HoloSync: Frame Synchronisation for Multi-Source Holographic Teleportation Applications

Sweta Anmulwar*, Ning Wang*, Vu San Ha Huynh*, Stewart Bryant*, Jinze Yang† , Rahim Tafazolli*
*Institute for Communication Systems 6G/5GIC, University of Surrey, Guildford, United Kingdom
{s.anmulwar, n.wang, v.huynh, s.bryant, r.tafazolli}@surrey.ac.uk
†Huawei Technologies Co., Ltd.
{yangjinze}@huawei.com

***Abstract*—Live holographic teleportation is an emerging media application that allows Internet users to communicate in a fully immersive environment. One distinguishing feature of such an application is the ability to teleport multiple objects from different network locations into the receiver's field of view at the same time, mimicking the effect of group-based communications in a common physical space. In this case, live teleportation frames originated from different sources must be precisely synchronised at the receiver side to ensure user experiences with eliminated perception of motion misalignment effect. For the very first time in the literature, we quantify the motion misalignment between remote sources with different network contexts in order to justify the necessity of such frame synchronisation operations. Based on this motivation, we propose HoloSync, a novel edge-computing-based scheme capable of achieving controllable frame synchronisation performances for multi-source holographic teleportation applications. We carry out systematic experiments on a real system with the HoloSync scheme in terms of frame synchronisation performances in specific network scenarios, and their sensitivity to different control parameters.**

***Index Terms*—Extended Reality (XR), Frame synchronisation, Multi-Source, Teleportation, Holographic-type Communication, Edge-computing**

I. Introduction

Holographic teleportation is a new type of extended reality (XR) application that allows humans or objects to be teleported live into the view space of remote receivers [1] [2]. This technology supports six-degree of freedom (6DoF) of object viewing (including three dimensions of head movement – yaw, pitch, and roll, as well as three dimensions of body movement – left-right, forward-backwards, and up-down), creating fully immersive experiences during a teleportation session [3]–[7]. In comparison to traditional video applications, today's holographic teleportation applications are still in their infancy, in the sense that large-scale communications over the global public Internet are difficult to support due to both the application's requirement for high data rates and the lack of agility in dealing with complex and uncertain network conditions during content transmissions [8]–[10]. It is a unique feature of holographic teleportation to simultaneously teleport people from multiple remote Internet locations to the same view space in an immersive fashion, mimicking the effect of group-based interactions in a common physical space. It is possible that such a feature will enable more advanced application scenarios in the future, such as distributed theatre/music performances at an Internet scale, where individual players simultaneously play at different network locations, and their holograms are transmitted in real-time to the audience, giving the perception that they are physically playing together in the common space in front of the audience [11]. In addition to the traditional network requirements for supporting holographic teleportation applications, such as high bandwidth, data synchronisation will be critical in supporting multi-source-based teleportation operations for ensuring user Quality-of-Experience (QoE) [12]. To avoid perceived motion misalignment between source objects, teleportation content frames originated from each source object with the same frame creation time must arrive at the receiver side within a stringent time window [13] [14]. It is expected that human tolerance to such motion misalignment varies depending on the application scenario. For example, in the case of casual chatting with minimal body movement, frame synchronisation error may not be a significant issue. However, in other scenarios with a greater degree of body movement, motion misalignment is more likely to be detected by human eyes on the viewer's side. There are multiple factors in the operational environment which may contribute to the motion misalignment, such as network distance/latency, path load conditions, frame production conditions, and end-to-end transport layer mechanisms applied [15] [16]. Extended from our recent work [17], we propose the HoloSync scheme for achieving frame synchronisation in live holographic teleportation applications. We systematically carry out our evaluation study on the frame synchronisation performances based on a real-life teleportation platform called LiveScan3D [18]. We emulate a wide range of Internet path condition scenarios and quantify the resulting misalignment of frame arrival time at the viewer side. In addition, we propose the HoloSync scheme for achieving frame synchronisation in live holographic teleportation applications. Specifically, we propose a 5G edge-computing [19] based frame synchronisation scheme in order to achieve bounded frame synchronisation errors, and such an error bound can be flexibly configured according to specific holographic teleportation application scenarios. In the context of such a scheme, we propose three different frame handling policies depending on the actual arrival time, namely simple forwarding, pairing with cached frame, and frame dropping.

The authors would like to acknowledge the support of University of Surrey's 5G/6GIC (http://www.surrey.ac.uk/5gic) members for this work.

This article has been accepted for publication in IEEE Transactions on Multimedia. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TMM.2022.3207280

Fig. 1: Handshake through real-life holographic teleportation platform

We perform systematic performance evaluations based on distinct network scenarios with regard to the frame statistics according to these three scenarios.

One real-life example of multi-source holographic teleportation is demonstrated in Figure 1. It shows two remote people performing virtual handshake actions with each other to be viewed by the audience wearing head-mounted devices such as Microsoft Hololens [20]. In such a simple application, motion misalignment introduced by different network distances from the two sources may lead to receiver-perceivable misalignment in the movement of their arms. The experiment was conducted at the Institute for Communication Systems, University of Surrey, UK.

It is worth mentioning that the purpose of this example is to indicate the importance of frame synchronisation in live-streaming of holographic teleportation content from multiple sources to common receivers. In addition to this, it is also necessary to achieve peering synchronisation between the two persons performing virtual handshake, even including haptics, but this is outside the scope of this paper. We highlight the major technical contributions from this paper as follows:

- For the very first time in the literature, we demonstrate the necessity of frame synchronisation operations when such an application is to be operated across the global Internet. We quantify the motion misalignment effect with different network contexts and prove that, without frame synchronisation operations, it can be in the order of seconds, leading to significantly suboptimal user experiences.
- We propose HoloSync, a sophisticated edge computing-based frame synchronisation scheme to eliminate user-perceived motion misalignment effects with different network scenarios. To the best of our knowledge, this is the first solution proposed to address the issue of motion misalignment in holographic teleportation applications. While we developed the HoloSync approach based on the representative application platform called LiveScan3D, the proposed synchronisation algorithm can certainly be applied to similar holographic teleportation applications based on live frame streaming mechanisms.
- We carry out comprehensive performance studies on HoloSync, including the evaluation of a wide range of performance metrics such as frame freshness statistics, playback latency and pairing error. We also analyse the performance sensitivity to the key control parameters used in the synchronisation scheme, namely frame waiting window size, and frame pairing approximation threshold. Through such evaluations, we show that the proposed HoloSync scheme is able to achieve assured user experiences by eliminating perceived motion misalignment effects.
- Based on an in-depth analysis of the results of the experiments, we further derive useful guidelines and policies for appropriately setting the system control parameters, including the synchronisation window and the timestamp approximation threshold, so that the overall teleportation performance can be tailored to specific use case scenarios and contexts.

The remainder of the paper is structured as follows. We discuss the related work in section II. In section III, we provide an overview of the holographic system and show the necessity for synchronisation. In section IV, we specify the proposed frame synchronisation scheme in detail. Section V focuses on the system-level description of the holographic teleportation platform using LiveScan3D as an example, followed by extensive real-world experiment results with full and reduced frames per second (FPS) performances at the source with a wide range of testing scenarios with different network contexts. It also discusses guidelines for setting synchronisation parameters. Finally, in section VI, we conclude the paper.

II. Related Work

In the research community, there is a huge interest in holographic-type communication. The latest work in holographic communication can be found in [21]–[29].

The authors of [21], discuss the technical networking challenges of enabling Holographic-Type Communication (HTC). A novel network architecture for dynamically establishing new flows with very low latency is presented by the authors. Experimentation is carried out on a Mininet-based emulated client-server network setup. According to the results, the fully distributed control plane has the lowest latencies in terms of flow setup, first segment download, and average per segment download.

In our recent work [22], we describe a novel Holographic Type Communication (HTC)-based teleportation platform. It demonstrates the advantage of locating the production server function close to the viewer side, as the risk of user experience degradation caused by long-distance paths between the local clients on the source side and the production server function on the receiver side can be mitigated by an intelligent frame buffering mechanism executed by the production server side.

The authors of [23] propose PCC-DASH (Point Cloud Compression – Dynamic Adaptive Streaming over HTTP), a standards-compliant method for HTTP adaptive streaming of scenes composed of multiple, dynamic point cloud objects. The authors' experiments with various rate adaptation methods, and the results show that buffer size is critical. A smaller buffer size improves accuracy and video quality while being affected by playout freezes.

Another paper [24], examines the impact of point cloud compression (PCC) and HTTP Adaptive Streaming (HAS) on the perceived quality of a scene. Subjects are shown volumetric test videos, and they rated volumetric videos lower in terms of perceived quality when compared to traditional HD or 4K videos. The experiment also discovers that subjective and objective metrics are highly correlated in the case of adaptive point cloud streaming. Nabin Kumar Karn et al. [25] propose a new quality-aware adaption mechanism for 3D MVD (Multi-view plus depth) over the Internet. It significantly improves the perceived video quality of 3D even in poor network conditions. Most of the 2D video conferencing applications use Real-time Transport Protocol (RTP) and Web Real-Time Communication (WebRTC) protocol [26] [27]. RTP protocol uses RTCP Sender Reports (SR) to synchronise the RTP flows from different users and it is not based on the concept of frame. SR contains sample clock i.e. RTP timestamp and NTP timestamp of the RTP packets. These two timestamps are paired to get the absolute time of the sample in a stream. The receiver needs to wait for the RTCP SR and Source Description (SDES) packets which contain a canonical end-point identifier (CNAME) to identify the user interface (sender/source) and then based on the SR reports synchronisation/mapping of flows is carried out. WebRTC also uses the RTCP SR reports to synchronise the flows [28]. Moving Picture Experts Group - Dynamic Adaptive Streaming over HTTP (MPEG-DASH) is another protocol used mainly for one-way streaming of video, but it is not used for interactive applications [29].

