QoE-Assured Live Streaming via Satellite Backhaul in 5G Networks

Chang Ge, *Member, IEEE,* Ning Wang, *Senior Member, IEEE,* Ioannis Selinis, Joe Cahill, Mark Kavanagh, Konstantinos Liolis, Christos Politis, Jose Nunes, Barry Evans, Yogaratnam Rahulan, Nivedita Nouvel, Mael Boutin, Jeremy Desmauts, Fabrice Arnal, Simon Watts, Georgia Poziopoulou

***Abstract*—Satellite communication has recently been included as one of the key enabling technologies for 5G backhauling, especially for the delivery of bandwidth-demanding enhanced mobile broadband (eMBB) applications in 5G. In this paper, we present a 5G-oriented network architecture that is based on satellite communications and multi-access edge computing (MEC) to support eMBB applications, which is investigated in the EU 5GPPP Phase-2 SaT5G project. We specifically focus on using the proposed architecture to assure Quality-of-Experience (QoE) of HTTP-based live streaming users by leveraging satellite links, where the main strategy is to realise transient holding and localization of HTTP-based (e.g., MPEG-DASH or HTTP Live Streaming) video segments at 5G mobile edge while taking into account the characteristics of satellite backhaul link. For the very first time in the literature, we carried out experiments and systematically evaluated the performance of live 4K video streaming over a 5G core network supported by a live geostationary satellite backhaul, which validates its capability of assuring live streaming users' QoE under challenging satellite network scenarios.**

***Index Terms*—HTTP live streaming, multi-access edge computing, network function virtualization, quality of experience, satellite communications, 5G networks**

I. Introduction

As 5G networks are gradually being rolled out worldwide, it is expected that users can begin consuming enhanced Mobile BroadBand (eMBB) applications on their mobile devices. Such applications, e.g., 4K and 8K video streaming as well as virtual or augmented reality applications, offer much better immersive experiences while requiring far higher data rate to support their smooth delivery to user devices. Therefore, one of the key objectives of 5G eMBB is to assure the Quality-of-Experience (QoE) of these applications' users in dynamic and challenging network scenarios. Recently, satellite communications have been introduced into 5G networks. Satellite plays a key role in 5G backhauling technologies, where one leading use case is to multicast video content due to its high efficiency and performance of supporting content delivery at scale to geographically-distributed locations [1]–[3].

The majority of video streaming applications nowadays follows the principle of HyperText Transfer Protocol, or HTTP-based adaptive streaming (e.g., MPEG-DASH or Apple HTTP Live Streaming - HLS), which divides a video sequence into multiple segments that are delivered in the format of sequenced files through HTTP requests. Such streams are delivered through unicast by default. For video-on-demand (VoD) applications, it is shown in [4], [5] that users' QoE can be assured through both preloading popular content a priori at 5G mobile edge sites with caching capabilities, and adaptive prefetching that utilizes parallel TCP (Transmission Control Protocol) connections to download video segments ahead of users' requests. However, in the scenario where a live stream is delivered through a 5G network with satellite backhaul, there are distinct challenges regarding the assurance of users' QoE in terms of network latency and packet errors. It is well known that due to the long network latency and relatively high packet error rate over the satellite backhaul, TCP is likely to experience poor performance due to its slow-start and retransmission mechanisms [5]. This is especially relevant to HTTP-based live streaming, as video segments are relatively small (typically less than 10MB). For this particular challenge, our objective is to tackle the feasibility of delivering 5G-oriented video content applications with assured user QoE in real geostationary satellite-enabled 5G core networks, which has never been attempted previously. Furthermore, through the recently standardized service-based architecture (SBA) of 5G networks, there are new opportunities to leverage satellite link's capability in a context-aware manner to provide QoE assurance.

In this paper, under the EU 5GPPP Phase-2 SaT5G (Satellite and Terrestrial network for 5G) project [6], we address the scenarios above by utilizing satellite communications as backhaul in a 5G network to support 4K video streaming applications with QoE assurance. We focus on HTTP-based live streaming scenario, where video content are generated on-the-fly at a content origin server and delivered to geographically distributed end-users through a 5G network with satellite backhaul. Specifically, we present a 5G SBA-based framework that provides QoE assurance in a context-aware manner. We envisage that stakeholders are involved in this scenario, i.e., 5G mobile network operator (MNO), video content provider (CP) and satellite network operator (SNO). In the proposed

Parts of this paper have been published in the Proceedings of the IEEE BMSB 2018, Valencia, Spain.

Chang Ge, Ning Wang, Ioannis Selinis, Barry Evans and Yogaratnam Rahulan are with 5GIC, Institute for Communication Systems, University of Surrey, Guildford, UK, GU2 7XH.

Joe Cahill and Mark Kavanagh are with VT iDirect Solutions Limited, Killarney, Ireland.

Nivedita Nouvel, Mael Boutin and Jeremy Desmauts are with Broadpeak, Cesson Sevigne, France.

Konstantinos Liolis, Christos Politis and Jose Nunes are with SES, Betzdorf, Luxembourg.

Fabrice Arnal is with Thales Alenia Space, Toulouse, France.

Simon Watts and Georgia Poziopoulou are with Avanti Communications, London, UK.

Digital Object Identifier:

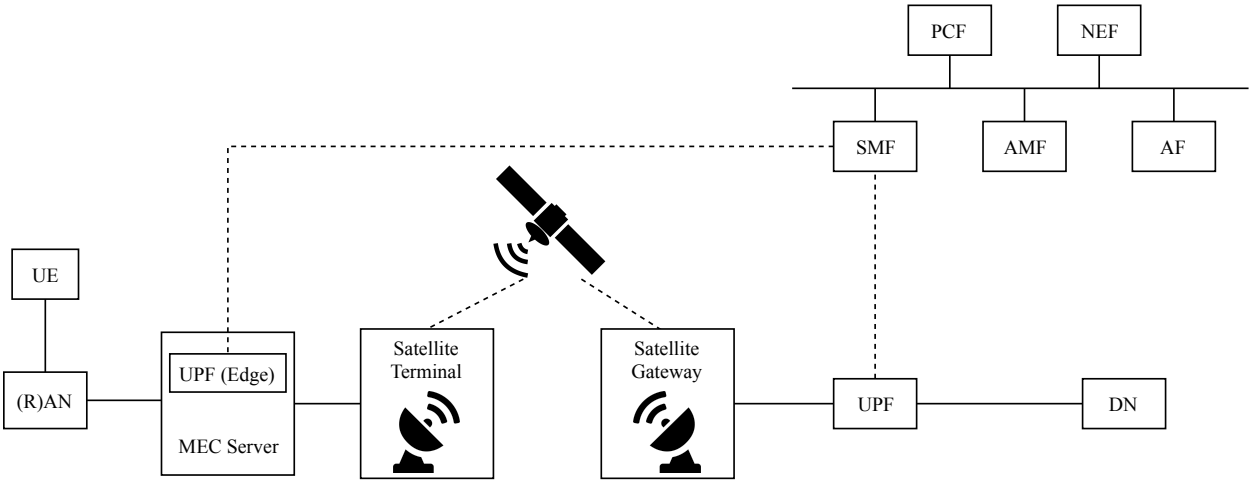

Fig. 1: System overview and architecture

framework, the 5G MNO virtualizes its computing and storage resources and leases them to CPs, where the latter can deploy their own virtualized network functions (VNFs) in multi-access edge computing (MEC) servers [7], [8]. Meanwhile, the SNO leases its satellite channel bandwidth resource to the 5G MNO, so that the latter uses it as a backhaul link in addition to the standard terrestrial backhaul. The framework's operations include the following aspects:

- For each user, its video requests are handled by a local CP-operated VNF that is hosted at the MEC server. Such a MEC server is envisaged to be located within proximity of the user to ensure low access latency. It also breaks the end-to-end (E2E) content delivery path into two segments, i.e., satellite backhaul and 5G Radio Access Network (RAN). This not only significantly reduces initial startup delay, but also offers better content delivery performance [4], [9].
- Each MEC server monitors the active live streaming sessions under its coverage, especially their real-time QoE status such as buffering and video quality. Based on these contexts, it performs necessary content operations such as transient segment holding [9] etc, which provide customized, context-aware QoE assurance on a per-user, per-session basis. Meanwhile, it may perform transport-layer performance enhancement techniques to complement the application-layer QoE assurance above. Furthermore, a MEC server aggregates requests from all sessions that consume the same live stream under its coverage. In other words, regardless of how many sessions are consuming a stream, only one flow needs to be established between that MEC server and the content origin, which effectively realises *application-layer multicast* in the last hop.
- Each CP monitors its live video streams' spatial popularity, and dynamically adjusts the method that a live stream is delivered from its origin server to the MEC servers. For example, if a live stream is detected to be popular among multiple distributed MEC servers, the CP may decide that it is more efficient for the content origin to use a *multicast*-based protocol such as FLUTE (File Delivery over Unidirectional Transport [10]) to disseminate the live stream files to multiple MEC servers, which is inherently supported over the satellite backhaul. If the stream's popularity drops, the CP may switch back to *unicast*-based delivery from origin to MEC servers. This realises context-aware *backhaul multicast* and ensures that backhaul content traffic consumes bandwidth efficiently.

The key contributions of this paper are as follows:

- This is the first system developed in the literature that utilizes both SBA-based 5G core network and satellite backhaul to support 4K HTTP-based live streaming applications with QoE assurance. Specifically, it leverages both the context awareness and flexibility that are enabled by 5G SBA architecture, as well as the multicast capability of satellite backhaul. It also utilizes virtualization technology to enable CPs to deploy their own VNFs in MEC servers at 5G mobile edge, which not only performs content operations such as transient segment holding, but also realises last-hop multicast at application layer. Overall, the proposed system assures live users' QoE while maintaining the video quality at or above 4K; it also ensures that video content are always delivered through the backhaul in the most efficient manner.
- This is the first time that a 5G core network and a real satellite communications link have been implemented and integrated as a holistic system, where the latter serves as the backhaul of the 5G network. The establishment of such a system means that it is possible to test the performance of MEC servers with content operations (such as transient segment holding) in terms of content delivery and QoE assurance through a real satellite backhaul.
- Based on the implemented system above, we have comprehensively evaluated the performance of our QoE assurance scheme in a wide variety of scenarios over-the-air using a satellite backhaul integrated with a 5G core network. The experiment outcome validates that even through satellite backhaul, the proposed scheme is able to guarantee a stalling-free live streaming experience while maintaining the video quality at 4K.

The remainder of this paper is organized as follows. Section II presents a high-level overview of the proposed system. Section III discusses in technical detail a number of implementation aspects of the system. Section IV evaluates the performance of the scheme in a real network with satellite backhaul. Section V concludes the paper.

II. System Overview

In this section, we provide a high-level overview of the proposed system architecture, which is shown in Figure 1. There are three stakeholders in the system, i.e., 5G MNO, SNO and CP. In summary, a live streaming user's E2E content delivery path involves an edge User-Plane Function (UPF, which typically runs on a MEC server), satellite backhaul and a core UPF before it reaches the CP's live content origin that is located in the Data Network (DN). The satellite backhaul involves a satellite terminal, the satellite link and a satellite gateway. The 5G MNO leases satellite channel capacity from the SNO to use it as a backhaul link. Note that although the MEC server is operated by the 5G MNO, its computing and storage resources are virtualized and leased to CPs who deploy their own VNFs to perform content operations such as caching and transient segment holding. As shown in Figure 1, a MEC server can host multiple VNFs that are operated by different CPs. Such an operational model has been adopted in the operator and content delivery industries since 2015 [11].

From the User Plane perspective, when a user sends an HTTP request in a live streaming session, it is first resolved to the corresponding CP's VNF within the MEC server. The VNF intercepts the request and acts as a reverse HTTP proxy. Through content operations such as transient segment holding, the MEC server can download video segments from their origin server and cache them locally before they are requested by users. This is achieved through manipulating the live stream's manifest file content and establishing multiple parallel TCP connections from the MEC server towards the origin server, and more details are described in Section III-A. If the requested video segment is already available at its local cache, it is served to the user immediately. If not, the VNF forwards the requests to the content origin and retrieves it on the user's behalf. Meanwhile, besides normal content operations such as caching, it also performs context-aware operations such as transient segment holding while subjecting to the CP's policy. More details on how such policies are established are described in Section III-B. The VNF is also responsible for monitoring and reporting application context (e.g., each user's real-time QoE status) periodically to the Control Plane.

Regarding the Control Plane, it is shown in Figure 1 that it adopts a SBA where all its elements communicate via a bus. This is in line with the latest 3GPP 5G system architecture [12]. In this work, we focus on five elements that are most relevant to our scheme, namely Application Function (AF), Policy Control Function (PCF), Session Management Function (SMF), Network Exposure Function (NEF) and Access and Mobility Management Function (AMF). While most Control Plane elements are operated by the MNO, AF is typically operated by third-party stakeholders such as CPs. Their functionalities are as follows:

- PCF is responsible for converting instructions from other Control Plane elements (e.g., SMF and AF) into policies that can be understood by CPs' VNFs.
- SMF is the direct "contact" point between the MEC server (and the VNFs within) and the Control Plane. On one direction, SMF receives policies from PCF and disseminates them to the VNFs where they will be enforced. One the other direction, SMF handles context updates and monitoring feedback from VNFs, and sends them to AF so that it can update its policies as necessary.
- AF has multiple functionalities. First, it provides context information to the MNO on how to identify each CP's traffic flow. Example criteria include destination IP address and port number. Second, it processes the monitored contexts that are reported by the MEC server via SMF, and adjusts content operation (e.g., caching or segment holding etc.) policies where necessary. Updated policies are sent to PCF first, where they are formatted and sent to SMF for dissemination and enforcement.
- NEF acts as a "bridge" between AF and other Control Plane elements. Because AF is operated by third-party stakeholders, the MNO needs to carefully control which aspects of the Control Plane are exposed to AF, and NEF is responsible for managing such exposure policies.
- AMF is mainly responsible for the management aspects of the Control Plane, such as authentication, authorization and user mobility management etc. For example, an AF needs to be first authenticated by AMF before it can interact with other Control Plane elements such as SMF.

