Reliable Provisioning of Low-Latency and High-Bandwidth Extended Reality Live Streams

Giap Le, Vinh Truong Hoang, Sifat Ferdousi, Andrea Marotta, Sugang Xu, Yusuke Hirota, Yoshinari Awaji, Massimo Tornatore, and Biswanath Mukherjee

***Abstract*—The networking industry is offering new services leveraging recent technological advances in connectivity, storage, and computing such as mobile communications and edge computing. In this regard, extended reality, a term encompassing virtual reality, augmented reality, and mixed reality, can provide unprecedented user experience and pioneering service opportunities such as: live concerts, sports, and other events; interactive gaming and entertainment; immersive education, training, and demos. These services require high-bandwidth, low-latency, and reliable connections, and are supported by next-generation ultra-reliable and low-latency communications in the vision of 6G mobile communication systems. In this work, we devise a novel scheme, called backup from different data centers with multicast and adaptive bandwidth provisioning, to admit reliable, low-latency, and high-bandwidth extended reality live streams in next-generation networks. We consider network services where contents are non-cacheable and investigate how backup services can be offered by different data centers with multicast and adaptive bandwidth provisioning. Our proposed service-provisioning scheme provides protection not only against link failures in the physical network but also against computing and storage failures in data centers. We develop scalable algorithms for the service-provisioning scheme and evaluate their performance on various complex network instances in a dynamic environment. Numerical results show that, compared to conventional service-provisioning schemes such as those seeking backup services from the same data center, our proposed service-provisioning scheme efficiently utilizes network resources, ensures higher reliability, and guarantees low latency; hence, it is highly suitable for extended reality live streams.**

***Index Terms*—Optical networks, adaptive bandwidth provisioning, dynamic multicast, edge computing, live stream.**

I. Introduction

With the adoption of recent technological advances in connectivity, storage, and computing such as 5G and beyond mobile communications and edge computing, the networking industry has started offering new services with unprecedented user experience. In this regard, virtual reality (VR), augmented reality (AR), and mixed reality (MR) have emerged as the first wave of killer applications [1]. VR, AR, and MR allow users to interact intuitively with the environment in six degrees of freedom (6DoF) and are typically referred to as *extended reality* (XR), a term encompassing all the three technologies. XR offers a set of innovative applications such as: live concerts, sports, and other events; interactive gaming; immersive education, training, and demos, just to name a few. These services require high-bandwidth, low-latency, reliable connections, and are supported by the so-called next-generation ultra-reliable and low-latency communications (xURLLC) in the vision of 6G mobile communication systems [2]. To meet these stringent requirements (e.g., latency within a few milliseconds, ms), a service-provisioning scheme must embrace the opportunities created by the recent technological advances.

To motivate our proposed service-provisioning scheme, in Fig. 1, we consider a dynamic scenario where, at time t_1, node 1 is requesting an XR live stream whose content can be retrieved from node 22. The requesting node 1 can be a multi-access edge computing (MEC) data center (DC), a base station, a central office, or an aggregation point in the access network from which data are ultimately delivered to end users via a wireless (e.g. 5G/6G/Wi-Fi) or a fixed network. Node 22 is a remote and powerful DC close to the live event, which processes multiple video streams from different shooting angles and produces the XR live stream. The XR live stream must be transmitted in quasi real-time on x-Gbps streams over a content delivery network (CDN) to edge DCs close to the requesting node to allow end users to enjoy immersive experiences. Note that, since immersive experiences are interactive (e.g., when an end user moves his/her head-mounted display, or HMD, the brain expects an instantaneous visual and aural update), low latency between the requesting node and edge DCs is critical. Hence, ideally, the requesting node should be served by the closest edge DC. Also, since this work considers XR live streams, contents are non-cacheable (i.e., network resources reserved for an XR live stream in an edge DC and along its multicast paths are released if there are no active users) [3]. In Fig. 1, we assume that nodes 1-18 belong to a metro network and are served by nearby edge DCs 5, 8, and 10 (which are referred to as *a local DC cluster*, aimed at reducing latency and partially or completely offloading storage and computing capability from HMDs for portability); nodes 19-22 belong to a backbone network.

Since XR live streams require reliable, high-bandwidth, and low-latency connectivity, an optical network is a suitable solution to transport data from a remote DC to edge DCs and from edge DCs to the requesting node. This optical network is exposed to many threats such as malicious attacks and natural and human-made large-scale disasters (e.g., earthquakes and hurricanes) [4]. To ensure reliability, restoration and protection schemes are traditionally used. In a restoration scheme, no resources are reserved in advance and the network must react to find an alternate path for a connection after a failure occurs on the working path. However, a restoration scheme requires long recovery time and provides no guarantee to restore a disrupted path; thus, it is not suitable for XR live streams for which latency and reliability are critical. In a protection scheme, extra network resources are reserved when a connection is provisioned. Conventionally, a pair of paths

Giap Le is with UC Davis and also with Ho Chi Minh City Open University; Vinh Truong Hoang is with Ho Chi Minh City Open University; Andrea Marotta is with the University of L'Aquila; Sugang Xu, Yusuke Hirota, and Yoshinari Awaji are with NICT; Massimo Tornatore is with Politecnico di Milano; Sifat Ferdousi and Biswanath Mukherjee are with UC Davis; Corresponding author: Giap Le, dgle@ucdavis.edu

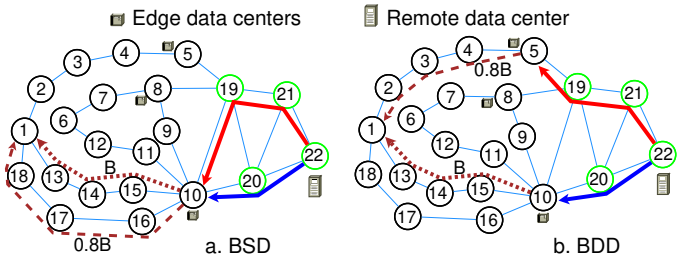

Fig. 1. Baseline service-provisioning schemes.

is provided to a connection: one is used to carry traffic during normal operation, referred to as primary path; and the alternate path, referred to as backup path, is reserved and will be activated after a failure occurs on the primary path. This protection scheme provides faster recovery and higher reliability; therefore, it is suitable for XR live streams.