III. HoloSync System Overview

A. Basic Background System

A basic single-source teleportation scenario is depicted in Figure 2, which includes all necessary system components, based on the LiveScan3D platform as an example. Multiple sensor cameras surrounding the person being teleported are responsible for capturing this person in a 3D hologram (for simplicity, only one camera is shown in Figure 2). The 3D hologram is captured as a point cloud, which is a collection of all the captured points. Each point represents the (X, Y, Z) co-ordinates, and the RGB (0-255) information associated with that point. We used Microsoft RGB-D Kinect v2 for experimentation, and each frame, which captures the whole scene contains approximately 144000 points, resulting in 17.38 Mb (or 2.16 MB) per frame. For transmitting a single point, 15 Bytes are needed; 3 Bytes for colour information (1 Byte each for R, G and B), and 4 Bytes each for X, Y and Z co-ordinates (float) i.e. 12 Bytes in total for the co-ordinates. Frames are produced at a rate of 30 Frames Per Second (FPS). In Livescan3D, there are two types of settings for the holographic experience; first to only teleport human bodies and second to teleport the whole scene. This paper specifically focuses on teleporting human bodies, in which case the data rate is less than 100Mbps for each teleported object. Additionally, less than 200kbps is required for the joints and skeleton, that can be neglected because its size is small compared to the point cloud. The payload size is dependent on encoding and actual codecs used might be different for other applications. Similarly, to our previous work [22], the bandwidth capacity of the link is 1 Gbps with an artificially introduced delay and packet loss ratio, mimicking the bandwidth over-provisioning scenario of the public Internet.

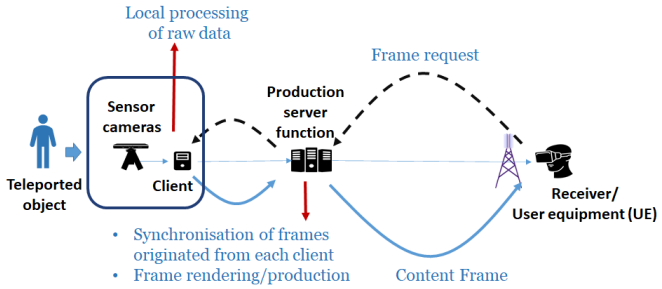

Fig. 2: Basic system diagram of the LiveScan3D holographical teleportation platform

Each camera is locally connected to a computer called *client*, which handles local data processing and continuous frame production. The production server, depicted in the centre of the diagram, is a specialised high-power computer node in charge of rendering final frames based on raw frames received from clients connected to the cameras surrounding that remote object. The output of the production server is a 3D holographic object that can be seen in 6DoF at the receiver side. It is also worth noting that, thanks to Network Function Virtualisation (NFV) [30] [31], the production server function can be virtualised and deployed flexibly at any location of the network, including a Multi-access Edge Computing (MEC) server in 5G. It is also important to have a time-synchronised network to get the correct frame production timestamps at the client side and to synchronise the frames at the production server. For time synchronisation Network Time Protocol (NTP) is used across all the connected devices. Figure 3 presents a more generalised scenario of group-based holographic teleportation operations in which multiple objects are teleported from different remote sites. As previously stated, from a network topology viewpoint, the production server becomes a central aggregation point where raw frames received from individual remote source clients are rendered on a per-object basis before being served to the receivers. In this case, the rendered frames, with full 3D views for each teleported object, are streamed in real-time to the receiver side over 5G new radio (NR). As a result, the receiver can see all the teleported objects in the common view field while also manipulating and adjusting each individual object, including its size, position, and orientation.

This article has been accepted for publication in IEEE Transactions on Multimedia. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TMM.2022.3207280

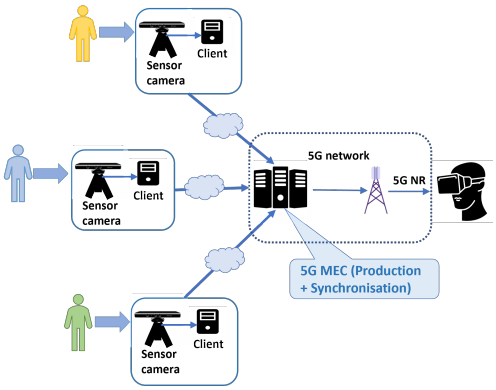

Fig. 3: 5G MEC for supporting Multi-source Holographic Teleportation

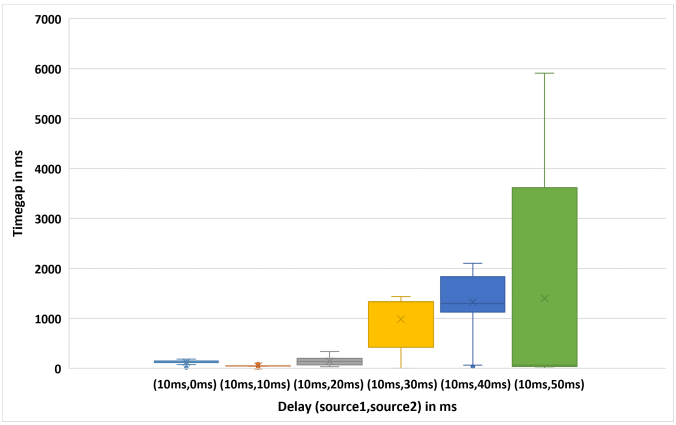

Fig. 5: Difference between frame pair with respect to increasing delay

In general, a MEC server is also responsible for synchronising outgoing frames on a per-teleported-object basis. Paths from multiple sources at different Internet locations to the MEC server can be different due to network latencies and traffic conditions. It is worth noting that the heterogeneous end-to-end network latency from different network locations naturally results in different data throughput performances, which can further impact the time to deliver a frame to the MEC server. In such a situation, the arrival times of raw frames coming from various sources during a live teleportation session are highly uncertain. In the context of 5G, the function of frame synchronisation (along with production) across multiple remote sources can be fulfilled by the MEC server in 5G, and it is able to provide the synchronisation service to all the potential receivers attached to it. [32]. Because of the more deterministic content data delivery capability of 5G new radio, once the frames are synchronised by the MEC node, there is a very low risk of synchronisation disruptions along their journey to the receiver's UE [33]–[35].

B. The necessity of frame synchronisation

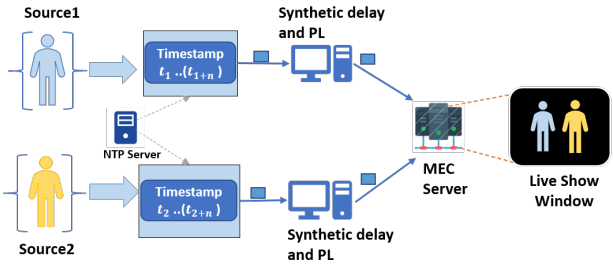

Fig. 4: Experimentation set-up for two sources

Now we justify the necessity for frame synchronisation operations through experiments without such a mechanism, as shown in Figure 4. The main purpose is to quantify the motion misalignment effect across sources at different network locations by natively streaming teleportation frames from them. Two remote sources are connected to the MEC server through emulated Internet paths with dedicated middleboxes for introducing synthetic delay and packet loss, as shown in Figure 4. We repeat the experiment with delay from 0ms to 50ms, with a constant packet loss of 0.01% which is a realistic setting in operational Internet environments [36] [37]. As shown in Figure 4, frames from the two sources, located at different network distances (in terms of latency), arrive at the MEC server before being forwarded to the user equipment (UE).

Through the experiment, we made the following observations:

- The average time gap between the frames is 122 ms, with a delay of 10ms on $Source_1$ and 0ms on $Source_2$, as shown on the X-axis of Figure 5. With the exception of (10ms, 10ms), the time gap between the frame pairs grows. For more challenging scenarios, ranging from (10ms, 30ms) to (10ms, 50ms), the average time gap increases to a range of 980ms to 1400ms, or approximately 1 second to 1.5 seconds. In the worst-case scenario, the gap can be more than 5 seconds, causing considerable motion misalignment between the two teleported objects, resulting in poor quality of user experience. The long playback latency incurred during long-distance streaming of holographic content was also observed in the state of the art [22], with real network locations across the whole Internet. The key reasons include the reduced end-to-end TCP throughput for streaming holographic content with long Internet distances, and the frame buffering mechanism for assuring FPS performances in long distance cases.
- If the network distance is large, the motion misalignment between sources can be on the order of seconds, as shown in Figure 5, which is unacceptable in terms of user QoE.
- The above observations indicate that frame synchronisation is required for multi-source teleportation applications in order to mimic group-like immersive experiences with teleported objects at various network locations.

