Enabling User Intent-based Network Path Adaptation for Live Volumetric Streaming

Peng Qian*, Ning Wang*, Foh Chuan Heng*, Carl Udora*, Rahim Tafazolli*
*5GIC&6GIC, Institute for Communication Systems (ICS), University of Surrey
Guildford, Surrey, U.K.
Email: {peng.qian, n.wang, c.foh, cu00029, r.tafazolli}@surrey.ac.uk

***Abstract*—Volumetric video is an emerging media application that enables the projection of people or objects into a virtual space in a real-time and immersive manner. Unlike traditional video, live volumetric video streaming in virtual space allows users to interact with teleported objects with a wide range of intentions. However, a significant technical challenge in this context is the need for real-time network adaptation driven by diverse user intents (e.g., user's movement in the virtual space), which may instantly change the streaming's network demand. To ensure a satisfactory perceived Quality of Experience (QoE) in the face of both intent and network condition uncertainty, we have developed a novel solution framework that allows offline user intent registration and online user intent capture with necessary path adaptation. This path adaptation module is empowered by a novel Multi-Arm Bandit (MAB) based path selection algorithm, with joint consideration of probed application delay and network congestion. Through real and extensive experiments, we have validated the effectiveness of our proposed framework in assuring user QoE under various network conditions and for different user intent scenarios.**

***Index Terms*—Volumetric streaming, extended reality (XR), edge computing, user intent**

I. Introduction

Volumetric video streaming is a novel immersive application that offers six degrees of freedom (6DoF) visual experience, allowing the user to engage in rich interactions to adjust viewing orientation, distance, and interested objects. However, to livestream a volumetric object that compromises a million points over a modern network faces unprecedented challenges. On the one hand, transmission and computation capabilities on both sides of the network and user devices have been overwhelmed by the unaffordable point scale, reaching up to GB/s levels [1]. On the other hand, end devices are being enhanced by interactive interfaces to capture human action (e.g., head motion, random walk, gesture, and eyeball movement). The dynamic and unpredictable nature of user intent behavior can significantly alter the application's network demands in terms of bandwidth and end-to-end latency [2]–[4]., leading to a degraded user experience when network resources fail to be adaptive to these rapid changes in data flow.

Fig.1 illustrates the scenario of a remote live volumetric scenario. One or multiple volumetric cameras placed at a remote location capture a car model, and the real-time raw frames are streamed to an edge node closer to the content source. Upon receiving the raw frames, the edge node combines them into a multi-viewpoint cloud frame and transmits them to the end user through a pre-allocated path in a public transport network. During live streaming, if the viewer is "far away" from a volumetric object in his field of view, streaming at a lower resolution is sufficient to deliver satisfactory image quality with a proper point density [3], [5]. Once the user gradually approaches the object, the original point density becomes visibly sparse, therefore a higher resolution is required to guarantee the same point density in the same area. Consequently, this demand for a high-resolution level can lead to elevated data rates, necessitating bandwidth increases by more than an order of magnitude in immersive applications [1]. In modern network management, to ensure efficient use of network resources and compliance with business constraints, the stream with the lowest resolution is always allocated to a path with limited resources [6], [7]. Therefore, verifying whether the other higher resolution levels can be delivered with a robust user experience on the current path or should be re-directing it to an optimal path with adequate resources becomes an essential task, especially upon the occurrence of a user intent requesting a higher resolution level. Towards this end, there are two technical challenges for enabling such intent-aware network adaptation. First, the network needs to timely capture the incoming user intent in immersive volumetric video applications and instantly understand the accurate new network requirements. Secondly, in modern edge/cloud backhaul links, disparities between multiple paths and random congestion/loss still exist due to geometric differences or varying loads [8], [9]. Therefore, the network should dynamically pair a potential viewer intent (e.g., required resolution levels) with the proper

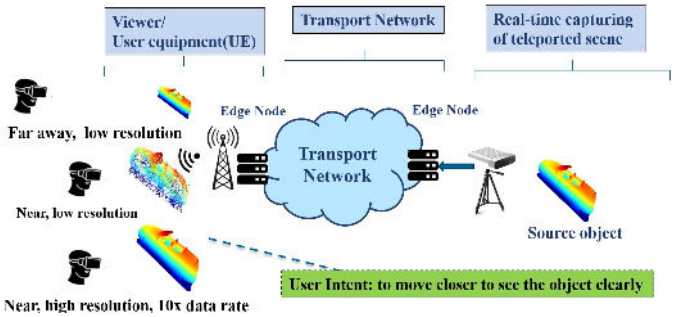


Fig. 1: Concept of remote live volumetric streaming with user intent change (movement)

ISBN 978-3-903176-63-8 © 2024 IFIP

paths that have sufficient resources, with the awareness of real-time link conditions.

These two challenges inspire us to propose a user intent-driven solution based on the Software-Defined Networking (SDN) paradigm. In this framework, user intents can be encoded as simple offline negotiated codepoints and categorized into groups that present different network resource demand levels. Through a well-defined API between the network edge and the user device, these codepoints can be inserted into customized Segment Routing over IPv6 (SRv6) packet header and expressed to the network edge whenever there is any on-the-fly change of user intent during volumetric teleportation session. In order to achieve seamless online path/intent pairing for optimized stream re-direction, the network edge leverages a lightweight probing mechanism to exchange tailored probing packet groups on candidate paths and also a Multi-Arm Bandit (MAB) path selection algorithm to obtain an optimal path for each registered user intent group, which can be immediately applied once a user intent is detected. We implemented the proposed framework and evaluated different user intent types with a wide range of network condition scenarios. The experiment results demonstrate that our proposed solution can successfully capture and identify a wide range of user intent types, thereby ensuring a satisfactory user Quality of Experience (QoE). This improvement is highlighted by significant enhancements in key performance metrics such as Frames Per Second (FPS), frame delay, and throughput, particularly in scenarios where delay and loss occur on specific network links.

This paper is organized as follows: In section II, we provide a literature review of the existing effort of optimizing volumetric content delivery and various network frameworks to enable application-aware information with the help of modern routing techniques. In section III, we present our framework with the awareness of instant user intent and corresponding probing and MAB routing algorithms. Then in section IV, we explain our implementation of the framework and evaluation results, and finally conclude this work in section V.