III. Implementation Aspects

A. Transient Segment Holding and Application-Layer Multicast at MEC Server

As mentioned in Section II, in order to assure live streaming users' QoE at the MEC server, context-aware transient segment holding (first proposed in [9]) is performed on a per-streaming-session-basis. In a nutshell, a live streaming client periodically requests the video stream's manifest file from the live origin, so that it can learn about newly-produced segments as soon as possible. Since all requests for stream manifest and video segment files are handled by the MEC server, if it holds back the availability of some segments from the client, the client would be given the false impression of the live origin's streaming progress and request segments that were produced a small while ago. This creates the opportunity for the MEC server to download those held-back segments from the live origin beforehand using parallel TCP connections, hence making them available locally before requested by the client. Intuitively, while holding more segments will provide better assurance on video segment localization, it also introduces higher live streaming latency at the client side. Therefore, the technical challenge is to minimize the number of held segments, while assuring that all video segments are localized before they are requested by clients. More details on how such optimization is performed over a satellite backhaul are specified in Section III-B.

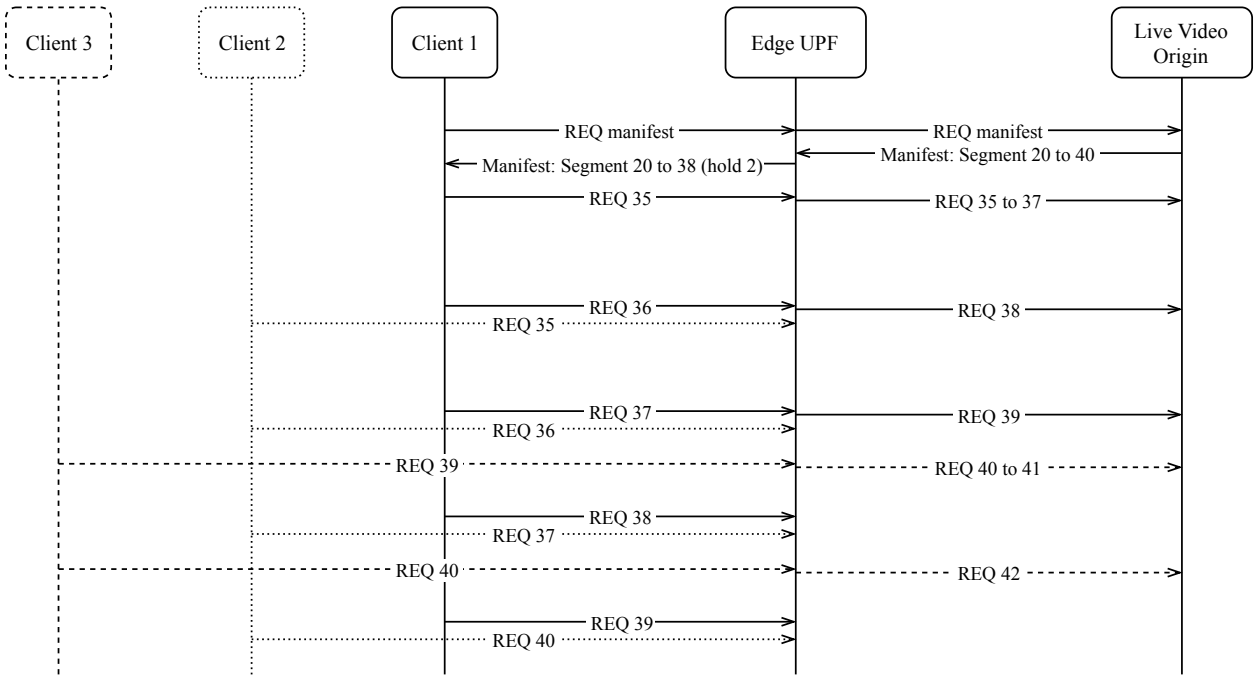

Fig. 2: Transient segment holding for multiple clients & application-layer multicast at MEC server

By default, the above techniques are performed on each individual live streaming session that is requested through the MEC. However, if a live stream is popular, it is expected that multiple clients will be consuming it simultaneously under a single MEC server's coverage. Although their streaming progresses are typically similar, there may be a difference of one or two segments between clients' progresses due to e.g., minor clock drifting on user devices. This is especially the case with shorter segment lengths, e.g., 2s or 4s. In order to handle such a situation, for each live stream, the MEC server always align its segment holding progress to the session with the most advanced streaming progress. This is illustrated in Figure 2. Note that for clarity, we omit all manifest requests (besides the first one) and all response flows in the figure.

In the beginning, the MEC server performs transient holding of two segments for client 1 that joins a live stream, where it opens three parallel TCP connections to download segments 35 to 37 simultaneously. Afterwards, client 2 joins the same stream, and its progress is one segment *behind* client 1. In this case, the MEC server directly serves the segments that were already made available locally for client 1, and there is no need for extra actions. Later, session 3 joins the same stream with its progress two segments *ahead* of session 1. In this case, the MEC server first serves segment 39 that was already pre-downloaded for client 1. Meanwhile, it downloads segments 40 and 41 to ensure that it always stays two segments ahead of client 3's progress, which also ensures that all other sessions will also have access to locally available segments for their subsequent requests. Note from Figure 2 that the MEC server adjusts its downloading schedule to match client 3's progress over client 1's as soon as it detects client 3 is the one with the most advanced progress.

It is worth noting that although not depicted in Figure 2, there are three possible scenarios regarding the availability of a video segment at the MEC server when it is requested by a user. It may a) have fully downloaded the segment already, or b) have started downloading it but the transmission is not finished yet; or c) have not started the download yet. Scenario b) is more likely to happen when the MEC server is using multiple parallel TCP connections to download several segments simultaneously, as their transmissions do not necessarily finish in order.

One key observation from Figure 2 is that each video segment file is only downloaded once by the MEC server from the live video origin. This is neither affected by the number of clients that consume a live stream, nor by how much their streaming progresses differ. Henceforth, our scheme has effectively realized application-layer multicast at the MEC server. It not only aggregates all clients' requests for each video stream into a single flow between itself and the live video origin, but also dynamically adjusts its transient segment holding schedule to ensure all clients always have access to locally available video segments. This is especially important for satellite backhaul where bandwidth resources are limited.

B. Establishing Transient Segment Holding Policies

As described above, the main challenge when the MEC server establishes its transient segment holding policies for each session is to determine the optimal (i.e., minimal) number of segments to be held. Such a problem has been formulated in [9], where the objective is formulated as follows:

$$\underset{x}{\operatorname{argmin}} \quad \frac{s_{\text{seg}}}{th_{\text{bh}} \cdot x} \leq l_{\text{seg}} \qquad (x = 1, 2, \dots) \tag{1}$$

subject to:

$$th_{\text{ran}} > b_{\text{seg}} \tag{2}$$

where x denotes the number of segments to be held. s_{seg} and l_{seg} refer to the size (in bytes) and the length (in seconds) of a video segment respectively. th_{bh} refers to the backhaul throughput between the MEC server and the live video origin, i.e., the throughput over the satellite link. Objective (1) means that each video segment's download duration must be shorter than its length, hence ensuring that it is available at the MEC server before it is requested by a client. Note that the overall aggregated backhaul throughput is $th_{\text{bh}} \cdot x$ because x parallel TCP connections will be used to download x segments simultaneously. Constraint (2) assumes that RAN throughput th_{ran} is always greater than the video segment's bitrate b_{seg}, which ensures that as long as a video segment is available at the MEC server, it can always be delivered to the client over RAN in time. In this work, we assume this constraint always holds true[1].