In Fig. 1.a, we first present a baseline service-provisioning scheme, named *backup from the same DC* (BSD) [5], where the XR live stream from the remote DC 22 is transmitted to the closest edge DC 10 using a primary path (blue line) and a backup path (red line). The selected edge DC 10 requires powerful computing capability and a large storage memory to host a local video server that receives the XR live stream from the remote DC and makes it locally available. Ultimately, the XR stream from the selected edge DC 10 is delivered to the requesting node using another pair of primary path (dotted line) and backup path (dashed line). This service-provisioning scheme guarantees that the requesting node has lowest latency (since it is served by the closest edge DC), recovery time is shorter (e.g., compared to a restoration scheme), and the connection is survivable against any single-link failure in the optical network (which is still the dominant failure scenario in an optical network [4], [6]). However, it provides no protection against failures of computing and storage resources in the selected edge DC (e.g., node 10) which is an important issue to ensure end-to-end network resilience [7].

In Fig. 1.b, we show another baseline service-provisioning scheme, named *backup from different DCs* (BDD) [8], where the XR live stream from the remote DC 22 is enabled in two different edge DCs 5 and 10. In addition to protection against a single-link failure in the optical network, this service-provisioning scheme also provides protection against failures of computing and storage resources in edge DCs. During normal operation, the requesting node is served by the closest edge DC 10 using the primary path (dotted line). If a failure occurring on the primary path or in the edge DC 10, the provisioned service is still guaranteed after the backup path (dashed line) is activated. Due to the scarcity of network resources in case a failure occurs, in this work, we consider degraded service where the bandwidth reserved on the primary path is partially protected (e.g., an XR live stream can be switched to a lower resolution depending on available bandwidth). In Fig. 1, while the bandwidth reserved on the primary path is B, the bandwidth reserved on the backup path is only $0.8B$. Some works, e.g., [9], have showed that this scheme also offers flexibility to allocate resources to different edge DCs rather than reserve a large amount of resources in a single edge DC.

In Fig. 1, high-bandwidth connections from the remote DC are required to make the XR live stream available in edge DCs. A service-provisioning scheme should exploit the recent technological advances in optical networks to improve resource utilization. In today's software-defined optical networks, wavelength selective switch-based (WSS) reconfigurable optical add-drop multiplexers (ROADMs) and optical cross-connects (OXCs) are widely adopted, e.g., in metro optical networks [10]. These components provide the network operator with a high level of flexibility to dynamically and remotely reconfigure optical channels. Depending on their architecture, some ROADMs or OXCs can cost-effectively support the multicast without significant modifications in their architecture [11]. In our scenario, for each XR live stream, the network must maintain a dynamic survivable multicast tree whose root is the remote DC close to the live event and consuming leaves are edge DCs with active users. When a request arrives at an edge DC without the desired XR stream, in some scenarios, a service-provisioning scheme should utilize multicast to enable the desired XR live stream in the target edge DC from a much closer node (which is not necessarily the root node).

Finally, a service-provisioning scheme should be aware of the unique trade-off between latency and bandwidth for XR live streams. In general, lower latency requires less bandwidth to meet certain user experience. This trade-off is related to the fact that, with lower latency, fewer number of viewpoints around the live event are required to be sent over the network. For instance, if the end-to-end latency between an end user and an edge DC is on the order of 1 ms, the required bandwidth for the XR live stream can be reduced to 100 Mbps. However, if this end-to-end latency is 20 ms, the required bandwidth for the XR live stream must be up to 400 Mbps to meet the same user experience [12]. In a specific scenario where latency is smaller than the requested value, it is essential for a service-provisioning scheme to adaptively reduce the reserved bandwidth while still ensuring user experience. We will illustrate how multicast and adaptive bandwidth provisioning can conserve network resources in Section III.

Our contributions in this work are as follows. We propose a service-provisioning scheme, named *BDD with multicast and adaptive bandwidth provisioning*, to fulfill XR live stream requests while ensuring strict requirements on reliability, latency, and bandwidth. Our service-provisioning scheme leverages multicast and adaptive bandwidth provisioning to improve network resource utilization. We develop scalable algorithms for the service-provisioning scheme and evaluate their performance on various complex network instances in a dynamic environment. Numerical results show that, compared to the baseline service-provisioning schemes such as BSD and BDD, our proposed service-provisioning scheme efficiently utilizes network resources, provides services to more users, guarantees higher reliability, and ensures low latency; hence, it is highly suitable for XR live streams.

The rest of this work is organized as follows. In Section II, we review related works. In Section III, we present our proposed scheme and develop scalable algorithms for it. In Section IV, we evaluate our proposed service-provisioning scheme. We conclude this work in Section V.

II. Related Works

To deliver XR live streams to end users, a CDN should include three major components: creation, transport, and edge delivery [13]. Most recent works on XR, e.g., in [14], are focused on the creation step where the authors proposed various solutions to reduce required bandwidth for 360 videos such as viewport-dependent and tile-based streaming.