II. RELATED WORKS

A. Point Cloud streaming and optimizations

To mitigate the challenges of high data transmission for volumetric streaming, recent advances have embraced edge computing to optimize content delivery based on viewer's interactions, such as viewer's location and viewpoint. The authors in [10] employ edge servers to convert volumetric videos into 2D views, reducing motion-photon latency through predictive modeling of user head movements. Similarly, the authors in [11] leverage machine learning techniques to split a large-sized frame into different sub-frames of different viewpoints and conduct customized selection according to the viewer's viewpoint. The implementation shows that this edge server can improve QoE by up to 85%. Parallelly, other Artificial Intelligence (AI) driven approaches focus on extracting essential video features offline [12], thus transmitting minimal data while preserving content integrity at the online phase. Further innovations include filtering techniques [5] that reduce data transmission to essential metadata, enabling high-quality content reconstruction on mobile devices. Advanced methodologies in [13], [14] employ down sampling and reinforcement learning to match features with network conditions effectively, while the authors in [14] explore semantic transmission for volumetric content, drastically reducing data volumes. However, by acknowledging the inevitable long delay (e.g., seconds even minutes for a single frame) for AI-based encoding and decoding [16] which is inherent in live streaming, the current focus shifts towards enabling lightweight compression strategies [1], [5] that facilitate real-time streaming without compromising QoE. However, these initial efforts lack the joint consideration of the varying bandwidth demands and inevitable fluctuations in network quality and user behavior, which are the key issues that our framework aims to address.

B. Network uncertainty and application-aware therapy in modern networks

In modern network topologies, uncertainties such as packet loss and latency are still widespread. To ensure that video applications deployed on top of these networks can achieve stable performance, a variety of routing-based methods and frameworks have been proposed and implemented. For instance, in a study based on multiple path measurements from the East to the West Coast of the United States [9], not only severe latency and packet loss fluctuations were observed, but also the default allocated path's performance can be 30% worse than another alternative path. Similarly, packet losses exceeding a 0.1% probability and latency jitters over 100ms were also observed in the global overlay Alibaba Content Delivery Network (CDN) network [8]. Towards a satisfactory streaming service experience, a centralized SDN orchestrates topology and link management, optimizing video streaming by algorithmically selecting dual alternative paths based on latency and packet loss metrics, significantly mitigating video stalling instances [8]. In the research findings of [17], targeting link failures or congestion can be observed in the topology of cloud networks. To overcome this network uncertainty, a method based on eBPF for detecting redundant links using simple ping packets over multiple paths was proposed. This allows network flows to be correctly and swiftly rerouted to alternative paths in response to link failures. Furthermore, architectures for application-aware rerouting based on SRv6's flexible header encapsulation have been investigated. For instance, the authors in [18] discuss an SDN-based architecture enabling end hosts to request customized paths, enhancing network efficiency through the innovative use of SRv6, which allows for programmable source routing by appending an IPv6 header with segment header extensions. This technique is further leveraged in [19], where SRv6's variable header space encapsulates application characteristics, facilitating network functions to deliver tailored application flows. Contrary to existing approaches, our approach enhances volumetric video streaming by using SRv6 for conveying encoded user intent requests, avoiding the reduced throughput by extra headers on the video data frame [19]. We also designed a special routing

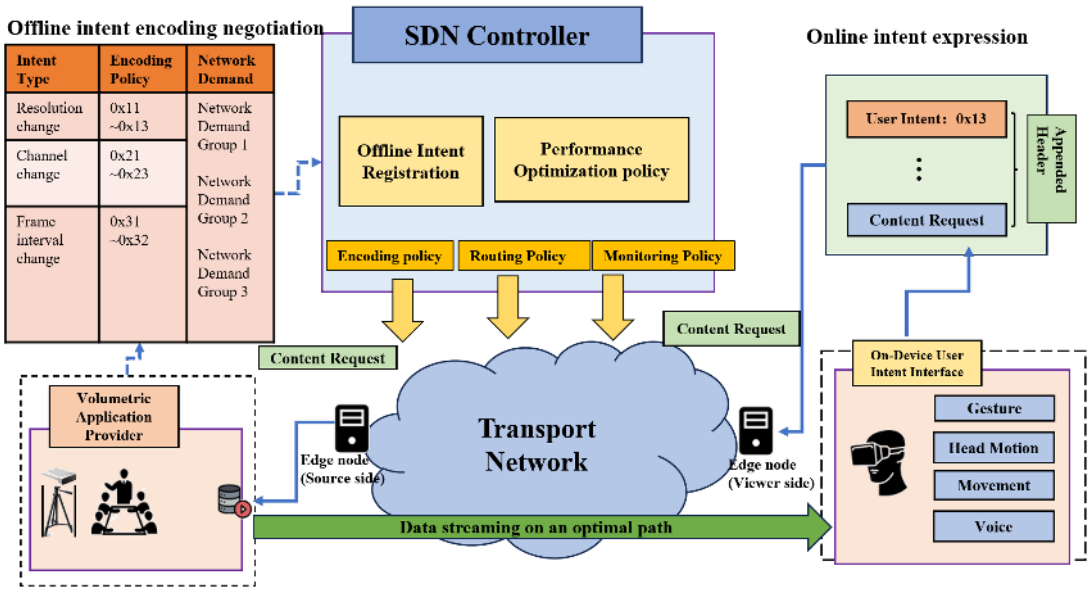


Fig. 2: The proposed User intent-aware framework

algorithm that adapts to different video resolution levels, user intents, and changing network conditions, providing a more customized solution.

III. INTENT-AWARE NETWORK ADAPTATION FRAMEWORK

In this section, we illustrate the proposed intent-aware framework, which aims to solve two key problems: 1) How to seamlessly capture user intent and understand its potential change to flow requirement. 2) How to apply a proper network treatment (e.g., deploy a proper routing path) for the incoming user intent instantly. Toward this end, we proposed an SDN-based framework with an offline intent registration interface and an online intent capture interface deployed at the network edge nodes, with necessary path redirection decisions according to real-time path conditions decided by edge nodes. (see Fig.2).

