionally, if any of the paths can have all packets delivered within the acceptable threshold, it will be immediately regarded as the path that can work solely to guarantee the performance, then the algorithm will stop and this path index i will be returned. The window size can be set according to the packet number per frame according to a frame size of different resolution levels.

IV. Implementation and Evaluation results

A. Implementation and evaluation tested

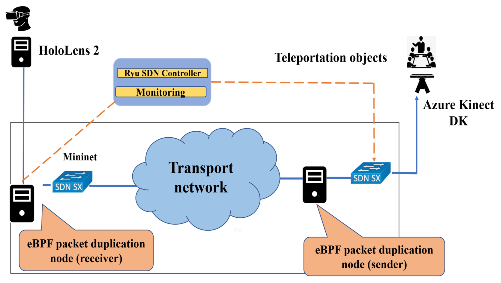
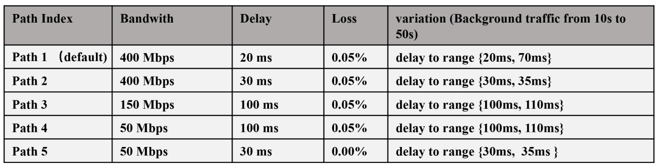


Fig. 4: The implemented test framework

In the implemented test platform (see Fig. 4), the Ryu controller [22] is adopted as the SDN controller, which enables the OpenFlow protocol and Restful API to allow communication between the SDN controller and eBPF nodes. The exchanged IPv4 signaling messages is to activate and modify the packet duplication function. Underlying nodes are created in a Mininet [23] environment, with two nodes designated as edge functions. We deploy a real HoloLens 2 helm for the viewer and a Microsoft Azure Kinect DK camera with an enhanced Livescan3d server [24] as the content source connected to the Mininet testbed. The captured content is a single adult standing 1 meter in front of the camera in a live live-streaming manner. The compression technique adopted is ZSTD [25] since can support low encoding and decoding latency for both on-demand and live streaming, while other approaches require unaffordable time to perform [30]. Regarding the transport network topology, we have drawn inspiration from recent reports in the industry [2], [3], [4] and designed a set of experimental paths to evaluate our framework and algorithms. The design criteria of disjoint paths in the transport network are based on the transmission performance of individual packets (e.g., 1500 Bytes) on these paths, categorised into three types: 1) Each packet is delivered within the required time with no fluctuations 2) Some packets are delivered on time, while others experience significant delays; 3) Almost all packets fail to be delivered within the required time (see in Table 1).Each experiment lasts for 60s and the background traffic starts to cause network variation from 10s to 50s. For the measured results we only show the record from 10s to 50s. By default, the flow is on path 1. Especially, in this emulated transport network, we exclude the case that no path set can support the required bandwidth/delay since this is a problem requiring network capacity expansion and should not be solved by network adaptation techniques. The tolerated delay is set to 5ms and the sliding window is set to 100. The TCP congestion control adopted is BBR [26].

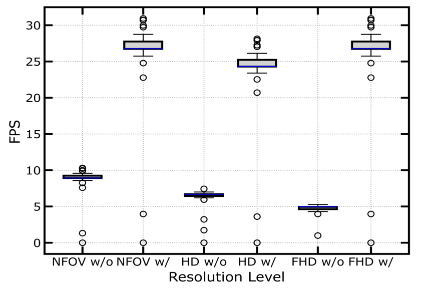
TABLE I: Caption for the table image



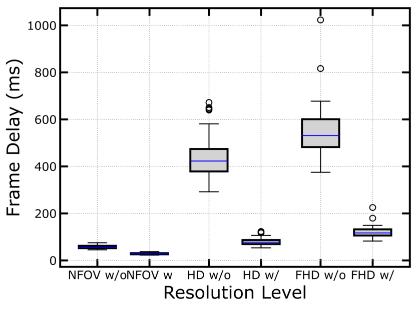
Path Index	Bandwith	Delay	Loss	variation (Background traffic from 10s to 50s)
Path 1 （default)	400 Mbps	20 ms	0.05%	delay to range {20ms, 70ms}
Path 2	400 Mbps	30 ms	0.05%	delay to range {30ms, 35ms}
Path 3	150 Mbps	100 ms	0.05%	delay to range {100ms, 110ms}
Path 4	50 Mbps	100 ms	0.05%	delay to range {100ms, 110ms}
Path 5	50 Mbps	30 ms	0.00%	delay to range {30ms, 35ms }