In this work, we focus on efficient solutions to transport data from a remote DC to edge DCs, and from edge DCs to, e.g., a MEC DC. Our solutions are applicable to various parts of the network including backbone, metro, backhaul, and fronthaul. Even though there are no similar works, as we mentioned in Section I, the underlying technologies for the proposed service-provisioning scheme are based on the multicast in the optical layer and adaptive bandwidth provisioning. Therefore, below, we review some works on these two technologies to position and motivate our work.

Regarding optical multicast, in [15], the authors proposed different approaches to provision a survivable multicast session using segment and path-pair protection. They validated the proposed approaches in a dynamic network environment where multiple multicast sessions are provisioned in wavelength-division multiplexing (WDM) mesh networks. In [16], the authors investigated shared protection for dynamic multicast sessions in survivable WDM mesh networks. They proposed a service-provisioning scheme, named multicast protection through spanning paths, where a primary multicast tree is protected by backup network resources which are shared between different multicast sessions. In [17], the authors proposed a solution where a multicast session is provisioned as the primary multicast tree with each of its segment being protected by another backup multicast tree. In [18], the authors studied multicast routing and distance-adaptive spectrum allocation in elastic optical (EO) networks with shared protection. They proposed a scheme to provision a set of static multicast trees which are survivable against a single-link failure using minimal required spectrum resources.

Regarding adaptive bandwidth provisioning, to the best of our knowledge, there are no works addressing the unique trade-off between latency and bandwidth for XR live streams to reduce required network resources (e.g. network bandwidth) while ensuring user experience. Most works to reduce bandwidth required for XR live streams are in the video creation step. For example, in [19], the authors considered a viewport-adaptive 360-degree video streaming system where a server prepares different video representations for a user to select based on the network condition. In [20], the authors proposed an adaptive method for 360-video streaming using scalable video coding to alleviate the restriction of buffer duration. In [21], the authors investigated various tile-based adaptive mechanisms to reduce required bandwidth and improve representation quality for equirectangular VR videos.

Most of the relevant works we reviewed above, e.g., [15]–[18], are focused on static survivable multicast trees. The authors proposed various solutions to protect multicast trees where each multicast tree has a static root and a fixed number of consuming leaves (i.e., destinations). Therefore, their solutions are more applicable to on-demand video streams where contents are periodically cached (e.g., in network low-traffic hours) in a set of edge DCs. In this work, we focus on non-cacheable contents where, for each XR live stream, the network must maintain a dynamic multicast tree whose root is the remote DC close to the live event and consuming leaves are edge DCs with active XR live streams. The dynamics of a multicast tree implies that the set of its consuming leaves and branches can significantly vary over time, depending on the availability of the requested XR stream in edge DCs. Moreover, each dynamic multicast tree must be survivable such that the desired XR live stream remains available for users against a single-link failure in the network or a failure in an edge DC. Below, we formally describe the problem and propose solutions for it.

III. Reliable Provisioning of Low-Latency and High-Bandwidth XR Live Streams

A. Multicast and Adaptive Bandwidth Provisioning

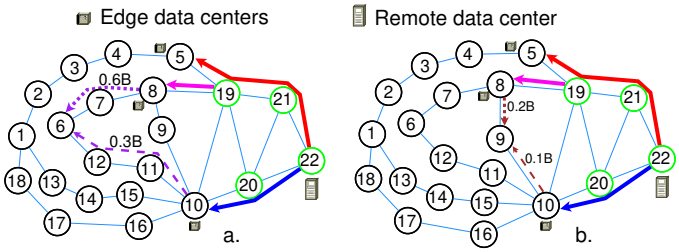

Fig. 2. Multicast and adaptive bandwidth provisioning for XR live streams.

In this section, we build upon BDD by incorporating multicast and adaptive bandwidth provisioning and propose a reliable service-provisioning scheme for low-latency and high-bandwidth XR live streams.

To illustrate our service-provisioning scheme, we consider the scenario in Fig. 2.a where, at time t_2, a request is arriving at node 6 for the XR live stream which is currently available in nodes 5 and 10 (e.g., to serve the requesting node 1 from time t_1 as in Fig. 1.b). For a fair comparison, the requested XR resolution is the same in the scenarios in Figs. 1 and 2. Instead of retrieving the desired content from the remote DC 22 (hence, requiring more network resources), the network can add a multicast branch from node 19 to make the XR live stream available also in node 8 (which is the closest edge DC to the requesting node 6). Compared the scenario in Fig. 1.b, the requested XR live stream is now available closer to the requesting node (i.e., the absolute/longer-path delay is approximately 4 and 3 hops for the scenarios in Figs. 1.b and 2.a, respectively). As a result, leveraging the unique trade-off between latency and bandwidth of XR live streams, our service-provisioning scheme can adaptively reduce reserved bandwidth while still ensuring user experience. As in Fig. 2.a, the bandwidths reserved on the primary and backup paths are $0.6B$ and $0.3B$, respectively (compared to B and $0.8B$ in Fig. 1.b). Moreover, consider a specific scenario as in Fig. 2.b where, at time t_3, another request is arriving at node 9 for the XR live stream which is already available in nodes 5, 8, and 10 (e.g., to serve the requesting node 1 from time t_1 as in Fig. 1.b and to serve the requesting

node 6 from time t_2 as in Fig. 2.a). Since the desired XR live stream is only one hop from the requesting node, in this scenario, our service-provisioning scheme can further reduce the reserved bandwidths on the primary and backup paths to $0.2B$ and $0.1B$, respectively. Below, we use *BDD with multicast and adaptive bandwidth provisioning* (BDD-MA) to denote our proposed service-provisioning scheme which supports both multicast and the capability to adaptively reduce reserved bandwidth while strictly ensuring user experience. In numerical results, we also use *BDD with multicast* (BDD-M) as a baseline service-provisioning scheme. Since a requesting node is primarily served by the closest edge DC, we will use the terms the closest DC and the primary DC interchangeably.