A. Negotiating user intent encoding policy at the offline phase

The offline interface between the live volumetric application platform and the network framework (see the left-up corner in Fig.2) is responsible for user intent registrations by using a negotiable encoding policy that represents each user intent. This policy comprises an 8-digit code in the SRv6 header's IPv6 flow label: the initial four digits categorize user intent types and the subsequent four denote network resource levels determined collaboratively by content providers and the network. From a network monitoring perspective, user intent behaviours can be generally categorized into the following types: 1) Change in frame resolution: during a live volumetric meeting, the focus of attention may shift from a low-resolution object to a higher-resolution object upon the host's request. Similarly, the application can adjust the object's resolution based on a user's proximity, increasing it for closer distances and vice versa. 2) Change in source channel: users may instantly enter another scenario, temporarily disconnecting their remote source and switching to another streaming connection. 3) Change in frame request interval: for specific object surveillance, users may request a lower frame rate initially but switch to a higher frame rate when a special event occurs. To trigger these intent types, modern extended reality devices like HoloLens 2 expose various interfaces to capture gestures, eyeball tracking, and voice commands. Therefore, it is feasible to aggregate and translate such human-related inputs into the coding level, and then encapsulate them in the encapsulated content request packet header. Regarding the network requirements for each resolution level, the application provider predefines the point density and then negotiates with the SDN controller to agree on the permissible network specifications. For instance, the low-resolution Near-Field-Of-View (NFOV) containing tens of thousands of points requires 60 Mbps and 30 ms latency for full FPS and satisfactory frame delay [1] and will be assigned intent as 0x11. The High Definition (HD) resolution level, which demands 300 Mbps and 20 ms latency for the same performance, can be marked as intent 0x12. Since various user intents might have similar network needs, they can be consolidated into fewer demand groups. For example, resolution changes (intent 0x1) and channel switches (intent 0x2) can be grouped if the demanding resolution is NFOV as 0x11 and 0x21.

By storing such user intents and network requirement mapping rules at the local intent registration function, the SDN controller can facilitate performance analysis history and optimization policy function to generate tailored flow monitoring rule and route optimization policy for each user intent type, and then instruct the underlying edge nodes to start to execute these policies accordingly. These policies can contain application information of frame size, frame interval of

each type of intent, and specific IP 5-tuple of registered flow, a predefined route set that fits each intent group, and hyper-parameters that will be used in the MAB routing probing and selection algorithm later. Meanwhile, the application provider can also offline update its application software on the user side to enable the SRv6 encapsulation feature along with the application message (e.g., HTTP message) once user intent triggers. In detail, the IPv6 address of the network edge node will be set as the target IPv6 address, with the 20-bit long flow label in the IPv6 head space to indicate the user intent type and corresponding network resource requirement level. It is worth noting that compared to other online application negotiated frameworks, this bit encoding design has several advantages: 1) First, encoding user behavior in offline negotiated coding policy can prevent the risk of exposing user bio-information (e.g., eye tracking, physical position) through online packet inference, while other methods also explicitly reveal detailed bandwidth and delay requirements on the packet header. 2) Second, directly mapping coding bits to pre-defined network resource levels can eliminate the online routing computation overhead, while other solutions [18], [19] still require online computing for incoming packets. 3) Initiating intent from the client side is sticking to the principle of mainstream application protocol. For instance, in HTTP/2, any push content from a server should require the client's acceptance in advance [20].

B. Processing online user intent and performing network adaptation

Then we use a user proximity-driven resolution change scenario to illustrate the online intent processing procedure in Fig.3. On the network side, by receiving intent encoding policy and target flow level IP 5-tuple information from the SDN controller, the two edge nodes deployed at the viewer side and source side are responsible for capturing incoming SRv6-based intent packets and performing necessary network path adaptation. For instance, once a user is moving closer to a target object and the application on the helm decides to change to Full HD resolution, it will send a packet with a 0x13 flow label to the viewer side edge node. Then the viewer side edge nodes will extract the IPv6 flow label and conduct intent verification. The intent verification policy is pre-instructed from the SDN controller since the SDN controller is able to load the user's network registration information like mobility, billing information, and registered QoS level through a proper interface.

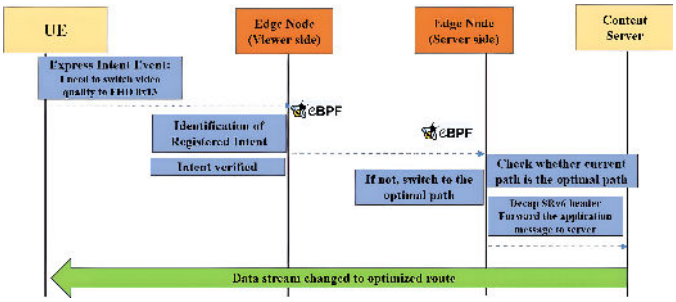


Fig. 3: Online User intent processing (quality switch)

Especially, towards a minimized parsing and forwarding time, the edge node adopts an XDP eBPF program [21] for parsing the incoming IPv6 header by inspecting the sk_buff struct. In detail, first, the edge node can locate $ethhdr$ via $skb->data$ object which indicates the layer 2 part of the packet, and locate IP layer protocol and position (e.g., IPv6 header in SRv6) via $ethhdr->protocol$. By moving the pointer to the SRv6 header, it can extract the flow label from the 12 to 31 bits, and match this to a predefined local intent encoding policy stored. If there are no predefined encoding bits, the SRv6 header will be removed, and the packet will be forwarded directly to the content source. If the content is matched and the XDP user space program which maintains the up-to-date user information successfully verifies such intent, it will be forwarded to the source side edge node with an XDP PASS action. The source-side edge node closer to the teleportation source is responsible for probing the pre-planned paths provided by the SDN controller and maintaining an optimal path set for different user intent types. When there is no intent packet received, it conducts path probing and executes an MAB routing algorithm to maintain an optimal path set for each user intent group. Once it receives the user intent packet (0x13), it will read the IPv6 flow part (following the same eBPF logic as the viewer side edge node) and perform the path change if the current path does not have adequate resources to guarantee the quality of Full High Definition (FHD) resolution level. Then it will decapsulate the SRv6 header and forward the original application message to the application platform.