In addition to network delay, there can be other factors such as transport layer mechanisms that may also affect

This article has been accepted for publication in IEEE Transactions on Multimedia. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TMM.2022.3207280

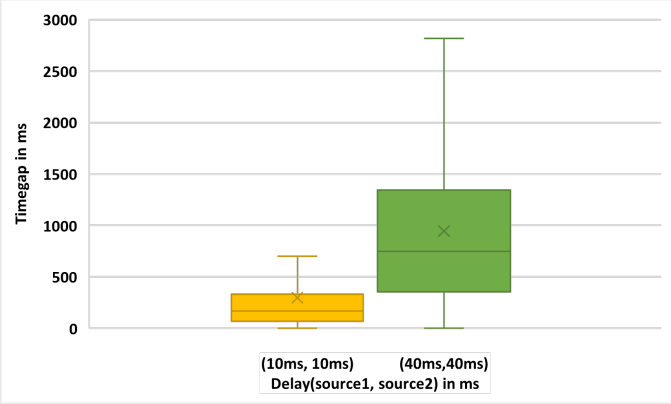

Fig. 6: Difference between frame pair due to heterogeneous TCP congestion control algorithms

the frame delivery performance at the receiver side. Today, a number of transport-layer congestion control algorithms are used in the public Internet. To understand the effect of heterogeneous transport-layer congestion control mechanisms on motion misalignment, we conducted the experiments on the same set-up as shown in Figure 3, but the TCP congestion control algorithms are different for $Source_1$ and $Source_2$. For $Source_1$, Bottleneck Bandwidth and Round-trip propagation time (BBR) was set as the congestion control algorithm and for $Source_2$, CUBIC was set as the congestion control algorithm. We conducted experiments by adding 10ms and 40ms delays with 0.01% packet loss for both the sources. The observations are as follows. In Figure 6, for the first case of (10ms, 10ms) network delay, the average frame pair time gap is 297ms, and it increases significantly to 945ms for the second case of (40ms, 40ms) delay. In the worst-case scenarios, the frame pair time gap can be 2800 ms which is expected to cause considerable motion misalignment between frames, and thus severely damaging the user experience. It indicates that if heterogeneous congestion control mechanisms are applied to the different network devices, it may cause motion misalignment between frames, even if the two sources have the same network distances/latencies.

IV. Detailed Specification of the HoloSync Framework

The terminologies and the frame synchronisation algorithm of the HoloSync scheme are explained in this section. Key terminologies related to synchronisation algorithm are explained in Section IV-A. The synchronisation algorithm is explained in full in Section IV-B.

A. Key terminologies

The terms used to specify the multi-source frame synchronisation are described below.

Timestamp: It is carried by each frame and is used to determine the time point when that frame is produced at the source. Individual frame timestamps from various sources are required inputs for frame synchronisation on the MEC server side.

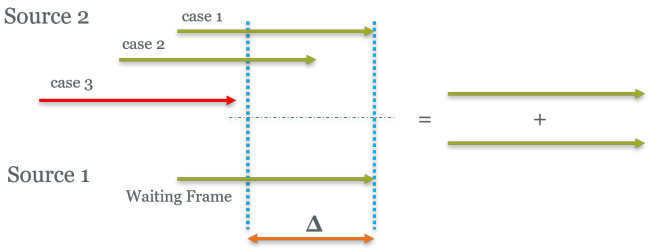

Fig. 7: Synchronisation window

Synchronisation window (Δ): To eliminate user-perceived motion misalignment, in the ideal case frames from different sources should be shown together if they have the same frame production timestamp. However, human eyes can tolerate small transient time gaps between multiple sources. This allows the teleportation system to wait for the frames that originate simultaneously from different sources but arrive at different times at the destination with a tolerable time gap between them. In particular, if two frames from different network locations sharing the same frame production timestamp arrive at different times because of network delay or other factors, the early arrived frame can wait for the late frame within a tolerable time window which we define as the synchronisation window Δ. It is shown in Figure 7 that once the frame from $Source_1$ arrives, it waits for the frame from $Source_2$. There are three cases for this scenario:

- Case 1: The frame from $Source_2$ arrives exactly at the same time, and can be naturally paired with that from $Source_1$.
- Case 2: The frame from $Source_2$ arrives late, but still within the synchronisation window Δ, and can still be paired with the corresponding frame from $Source_1$ with a tolerable delay gap between them.
- Case 3: The frame from $Source_2$ arrives after the synchronisation window expires, and hence it cannot be paired with the corresponding frame from $Source_1$.

Frame paring approximation threshold (γ): Cameras at individual sources may produce discrete frames at different time points (offset) even with fixed time intervals, in which case there is no guarantee that frames from these sources will share exactly the same frame timestamp in order to be paired. Such an offset is more obvious if these sources have different frame generation rates. As a result, we need an additional level of approximation in order to pair the frames together which have sufficiently small-time offset between their timestamps. Specifically, if the timestamps between frames coming from different sources are sufficiently close to each other (defined by the threshold γ), then the synchronisation function will treat them as if they have the same timestamp. In this sense, γ is also referred as the approximation threshold in the rest of the paper. For instance, if frames from the two sources have timestamps x and y respectively, then the difference between

these two timestamps $|x - y|$ is compared with γ. In case the difference is smaller than the approximation threshold, then the synchronisation function makes the approximation that they have the same timestamp. Based on this, the synchronisation algorithm is able to apply Δ to determine whether the two frames can be paired depending on their arrival time at the MEC server.

B. Synchronisation mechanism

The proposed frame synchronisation algorithm in HoloSync is executed at the MEC server on the receiver side in the context of 5G. Frame production timestamps are used in the synchronisation process. For the sake of simplicity, just two sources, $Source_1$ and $Source_2$ are considered, but the proposed synchronisation strategy can be used in a scenario with more than two sources, which will be explained later. In addition, the common practice of performing teleportation with immersive experiences requires multiple (e.g. 4) cameras surrounding the object to be teleported. To simplify the explanation, we denote all these cameras with one source camera from each remote location, as all of them share common physical paths to the MEC server with the same network treatment. We denote the latest frame production timestamps of $Source_1$ and $Source_2$ as t_{S1} and t_{S2}, respectively. For example, if a frame from $Source_1$ arrives first, it is referred to as Frame1, and it waits for the frame from $Source_2$ with the same frame timestamp. Elapsed time is the amount of time that passes after Frame1 starts waiting for a frame from $Source_2$. If in case the frame from $Source_2$ does not arrive at the MEC server, then previously cached frames from $Source_2$ are paired with the Frame1. The caching operation of incoming frames is based on a simple moving window that makes sure that the most recent incoming frames are transiently cached at the MEC server, which are later deleted on becoming obsolete.

Frames which are processed by the synchronisation function at the MEC server are delivered to the UEs (hololenses) attached to the MEC server. These processed frames are retained at the server and are referred to as cached frames. If the expected frame from the second source does not arrive inside the synchronisation window, the cached frame is coupled with the waiting frame if the time gap is within the approximation threshold. The timestamp of the most recent frame received from $Source_2$ is denoted by t_{S2}, and the timestamp of the previously cached frame from this source is denoted by (t'_{S2}). Frames may arrive late to the MEC server as a result of delays, packet loss, bandwidth fluctuations, or other conditions along the way. In that situation, a cached frame can be leveraged to achieve motion alignment if it is within the specified time gap γ.

We aim to increase the proportion of synchronised frames by applying the synchronisation algorithm instead of having a non-synced frame. The outcome of the frame synchronisation operation can be classified into the following three categories:

- *Fresh frames* - Frames with the same timestamp that arrive at the MEC server within the synchronisation window (Δ) from all connected sources.
- *Half fresh frames* - If a frame from one of the sources does not arrive within the synchronisation window, instead of discarding the frame, the system pairs a previously cached frame with the waiting frame. For instance, consider that the latest frame from $Source_1$ with timestamp t_{S1} arrives, and it waits for the frame from $Source_2$. If the expected frame from $Source_2$ does not arrive within the synchronisation window, a previously cached frame with a timestamp t'_{S2} is paired with it if the time difference between the frame pair is less than or equal to γ.
- *Dropped frames* - Frames that arrive at the MEC server which are neither fresh nor half-fresh frames are dropped, and are not used.

The basic working mechanism of HoloSync is as follows. It is assumed that the frame from $Source_1$ (*Frame1*) arrives first, and is queued in the waiting queue. Now it waits for a frame from $Source_2$ (*Frame2*) with the "same" timestamp as that of $Source_1$ (determined by γ). If the expected frame arrives within the synchronisation window Δ, the algorithm checks the time gap between the frame pair, and forwards the frame pair if it is smaller than or equal to γ. Further, it caches the frame pair for future pairing in case the next expected frame does not arrive in time. If the synchronisation window expires, and no latest frame from $Source_2$ is received, the algorithm searches for a cached frame from $Source_2$ (*Frame2'*) which is within the frame pair approximation threshold. If such a cached frame exists, the algorithm checks whether the time gap between *Frame1* and the cached $Source_2$ frame (*Frame2'*) is within the γ threshold or not. If it satisfies the condition, then *Frame1* is cached, and the frame pair (*Frame1* and *Frame2'*) is forwarded. Otherwise, *Frame1* is dropped for pairing, but it is still cached. The proposed algorithm naturally supports more general scenarios with more than two sources, with linearly increasing complexity. Specifically, the algorithm only needs to identify the earliest arrived frame (denoted by Frame 1), and then iteratively apply the algorithm on the corresponding (late) frame from each of the additional sources. That is, each late frame can be denoted by Frame 2 in the algorithm above.