B. Problem Statement

We consider a content service provider (CSP) with a CDN represented by a network graph $G(\mathcal{V}_t, \mathcal{E}_t)$ where $\mathcal{V}_t$ is the set of nodes with available resources (i.e., throughput, computing, or storage) and $\mathcal{E}_t$ is the set of links with available bandwidth at the request arrival time (i.e., denoted by t). Since XR live stream requests arrive, hold, and depart, the sets $\mathcal{V}_t$ and $\mathcal{E}_t$ may vary over time. Within the network graph, we use $\mathcal{S}$ to denote the set of XR live streams offered by the CSP and $\mathcal{R}$ to denote the set of remote DCs close to the locations of live events ($\mathcal{R} \subset \mathcal{V}_t$).

In this work, each incoming XR live stream request, or Ω, is characterized by a tuple, $\Omega = (r, s, t, u, v, w, x, y)$, where r is the remote DC which is originally hosting the requested XR live stream (i.e., $r \in \mathcal{R}$), s is the requested XR live stream ID (i.e., $s \in \mathcal{S}$), and t is arrival time (sec), u is the requested bandwidth (Gbps) in normal operation (i.e., bandwidth reserved on the primary path from the primary edge DC to the requesting node), v is the degraded bandwidth (Gbps) in case the primary path is disrupted (i.e., bandwidth reserved on the backup path from the backup edge DC to the requesting node), w is the holding time (minute) of the stream, x is the required latency (ms), and y is the requesting node. Within CDN, we use D to denote the list of nearby edge DCs in an ascending order of distance to the requesting node and $\mathcal{D}$ to denote the set D (i.e., $\mathcal{D}$ is a set of D). Without loss of generality, $\mathcal{D} \cap \mathcal{R} = \varnothing$ (i.e., edge DCs are not remote DCs). We use $\mathcal{M}_s$ to denote the set of nodes belonging to the multicast tree for the XR live stream s. Moreover, when a stream is added to a DC, the DC must reserve a certain amount of computing resources (e.g., equal to the bit rate of the stream, either in a remote DC or an edge DC) and storage resources (e.g., for buffering in an edge DC).

The problem of reliable provisioning of low-latency and high-bandwidth XR live streams can be formally stated as follows. *Given* the network graph at the request arrival time $G(\mathcal{V}_t, \mathcal{E}_t)$, the XR live stream request $\Omega = (r, s, t, u, v, w, x, y)$, and D (which is the list of available edge DCs in a cluster), *admit* the XR live stream request *such that* the requested bandwidth in normal operation u, the degraded bandwidth v (in case the primary path is disrupted), and the required latency x are fulfilled. The feasible solution is subject to the constraints on the available bandwidth of each physical link, computing and storage capacity in each selected edge DC, throughput in each transit node, and the survivability of paths. We use BDD-MA (as in Fig. 2) to fulfill this request in which, for each arrival, we consider one of the following scenarios.

First, we consider the scenario where the requested XR live stream is available in the closest edge DC (the first DC in D). This scenario implies that the same XR live stream is also available in another edge DC in the same cluster, since it is initially available in two different edge DCs (as in Fig. 1.b). In this case, BDD-MA is reduced to finding a pair of link-disjoint paths from the first and second closest edge DCs with the desired XR live stream to the requesting node and adaptively reserving bandwidth on each path, throughput in each transit node, and computing and storage resources in each serving DC. We will elaborate on this scenario in `Algorithm` 1.

Second, we consider the scenario where the requested XR live stream is unavailable in the closest edge DC but available in multiple locations in the DC cluster. In this scenario, BDD-MA performs the following steps: a) find the closest node in the multicast tree (i.e., $\mathcal{M}_s$) and make the desired XR live stream available in the closest edge DC (to serve as the primary edge DC) by adding an additional multicast branch, b) find a pair of link-disjoint paths from the primary edge DC (now with the active requested XR live stream) and the backup edge DC (which must not be the source of the additional multicast branch for survivability) to the requesting node and adaptively reserve bandwidth on each path (including the additional multicast path), throughput in each transit node, and computing and storage resources in each serving DC. We will elaborate on this scenario in `Algorithm` 2.

Third, we consider the scenario where the requested XR live stream is only available in a remote DC close to the live event. In this scenario, BDD-MA performs the following steps: a) find a pair of link-disjoint paths from the first and second edge DCs in D (i.e., as the primary and backup edge DCs) to the requesting node to serve as the primary and backup paths, b) make the desired XR live stream available in the primary and backup edge DCs using another pair of link-disjoint paths from the remote DC and adaptively reserve bandwidth on each path, throughput in each transit node, and computing and storage resources in each serving DC. We will elaborate on this scenario in `Algorithm` 3.

Before further elaborating on `Algorithms` 1, 2, and 3, we provide a latency model and a scheme to adaptively reduce network resources offered to an XR live stream request while ensuring user experience.