C. Online path probing and selection algorithm

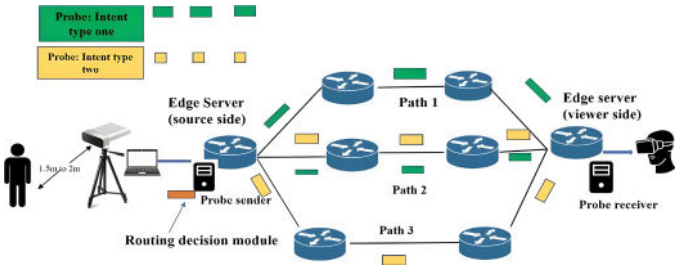


Fig. 4: path probing for user intent types

Subsequently, we delve into the path probing and selection mechanisms, deployed at both the source side and viewer side edge nodes, to periodically probe path conditions and maintain a re-direction path set by a MAB algorithm. We first examine the path probing mechanism, where the source edge node sends tailored TCP packets to the viewer node, recording their delay and transport layer data as input for routing selection (see Fig.4). Due to rare user intent expressions, gathering path performance data for each intent group from infrequent online user samples is impractical. Moreover, with the inherent congestion and variability in today's cloud/edge paths [8], [9], [17], timely capturing path information is crucial for real-time routing decisions. Towards this end, we developed a path probing mechanism enabling the source edge node to collect real-time path data by periodically sending customized packets to

the viewer node, tailored to match the flow patterns of various user intent groups through adjustments in packet size, number, and interval. For example, at the lowest resolution, NFOV has 75K points/frame [1]; for the streaming requirement of 30 FPS, frames are sent consecutively with a 33 ms interval. To reduce the traffic volume of path probing, we keep the same request interval but each probing frame is only 1/30 of a single volumetric video frame (e.g., 13KB per probing frame). Conversely, for higher resolutions like Full HD at 8.46 million points/frame and 15 FPS, the source node sends two frames 66 ms apart, each about 42 KB. Different probing patterns are employed because low-resolution packets may not detect link variations due to their size or FPS, while larger high-resolution packets are more susceptible to or can be affected by network congestion [1]. In detail, the edge node at the source side will periodically send probing packets to the viewer side edge node, printing and saving real-time transport-layer congestion control information (e.g., re-ordered packet number from the kernel), and the application layer frame delay as sensed path information.

Algorithm 1: UCB routing algorithm with moving average window tracking

Data: path set I, Intent network group J
Result: Selected path for each user intent $P(t)$
Initialize: $N_{ij}(t)$: the number of path i has been selected up for user intent group j at round t
$R_{ij}(t)$: the sum of rewards of path i for intent j at round t
$S_{ij}(t)$: Normalized sample value that contains probing result on each path i
$\sigma(S_{ij}(t))$: Standard deviation of $S_{ij}(t)$
$P_j(t)$: Path index in path set I should be deployed for user intent group j
W: size of moving window
α and β: hyperparameters to adjust UCB value
for $t = 1, 2, \ldots$ **do**
  for *intent type* j *in* J **do**
    for *path* i *in* I **do**
      Calculate the average reward: $r_{ij}(t) = \frac{R_{ij}(t)}{N_{ij}(t)} - \alpha \cdot \sigma(S_{ij}(t))$ of path i in window $[t - W, t)$
      Calculate the confidence interval $[r_{ij}(t) - \Delta_{ij}(t), r_{ij}(t) + \Delta_{ij}(t)]$ in window $[t - W, t)$ by obtaining $\Delta_{ij}(t) = \sqrt{\frac{2 \ln T}{N_j(t)}}$
      Update the optimal path i for intent j that has the maximum UCB: $P_{ij}(t) = \arg\max_i \left(\max\left(r_{ij}(t), 0\right) + \beta \cdot \Delta_{ij}(t)\right)$
      Send probing frame of intent group j on path $P_{ij}(t)$, observe $S_{ij}(t)$
      Update $R_{ij}(t)$, increase $N_{ij}(t)$ by 1

Algorithm 1 shows the proposed moving average Upper Confidence Bound (UCB) routing selection algorithm that will be executed at the source side edge node to decide an optimal path. Traditionally, the UCB algorithm can balance exploitation and exploration, and here we adopt it to decide whether to stick on the current path or switch to another path. It chooses the path with the highest upper confidence bound, using the average reward $\bar{r}_{ij}(t) + \Delta_i(t)$ for exploration and the selection frequency $\Delta_i(t)$ for exploitation at round t [24]. However, in the context of modern network scenarios, the UCB algorithm faces the challenge of non-stationary network variation [8], [9], [17]. This means that the characteristics of the network such as the available bandwidth, latency, and packet loss rate, may present temporal variation, making it difficult to maintain an accurate estimate of the expected return for each path. To overcome this challenge, we introduce a moving window-based average value and standard deviation in the reward part to capture potential perceived network and application fluctuations.

$$S_{ij}(t) = \delta \left(1 - \frac{\text{Retrans}_{ij}(t)}{\text{Retrans\_Max}}\right) + \gamma \left(1 - \frac{\text{Frame\_Delay}_{ij}(t)}{\text{Frame\_Delay\_Max}}\right) \quad (1)$$

Regarding each notation in the algorithm, the topology information set is denoted as I, including candidate paths for each user intent group. Utilizing global topology data and user intent needs, the SDN controller selects tailored paths for each group. $N_{ij}(t)$ is the number of paths i has been selected for user intent group j. $R_{ij}(t)$ is the sum of rewards of path i for intent group j, and $S_{ij}(t)$ is the raw probing feedback that is normalized by a predefined function, then the standard deviation of raw probing feedback $\sigma(S_{ij}(t))$ will be obtained for each round. The raw probing feedback contains the number of retransmission packets and frame delay which is calculated as normalized and weighted linear sum function as: The design of $S_{ij}(t)$ aims to integrate feedback from both application and transport layers for each intent group. It tracks retransmission counts from the transport layer, indicating link variations that may activate congestion control measures, potentially impacting user performance [1]. The frame delay is an application layer metric more specifically related to frame size, link capacity, and end-to-end delay. For each intent group type, an empirical pre-defined maximum value can be set according to the offline intent knowledge model at the SDN controller, in order to clip them into the range [0,1]. The coefficients δ and γ are 0.2 and γ to 0.8.