V. Experiment setup and performance results

A. Experimentation Set-up

In this section, we evaluate the synchronisation performance across two sources with different network contexts. In particular, we aim to study the sensitivity of the actual application performance metrics to the two key control parameters Δ and γ. The experiments with the synchronisation function are still based on the same setup as in Figure 4, apart from that the HoloSync function is embedded at the MEC server side. The set-up consists of the following elements:

- Two KINECT V2 sensor cameras, each connected to one PC which works as a client to process and send the frames to the MEC server using asynchronous Socket APIs.
- One MEC server for frame synchronisation on the receiver side. System configuration of clients and server is

Algorithm 1 Synchronisation Algorithm

Assumption : Current frame from $Source_1$ arrives first (*Frame1* with timestamp t_{S1})
Enqueued *Frame1* into the waiting queue
Frame2 ← Current frame from $Source_2$ (with timestamp t_{S2})
Frame2' ← Previous frame from $Source_2$ (with timestamp t'_{S2})
while ((*ElapsedTime* $\leq$ Δ) AND (*Frame2.Arrived*== false)) **do**
    Wait
end while
if ((*Frame2.Arrived* == true) AND ($\left|t_{S1} - t_{S2}\right| \leq \gamma$)) **then**
    Pair Frame1 with Frame2
    Cache Frame1 and Frame2
    Forward paired frames to user device
else if ((*Frame2'.Exists* == true) AND ($\left|t_{S1} - t'_{S2}\right| \leq \gamma$)) **then**
    Pair Frame1 with Frame2'
    Cache Frame1
    Forward paired frames to user device
else
    Drop Frame1 for pairing
end if

as follows: Operating System (OS): Microsoft Windows 10 Pro Processor: Intel(R) Core(TM) i7-7700 CPU @ 3.60GHz, 3600 Mhz, 4 Core(s), 8 Logical Processor(s). Random Access Memory (RAM): 16 GB

- Two PCs for adding en-route synthetic delay and packet losses are referred to as 'middleboxes'. At the middlebox, a Netem script is used to add synthetic delay and packet loss. Network routes are added to the routing table of the clients to take the specified path to the server through the middlebox. One middlebox is configured to add a delay of 10ms and a packet loss of 0.01%. The other middlebox is set to add a 50ms delay and 0.01% packet loss. This focused testing case, shown in Figure 5, is effectively the most challenging scenario, where the misalignment time gap between the two sources can be more than 5 seconds without any frame synchronisation mechanism in place.
- The transport-layer protocol is TCP, which is the default for LiveScan3D, and the congestion control algorithm applied is CUBIC when we evaluate the performance with different network distances in sections V-B1 and V-B2. In addition, we also investigate the scenario that different sources apply different TCP-based congestion control algorithms, and in this case, one source is based on CUBIC and the other is based on BBR in section V-B3.
- A local NTP server is used for time synchronisation between all the clients and the MEC server. It is important to have time synchronisation across all the entities in order to ensure synchronisation based on accurate frame production timestamps. To achieve that, NTP is used to synchronise the system time of all the platform components.

B. Performance Evaluation

Without loss of generality, our experiments include scenarios with the same source FPS and different FPS at the two sources. Specifically, the following scenarios are considered:

(1) Both the sources produce frames at 30 FPS, which can be considered as the ideal scenario with the full FPS capacity at the camera side.

(2) One source produces frames at 15 FPS, and the second source produces frames at 30 FPS : Since one of the sources produces frames at a lower rate i.e.,15 frames per second, it causes a higher time gap between consecutive frames, and fewer frames are available for pairing, thus affecting the setting of γ in the algorithm. In both scenarios, experiments were repeated for different approximation threshold (γ) values of 10ms, 50ms and 100ms respectively. The synchronisation window (Δ) is changing from 5ms to 50ms for each value of γ.

(3) Using heterogeneous congestion control mechanisms on the two sources. Specifically, one source is based on BBR for congestion control, and the other source is based on CUBIC. Both sources produce frames at the full capacity of 30FPS.

As mentioned previously, the proposed synchronisation algorithm is not designed for any specific codec, resolution, or compression mechanism, but it can be applied generically across different application configurations. In this paper, we evaluate the relevant performance based on a particular application scenario based on a fixed resolution level and without introducing compression.

- *Fresh/Half-fresh/Dropped frame ratios*: This is the statistical distribution of all the frames sent from the sources into the three categories as the outcome of each experiment instance.
- *Pairing error*: This is defined as the average time gap between successfully paired frames. Ideally, two frames having exactly the same frame production timestamp should be paired to completely avoid the motion misalignment. However, this is not possible in real life scenarios as the frames are produced at specific time intervals at different sources. Pairing error indicates the average error while pairing the fresh and half-fresh frames based on different approximations set by control parameter values. Figure 8 considers five frames each from $Source_1$ and $Source_2$. Let's assume that the sources produce frames at 30 FPS and consecutive frames from each source have 33ms of time gap between them. Ideally, the first frame from $Source_1$ should be paired with the first frame from $Source_2$ having the same timestamp. If it's not possible due to various reasons, the first frame from $Source_1$ is paired with the second frame from $Source_2$, and the time gap between the frame pair is 33ms. Similarly, if the first frame from $Source_1$ is paired with the third frame from

This article has been accepted for publication in IEEE Transactions on Multimedia. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TMM.2022.3207280

$Source_2$, the time gap will be 66ms i.e. 66ms pairing error.

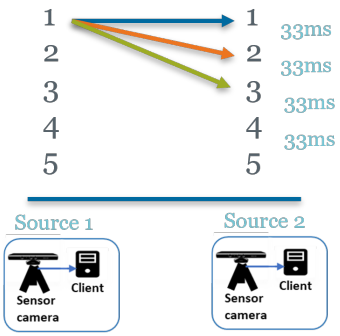

Fig. 8: Pairing error

- *Playback Latency*: This is defined as the actual time gap between the frame production timestamp at the source and the time point when it is displayed to the user.

1) Performance evaluation with 30 FPS from both sources: For all values of γ and varying Δ, the average playback latency and pairing error are shown in Figure 10 and 11 respectively.

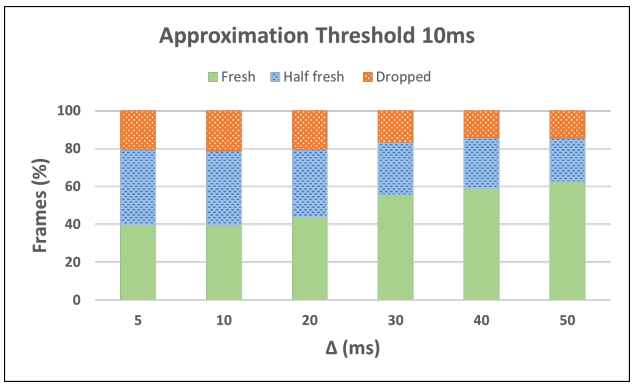

Fig. 9: Statistical distribution of frames (γ=10ms)

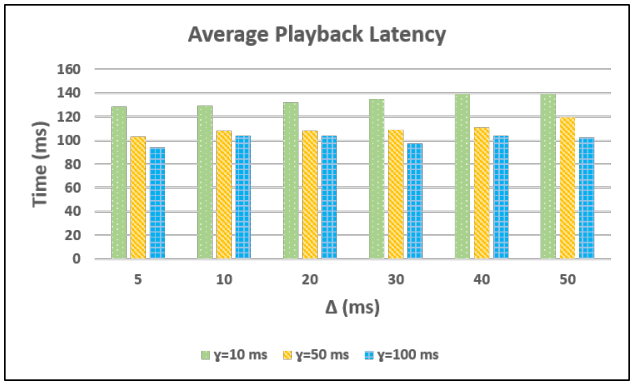

Fig. 10: Average playback latency for γ values

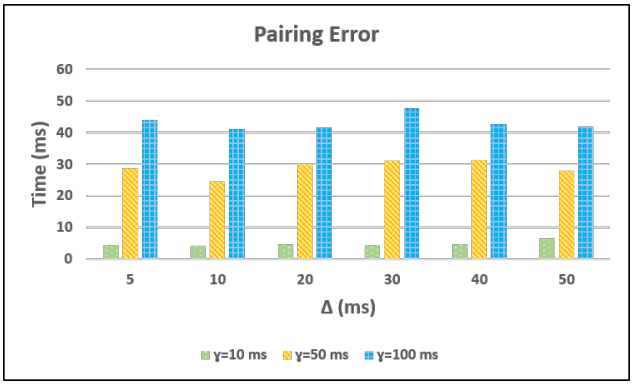

Fig. 11: Pairing error for γ values

*a) γ **is set at 10ms**:* This is the most stringent scenario of threshold setting for frame pairing approximation. With a fixed γ, we evaluate the application performance against different synchronisation window sizes Δ. It is observed in Figure 9 that the number of fresh frames increases significantly as the synchronisation window relaxes from 5ms to 50ms. In particular, for the case of 5ms and 10ms, the proportion of half fresh frames is almost equal to the number of fresh frames. It is because the frames have a low time margin to wait for frames from other sources, due to the stringent synchronisation window, and after the synchronisation window expires, the waiting frame searches and pairs with a qualified cached frame based on the approximation threshold. On the other hand, the proportion of fresh frames for the most relaxed Δ of 50ms is only around 60%. It is because of the reduced FPS at the receiver side, as one of the sources is 50ms away. Stringent γ coupled with the reduced FPS affects the performance of the application. As compared to the 10ms case, the number of fresh frames increase by 157% for the 50ms case. For a synchronisation window between 30ms to 50ms, the proportion of fresh frames increases as the synchronisation window increases. Figure 10 shows the average playback latency values for three approximation threshold values (γ) 10ms, 50ms and 100ms with varying synchronisation windows (Δ) from 5ms to 50ms. As shown in Figure 10, the average playback latency increases from 128ms to 138ms as the synchronisation window increases. Average pairing error is 4.44ms as shown in Figure 11. It means that the user will experience a motion misalignment of 4.44ms while using the application.