B. Performance comparison between enabling and disabling packet duplication function

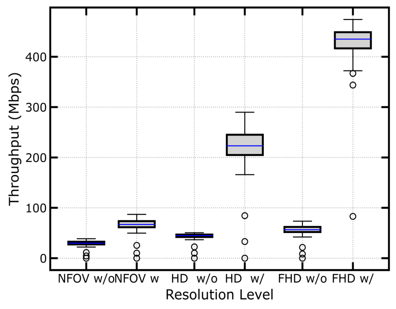
First, we compare the performance improvement over three different resolution levels, which are Narrow Field of View (NFOV), High Definition (HD), and Full High Definition (FHD) (see Fig. 5a, Fig. 5b, Fig. 5c). Obviously, our algorithm can retain almost full FPS performance over all resolution levels, with ignorable variation observed. In contrast, when the flow is confined to a single path but experiences fluctuations, higher-resolution settings become increasingly susceptible to variations in network links. For instance, the FHD resolution level may see a drastic drop in FPS to as low as 3, significantly deteriorating the user experience. The reason behind this huge performance improvement is that the mainstream transport-layer congestion control algorithms rely on per-round estimated delivery rate and congestion to decide the sending rate for the next round. However, RTT variation will affect their inner maintained variables like round-trip propagation time and its standard deviation calculation which then constrain the sending rate at a very low level. Furthermore, the rendering of each frame will be interrupted by the packet retransmission and delay, therefore significantly hurting the user-perceived frames. In contrast, duplicating packets on multiple paths can mitigate such variation by utilising the best path in the transport network at every round. Therefore, the in-time data packet and the corresponding acknowledgement packet can retain the rate estimation to a satisfactory level. Apart from FPS, we continue to compare the frame delay, which is defined as the timestamp that the content source generates a frame so the frame can be fully received and rendered at the client side. Different from FPS which the user device can use a buffer [25] to temporarily absorb the packet insufficiency, any network variation will be directly embodied in the frame delay performance once no network therapy can be applied. Fig. 5b shows the comparison between enabling and disabling packet duplication across different resolution levels. Apparently, the frame delay can be retained at a low level with the help of packet duplication, while the observed variation can be attributed to its varied size, which



(a) Calculated UCB value of each path (under delay variation)

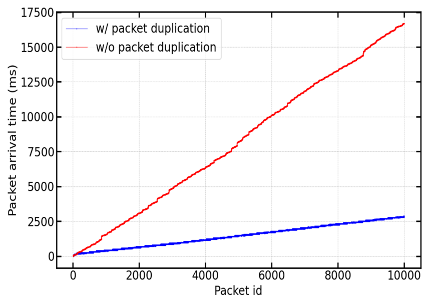


(b) The reward of each path (under delay variation)

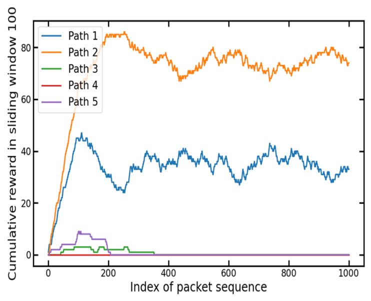


(c) Selected path index (under delay variation)

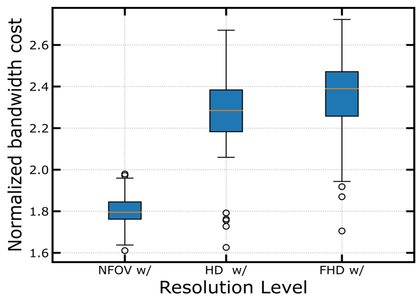
Fig. 5: Performance comparison between enable/disable eBPF duplication



(a) Packet arrival time comparison



(b) Cumulative reward comparison



(c) Normalised bandwidth cost comparison

Fig. 6: Extended analysis of Packet Duplication function

may require different computation and transmission times. Regarding the throughput comparison (see Fig. 5c), around 7 times improvement can be observed at the FHD level due to its high point density, while the difference in NFOV level is less significant.