The algorithm initiates by sending probing frames along each candidate path from the source node to set up the path parameter matrices. When the source side node starts this algorithm, there will be a probing frame group sent on each candidate path to initialize each path's parameter matrix. To obtain the UCB value of each path, $R_{ij}(t)$, $S_{ij}(t)$ and $\sigma(S_{ij}(t))$ will be calculated within a moving window-based manner (e.g., $[t - W, t)$) where W denotes the size of the sliding window). For instance, $R_{ij}(t) = \sum_{k=t-W}^{t} S_{ij}(k)$ is the cumulative reward in a given window size w for user intent group j on path i. This moving average window can help to gradually forget the path's past historical performance; therefore the algorithm can capture more temporal features of a link and then make more informed decisions when selecting the optimal path. The variance obtained is a subtractive value to reduce the reward of the current path, indicating that continuous variation can lead to a reduction of the probability that

the path will be selected. The moving window is adjustable depending on the network operator's knowledge of its link variation frequency. The number of an arm has been selected during a given sliding window will be used to calculate the $\Delta_{ij}(t) = \sqrt{\frac{2\ln T}{N_{ij}(t)}}$ which indicates that the less an arm has been selected, the larger the value of $\Delta_{ij}(t)$ will be to boost the probability that this arm can be explored in the future. Then during every probing round, the path with the highest upper-bound confidence $\bar{r}_{ij}(t) + \Delta_{ij}(t)$ will be selected as the optimal path and stored in $P_{ij}(t)$, which denotes the path index with maximum UCB value in the path set I that should be deployed for user intent group j. Once the optimal path is obtained, the edge function will send corresponding probing frames on the selected path, also increase the selected path number $N_{ij}(t)$ and update the observed reward. It is worth noting that we only consider the application flow state change triggered by user intent, leaving the resolution change caused by varying network conditions as our future work.

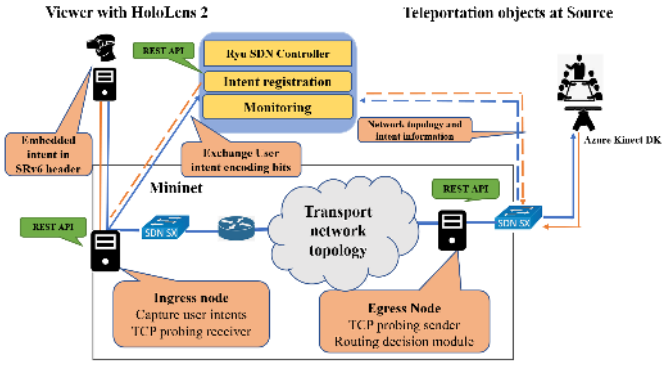


Fig. 5: Implementation of the proposed framework

IV. EVALUATION RESULTS

A. Implementation of the proposed framework

1) We utilized the Ryu controller [24] with the OpenFlow protocol and Restful API to manage the SDN, gathering topology and path data in the transport network simulated in Mininet [25]. The network includes two edge nodes: one for real-time intent packet parsing via XDP eBPF with SRv6, and another for SRv6 decapsulation and rerouting. These nodes use a non-blocking TCP socket model with a window size of 30 and tuning constants α and β set to 2 and 0.4, respectively (see Fig.5).

2) Intent expression interface at HoloLens 2: We have implemented a practical system on HoloLens2 to stream the captured real-time 6-DoF content to be transmitted and rendered on the HoloLens 2 side. Additionally, a dedicated intent tracing module is deployed within HoloLens 2 (e.g., position to the target object), and it can export user intent to a locally connected Ubuntu machine. Then the local Ubuntu machine can send a message with the SRv6 header and modify the corresponding IPv6 flow label to indicate the remote edge node on behalf of the HoloLens 2.

3) Live streaming source: We integrated a Microsoft Azure Kinect DK camera with modified Livescan3D code. A novel feature dynamically adjusts camera resolution and depth based on control messages, optimizing for scenarios with a single subject 1 meter away. We evaluated performance based on throughput, frames per second (FPS), and frame delay—defined as the latency from frame creation to receipt by the edge node. Assuming minimal access path cost, we focused on the transport network as the primary bottleneck. Edge-to-user traffic optimization and 2D meta-context reconstruction significantly reduce data load. We used ZSTD [27] compression, superior to other methods for live streaming delays [15].

B. Key application performance comparisons

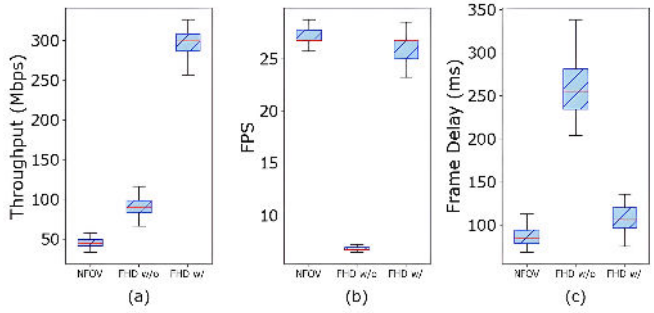
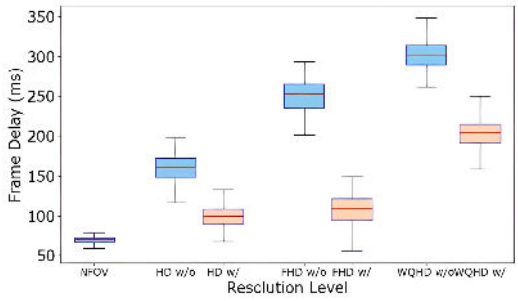


Fig. 6: Key performance comparison

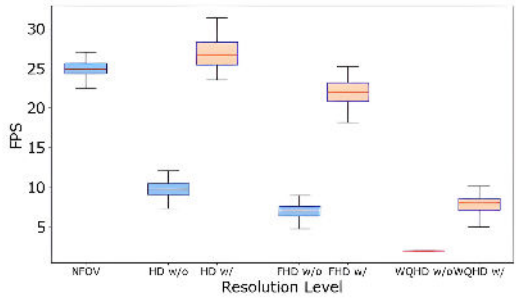
We generated 20 unique path sets, each with 5 disjoint paths based on specified bandwidth (350, 150, 50 Mbps), delay (10, 20, 100 ms), and loss (0.0%, 0.10%, 0.15%) parameters. For each topology, there is at least one path that can guarantee a satisfactory performance for all resolution levels. Since the bandwidth of 50Mbps is sufficient for the lowest resolution level NFOV, therefore we allocate the path with the lowest delay to the live volumetric flow. Each experiment lasts for 240 seconds and the resolution change request happens at the 60th second. We enabled and disabled our framework for each experiment setting and recorded the key performance metrics for the higher resolution level. Fig.6 depicts the performance comparison when a user switches to an FHD resolution level with and without our framework. As the FHD resolution level requires a much higher point density (more than 10 times than NFOV), therefore without enabling our intent-aware path adaptation framework, the default path with inadequate bandwidth will degrade the perceived FPS and frame delay. Furthermore, this poor FPS and around 250 ms frame delay can not even allow the user to understand the motion displayed in the live volumetric streaming scenario. In contrast, with the help of the proposed framework, satisfactory performance can be guaranteed. This is because at the beginning 60th seconds, the MAB algorithm can successfully select and update a proper path for FHD resolution level, even though the target path may present various properties across different topology settings.