*b) γ **is set at 50ms**:* For this case, the approximation threshold is relaxed from 10ms to 50ms. As shown in Figure 12, there is a smaller number of half fresh frames as compared to that in Figure 9. The number of fresh frames increases significantly as compared to the previous case of 10ms approximation threshold. We conclude that the performance is less sensitive to Δ as compared to the previous case due to relaxed γ. The proportion of fresh frames increases slightly for higher values of Δ like 40ms and 50ms. From Figure 10 it can be observed that the average latency increases from

This article has been accepted for publication in IEEE Transactions on Multimedia. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TMM.2022.3207280

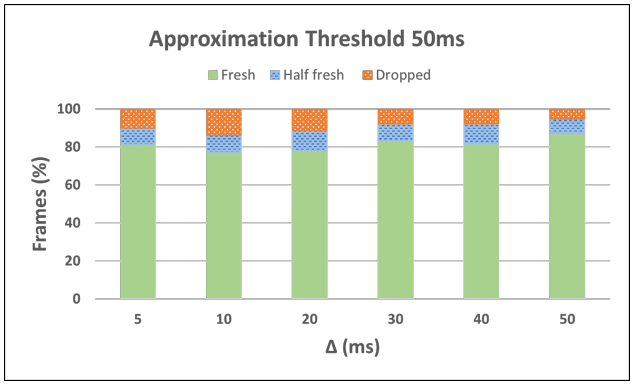

Fig. 12: Statistical distribution of frames (γ=50ms)

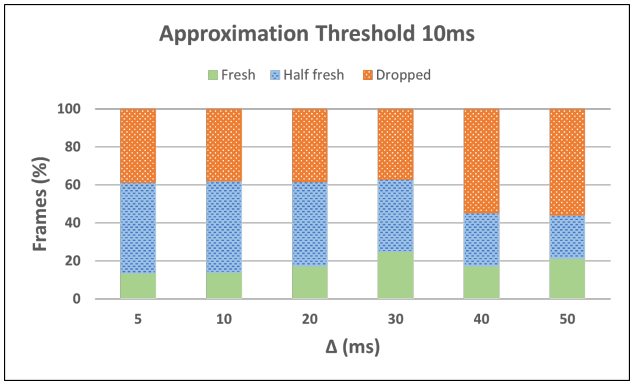

Fig. 14: Statistical distribution of frames (γ=10ms)

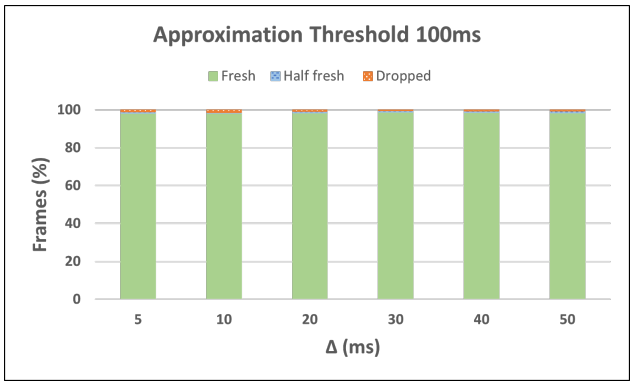

Fig. 13: Statistical distribution of frames (γ=100ms)

102ms to 119ms for increasing synchronisation windows of 5ms to 50ms. The average pairing error, i.e., average time gap between the frames is 28.89ms as shown in Figure 11. As compared to the previous case of γ = 10ms, pairing error increases from 4.44ms to 28.89ms for 50ms γ.

c) γ ***is set at 100ms****:* As shown in Figure 13, there are negligible amounts of half fresh frames, and almost all the frames are fresh frames because of a higher approximation threshold of 100ms i.e. higher number of feasible frames for pairing. However, one of the sources is 50ms away, causing reduced FPS at the receiver side. It indicates that if the approximation threshold is set high enough, the size of the synchronisation window does not matter. However, there is a trade-off as the approximation threshold increases, the pairing error also increases i.e., motion misalignment increases, degrading the quality-of-experience of the user. It is important to consider the pairing error before selecting the approximation threshold value for the application. It can be observed from Figure 10 that the average playback latency increases from 94.26ms to 102.35ms when the synchronisation window increases from 5ms to 50ms. Figure 11 shows that the average paring error is 43.16ms, which is much higher than that for 10ms and 50ms approximation thresholds. This indicates that an approximation threshold of 10ms provides a lower pairing error, that is, lower motion misalignment, and an approximation threshold of 100ms provides a higher pairing error, that is, higher motion misalignment. From Figure 10 we conclude that average playback latencies are higher (green bar) for lower values of approximation threshold. For instance, in Figure 10, for a 5ms synchronisation window, average latency for an approximation threshold of 10ms is 128.67ms, for 50ms it is 102.96ms, and for 100ms it decreases to 94.26ms.

It is because, whenever the approximation threshold is stringent like 10ms, there are very few frames that are qualified to be paired with the waiting frame, and as soon as the synchronisation window expires, the waiting frame needs to look for a cached frame, which has a higher time gap with the waiting frame as it is stored in the cached frames for a long time. Although the synchronisation algorithm ensures that the time gap between the paired frames lies within the approximation threshold, for cases where the waiting frame looks in the cached frames to find a suitable frame (i.e. within the approximation threshold), the time gap is close to the approximation threshold i.e. at the higher end. If there is a higher number of half fresh frames, it will collectively increase the playback latency.

2) Performance evaluation with different FPS from two sources: Now we evaluate the same application performance in a non-ideal environment where one of the sources has a reduced FPS performance of 15, while the other source remains constant at 30 FPS. For all values of γ and varying Δ, average playback latency and pairing error are shown in Figure 15 and 16 respectively.

a) γ ***is set at 10ms****:* As shown in Figure 14, the number of fresh frames increases as Δ increases from 5ms to 30ms. The number of half fresh and dropped frames is almost constant for Δ values from 5ms to 20ms, with the Δ value of 30ms having the highest number of fresh frames. For the cases of Δ value between 40ms and 50ms, there is a higher number of dropped frames and a lower number of half fresh frames, because the waiting frame has a smaller number of frames to match within the synchronisation window, and even if the expected frame arrives, the time gap is above the approximation threshold. Once the synchronisation window expires, the waiting frame searches for the cached frames, and does not find a suitable match. This is because a frame is cached once it is paired with a suitable frame, and dequeued

This article has been accepted for publication in IEEE Transactions on Multimedia. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TMM.2022.3207280

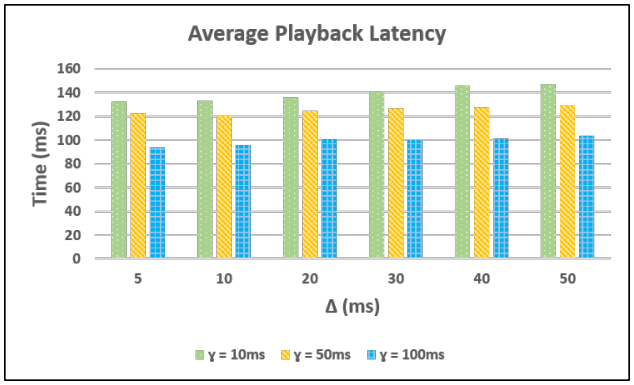

Fig. 15: Average playback latency for γ values

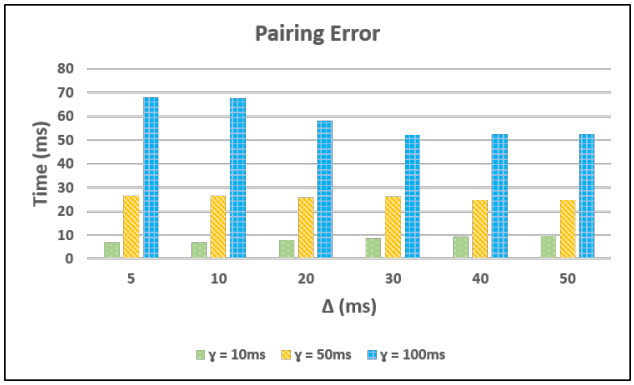

Fig. 16: Pairing error for γ values

for the next hop. As a result, for higher Δ values, frames are dequeued at a greater interval, implying that frames in the cached frame list are much older than the waiting frame and do not match. Consequently, the waiting frame is dropped instead of being paired with the cached frame. We conclude that for a stringent approximation threshold, higher values of Δ do not help. Instead, it has an adverse effect on the synchronised frame proportion. This is because the system spends more waiting time with higher Δ, and thus may waste time in identifying a matching cached frame. It is also noted from Figure 14 that the performance is less sensitive to Δ, since for all the values of Δ the maximum percentage of fresh frames lies between 15% to 25% for fresh frames.