C. Extended analysis of packet duplication function

A more detailed packet arrival time comparison between enabling and disabling packet duplication of NFOV resolution level can be found in Fig. 6a. Without duplication, random and excessive delays for each packet significantly degrade user experience, impacting frame delay and FPS. However, once link variation is mitigated, packet delivery latency remains low and stable. This leads to robust delivery rate growth without perceived link variation. In Fig. 6b, we depict the cumulative reward of five deployed paths, with a sliding window of 100 packets and a tolerated delay of 5 ms. According to our algorithm, a packet arriving within the tolerated delay receives a reward of one. Path 4, lacking resources for packet delivery, is removed from the path set after 200 windows. Paths 3 and 5, despite contributing initially, perform poorly due to high variation and are subsequently eliminated. Although path 2 is identified as the best, its cumulative reward is still less than 80 within a window, necessitating the retention of default path 1 to ensure robust performance. We calculate the normalised bandwidth cost of packet duplication by summing all transmitted packets. Higher resolutions necessitate more paths to ensure stability due to their larger size and sensitivity to link variation. Enabling packet duplication results in 1.8 times the bandwidth cost for NFOV and 2.4 times for FHD compared to no duplication, as higher resolutions are more susceptible to variation. Paths consistently delivering packets outside the acceptable range are promptly eliminated for lower resolutions, whereas higher resolutions require more rounds to leverage multiple paths, reverting to a single path once variation subsides. The sliding window size and tolerated delay range can be tailored based on historical variation records in the transport network for different resolution levels.

D. Comparison with path redirection function

Then this temporary bandwidth cost inspires us to compare the packet duplication function with the path redirection function, which can select only one path with the least RTT once network variation happens (see Fig. 7). The Y axis shows the throughput improvement compared to no packet duplication applied. Generally, packet redirection achieves less improvement than packet duplication, although it can reduce the traffic volume to approximately the same as the original quantity. However, the performance that a single path can achieve significantly depends on the real-time condition of a network which is highly uncontrollable. In an industrial network, all paths will perform variance and unpredictable conditions [2], [3], [4], and even it will take several seconds to find an optimal path in a large topology case [2]. Therefore, the additional bandwidth cost by path duplication is not only to improve the throughput but also can provide a guaranteed performance under such unpredictable network fluctuations, especially for higher resolution levels which are observed to have larger throughput improvement. The effectiveness of packet duplication versus

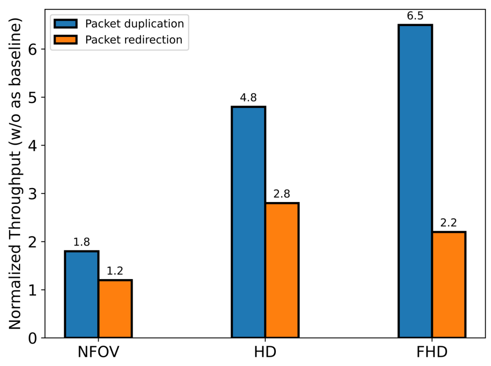


Fig. 7: Comparison with path redirection

redirection hinges on path properties. In practical networks, paths fall into two categories: those meeting video resolution requirements independently and those needing multiple paths to work together for performance. In the former, our algorithm efficiently trims redundant paths, minimising overhead. However, redirecting flow to a single path in the latter case may not ensure satisfactory performance.

V. Conclusion

The emerging volumetric streaming application is sensitive to network fluctuations due to its large frame size and high FPS requirements. To ensure consistent user QoE, we propose an eBPF-based packet duplication function. This function uses kernel TC/XDP packet cloning and forwarding to send volumetric packets on separate paths, achieving up to 7x throughput improvements. Additionally, a path elimination algorithm reduces unnecessary paths based on early real-time duplicate packets, saving bandwidth costs. Evaluations have verified its performance across various scenarios.

VI. Acknowledgment

This project is sponsored by the EU Horizon Europe SPIRIT project (Grant Agreement 101070672, https://www.spirit-project.eu/). The authors would also like to acknowledge the support of 5GIC industry members (5GIC) (http://www.surrey.ac.uk/5gic) for this work.