In objective (1), besides the variable x which needs to be minimized, the only field that is not directly known is th_{bh}. In other words, we need to model the throughput that the MEC server gets when downloading a video segment from the live origin via the satellite link. In this work, we follow the modeling approach in [9] that is based on the recently-proposed BBR (Bottleneck-Bandwidth and Round-trip Latency) as the TCP congestion control mechanism [13], [14], which is due to its superior performance over links with large bandwidth-delay product (BDP) as well as relatively high packet error rates such as satellite links. What is different from [9] is that while fixed backhaul network exhibits a very stable BDP, satellite backhaul has distinct BDP characteristics. We have performed measurements of latency and jitter over a real satellite link for one hour, and the round-trip time (RTT) results are plotted in Figure 3. It is observed that while not as stable as fixed backhaul links, the satellite latency is still relatively stable as 99.7-percentile of the results fall within 540ms to 580ms. Statistically, the latency measurement has a mean of 560.03ms with a standard deviation of 13.02ms. The bandwidth on a satellite link is relatively stable as well. Therefore, we make the following assumptions:

- Each video segment's size is relatively small, and can usually be fully delivered within 10-20 RTTs based on our measurements above.
- The backhaul RTT is relatively stable with low jitter, hence the overall BDP does not fluctuate significantly.
- Packet loss over the satellite backhaul does not affect BBR's performance, since BBR is a rate-based (instead of loss-based) congestion control mechanism and adjusts its sending rates via BDP measurement.

Based on these assumptions, we model th_{bh} over the satellite backhaul link as follows. First, for each video segment download session, the number of bits that is transmitted during the TCP BBR's startup phase is modeled as s_{startup}:

[1]It is explained comprehensively in [9] what the MEC server can do if it does not hold.

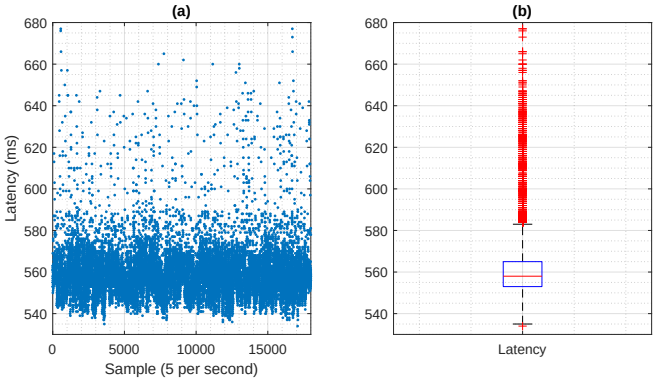

Fig. 3: Satellite backhaul latency over one-hour

$$s_{\text{startup}} \approx \frac{mss(1-(\frac{2}{\ln(2)})^{d_{\text{startup}}})}{1-\frac{2}{\ln(2)}} \tag{3}$$

where mss refers to TCP's maximum segment size (e.g., 1460 bytes), and d_{startup} denotes the time duration that is spent in BBR's startup phase. Note that by default, BBR begins by increasing its sending rate with a factor of $2/ln(2)$ and stops when no more additional bandwidth is found after three RTTs. Therefore, we have

$$d_{\text{startup}} = (log_2\text{BDP} + 3) \cdot \text{RTT} \tag{4}$$

where BDP is estimated by multiplying satellite link bandwidth bw_{bh} by RTT.

Since the satellite backhaul's BDP is generally stable, the total transmission duration of a video segment d_{seg} is modeled as:

$$d_{\text{seg}} \approx d_{\text{startup}} + \frac{(s_{\text{seg}} - s_{\text{startup}})}{bw_{\text{bh}}} \tag{5}$$

which indicates that the remaining amount of bits that were not transmitted during the startup phase will be downloaded at bw_{bh}. The overall backhaul throughput can be hence modeled as:

$$th_{\text{bh}} = \frac{s_{\text{seg}}}{d_{\text{seg}}} \tag{6}$$

Substituting into the objective function (1), we get:

$$\underset{x}{\text{argmin}} \quad x \geq \frac{1}{l_{\text{seg}}}\left(d_{\text{startup}} + \frac{(s_{\text{seg}} - s_{\text{startup}})}{bw_{\text{bh}}}\right) \tag{7}$$

It is worth noting that while bandwidth is typically over-provisioned in fixed backhaul links and is in the order of hundreds to thousands of Mbps, it is much lower in a satellite link. For example, as we will show in Section IV, the TCP throughput that can be maximally achieved over a typical 20MHz Ku-band satellite channel is generally less than 60Mbps. Furthermore, when transmitting over such a bandwidth-limited link, opening multiple parallel TCP connections will lead to degraded per-connection throughput performance and even channel congestion. Therefore, although transient segment holding's strategy is to use parallel TCP connections to compensate a single connection's poor performance,

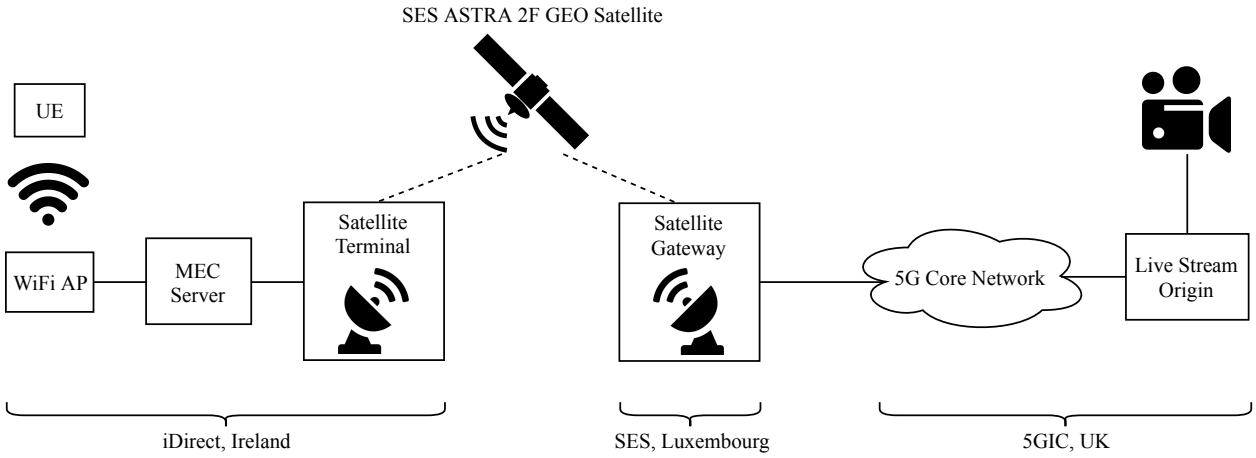

Fig. 4: Experiment setup: live 4K streaming via satellite backhaul

cautions must be taken to avoid opening too many parallel TCP connections from over-saturating the satellite channel and causing congestion on the link. In Section IV, we will evaluate the minimum number of parallel TCP connections that are needed under various scenarios, as well as multiple TCP connections' impacts on throughput performance.

C. Multicast over Satellite Backhaul

So far, we have described how each MEC server aggregates its downstream requests from clients, and establishes one transmission flow per live stream towards the live origin. Meanwhile, it is expected in a 5G network that MEC servers are deployed in a distributed manner among potentially numerous mobile edge sites. If a live stream is popular among multiple MEC servers, they would be delivered from the live origin to every MEC server via unicast by default, which multiplies the traffic on the satellite backhaul with limited bandwidth resource.