In Figure 15, average playback latency increases from 132ms to 151ms when the synchronisation window is relaxed from 5ms to 50ms. It is observed from Figure 16 that user-perceived motion misalignment, i.e. pairing error, is minimised to 8.33ms. In case of FPS 30 scenario for γ of 10ms, the pairing error was 4.44ms, which is almost half of the pairing error of this error in Figure 16 i.e., 8.34ms. It shows that different FPS at the sources have an effect on the performance of the application.

b) γ ***is set at 50ms:*** It is observed from Figure 17 that as Δ increases, the proportion of fresh frames also increases as the waiting frames have a higher margin to wait for the expected frame. As compared to the previous case of 10ms γ,

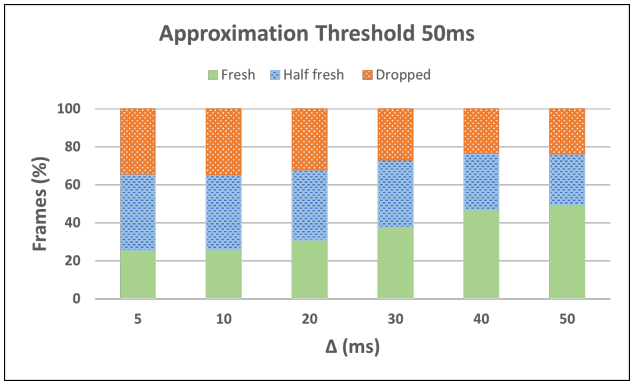

Fig. 17: Statistical distribution of frames (γ=50ms)

there is a higher number of fresh frames for all the values of Δ. So, it can be inferred that due to a higher value of γ equal to 50ms, there is a higher number of fresh frames as compared to 10ms γ as shown in Figure 14, and the performance is less sensitive to Δ. It is observed that the number of half fresh frames and dropped frames decreases with increasing value of Δ. However, for the cases 40ms and 50ms, the number of half fresh frames decreases because there are fewer suitable cached frames available. In Figure 15, the average playback latency increases from 122ms to 129ms as the synchronisation window relaxes. Figure 16 displays the average pairing error of 25.7ms, and it decreases slightly with increasing value of Δ. It is because, as the synchronisation window increases, the waiting frame has a higher margin to wait for the expected frame, and the approximation threshold is also higher. Because of these two reasons, the waiting frame is most likely to find the suitable frame pair, resulting in a lower pairing error. Effectively, the waiting frame does not have to look into the cached frames for a suitable frame, which could increase the pairing error.

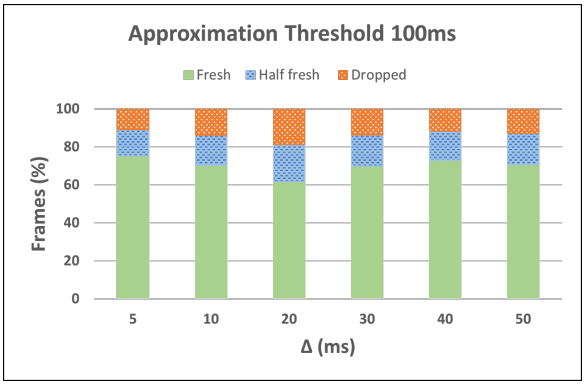

Fig. 18: Statistical distribution of frames (γ=100ms)

c) γ ***is set at 100ms:*** As shown in Figure 18, there is almost the same number of fresh frames irrespective of the value of Δ. As compared to the previous cases of γ value of 10ms and 50ms, the performance becomes even less sensitive to Δ in this case. Based on this observation, We infer that γ is important for the application performance.

This article has been accepted for publication in IEEE Transactions on Multimedia. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TMM.2022.3207280

However, increasing the value of γ would increase the pairing error which would deteriorate the experience of the user while using the application by adding higher motion misalignment. It can be noted from Figure 15, that average playback latency increases with increasing values of Δ values. For instance, in the case of 10ms γ and 5ms Δ, the playback latency is 132.2ms, and for the same value of γ and 50ms Δ, playback latency increases to 147.5ms. It is due to the fact that the frame waits for a longer period when Δ is set to 50ms than when Δ is 5ms. It can be observed from Figure 15 that for a lower value of γ, the range of playback latencies is higher than for higher values of γ e.g. 50ms and 100ms. It is because for a lower value of γ, the approximation threshold is very stringent and very few frames would qualify as a fresh frame, therefore the waiting frame needs to look into the cached frame which would be on the higher end of the approximation threshold, which increases the playback latency. Average pairing error is 58.5ms, hence paired frames will have an average time gap of 58.5ms. Pairing error, as shown in Figure 16, decreases as Δ increases because there is a greater number of suitable frames available for the waiting frame. It can be observed from Figure 16 that for the case of 10ms γ, pairing error gradually increases as the value of Δ increases because of the stringent γ, there is a fewer or zero number of frames to match, and as Δ increases the frame waits even longer before looking into the cached frames, resulting in an increased pairing error. In contrast, for the case of 50ms γ, pairing error decreases as Δ increases and because of relaxed γ a higher number of frames will be available to match, and the waiting frame does not have to look into the cached frames. The same trend is observed for 100ms γ.

3) Performance evaluation with heterogeneous congestion control algorithms: Now, we evaluate the performance of the synchronisation algorithm with Δ = 5ms, which is the most stringent setting for Δ, and relax it to 50ms. The approximation threshold (γ) is set to 10ms, 50ms, and 100ms for each Δ value. $Source_1$ and the middlebox connected to it, use the BBR congestion control algorithm as shown in Figure 4. $Source_2$, and the middlebox along the path, use CUBIC congestion control algorithm. Synthetic delay added to the routes on both sides is 40ms with 0.01% packet loss. Based on this experiment setup, we focus on the synchronisation performances based on different TCP congestion algorithms at sources while they share the same or similar network conditions (network delay and packet loss).

*a) γ **is set at 10ms**:* This is the most stringent scenario for parameter setting. In Figure 21, almost all the frames are dropped by the MEC server. For 50ms Δ, there are negligible amounts of fresh (0.15%) and half fresh (0.09%) frames, and there are no fresh frames for 5ms Δ. It is because none of the frames could find a suitable match within the synchronisation window or in the cached frames because of the low FPS performance at the MEC server. The low FPS is due to the network distances of the sources from the MEC server. Low FPS means that consecutive frames have a higher time gap between them, making it difficult for the waiting frame to find a suitable frame below the approximation threshold. It shows that a stringent γ setting may damage the performance of the application. Average playback latency for 50ms Δ, is 144ms, and pairing error is 10ms. For 5ms Δ, there are no fresh or half fresh frames. So, playback latency and pairing error can not be calculated. Thus, we could not plot it in the figure for comparison.

*b) γ **is set at 50ms**:* It is observed from Figure 22 that the number of fresh frames significantly increases to 95% for this case as compared to the previous case of 10ms γ. This is irrespective of the value of Δ i.e. whether it is 5ms or

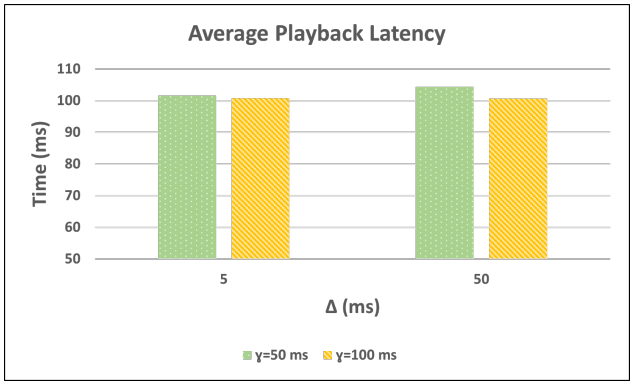

Fig. 19: Average playback latency for γ values

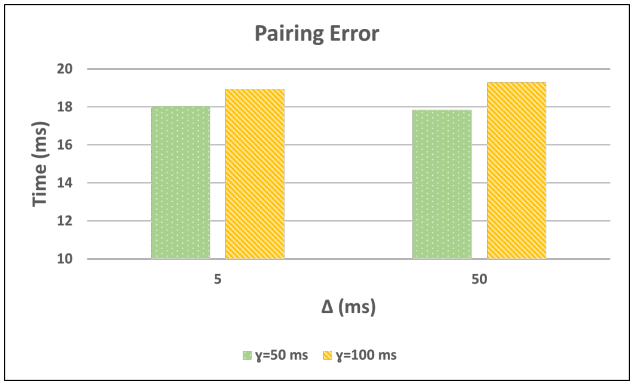

Fig. 20: Pairing error for γ values

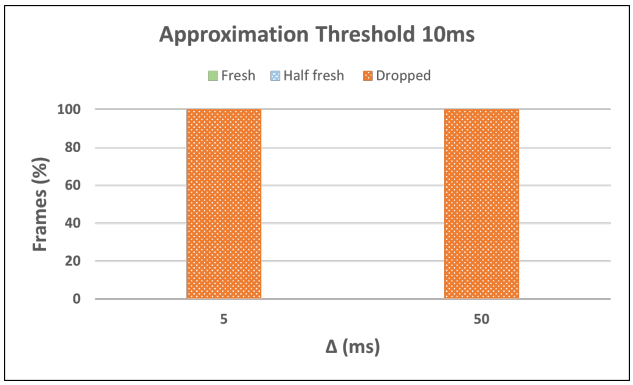

Fig. 21: Statistical distribution of frames (γ=10ms)