References

[1] Qian, Peng, et al. "Remote Production for Live Volumetric Teleportation Applications in 5G Networks." IEEE transactions on broadcasting 68.2 (2022): 451-463.
[2] Li, Jinyang, et al. "LiveNet: a low-latency video transport network for large-scale live streaming." Proceedings of the ACM SIGCOMM 2022 Conference. 2022.
[3] Birge-Lee, Henry, Maria Apostolaki, and Jennifer Rexford. "It takes two to tango: cooperative edge-to-edge routing." Proceedings of the 21st ACM Workshop on Hot Topics in Networks. 2022.
[4] Jadin, Mathieu, et al. "Leveraging eBPF to Make TCP Path-Aware." IEEE Transactions on Network and Service Management (2022).
[5] eBPF, [Online]: https://ebpf.io
[6] Aubry, François, et al. "Traffic duplication through segmentable disjoint paths." 2015 IFIP Networking Conference (IFIP Networking). IEEE, 2015.
[7] Jadin, Mathieu, et al. "Leveraging eBPF to Make TCP Path-Aware." IEEE Transactions on Network and Service Management (2022).
[8] Parola, Federico, et al. "Comparing User Space and In-Kernel Packet Processing for Edge Data Centers." ACM SIGCOMM Computer Communication Review 53.1 (2023): 14-29.
[9] Vieira, Marcos AM, et al. "Fast packet processing with ebpf and xdp: Concepts, code, challenges, and applications." ACM Computing Surveys (CSUR) 53.1 (2020): 1-36.
[10] Shahinfar, Farbod, et al. "Automatic Kernel Offload Using BPF." Proceedings of the 19th Workshop on Hot Topics in Operating Systems. 2023.
[11] Enberg, Pekka, Ashwin Rao, and Sasu Tarkoma. "Partition-aware packet steering using XDP and eBPF for improving application-level parallelism." Proceedings of the 1st ACM CoNEXT Workshop on Emerging in-Network Computing Paradigms. 2019
[12] Li J, Zhang C, Liu Z, et al. Joint communication and computational resource allocation for QoE-driven point cloud video streaming[C]//ICC 2020-2020 IEEE International Conference on Communications (ICC). IEEE, 2020: 1-6.
[13] Zhang, Anlan, et al. "YuZu:Neural-Enhanced Volumetric Video Streaming." 19th USENIX Symposium on Networked Systems Design and Implementation (NSDI 22). 2022.
[14] Huang, Yakun, et al. "Towards Holographic Video Communications: A Promising AI-driven Solution." IEEE Communications Magazine (2022).
[15] B. Felix, I. Steuck, A. Santos, S. Secci, and M. Nogueira, "Redundant packet scheduling by uncorrelated paths in heterogeneous wireless networks," in 2018 IEEE Symposium on Computers and Communications (ISCC), 2018, pp. 00 498–00 503.
[16] E. Dong, M. Xu, X. Fu, and Y. Cao, "A loss aware MPTCP scheduler for highly lossy networks," Computer Networks, 2019
[17] Zheng, Zhilong, et al. "Xlink: Qoe-driven multi-path quic transport in large-scale video services." Proceedings of the 2021 ACM SIGCOMM 2021 Conference. 2021.
[18] De Coninck, Quentin, and Olivier Bonaventure. "Multipath quic: Design and evaluation." Proceedings of the 13th international conference on emerging networking experiments and technologies. 2017.
[19] Michel, François, Quentin De Coninck, and Olivier Bonaventure. "QUIC-FEC: Bringing the benefits of Forward Erasure Correction to QUIC." 2019 IFIP Networking Conference (IFIP Networking). IEEE, 2019.
[20] Shen, Meng, et al. "DeepQoE: Real-time measurement of video qoe from encrypted traffic with deep learning." 2020 IEEE/ACM 28th International Symposium on Quality of Service (IWQoS). IEEE, 2020.
[21] Ryu SDN controller [Online]: https://ryu-sdn.org/
[22] Mininet: An Instant Virtual Network on Your Laptop (or Other PC) – [Online]: Mininet http://mininet.org
[23] GitHub - MarekKowalski/LiveScan3D: LiveScan3D is a system designed for real time 3D reconstruction using multiple Azure Kinect or Kinect v2 depth sensors simultaneously at real time speed.
[24] Collet, Yann, and Murray Kucherawy. Zstandard Compression and the application/zstd Media Type. No. rfc8478. 2018.
[25] Selinis, Ioannis, et al. "On the Internet-scale streaming of volumetric-type content with assured user quality of experiences." 2020 IFIP networking conference (networking). IEEE, 2020.
[26] Cardwell, Neal, et al. "BBR: Congestion-based congestion control." Communications of the ACM 60.2 (2017): 58-66.
[27] Selinis, Ioannis, et al. "On the Internet-scale streaming of volumetric-type content with assured user quality of experiences." 2020 IFIP networking conference (networking). IEEE, 2020.
[28] Zhang, Jia, et al. "Reducing Mobile Web Latency Through Adaptively Selecting Transport Protocol." IEEE/ACM Transactions on Networking (2023).
[29] Qian, Peng, Ning Wang, and Rahim Tafazolli. "Achieving robust mobile web content delivery performance based on multiple coordinated QUIC connections." IEEE Access 6 (2018): 11313-11328.
[30] Yang, Mengyu, et al. "A Comparative Measurement Study of Point Cloud-Based Volumetric Video Codecs." IEEE Transactions on Broadcasting (2023).