C. Model of Latency Budget and Offered Bandwidth

In this section, we elaborate on the latency model and consider the unique trade-off between latency and bandwidth of XR live streams to potentially reduce bandwidth offered to a request while still ensuring user experience. To estimate the end-to-end latency between an end user to the serving edge DC, we assume that a mobile network is used as the access network and adopt the following model for latency budget [22]:

$$T_{e2e} = T_{Rad} + T_{Proc} + T_{Rend} + T_{Prop}, \quad (1)$$

where T_{e2e}, T_{Rad}, T_{Proc}, T_{Rend}, and T_{Prop} are the total end-to-end latency, radio transmission time between the antenna and user equipment, processing time (e.g., in centralized units

(CUs) and distributed units (DUs)), rendering and refresh time, and propagation delay, respectively. In our analysis, we focus on T_{Prop}, i.e., the propagation delay between the requesting node (e.g., a base station) and edge DCs (which includes mid-haul and front-haul networks). In fact, for a specific combination of the radio access technology (e.g., 5G), a network configuration (e.g., a placement of CUs and DUs), and an XR resolution (e.g., advanced VR), T_{Rad}, T_{Proc}, and T_{Rend} are fixed. These terms will be further improved in next-generation networks (e.g., 6G). Based on the overall end-to-end latency obtained in Eqn. (1), we use a linear model [12] (latency vs. offered bandwidth) to adaptively offer less bandwidth while still guaranteeing user experience, i.e.,

$$u' = u * T_{e2e} \,/\, x, \tag{2}$$

where u' is the offered bandwidth (Gbps). Since $T_{e2e} \leq x$ (otherwise the XR live stream request is rejected), the offered bandwidth is equal to or less than the requested bandwidth.

D. Algorithms For Reliable Provisioning of Low-Latency and High-Bandwidth XR Live Streams

This section is divided into three subsections, in each of which we consider one of the scenarios introduced in Section III-B and provide the pseudo codes of the corresponding `Algorithms` 1, 2, and 3. In all the three algorithms, we use the Dijkstra's algorithm [23] and the Bhandari's algorithm [24] to find the shortest path and all link-disjoint paths with minimal total length from one node to another node in a graph. We sort the output link-disjoint paths in an ascending order with respect to their lengths. For convenience, we use p_b^a and B_b^a to denote the shortest path and the list of link-disjoint paths with minimal total length from nodes a to b, respectively. To find all link-disjoint paths from one node to a set of nodes (e.g., a set of edge DCs), we introduce an auxiliary graph by leveraging a dummy node (denoted by d) and several dummy links (i.e., `get_aux_graph` in `Algorithm` 3). The reader can refer to [9] for how to build the auxiliary graph. Below, we use d_p, d_b, and d_m to denote the primary edge DC (i.e., d_p is the closest edge DC to the requesting node), the backup edge DC, and the multicast source node (i.e., the source of the additional multicast branch, if needed, to make the XR live stream available in d_p), respectively ($d_p \in \mathcal{D}$, $d_b \in \mathcal{D}$, $d_m \in \mathcal{M}_s$). Moreover, for each edge DC with an active XR live stream, we use, e.g., $p_{d_p}^r$ to denote the updated multicast path from the root (i.e., r) to the primary edge DC (i.e., d_p). All the three algorithms take $G(\mathcal{V}_t, \mathcal{E}_t)$, $\Omega = (r, s, t, u, v, w, x, y)$, and D as inputs; and return Δ, Σ, Φ, and Ψ as the hash maps representing: bandwidth reserved on each link, throughput reserved on each transit node, computing resources reserved in each serving DC, and storage resources reserved in each serving DC, respectively. In case there are insufficient network resources or the required latency is not fulfilled, the incoming XR live stream request is rejected and no reservation is made (i.e., $\Delta = \Sigma = \Phi = \Psi = \varnothing$). Note that the following algorithms are derived for a single XR live stream request considering the updated network graph. To obtain the numerical results reported in this work, we integrate them into a dynamic simulation framework as in Fig. 4. We will elaborate on the dynamic simulation framework later.

Algorithm 1: XR live stream available in the closest edge DC.

Input: $G(\mathcal{V}_t, \mathcal{E}_t)$, $\Omega = (r, s, t, u, v, w, x, y)$, D
Output: Δ, Σ, Φ, Ψ
Data: $\Delta = \Sigma = \Phi = \Psi = \varnothing$

1 $d_p = D[0]$
2 **for** i in $[0, |D| - 1]$ **do**
3 **if** s in $D[i]$ & $D[i] \neq d_p$ & $p_{D[i]}^r \cap p_{d_p}^r = \varnothing$ **then**
4 $d_b = D[i]$; break
5 $B_y^{d_p}$ = get_disjoint_paths($G(\mathcal{V}_t, \mathcal{E}_t), d_p, y$)
6 **for** i in $[0, |B_y^{d_p}| - 1]$ **do**
7 **if** $B_y^{d_p}[i] \cap p_{d_b}^r = \varnothing$ **then** $p_y^{d_p} = B_y^{d_p}[i]$; break
8 $B_y^{d_b}$ = get_disjoint_paths($G(\mathcal{V}_t, \mathcal{E}_t), d_b, y$)
9 **for** i in $[0, |B_y^{d_b}| - 1]$ **do**
10 **if** $B_y^{d_b}[i] \cap p_{d_p}^r = \varnothing$ & $B_y^{d_b}[i] \cap p_y^{d_p} = \varnothing$ **then**
11 $p_y^{d_b} = B_y^{d_b}[i]$; break
12 **if** $p_y^{d_p}$ & $p_y^{d_b}$ & *resources are sufficient* **then**
13 $T_{Prop} = T_{abs}(p_y^{d_p}, p_y^{d_b})$; $T_{e2e} \leftarrow T_{Prop}$
14 **if** $T_{e2e} \leq x$ **then**
15 $u' = u * T_{e2e} \,/\, x$
16 update Δ, Σ, Φ, and Ψ
17 return Δ, Σ, Φ, Ψ