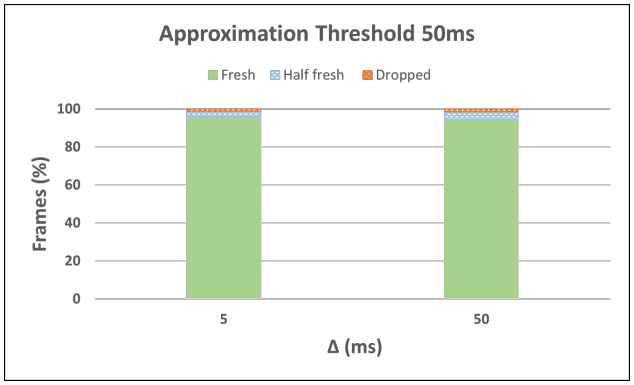

Fig. 22: Statistical distribution of frames (γ=50ms)

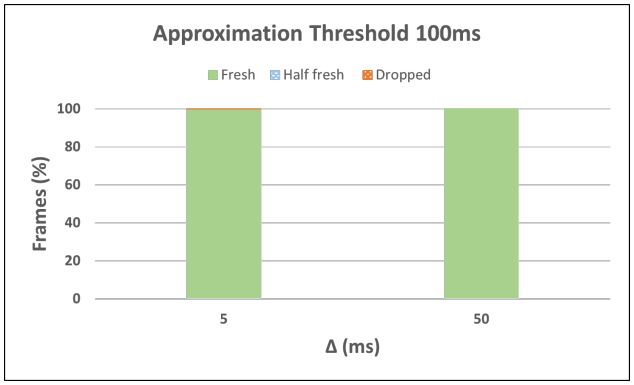

Fig. 23: Statistical distribution of frames (γ=100ms)

50ms, the number of fresh frames is almost the same. The percentage of half fresh and dropped frames are approximately 4% and 1% respectively. Figure 19 shows that the average playback latency for 5ms and 50ms Δ is around 102ms. Figure 20 shows that the pairing error is 18ms for both values of 5ms and 50ms Δ. That means the user will perceive a motion misalignment of 18ms. We conclude that the performance of the application is less sensitive to the synchronisation window setting as compared to the approximation threshold.

*c) γ **is set at 100ms**:* In Figure 23, it is shown that the number of fresh frames is almost 100%. There are negligible amounts of half fresh and dropped frames for 5ms Δ and there are no dropped or half fresh frames for 50ms Δ. Average playback latency and pairing error for both cases of Δ are 100ms and 19ms respectively as shown in Figure 19 and Figure 20. In this case, it is observed that the performance of the application is more sensitive to the approximation threshold as compared to the synchronisation window.

C. Guidance on setting control parameters

According to the evaluations, there are two control parameters for synchronisation of frames, namely synchronisation window (Δ) and approximation threshold (γ). As far as γ is concerned, its setting should take into account the frame rate on the camera side. Specifically, if any camera side FPS is low, then the time gap between consecutive frames becomes larger. If a low (γ) value is set, it will lead to the failure of frame pairing and γ needs to be relaxed to get higher user-perceived FPS causing higher pairing error which degrades the quality of user experience. In the case of setting γ, there is a trade-off between user-perceived FPS and pairing error. Similarly, we also observe that long network distances between the source and the MEC server may also cause reduced FPS performance. Again, in this case, γ should be relaxed to match the time gap between consecutive frames on the source side. As a result, the user will perceive higher motion misalignment.

According to the nature of the holographic applications, they can be classified into interactive and non-interactive ones. As far as the setting of Δ is concerned, it will be dependent on the playback latency requirements from the application. There is a trade-off between average playback latency and user-perceived FPS. Multi-party interactive applications require low average playback latency. To achieve this, the setting of Δ should be sufficiently low so that early arrived frames do not have to wait for a long time, and frames will be processed faster at the MEC server. The application user will perceive lower playback latency, i.e. faster responses from other application users. In the case of non-interactive applications, Δ could be relaxed and the early arrived frames could afford to wait longer for suitable frames to be paired, but it will increase the average playback latency. The average playback latency is largely dependent on Δ but is also subject to FPS performance at the receiver side. This is because, if FPS is lower, consecutive frames have a higher time gap, and the early arrived frame needs to wait longer to get the suitable frame within the allowed Δ setting.

Finally, the coordination between γ and Δ is important to achieve the desired synchronisation performances. If Δ is set to a low value and γ is set to a very higher value, then the average playback latency is lower, but this is at the expense of easily perceived motion misalignment which will degrade the quality of user experience. Conversely, if Δ is high while γ is too low, then the average playback latency will be higher, but the downside is potentially a high chance of having frames dropped.

VI. Conclusion

In this paper, we proposed HoloSync, a novel frame synchronisation scheme for supporting live holographic teleportation operations, involving multiple remote sources at different network locations. The motivation is based on our observation of significant motion misalignment effects when remote sources are teleported from different Internet locations. For the very first time in the literature, our proposed scheme is able to substantially reduce such misalignment in a controllable manner in order to improve user experiences in such emerging applications.

We showed that after applying the synchronisation algorithm, motion misalignment decreases significantly. However, it is dependent on various parameters like the synchronisation window and the approximation threshold. To further investigate the sensitivity of the performance of the application

to these parameters, in terms of average playback latency, pairing error, and frame statistics, we tested the holographic teleportation platform with full and reduced FPS capacity at the sources. The experimentation results show that there are fewer synchronised frames for stringent approximation threshold values, which can be increased by increasing synchronisation window. The synchronisation window has a smaller effect on frame statistics for higher approximation threshold values. We guided on setting control parameters based on the type of applications and user requirements. Last but not least, we specifically analysed the trade-off between different user-perceived performances with regard to the setting of the synchronisation window and approximation threshold. In our future work, we will carry out additional experiments to evaluate the trade-offs between the synchronisation and adaptation to the resolution and colour depth as part of the subjective QoE analysis.

REFERENCES

[1] S. Mann, T. Furness, Y. Yuan, J. Iorio, and Z. Wang, "All reality: Virtual, augmented, mixed (x), mediated (x,y), and multimediated reality," 04 2018.

[2] A. Maimone, A. Georgiou, and J. Kollin, "Holographic near-eye displays for virtual and augmented reality," *ACM Transactions on Graphics*, vol. 36, pp. 1–16, 07 2017.

[3] S. Orts-Escolano *et al.*, "Holoportation: Virtual 3d teleportation in real-time," in *Proceedings of the 29th Annual Symposium on User Interface Software and Technology*, ser. UIST '16. New York, NY, USA: Association for Computing Machinery, 2016, p. 741–754. [Online]. Available: https://doi.org/10.1145/2984511.2984517

[4] B. Han, Y. Liu, and F. Qian, "Vivo: visibility-aware mobile volumetric video streaming," 09 2020, pp. 1–13.

[5] J.-B. Jeong, S. Lee, I.-W. Ryu, T. T. Le, and E.-S. Ryu, *Towards Viewport-Dependent 6DoF 360 Video Tiled Streaming for Virtual Reality Systems*. New York, NY, USA: Association for Computing Machinery, 2020, p. 3687–3695. [Online]. Available: https://doi.org/10.1145/3394171.3413712

[6] F. Qian, B. Han, J. Pair, and V. Gopalakrishnan, "Toward practical volumetric video streaming on commodity smartphones," in *Proceedings of the 20th International Workshop on Mobile Computing Systems and Applications*, ser. HotMobile '19. New York, NY, USA: Association for Computing Machinery, 2019, p. 135–140. [Online]. Available: https://doi.org/10.1145/3301293.3302358

[7] J.-B. Jeong, S. Lee, I.-W. Ryu, T. T. Le, and E.-S. Ryu, *Towards Viewport-Dependent 6DoF 360 Video Tiled Streaming for Virtual Reality Systems*. New York, NY, USA: Association for Computing Machinery, 2020, p. 3687–3695. [Online]. Available: https://doi.org/10.1145/3394171.3413712

[8] R. Li, "Towards a new internet for the year 2030 and beyond," Third Annual ITU IMT-2020/5G Workshop and Demo Day Geneva, Switzerland July 18, 2018. [Online]. Available: https://www.itu.int/en/ITU-T/Workshops-and-Seminars/201807/Documents/3_Richard%20Li.pdf

[9] R. Li and Y. Miyake, "New services and capabilities for network 2030: Description technical gap and performance target analysis," Doc. NET2030-O-027 in FOCUS GROUP ON TECHNOLOGIES FOR NETWORK 2030 2019. [Online]. Available: http://handle.itu.int/11.1002/pub/81444c78-en

[10] E. Ramadan, A. Narayanan, U. Dayalan, R. Fezeu, F. Qian, and Z.-L. Zhang, "Case for 5g-aware video streaming applications," 08 2021, pp. 27–34.