In this work, we propose that if a live stream is being consumed by multiple MEC servers simultaneously, the live stream's origin should deliver the segment via multicast, instead of unicast, to these MEC servers. The key rationale here is to significantly reduce traffic volume over the satellite backhaul for popular live streams [15]. This is especially beneficial to 5G networks where MEC servers are envisaged to be deployed close to end-users and hence, a significant number of MEC servers will be required. Furthermore, in the case of live streaming, the MEC servers' requesting progress are likely to be very similar to each other's, making it an ideal scenario for push-based file deliveries. Existing multicast-based content delivery mechanisms such as [10] and [16] can be used to transmit the files while assuring file integrity, hence eliminating the risk of corrupting video frames.

As mentioned in Section II, each MEC server regularly monitors and reports popularities of live streams under its coverage to SMF in the control-plane, which then forwards such context updates to the AF that is operated by CPs. If the AF detects that a live stream is being consumed by multiple MEC servers, it will send a policy update to the corresponding MEC servers via PCF and SMF, which contains instructions for them to join the respective live stream's multicast group at the origin. For example, the AF may tell the MEC servers about which live streams are available through which multicast groups, as well as optional recommendations or policies on which groups they should join. Note that the methodology above is just one example of how to achieve the proposed scheme, and the CP may flexibly determine the specific operating policy on this. For example, the CP may advertise one multicast group per live stream, or it may multicast multiple live streams simultaneously if they are often consumed together.

IV. Performance Evaluation

In this section, we evaluate the performance of transient segment holding mechanism and its QoE assurance in a variety of scenarios using a real satellite backhaul.

A. Experiment Setup

The experiment setup is illustrated in Figure 4. At the client side (site A), a laptop uses Chrome browser (version 68) with hls.js v0.11.1 as the HLS (HTTP Live Streaming) client. It connects to a local WiFi access point with an attached MEC server, where the transient segment holding function is deployed as a VNF. More specifically, it is implemented by customizing and extending a Jetty HTTP/2 web server in Java, which realizes all functionalities that are described in Section III.

The MEC server is connected to a satellite terminal that is located at iDirect's premise in Ireland. The satellite terminal communicates with a satellite gateway (SES Teleport at Betzdorf, Luxembourg) through SES's owned and operated in-orbit geostationary satellite ASTRA 2F (28.2°E) [17] that operates at Ku-band, and the channel bandwidths are 26MHz and 6MHz for downlink and uplink respectively. This leads to a maximum backhaul TCP throughput of around 60Mbps. The satellite channel is reserved for our experiments during this period. The satellite gateway is directly connected to the 5G core network that is hosted at 5G Innovation Centre (5GIC), University of

Surrey, UK. Note that such a topology is due to the facility availability at SaT5G project partners. All experiments were conducted on 24th and 25th September 2018 under clear-sky conditions. A Nokia OZO+ camera is also deployed at 5GIC which outputs 360° monoscopic 4K video at 30FPS and around 3Gbps. The raw video feed is then compressed at a local Matrox Maevex 6100 encoder card, where it is encoded into two RTSP streams with bitrates of 10Mbps and 20Mbps respectively. Constant Bitrate (CBR) is used on both streams and keyframes are inserted every one second. At the live stream origin, each RTSP stream is packaged into three HLS streams with segment lengths of 2s, 5s and 10s respectively, which creates a total of 6 stream scenarios. The HLS streams are delivered via HTTPS to the hls.js client above using TCP as the underlying protocol, which employs BBR as its congestion control mechanism. All streaming sessions are terminated after five minutes worth of video have been streamed.

B. Performance Metrics

In this work, we evaluate the following QoE metrics:

- Initial startup delay: the duration a client spends waiting before the video starts streaming
- Buffering: given the same 5-minute streaming duration, how long does the streaming stall
- Live stream latency: the gap between the client's streaming progress and the live origin's production progress

Under each of the 6 stream scenarios, we begin by evaluating hold-0 scheme's performance. Hold-0 effectively means that the MEC server plainly breaks the E2E connection into two parts (i.e., RAN and backhaul) and does not hold any video segment's availability from the client. With the 2-part E2E connection, we then increase the number of held segments (denoted by hold-x) and evaluate their performances accordingly. As x increases, the optimal x is determined if it meets the following criteria:

1) The client streamed for 5 minutes without experiencing any stalling.
2) The minimal amount of live stream latency is introduced while meeting criteria 1).

Besides the QoE metrics, we also evaluate download throughputs that are experienced by the client and the MEC server respectively. The former indicates how many segments are successfully downloaded by the MEC server beforehand, and the latter provides more insight into the satellite backhaul link's TCP performance under various scenarios. Both metrics are measured on a per-video-segment basis.

Note that in this work, we do not evaluate the E2E scenario where the client directly streams video from the live origin without passing through any MEC server. This is because it is already verified in [9] that E2E streaming always lead to significantly higher initial startup delay due to TCP's slow-start and retransmission mechanisms.

C. Satellite Backhaul Throughput Performance

We begin by evaluating the satellite backhaul's throughput performance, which indicates the average data rate that the MEC server experienced when it downloads each video segment from the live origin over the satellite link.

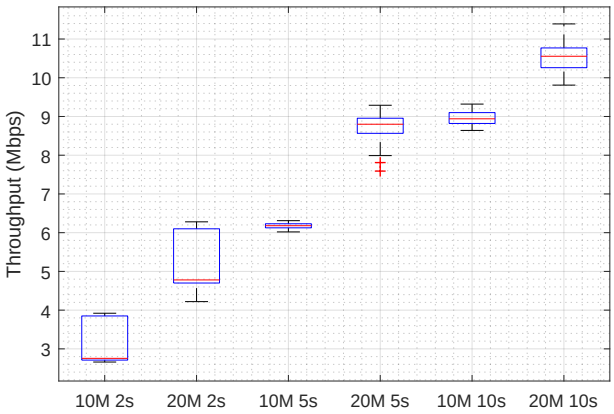

Fig. 5: Satellite backhaul throughput: all streaming scenarios with hold-0 scheme

We first evaluate the effect of video segment size (in bytes) and length (in seconds) on backhaul throughput. In Figure 5, we plot the 95-percentile backhaul throughput results under the hold-0 scheme. Note from the x-axis that we sort the streaming scenarios in ascending orders of segment size and segment length respectively. It is directly observed from Figure 5 that larger segment size leads to higher backhaul throughput, which verifies that file size is the bottleneck of TCP performance over the satellite link. Furthermore, if we compare each pair of streaming scenarios with similar segment sizes, the one with shorter segment length always exhibit higher variance. For example, 20Mbps-2s scenario's throughput has standard deviation of 0.69Mbps, while 10Mbps-5s scenario's standard deviation is only 0.07Mbps. This is because shorter segment size means more frequent requests and hence more bursty data transmissions, which is more prone to the satellite channel fluctuation in terms of latency and packet errors. During our experiments, out of the 27,591,951 TCP packets that were sent via the satellite backhaul, there were 133,930 TCP retransmission events which accounts for 0.49% of all packets. This is a significant value in terms of TCP retransmissions and verifies our statement above.

We then evaluate the effect of transient segment holding operation on backhaul throughput. In Figure 6, within each streaming scenario, we plot the 95-percentile backhaul throughput results under each holding scheme. First, it is observed that hold-0 and hold-1 schemes experience similar backhaul throughputs, as they both utilize just one TCP connection over the satellite link. Second, it is shown that as more parallel TCP connections are opened, *each* TCP connection experiences lower throughput in general. This verifies our earlier statement in Section III-B that in a channel with limited bandwidth (e.g., satallite link), the number of parallel TCP connections should be limited to avoid causing congestion over the channel.