C. Impact of different resolution levels

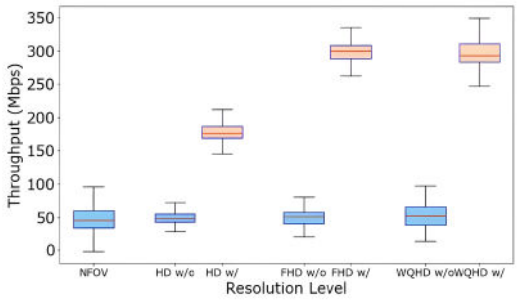
Then we evaluate the resolution change to be more diverse to include HD, FHD, and Wide Quad High Definition (WQHD) levels, following the same topology sets of above evaluation. A common observation is that improved



(a) Frame Delay Comparison



(b) FPS comparison



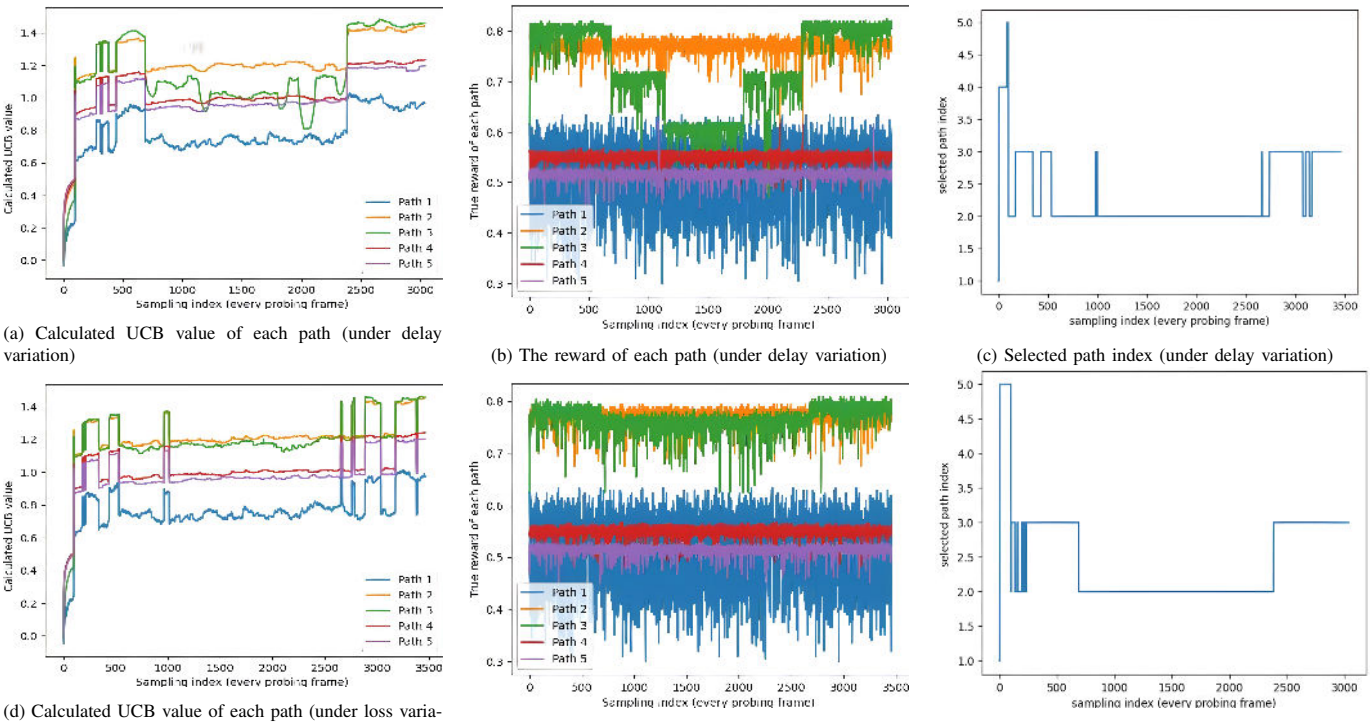
(c) Throughput Comparison

Fig. 7: Comparison between different resolution levels

performance can be observed across all resolution levels. Both HD and FHD resolution levels can obtain full FPS performance and around 100ms frame delay, although in some cases they can not capture the full bandwidth provided by the path. This is because HD resolution is more tolerant in terms of path selection; even if there is packet loss due to congestion on some high-bandwidth links, it does not affect its ability to achieve satisfactory performance. Moreover, as the bandwidth required for HD resolution in our experiments is around 180 Mbps, its throughput does not consume all available bandwidth. In comparison, FHD requires strict high bandwidth to achieve satisfactory performance, which means that any packet loss or additional latency on the path can cause fluctuations in its performance. Additionally, we have incorporated WQHD resolution, and although its performance has significantly improved compared to before path switching was enabled, it still falls far short of a satisfactory user experience. In such cases, network operators should explicitly refuse the registration of such resolutions during offline intent registration, and application providers should also update their terminal software accordingly to ensure that unregistered resolutions are not emitted from user devices.