1) XR live stream is available in the closest edge DC

In `Algorithm` 1, we present the pseudo code to provision an incoming request in case the XR live stream is available in the closest edge DC (s in d_p where $d_p = D[0]$ as the closest/primary edge DC to the requesting node). This scenario implies that the requested XR live stream is also available in another edge DC in the same cluster. From lines 1 to 4, the algorithm finds the second-closest edge DC (to the requesting node) with the active requested XR live stream to serve as the backup edge DC (d_b). For survivability, the multicast path from the remote DC to the backup edge DC must be link-disjoint with the multicast path from the remote DC to the primary edge DC (i.e., $p_{d_b}^r \cap p_{d_p}^r = \varnothing$). In line 5, the algorithm uses the Bhandari's algorithm to compute all link-disjoint paths from the primary edge DC to the requesting node (i.e., $B_y^{d_p}$). From lines 6 to 7, the algorithm finds the primary path from the primary edge DC to the requesting node which is the shortest path in $B_y^{d_p}$. To guarantee survivability, the primary path from the primary edge DC to the requesting node must be link-disjoint with the multicast path from the root to the backup edge DC (i.e., $p_y^{d_p} \cap p_{d_b}^r = \varnothing$). To find the backup path from the backup edge DC to the requesting node, the algorithm uses the Bhandari's algorithm to compute all link-disjoint paths from the backup edge DC to the requesting node (i.e., $B_y^{d_b}$, in line 8). The backup path from the backup edge DC to the requesting node is the shortest path in $B_y^{d_b}$ which is link-disjoint (for survivability) with both the multicast path from the root to the primary edge DC (i.e., $p_y^{d_b} \cap p_{d_p}^r = \varnothing$) and the primary path from the primary edge DC to the requesting node (i.e., $p_y^{d_b} \cap p_y^{d_p} = \varnothing$). If the pair of primary and backup paths is found, the algorithm verifies that the network resources are sufficient along the paths and also in the selected edge DCs, estimates the paths' absolute delay (i.e., the delay of the longer path between the primary and backup paths), and

Algorithm 2: XR live stream available in the DC cluster.

Input: $G(\mathcal{V}_t, \mathcal{E}_t)$, $\Omega = (r, s, t, u, v, w, x, y)$, D
Output: Δ, Σ, Φ, Ψ
Data: $\Delta = \Sigma = \Phi = \Psi = \varnothing$
1 d_p = $D[0]$; $L_s = \infty$
2 **for** m in $\mathcal{M}_s$ **do**
3 $p^m_{d_p}$ = get_shortest_path($G(\mathcal{V}_t, \mathcal{E}_t)$, m, d_p)
4 **if** get_length($p^m_{d_p}$) $< L_s$ **then**
5 $d_m = m$; L_s = get_length($p^m_{d_p}$)
6 $p^{d_m}_{d_p}$ = get_shortest_path($G(\mathcal{V}_t, \mathcal{E}_t)$, d_m, d_p);
7 **for** i in $[0, |D| - 1]$ **do**
8 **if** $D[i] \neq d_p$ & s in $D[i]$ & $p^r_{d_p} \cap p^r_{D[i]} = \varnothing$ **then**
9 $d_b = D[i]$; break
10 $B^{d_p}_y$ = get_disjoint_paths($G(\mathcal{V}_t, \mathcal{E}_t), d_p, y$)
11 **for** i in $[0, |B^{d_p}_y| - 1]$ **do**
12 **if** $B^{d_p}_y[i] \cap p^r_{d_b} = \varnothing$ **then** $p^{d_p}_y = B^{d_p}_y[i]$; break
13 $B^{d_b}_y$ = get_disjoint_paths($G(\mathcal{V}_t, \mathcal{E}_t), d_b, y$)
14 **for** i in $[0, |B^{d_b}_y| - 1]$ **do**
15 **if** $B^{d_b}_y[i] \cap p^r_{d_p} = \varnothing$ & $B^{d_b}_y[i] \cap p^{d_p}_y = \varnothing$ **then**
16 $p^{d_b}_y = B^{d_b}_y[i]$; break
17 **if** $p^{d_p}_y$ & $p^{d_b}_y$ & *resources are sufficient* **then**
18 $T_{Prop} = T_{abs}(p^{d_p}_y, p^{d_b}_y)$; $T_{e2e} \leftarrow T_{Prop}$
19 **if** $T_{e2e} \leq x$ **then**
20 $u' = u * T_{e2e} \,/\, x$
21 update Δ, Σ, Φ, and Ψ
22 return Δ, Σ, Φ, Ψ

computes end-to-end latency using the model in Eqn. (1) (lines 12-13). If the end-to-end latency fulfills the requirement (i.e., $T_{e2e} \leq x$), the incoming XR live stream request is admitted and the algorithm updates and returns the bandwidth reserved on each link, throughput reserved in each transit node, computing and storage resources reserved in each serving DC using the hash maps Δ, Σ, Φ, and Ψ (lines 14-17).