It is worth noting that as more parallel TCP connections are opened, even though *each* connection experiences slightly degraded performance, the overall throughput that is aggre-

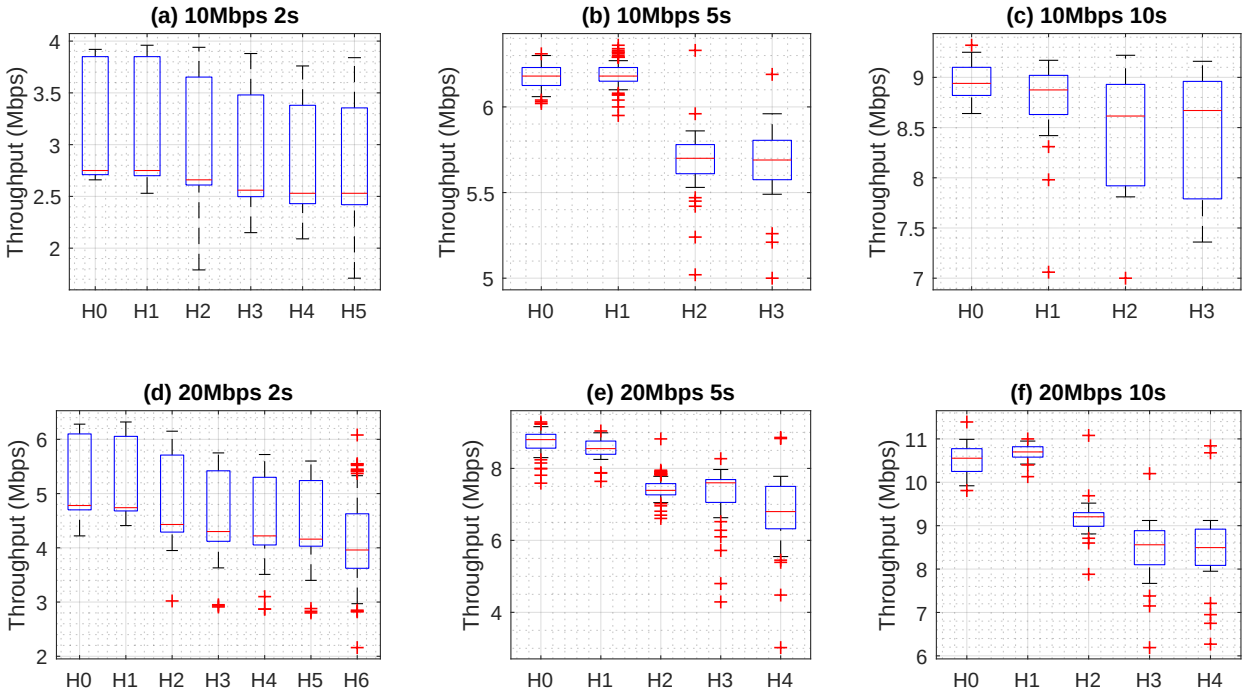

Fig. 6: Satellite backhaul per-TCP-connection throughput: all streaming and holding scenarios, where H0 refers to hold-0 scheme etc.

gated over *all* connections still increases significantly. This is because the parallel TCP connections are used to download multiple video segments simultaneously. Take the 10Mbps-2s scenario as an example, the hold-0 to hold-5 schemes' *aggregated* throughputs' medians are 2.76Mbps, 2.75Mbps, 5.32Mbps, 7.68Mbps, 10.12Mbps and 12.65Mbps respectively.

D. Client-Perceived Throughput Performance

We now evaluate the throughput that is actually perceived by the client, i.e., the data rate that the client experiences when downloading each video segment. Recall from Section III-A that for a segment download session, its client-perceived throughput takes into account both backhaul throughput (if the segment was not downloaded in time by the MEC server before being requested) and RAN throughput. The 95-percentile results under each streaming scenario and each holding scheme are plotted in Figure 7.

First, it is observed that under all streaming scenarios, hold-0 and hold-1 schemes produced very similar client-perceived throughput results, which match their performance over the backhaul. This shows that holding one segment over the satellite backhaul provides little benefit for the client, as the throughput over a single TCP connection is significantly lower than the required video bitrate. As the number of held segments increases, in general the overall client-perceived throughput increases as a higher proportion of video segments are downloaded to MEC server beforehand.

Recall from Section IV-C and Figure 6 that in some scenarios, although holding too many segments can cause each TCP connection's throughput to decrease, the overall throughput that is aggregated over all connections still increases. This statement is further verified in the client-perceived throughput results. Take the 20Mbps-5s stream as an example, the hold-3 and hold-4 schemes produced median backhaul (per-connection) throughputs of 7.6Mbps and 6.8Mbps respectively, which is shown in Figure 6(e). However, their aggregated backhaul throughputs are 22.8Mbps and 27.2Mbps respectively, which led to their median client-perceived throughputs of 61.4Mbps and 187Mbps respectively as reflected in Figure 7(e). This verifies that despite the trade-off between per-connection performance and the number of held segments, the overall benefit that is brought by utilizing parallel TCP connections to boost aggregated backhaul throughput is still significant.

E. Client QoE Performance

After evaluating the throughput performance and discussing their patterns, we now assess the clients' QoE results while focusing on the KPIs that are listed in Section IV-B, i.e., initial delay, buffering duration and live streaming latency. We list detailed statistical results in Table I. Note that in the table, we also include each scenario's mean client-perceived throughput and the percentage of video segments that are successfully downloaded by MEC server before they are requested by clients. We believe these two metrics provide additional insights into the QoE performance evaluation. For each streaming scenario in Table I, we highlight two columns due to their significance. The first highlighted column contains hold-0 scheme that serves as the performance benchmark, and the second highlighted column contains the scheme that holds the *optimal* number of segments for the corresponding streaming scenario.