[11] S.-H. Jun and J.-H. Kim, "5g will popularize virtual and augmented reality: Kt's trials for world's first 5g olympics in pyeongchang," in *Proceedings of the International Conference on Electronic Commerce*, ser. ICEC '17. New York, NY, USA: Association for Computing Machinery, 2017. [Online]. Available: https://doi.org/10.1145/3154943.3154947

[12] ITU-T Technical Report , "Representative use cases and key network requirements for network 2030," January 2020. [Online]. Available: https://www.itu.int/dms_pub/itu-t/opb/fg/T-FG-NET2030-2020-SUB.G1-PDF-E.pdf

[13] FG-NET-2030, "Network 2030 a blueprint of technology, applications and market drivers towards the year 2030 and beyond." [Online]. Available: https://www.itu.int/en/ITU-T/focusgroups/net2030/Documents/White_Paper.pdf

[14] "Google VR – fundamental concepts." [Online]. Available: https://developers.google.com/vr/discover/fundamentals

[15] J. Park, P. A. Chou, and J.-N. Hwang, "Volumetric media streaming for augmented reality," in *2018 IEEE Global Communications Conference (GLOBECOM)*, 2018, pp. 1–6.

[16] J. Park, P. Chou, and J.-N. Hwang, "Rate-utility optimized streaming of volumetric media for augmented reality," *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, vol. PP, pp. 1–1, 02 2019.

[17] S. Anmulwar, N. Wang, A. Pack, V. S. Ha Huynh, J. Yang, and R. Tafazolli, "Frame synchronisation for multi-source holograhphic teleportation applications - an edge computing based approach," in *2021 IEEE 32nd Annual International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC)*, 2021, pp. 1–6.

[18] M. Kowalski, J. Naruniec, and M. Daniluk, "Livescan3d: A fast and inexpensive 3d data acquisition system for multiple kinect v2 sensors," in *2015 International Conference on 3D Vision*, 2015, pp. 318–325.

[19] Q.-V. Pham, F. Fang, V. N. Ha, M. J. Piran, M. Le, L. B. Le, W.-J. Hwang, and Z. Ding, "A survey of multi-access edge computing in 5g and beyond: Fundamentals, technology integration, and state-of-the-art," *IEEE Access*, vol. 8, pp. 116 974–117 017, 2020.

[20] Microsoft Hololens. [Online]. Available: http://handle.itu.int/11.1002/pub/81444c78-en

[21] A. Clemm, M. T. Vega, H. K. Ravuri, T. Wauters, and F. D. Turck, "Toward truly immersive holographic-type communication: Challenges and solutions," *IEEE Communications Magazine*, vol. 58, no. 1, pp. 93–99, 2020.

[22] I. Selinis, N. Wang, B. Da, D. Yu, and R. Tafazolli, "On the internet-scale streaming of holographic-type content with assured user quality of experiences," in *2020 IFIP Networking Conference (Networking)*, 2020, pp. 136–144.

[23] J. van der Hooft, T. Wauters, F. De Turck, C. Timmerer, and H. Hellwagner, "Towards 6dof http adaptive streaming through point cloud compression," in *Proceedings of the 27th ACM International Conference on Multimedia*, ser. MM '19. New York, NY, USA: Association for Computing Machinery, 2019, p. 2405–2413. [Online]. Available: https://doi.org/10.1145/3343031.3350917

[24] J. van der Hooft, M. T. Vega, C. Timmerer, A. C. Begen, F. De Turck, and R. Schatz, "Objective and subjective qoe evaluation for adaptive point cloud streaming," in *2020 Twelfth International Conference on Quality of Multimedia Experience (QoMEX)*, 2020, pp. 1–6.

[25] N. K. Karn, H. Zhang, and F. Jiang, "User-perceived quality aware adaptive streaming of 3d multi-view video plus depth over the internet," *Multimedia tools and applications*, vol. 77, no. 17, pp. 22 965–22 983, 2018.

[26] C. Perkins and T. Schierl, "Rapid Synchronisation of RTP Flows," RFC 6051, Nov. 2010. [Online]. Available: https://www.rfc-editor.org/info/rfc6051

[27] C. Perkins, M. Westerlund, and J. Ott, "Media Transport and Use of RTP in WebRTC," RFC 8834, Jan. 2021. [Online]. Available: https://www.rfc-editor.org/info/rfc8834

[28] webRTC, 2022. [Online]. Available: https://www.w3.org/TR/webrtc/

[29] MPEG-DASH, 2022. [Online]. Available: https://mpeg.chiariglione.org/standards/mpeg-dash

[30] R. Mijumbi, J. Serrat, J.-L. Gorricho, N. Bouten, F. De Turck, and R. Boutaba, "Network function virtualization: Stateoftheart and research challenges," *IEEE Communications Surveys Tutorials*, vol. 18, no. 1, pp. 236–262, 2016.

[31] B. Han, V. Gopalakrishnan, L. Ji, and S. Lee, "Network function virtualization: Challenges and opportunities for innovations," *IEEE Communications Magazine*, vol. 53, no. 2, pp. 90–97, 2015.

[32] ETSI White Paper, "MEC in 5g networks," 2018. [Online]. Available: https://www.etsi.org/images/files/ETSIWhitePapers/etsi_wp28_mec_in_5G_FINAL.pdf

[33] F. W. Vook, A. Ghosh, E. Diarte, and M. Murphy, "5g new radio: Overview and performance," in *2018 52nd Asilomar Conference on Signals, Systems, and Computers*, 2018, pp. 1247–1251.

[34] "5g low latency requirements," 2021. [Online]. Available: https://developers.google.com/vr/discover/fundamentals
[35] Qualcomm, "Everything you need to know about 5g," 2019. [Online]. Available: https://www.qualcomm.com/5g/what-is-5g
[36] 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, 2017.
[37] "Sprint Network," https://www.sprint.net/tools/sla-performance/sl.

Sweta Anmulwar received her M.Tech. degree in wired and wireless communications from Savitribai Phule Pune University in 2014. She is a PhD researcher at the University of Surrey working on frame synchronisation for holographic applications. Sweta worked in computer networking, software development, and wireless communication at C-DAC, India from 2014 to 2019. She has published in multiple venues including IEEE PIMRC, IEEE WPMC, IEEE Transaction on Broadcasting, Elsevier, and Springer and conducted various tutorials at IEEE conferences. Her main research interests include extended reality, SDN, IoT and future networks.

Ning Wang (SM'17) obtained his PhD degree in Electronic Engineering from the Centre for Communication Systems Research at University of Surrey where he has been working as a professor. Professor Wang is currently leading a research team focusing on 5G and beyond networking and applications. He has published over 150 research papers at prestigious international conferences and journals. His main research areas include future networks, multimedia networking, network management and control and QoS/QoE assurances.

Vu San Ha Huynh received the B.Sc. (Hons) degree in computer science from the University of Nottingham in 2016 and the PhD degree from the University of Nottingham in 2021. He has authored in venues including IEEE Transactions on Broadcasting, IEEE Access, ACM MobiCom, CLOSER, PECCS, IEEE WiMob, IEEE PIMRC, IEEE FMEC and IEEE IWCMC. His research interests include intelligence in the network transport layer, mobile edge cloud and fog computing, distributed data caching services, mobile heterogeneous opportunistic networks, large-scale internet service architectures, Internet of Things and network science.

Stewart Bryant is a Visiting Professor at the University of Surrey. He is co-chair of the IETF PALS working group and a former IETF Routing Area Director. Stewart is a specialist in Internet forwarding and routing technologies. He is an author of 44 IETF RFCs and an inventor of over 80 patents in this area. Stewart's current research interest is in making packet switching networks provide deterministic behaviour. In particular, he is working on MPLS methods for adding ancillary data into the data plane.

Regius Rahim Tafazolli Fellow of the Royal Academy of Engineering, IET, WWRF and Regius Professor of Electronic Engineering; Professor of Mobile and Satellite Communications; Founder and Director of 5GIC, 6GIC and ICS (Institute for Communication Systems) at the University of Surrey. He has over 30 years of experience in digital communications research and teaching. He has authored and co-authored more than 1000 research publications and is regularly invited to deliver keynote talks and distinguished lectures to international conferences and workshops. He was advisor to the Mayor of London (Boris Johnson) on London 2050 Infrastructure.

Jinze Yang , received his B.Eng. degree in Internet of Things Engineering and the Ph.D. degree in Electronic Engineering from the Queen Mary University of London (QMUL) in 2015 and 2020, respectively. He joined the Huawei Technologies Ltd., China in 2020. He is currently a senior engineer in Huawei Ltd. His research interests include data centre networks, Internet of things networks, information-centric networks, and ad-hoc networks.