2) XR live stream is available in the DC cluster

In this scenario, the requested XR live stream is unavailable in the closest edge DC (s not in d_p where $d_p = D[0]$ as the primary/closest DC) but available in multiple locations in the local DC cluster. `Algorithm 2` exploits the multicast functionality in the optical layer to make the requested XR live stream available in the closest edge DC. From lines 1 to 5, the algorithm finds the closest node (to the closest edge DC) in the multicast tree, or $\mathcal{M}_s$, to serve as the source node of the additional multicast branch. If the multicast source node (i.e., d_m) is found, the algorithm uses the Dijkstra's algorithm to find the additional multicast path, $p^{d_m}_{d_p}$, as the shortest path from the multicast source node to the closest edge DC (in line 6). From lines 7 to 9, the algorithm finds another edge DC with the active requested XR live stream (rather than the primary edge DC or the multicast source node) to serve as the backup edge DC. To guarantee survivability, the multicast path for the backup edge DC must be link-disjoint with the multicast path for the primary edge DC (i.e., $p^r_{d_b} \cap p^r_{d_p} = \varnothing$). To find the primary path from the primary edge DC to the requesting node, from lines 10 to 12, the algorithm uses the Bhandari's algorithm to compute all link-disjoint paths from the primary edge DC to the requesting node (i.e., $B^{d_p}_y$). The primary path from the primary edge DC to the requesting node (i.e., $p^{d_p}_y$) is the shortest path in $B^{d_p}_y$ which is link-disjoint with the multicast path from the remote DC to the backup edge DC (i.e., $p^{d_p}_y \cap p^r_{d_b} = \varnothing$). To find the backup path from the backup edge DC to the requesting node, in line 13, the algorithm uses the Bhandari's algorithm to compute all link-disjoint paths from the backup edge DC to the requesting node (i.e., $B^{d_b}_y$). For survivability, the backup path from the backup edge DC to the requesting node (i.e., $p^{d_b}_y$) is the shortest path in $B^{d_b}_y$ which is link-disjoint with both the multicast path from the remote DC to the primary edge DC and the primary path from the primary edge DC to the requesting node (i.e., $p^{d_b}_y \cap p^r_{d_p} = \varnothing$ and $p^{d_b}_y \cap p^{d_p}_y = \varnothing$, lines 14-16). If the pair of primary and backup paths and the additional multicast path are found, the algorithm verifies that the network resources are sufficient along the paths and also in each selected DC, estimates the absolute delay of the primary and backup paths, and computes the end-to-end latency using the model in Eqn. (1) (lines 17-18). If the end-to-end latency fulfills the required latency, the incoming XR live stream request is admitted and the algorithm updates and returns the bandwidth reserved on each link, throughput reserved on each node, computing and storage resources reserved in each serving DC using the hash maps Δ, Σ, Φ, and Ψ (line 19-22).

3) XR live stream is only available in a remote DC

In `Algorithm 3`, we consider the scenario where the requested XR live stream is unavailable in the local DC cluster and it must be retrieved from a remote DC close to the location of the live event. In this case, the algorithm selects the first and second closest edge DCs to the requesting node in D to serve as the primary and backup edge DCs (in line 1). Once the primary and backup edge DCs are found, in line 2, the algorithm builds an auxiliary graph leveraging a dummy node (i.e., d), the primary and backup edge DCs (i.e., d_p and d_b), and a few dummy links. In lines 3-4, the algorithm uses the auxiliary graph and the Bhandari's algorithm to compute the primary and backup paths from the primary and backup edge DCs to the requesting node. If the pair of the primary and backup paths is found, in the next steps, the algorithm uses the Bhandari's algorithm to compute all link-disjoint paths from the remote DC to the primary edge DC (i.e., $B^r_{d_p}$, in line 6). To guarantee that the requested XR live stream is not concurrently disrupted in the primary and backup edge DCs, the multicast path from the remote DC to the primary edge DC (i.e., $p^r_{d_p}$) is the shortest path in $B^r_{d_p}$ which is link-disjoint with the backup path from the backup edge DC to the requesting node (i.e., $p^r_{d_p} \cap p^{d_b}_y = \varnothing$, in lines 7-8). To find a valid multicast path from the remote DC to the backup edge DC (i.e., $p^r_{d_b}$), in line 9, the algorithm computes all link-disjoint paths from the remote DC to the backup edge DC (i.e., $B^r_{d_b}$). For survivability, $p^r_{d_b}$ is the shortest path in $B^r_{d_b}$ which is link-disjoint with both the primary path from the primary edge DC to the requesting node and the multicast path from the remote DC to the primary edge DC (i.e., $p^r_{d_b} \cap p^{d_p}_y = \varnothing$ and $p^r_{d_b} \cap p^r_{d_p} = \varnothing$, lines 10-12). If there are sufficient network resources along the paths

Algorithm 3: XR live stream available in a remote DC.

Input: $G(\mathcal{V}_t, \mathcal{E}_t)$, $\Omega = (r, s, t, u, v, w, x, y)$, D
Output: Δ, Σ, Φ, Ψ
Data: $\Delta = \Sigma = \Phi = \Psi = \varnothing$
1 d_p = $D[0]$; d_b = $D[1]$
2 $G_d(\mathcal{V}_t, \mathcal{E}_t)$ = get_aux_graph($G(\mathcal{V}_t, \mathcal{E}_t), \{d_p, d_b\}, d$)
3 B_y^d = get_disjoint_paths($G_d(\mathcal{V}_t, \mathcal{E}_t), d, y$)
4 $p_y^{d_p} = B_y^d[0]$; $p_y^{d_b} = B_y^d[1]$
5 **if** $p_y^{d_p}$ & $p_y^{d_b}$ **then**
6 $B_{d_p}^r$ = get_disjoint_paths($G_d(\mathcal{V}_t, \mathcal{E}_t), r, d_p$)
7 **for** i in $[0, |B_{d_p}^r| - 1]$ **do**
8 **if** $B_{d_p}^r[i] \cap p_y^{d_b} = \varnothing$ **then** $p_{d_p}^r = B_{d_p}^r[i]$; break
9 $B_{d_b}^r$ = get_disjoint_paths($G_d(\mathcal{V}_t, \mathcal{E}_t), r, d_b$)
10 **for** i in $[0, |B_{d_b}^r| - 1]$ **do**
11 **if** $B_{d_b}^r[i] \cap p_y^{d_p} = \varnothing$ & $B_{d_b}^r[i] \cap p_{d_p}^r = \varnothing$ **then**
12 $p_{d_b}^r = B_{d_b}^r[i]$; break
13 **if** $p_{d_p}^r$ & $p_{d_b}^r$ & *resources are sufficient* **then**
14 $T_{Prop} = T_{abs}(p_y^{d_p}, p_y^{d_b})$; $T_{e2e} \leftarrow T_{Prop}$
15 **if** $T_{e2e} \leq x$ **then**
16 $u = u * T_{e2e} / x$
17 update Δ, Σ, Φ, and Ψ
18 return Δ, Σ, Φ, Ψ

(i.e., the primary path, the backup path, and the paths from the remote DC to the primary and backup edge DCs) and in the primary and backup edge DCs, the algorithm estimates the absolute propagation delay of the primary and backup paths and computes the end-to-end latency using the model in Eqn. (1) (lines 13-14). If the end-to-end latency fulfills the required latency, the incoming XR live request is admitted and the algorithm updates the bandwidth reserved on each link, throughput reserved on each node, computing and storage resources reserved in each serving DC using the hash maps Δ, Σ, Φ, and Ψ, respectively (lines 15-18).