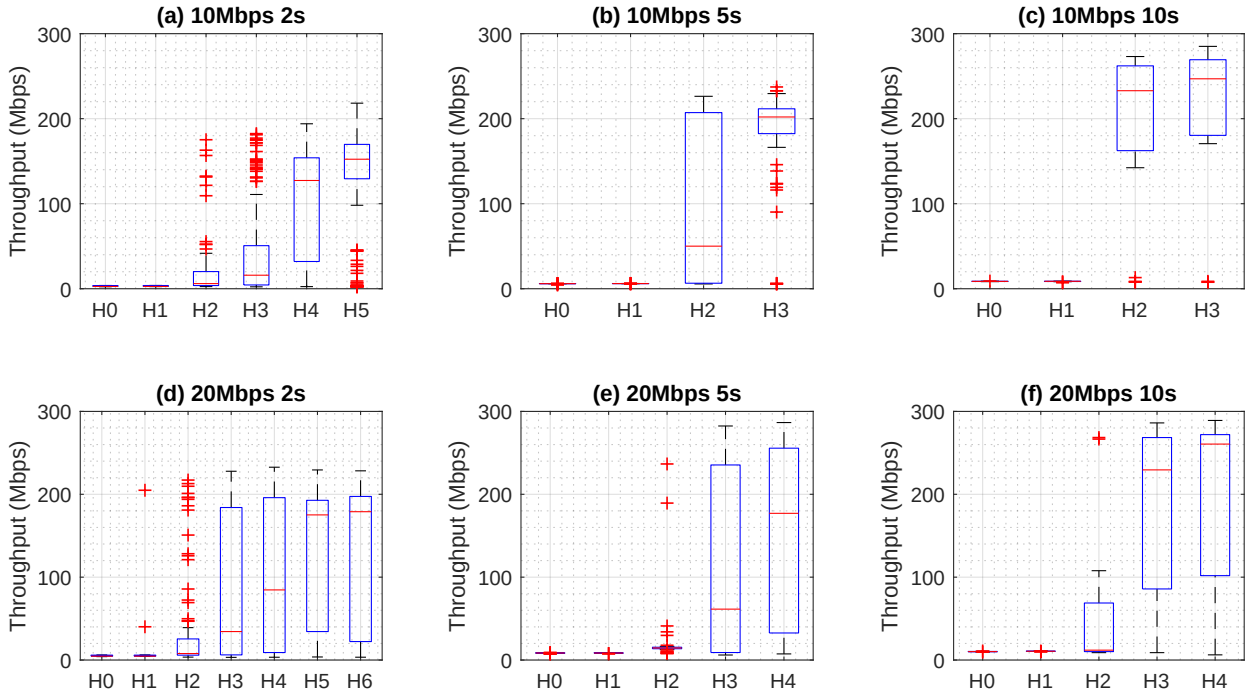

Fig. 7: Client-perceived throughput: all streaming and holding scenarios, where H0 refers to hold-0 scheme etc.

The first observation is that under all scenarios, all holding schemes produced almost the same initial startup delay (mostly 2.6s to 2.7s). The initial delay is calculated at the client side using the first request's timestamp and the time that the first video segment is decoded successfully. This observation verifies that breaking E2E content delivery path into two segments at the MEC server can assure initial startup delay, as earlier works showed that video streams often experience initial delay of more than 10s to 20s when being consumed E2E [9]. Since an initial delay of 10s or 20s will cause around 15% or 35% of users to quit watching a stream before it even starts [18], being able to maintain a sub-3s initial delay provides an important assurance for live streaming users' QoE.

We now look into the 10Mbps-2s scenario's results. First, both hold-0 and hold-1 schemes produced a mean client-perceived throughput of only 3.1Mbps, which is significantly lower than the required 10Mbps. As a result, both schemes caused the streaming to stall for a total of over 10 minutes (644.8s and 641.2s respectively) while watching 5-minute worth of video. We identify hold-4 to be the optimal scheme in this case, because it is the scheme that produces zero buffering as well as minimal live streaming latency. Although hold-5 produces a higher mean throughput (135.7Mbps compared to hold-4's 97.3Mbps) and is able to pre-download 84.5% (compared to hold-4's 53.7%) of all video segments, it incurs a higher live stream latency as well due to the extra segment that is transiently held by the MEC server.

Under both 10Mbps-5s and 10Mbps-10s scenarios, only 2 segments need to be held to provide a stalling-free streaming experience. This is because the larger segment file size leads to improved TCP performance, which means less parallel TCP connections are needed to bring the aggregated backhaul throughput over the 10Mbps threshold. Meanwhile, note that under the 10Mbps-2s, 5s and 10s scenarios, their optimal holding schemes incurs live streaming latencies of 10.6s, 12.7s and 22.7s respectively. Therefore, given the same video quality (10Mbps) and QoE assurance (i.e., no buffering), despite the TCP performance benefit that is introduced by longer segment length, using 2s segments still incurs the least amount of live latency.

Under the 20Mbps streaming scenarios, the performance patterns are generally similar to the ones under the 10Mbps scenarios. The 20Mbps-2s, 5s and 10s scenarios require 6, 4 and 3 segments to be held respectively to assure the client's QoE. Meanwhile, they incur live streaming latencies of 14.5s, 22.6s and 32.9s respectively. Therefore, even with the higher bitrate requirement (20Mbps), we still establish the same observation that using 2s segments incurs the lowest live latency while assuring a stalling-free streaming session.

Based on the discussions above, we establish the following key observations and conclusions:

- We have successfully validated the effectiveness of our proposed transient segment holding mechanism regarding the assurance of HTTP live streaming clients' QoE over a real satellite backhaul. Furthermore, we use six typical streaming scenarios in real experiments to provide practical guidelines on how to establish segment holding policies in real networks.
- We have verified that file size (in bytes) is the bottleneck of TCP performance over a satellite backhaul. Furthermore, if multiple streaming scenarios have similar video segment sizes, the ones with longer segment length (in seconds) produces better TCP performance.
- We have verified that as more parallel TCP connections

TABLE I: Statistics on key performance metrics' results

		Holding Scheme						
		Hold-0	Hold-1	Hold-2	Hold-3	Hold-4	Hold-5	Hold-6
10Mbps 2s	Client-Perceived Throughput (Mbps)	3.1	3.1	17.8	40.8	97.3	135.7	-
	Total Backhaul Throughput (Mbps)	3.2	3.2	6.1	8.7	11.3	14.1	-
	Initial Delay (s)	2.7	2.7	2.6	2.6	2.6	2.6	-
	Buffering Duration (s)	644.8	641.2	201.6	52.7	0	0	-
	Live Streaming Latency (s)	644.8	643.2	205.6	58.7	8	10	-
	Prefetched Segments (%)	0%	0%	4.7%	16.8%	53.7%	84.5%	-
10Mbps 5s	Client-Perceived Throughput (Mbps)	5.9	6.1	106.3	178.3	-	-	-
	Total Backhaul Throughput (Mbps)	6.2	6.2	11.4	17.0	-	-	-
	Initial Delay (s)	2.6	2.5	2.7	2.8	-	-	-
	Buffering Duration (s)	197.4	187.2	0	0	-	-	-
	Live Streaming Latency (s)	197.4	192.2	10	15	-	-	-
	Prefetched Segments (%)	0%	0%	47.5%	88.1%	-	-	-
10Mbps 10s	Client-Perceived Throughput (Mbps)	8.7	8.6	190.6	212.7	-	-	-
	Total Backhaul Throughput (Mbps)	8.8	9.0	17.0	25.4	-	-	-
	Initial Delay (s)	2.8	2.6	2.7	2.7	-	-	-
	Buffering Duration (s)	44.9	42.1	0	0	-	-	-
	Live Streaming Latency (s)	44.9	52.1	20	30	-	-	-
	Prefetched Segments (%)	0%	0%	79.3%	89.7%	-	-	-
20Mbps 2s	Client-Perceived Throughput (Mbps)	5.1	6.7	29.3	83.1	102.6	126.9	122.3
	Total Backhaul Throughput (Mbps)	5.2	5.3	9.7	13.9	18.1	22.5	24.6
	Initial Delay (s)	2.6	2.7	2.7	2.6	2.7	2.6	2.5
	Buffering Duration (s)	838.2	806.2	325.4	152.3	62.9	3.3	0
	Live Streaming Latency (s)	838.2	808.2	329.4	158.3	70.9	13.3	12
	Prefetched Segments (%)	0%	0.7%	7.4%	34.2%	42.9%	57.4%	58.4%
20Mbps 5s	Client-Perceived Throughput (Mbps)	8.4	8.4	21.6	101.9	153.6	-	-
	Total Backhaul Throughput (Mbps)	8.6	8.7	14.9	22.0	27.2	-	-
	Initial Delay (s)	2.7	2.5	2.8	2.5	2.6	-	-
	Buffering Duration (s)	399.7	400.2	112.7	10.9	0	-	-
	Live Streaming Latency (s)	399.7	405.2	122.7	25.9	20	-	-
	Prefetched Segments (%)	0%	0%	1.7%	29.3%	55.2%	-	-
20Mbps 10s	Client-Perceived Throughput (Mbps)	10.1	10.6	51.9	188.8	196.8	-	-
	Total Backhaul Throughput (Mbps)	10.5	10.7	18.4	25.2	33.8	-	-
	Initial Delay (s)	2.6	2.7	2.9	2.9	2.5	-	-
	Buffering Duration (s)	279.5	258.1	24.5	0	0	-	-
	Live Streaming Latency (s)	279.5	268.1	44.5	30	40	-	-
	Prefetched Segments (%)	0%	0%	13.8%	71.4%	75.9%	-	-

are opened over a satellite backhaul link, each *individual* connection experiences degraded throughput. However, such degradation does not affect the fact that the *aggregated* throughput over all connections still increases.