D. Performance under network variation (loss and delay)

As common network variation in the industrial level network is still observed in delay and packet loss [8], [9], we emulate such time-variant links and evaluate whether our algorithm can timely capture such variation and maintain an optimal path. Path set in this experiment is Path 1: 150Mbps, 20ms, 0.15%, Path 2: 350Mbps, 10ms, 0.05%, Path 3: 350Mbps, 10ms,0.00%, Path 4: 50Mbps, 20ms, 0.00%, Path 5: 50Mbps, 20ms, 0.05%. We manually introduce background traffic on path 3 to add delay and loss at around the 600th sampling index and double it at the 1100th index sampling index, then asymmetrically reduce them at the 1600th and 2100th sampling index, respectively. The probed reward of each path is displayed in Fig.8a and Fig.8d. Then we show the probing and routing selection results generated by our algorithm in Fig.8b, Fig.8e, Fig.8c, Fig.8f. At the beginning, it will take around 200 sampling rounds to identify and converge to an outperforming path (e.g., path 3), especially during the competition between Path 2 and Path 3. We attribute this initial converge phase to several reasons. First, the UCB algorithm has the expected total regret upper bound approximately as $O(\sqrt{KT\log T})$ [24], where K is the path number and T is the current round. There it will take a short while for each path to be explored in the initial phase. Second, as the frame size of a given resolution also varies between 412KB to 625KB, it will take a few rounds for the algorithm to correctly recognize and then distinguish the empirical average reward on these two paths (as see Fig.8b and Fig.8c). However, once the background traffic surges on path 3, its reward becomes highly variant. After the observable reward drop, the algorithm adds path 2 as the best path. This is because path 2 has a more stable average reward and due to its less selected count, the joint effect of the exploration and exploitation parts brings this path to have the highest UCB value (see Fig.8b). It is worth mentioning that once Path 3 gradually recovers from the high load and longer delay, the algorithm will continuously leverage its probing mechanism for a hundred probing samples to ensure true path performance recovery before it can trigger the path switch back to Path 3 (see in Fig.8c). In contrast, in the beginning, once the path starts to drop, the probing mechanism will instantly capture the variance to trigger path switching (e.g., only one probing interval by examining the log file), therefore minimising the probability that the new intent arrives in this gap. Regarding the performance improvement evaluated by triggering an intent to change resolution from NFOV to HD, the frame delay and FPS can be retained as satisfied levels once user intent for the HD resolution level arrives. Next, we emulate random loss and its effect on path selection and switching of HD resolution (see Fig.8d, Fig.8e, and Fig.8f). Obviously, the probed reward of path 3 presents relatively lower variation under random loss. This is because the transport layer congestion control algorithm



(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)

(d) Calculated UCB value of each path (under loss variation)

(e) The reward of each path (under loss variation)

(f) Selected path index (under loss variation)

Fig. 8: Detail information of path probing and route selection under loss and delay variation

BBR algorithm does not rely on packet loss as a congestion signal, random packet loss However, we can still observe some fluctuations in path 3, due to TCP BBR's periodic congestion control window drop to enter probe RTT state to sense a more accurate congestion control window once network fluctuation is detected [27]. These fluctuations can be correctly captured because we consider packet retransmission in our sampling (see Fig.8b), and consequently path switch can be triggered correspondingly. From the application's perspective, packet retransmission and reordering have a negative impact on the parsing and processing at the application layer. Therefore, our policy still chooses to switch to path 2 when these occur and switch back to path 3 after detecting that the packet loss has disappeared and the path has stabilized.

E. Comparison between MAB algorithms

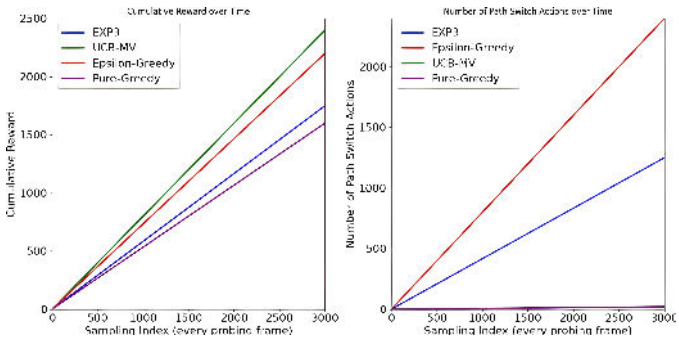


Fig. 9: Performance comparison between different MAB algorithms

Then we compare the proposed intent-based UCB moving average and variance (UCB-MV) algorithm with other algorithms including greedy, e-greedy, and EXP3 [29]. Intuitively, the exploration part of the algorithm should timely capture another stable path while network variation occurs. The e-greedy implemented uses a fixed parameter of 0.2 to allow other paths to be explored randomly, the pure-greedy algorithm always follows the best reward arm with minimized exploration, and similarly, EXP3 updates the probability for each candidate path according to the observed reward and is natively capture the non-stationary scenario. The first evaluation metric is the cumulative reward of each algorithm under delay variation. (see Fig.9). Apparently, the adopted UCB-MV algorithm achieves the best cumulative reward in around 3000 sampling index among all algorithms, although epsilon-greedy can achieve close performance as UCB-MV. However, as these four curves start to perform differently at around the 600th sampling index that the delay variation happens, only the UCB-MV algorithm can immediately identify this change, maintaining the optimal growth rate among the four curves. The second evaluation metric is the number of path switches applied to the network which can cost unnecessary network configuration resources. Similarly, the epsilon-greedy algorithm with proper exploration parameters (e.g., 0.2) would allow each arm to be randomly selected. However, in a practical networking routing system, this can cause unnecessary network configuration costs. In contrast, the UCB algorithm's exploration is based on reward information gathered from calculated reward and the number of selected, it can ensure only the path with noticeable

reward will be explored.

F. Impact of congestion control algorithms

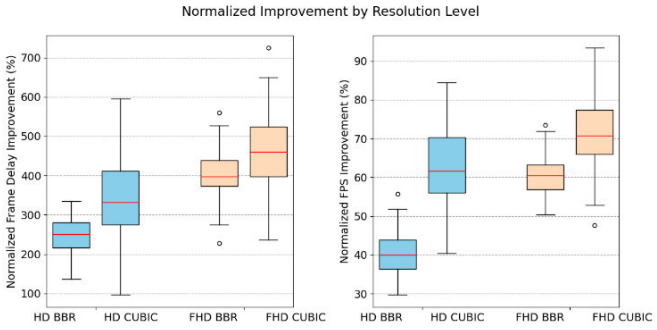


Fig. 10: Performance comparison between different congestion algorithms

Finally, we compare the impact of different congestion control algorithms, such as CUBIC [30] and BBR. CUBIC uses explicit packet loss as its congestion signal, unlike BBR which estimates per-round delivery rates. Thus, applications using CUBIC are more sensitive to network variations. Fig.10 illustrates that path-switching algorithms significantly benefit CUBIC, showing a larger percentage improvement in resolution levels, although with more variability. This also brings the task for network operators to be aware of the congestion control algorithm used by application providers during the offline intent registration phase, then to wisely set the related parameters of their network adaptation policies.