- We have observed that given the same video quality requirement, even though longer segment length (i.e., larger file size) means better performance over single TCP connections, the 2s-segment scenarios still incur the least amount of live streaming latency even when more segments need to be held. This provides valuable insights into practical streaming operations, as we have verified that there is no need to use longer segment length in the hope of improving TCP performance, as it may even further degrade live streaming latency.

V. Conclusion

In this work, we have proposed and developed for the first time in the literature a system that provides QoE assurance for HTTP-based live streaming applications in 5G networks with satellite backhaul. Specifically, it leverages the context awareness and flexibility that are enabled by the 5G service-based architecture (SBA) and the satellite backhaul's multicast capability. We envisage that through virtualization technologies, third-party stakeholders such as content providers (CPs) are able to deploy their own virtual network functions (VNFs) in multi-access edge computing (MEC) servers at 5G mobile edge. These VNFs perform the transient segment holding technique that uses multiple parallel TCP connections to compensate the suboptimal TCP performance over satellite backhaul. Furthermore, we propose that if a video stream is popular among multiple MEC servers, its video segment files should be delivered from the CP's origin server to multiple MEC servers simultaneously using multicast-based file delivery protocols. These techniques ensure effectiveness and efficiency of file delivery as well as QoE assurance.

We have systematically evaluated the proposed scheme's performance over a real satellite backhaul link under a wide variety of practical scenarios. Through these innovative over-the-air experiments, we have validated the effectiveness of the proposed QoE assurance scheme. Furthermore, we provide

practical insights into not only the intrinsic behaviour of TCP over a satellite backhaul under live streaming scenarios, but also the establishment of transient segment holding policies under different streaming settings. These further pave the way for the successful integration of satellite backhaul into 5G networks.

ACKNOWLEDGEMENT

This work is funded by EU H2020 SaT5G project under grant number 761413.

The authors would also like to acknowledge the support of the University of Surrey's 5G Innovation Centre (5GIC) (http://www.surrey.ac.uk/5gic) members for this work.

REFERENCES

[1] 3GPP. TS 22.261: Service requirements for next generation new services and markets. [Online]. Available: http://www.3gpp.org/DynaReport/22261.htm

[2] K. Liolis, A. Geurtz, R. Sperber, D. Schulz, S. Watts, G. Poziopoulou, B. Evans, N. Wang, O. Vidal, B. Tiomela Jou, M. Fitch, S. Diaz Sendra, P. Sayyad Khodashenas, and N. Chuberre, "Use cases and scenarios of 5G integrated satellite-terrestrial networks for enhanced mobile broadband: the SaT5G approach," *International Journal of Satellite Communications and Networking*, pp. 1–22.

[3] K. Liolis, A. Geurtz, R. Sperber, D. Schulz, S. Watts, G. Poziopoulou, B. Evans, N. Wang, O. Vidal, B. T. Jou, M. Fitch, S. S. Diaz, P. S. Khodashenas, and N. Chuberre, "Satellite use cases and scenarios for 5G eMBB," in *Satellite Communications in the 5G Era*. IET, 2018, pp. 25–60.

[4] C. Ge, N. Wang, G. Foster, and M. Wilson, "Toward QoE-assured 4K video-on-demand delivery through mobile edge virtualization with adaptive prefetching," *IEEE Transactions on Multimedia*, vol. 19, no. 10, pp. 2222–2237, Oct 2017.

[5] N. Wang, N. Nouwell, C. Ge, B. Evans, Y. Rahulan, M. Boutin, J. Desmauts, K. Liolis, C. Politis, S. Votts, and G. Poziopoulou, "Satellite support for enhanced mobile broadband content delivery in 5G," in *2018 IEEE International Symposium on Broadband Multimedia Systems and Broadcasting (BMSB)*, June 2018, pp. 1–6.

[6] SaT5G Project - Horizon 2020. [Online]. Available: http://sat5g-project.eu/

[7] ETSI. Mobile Edge Computing - a key technology towards 5G. [Online]. Available: http://www.etsi.org/images/files/ETSIWhitePapers/etsi_wp11_mec_a_key_technology_towards_5g.pdf

[8] ——. MEC in 5G networks. [Online]. Available: https://www.etsi.org/images/files/ETSIWhitePapers/etsi_wp28_mec_in_5G_FINAL.pdf

[9] C. Ge, N. Wang, W. K. Chai, and H. Hellwagner, "QoE-assured 4K HTTP live streaming via transient segment holding at mobile edge," *IEEE Journal on Selected Areas in Communications*, vol. 36, no. 8, pp. 1–15, 2018.

[10] T. Paila, R. Walsh, M. Luby, V. Roca, and R. Lehtonen, "RFC 6726: FLUTE - File Delivery over Unidirectional Transport," 2012.

[11] "Akamai Introduces Predictive Video Over Cellular Capabilities." [Online]. Available: https://www.akamai.com/uk/en/about/news/press/2015-press/akamai-introduces-predictive-video-over-cellular-capabilities.jsp

[12] 3GPP. TS 23.501 - system architecture for the 5G system. [Online]. Available: http://www.3gpp.org/DynaReport/23501.htm

[13] N. Cardwell, Y. Cheng, C. S. Gunn, S. H. Yeganeh, and V. Jacobson, "BBR: Congestion-based congestion control," *Queue*, vol. 14, no. 5, pp. 50:20–50:53, Oct. 2016.

[14] ——. (March 2017) BBR congestion control: an update. [Online]. Available: https://www.ietf.org/proceedings/98/slides/slides-98-iccrg-an-update-on-bbr-congestion-control-00.pdf

[15] CEPT. ECC Report 280 - satellite solutions for 5G. [Online]. Available: https://www.cept.org/files/9522/Draft%20ECC%20Report%20280%20on%20satellite%20solutions%20for%205G_for%20web.docx

[16] L. Pardue and R. Bradbury, "Hypertext Transfer Protocol (HTTP) over multicast QUIC," Internet Engineering Task Force, Internet-Draft draft-pardue-quic-http-mcast-03, Aug. 2018, work in Progress. [Online]. Available: https://datatracker.ietf.org/doc/html/draft-pardue-quic-http-mcast-03

[17] SES ASTRA 2F Satellite. [Online]. Available: https://www.ses.com/our-coverage/satellites/344

[18] S. S. Krishnan and R. K. Sitaraman, "Video stream quality impacts viewer behavior: Inferring causality using quasi-experimental designs," *IEEE/ACM Transactions on Networking*, vol. 21, no. 6, pp. 2001–2014, Dec 2013.