V. Conclusion

In this work, we explore performing real-time network path adjustments driven by user behaviours and network conditions, for optimal performance based on user intents. Using offline intent negotiation, the network sets fixed encoding for different behaviours and requirements, while an online edge interface can capture real-time behaviours. In a fluctuating network environment, we introduce a moving averaged UCB algorithm to detect and maintain an optimal path table for various user intents. The evaluation results validated through software implementations and measurements, demonstrates stable and improved QoE performance, once the proposed framework is enabled.

Acknowledgment

This project is sponsored by SPIRIT project (Grant Agreement 101070672, https://www.spirit-project.eu/).

References

[1] Qian, Peng, et al. "Remote Production for Live Holographic Teleportation Applications in 5G Networks." IEEE transactions on broadcasting 68.2 (2022): 451-463.

[2] Han, Bo, Yu Liu, and Feng Qian. "ViVo: Visibility-aware mobile volumetric video streaming." Proceedings of the 26th annual international conference on mobile computing and networking. 2020.

[3] Seufert, Michael, Sarah Wassermann, and Pedro Casas. "Considering user behavior in the quality of experience cycle: Towards proactive QoE-aware traffic management." IEEE Communications Letters 23.7 (2019): 1145-1148.

[4] Qian, Peng, Ning Wang, and Rahim Tafazolli. "User Intent Driven Path Switching in Video Delivery-An Edge Computing Based Approach." IEEE INFOCOM 2022-IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS). IEEE, 2022.

[5] Liu, Yu, et al. "Vues: practical mobile volumetric video streaming through multiview transcoding." Proceedings of the 28th Annual International Conference on Mobile Computing And Networking. 2022.

[6] He, Lin, et al. "Enabling Application-Aware Traffic Engineering in IPv6 Networks." IEEE Network 36.2 (2022): 42-49.

[7] da Costa Filho, Roberto Irajá Tavares, et al. "Scalable QoE-aware path selection in SDN-based mobile networks." IEEE INFOCOM 2018- IEEE Conference on Computer Communications.

[8] Li, Jinyang, et al. "LiveNet: a low-latency video transport network for large-scale live streaming." Proceedings of the ACM SIGCOMM 2022 Conference. 2022.

[9] 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.

[10] Gül, Serhan, et al. "Low-latency cloud-based volumetric video streaming using head motion prediction." Proceedings of the 30th ACM Workshop on Network and Operating Systems Support for Digital Audio and Video. 2020.

[11] Liu, Yu, et al. "Vues: practical mobile volumetric video streaming through multiview transcoding." Proceedings of the 28th Annual International Conference on Mobile Computing And Networking. 2022.

[12] Zhang, Anlan, et al. "YuZu: Neural-Enhanced Volumetric Video Streaming." 19th USENIX Symposium on Networked Systems Design and Implementation (NSDI 22). 2022.

[13] Huang, Yakun, et al. "AI-transfer: Progressive AI-powered transmission for real-time point cloud video streaming." Proceedings of the 29th ACM International Conference on Multimedia. 2021.

[14] Zhu, Yuanwei, et al. "A semantic-aware transmission with adaptive control scheme for volumetric video service." IEEE Transactions on Multimedia (2022).

[15] Guan, Y., Hou, X., Wu, N., Han, B., and Han, T. "MetaStream: Live Volumetric Content Capture, Creation, Delivery, and Rendering in Real Time," in Proceedings of the 29th Annual International Conference on Mobile Computing and Networking, 2023, pp. 1–15.

[16] Yang, Mengyu, et al. "A Comparative Measurement Study of Point Cloud-Based Volumetric Video Codecs." IEEE Transactions on Broadcasting (2023).

[17] Jadin, Mathieu, et al. "Leveraging eBPF to Make TCP Path-Aware." IEEE Transactions on Network and Service Management (2022).

[18] Lebrun, David, et al. "Software Resolved Networks: Rethinking Enterprise Networks with IPv6 Segment Routing." In Proceedings of the Symposium on SDN Research (SOSR '18). ACM, 2018, Article 6.

[19] Miyasaka, Takuya, Yuichiro Hei, and Takeshi Kitahara. "NetworkAPI: An in-band signaling application-aware traffic engineering using srv6 and is anycast." NAI '20: Proceedings of the Workshop on Network Application Integration/CoDesign.

[20] "Hypertext Transfer Protocol Version 2 (HTTP/2)." [Online; Accessed 2024-Apr-29]. Available: https://datatracker.ietf.org/doc/html/rfc7540.

[21] "eBPF." [Online; Accessed 2024-Apr-29]. Available: https://ebpf.io.

[22] Auer, P. "Using confidence bounds for exploitation-exploration trade-offs." Journal of Machine Learning Research 3 (2002): 397-422.

[23] Auer, Peter, et al. "Finite-time analysis of the multiarmed bandit problem." Machine learning 47.1 (2002): 235-256.

[24] "Ryu SDN controller." [Online; Accessed 2024-Apr-29]. Available: https://ryu-sdn.org/.

[25] "Mininet: An Instant Virtual Network on Your Laptop (or Other PC)." [Online; Accessed 2024-Apr-29]. Available: http://mininet.org.

[26] Cardwell, Neal, et al. "BBR: congestion-based congestion control." Communications of the ACM 60.2 (2017): 58-66.

[27] "ZSTD compression." [Online; Accessed 2024-Apr-29]. Available: https://github.com/facebook/zstd.

[28] Auer, Peter, et al. "The non-stochastic multiarmed bandit problem." SIAM Journal of Computing 32.1 (2003).

[29] Ha, Sangtae, Injong Rhee, and Lisong Xu. "CUBIC: a new TCP-friendly high-speed TCP variant." ACM SIGOPS operating systems review 42.5 (2008): 64-74.

[30] Kowalski, Marek, Jacek Naruniec, and Michal Daniluk. "Livescan3d: A fast and inexpensive 3d data acquisition system for multiple kinect v2 sensors." 2015 international conference on 3D vision. IEEE, 2015.