E. Complexity Analysis

In `Algorithms` 1, 2, and 3, the number of nearby edge DCs in the cluster (i.e., $|\mathcal{D}|$) is deterministic. Moreover, the number of link-disjoint paths from one node to another node in a network graph found by the Bhandari's algorithm (i.e., $|B_y^{d_p}|$, $|B_y^{d_b}|$, $|B_y^d|$, $|B_{d_p}^r|$, or $|B_{d_b}^r|$) is also deterministic (e.g., the number of link-disjoint paths must not exceed the nodal degree of the source and destination nodes). Therefore, the time complexity of `Algorithms` 1, 2, and 3 heavily depends on the time complexity of the Dijkstra's and Bhandari's algorithms and the number of nodes in multicast trees. In this study, we use the Dijkstra's algorithm and a binary min-heap data structure to find the shortest path from one node to another node in a network graph whose time complexity is $\mathcal{O}\left[(|\mathcal{V}_t| + |\mathcal{E}_t|) * \log |\mathcal{V}_t|\right]$. The Bhandari's algorithm uses the Dijkstra's algorithm as the underlying algorithm to find the shortest path from one node to another node, updates the network graph, and repeats this loop until no more link-disjoint paths are found. Since updating of a network graph is linear with the number of nodes and links (i.e., $\mathcal{O}\left(|\mathcal{V}_t| + |\mathcal{E}_t|\right)$), the time complexity of the Bhandari's algorithm is $\mathcal{O}\left[K * (|\mathcal{V}_t| + |\mathcal{E}_t|) * (1 + \log |\mathcal{V}_t|)\right]$ where K is the number of link-disjoint paths. To find the multicast source node (which is a node belonging to the multicast tree and closest to the closest edge DC) as in `Algorithm` 2 (lines 1-5), in the worst case, the algorithm must compute the shortest paths from every node in the graph to the closest edge DC; hence, the time complexity is $\mathcal{O}\left[|\mathcal{V}_t| * (|\mathcal{V}_t| + |\mathcal{E}_t|) * \log |\mathcal{V}_t|\right]$. If we omit non-dominant terms and deterministic constants, the time complexity of `Algorithms` 1, 2, and 3 is $\mathcal{O}\left[|\mathcal{V}_t| * (|\mathcal{V}_t| + |\mathcal{E}_t|) * \log |\mathcal{V}_t|\right]$. Even though the problem of admitting a single XR live stream can be formulated as a combinatorial optimization problem (which is proved to be NP-hard [25]), an exact solving method (e.g., Integer Linear Programming) is infeasible in a dynamic network environment (such as in this work) where an instant solution for an incoming request must be found.

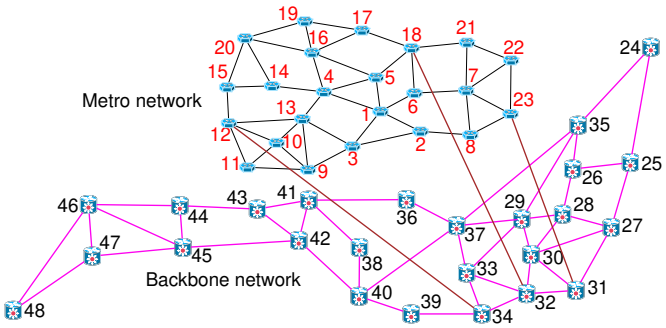

Fig. 3. The physical network including Tokyo23 and JPN25.

IV. Illustrative Numerical Results

A. Physical Network

To evaluate our proposed service-provisioning scheme, we consider a CDN including a metro network and a backbone network (Fig. 3). The metro network is the Tokyo23 covering an urban area up to tens of kilometers in diameter [26]. It consists of 43 100-Gbps bi-directional links and 23 nodes (1-23), with each node located at each ward office building in the Tokyo metropolitan area. The backbone network is the Japan Photonic Network with 25 nodes (24-48, in major cities) and 43 400-Gbps bi-directional links (JPN25) [27]. The metro network is connected to the backbone network through nodes 31 (Chiba), 32 (Tokyo), and 34 (Yokohama). In the composite network, we use $\mathcal{D} = \{1, 7, 12, 13, 16\}$ and $\mathcal{R} = \{24, 27, 37, 41, 43, 48\}$ to denote the set of edge DCs in a local cluster in the metro network and the set of remote DCs in the backbone network. Each edge DC has a storage of 100 TB and the computing capability to support XR traffic of 240 Gbps. Remote DCs are more powerful with each one having storage of 1000 TB and the computing capability to process XR traffic up to 640 Gbps. Each transit node in metro and backbone networks has a maximum capacity of 500 and 1000 Gbps, respectively.

B. Simulation Setup

We consider a CSP with 1000 available XR live streams (i.e, $\mathcal{S} = \{1\text{-}1000\}$) in the catalog whose popularity is modeled following a Zipf distribution [22]. XR live stream requests, $\Omega = (r, s, t, u, v, w, x, y)$, are generated following the characteristics of live streaming workloads publicly available in [28],

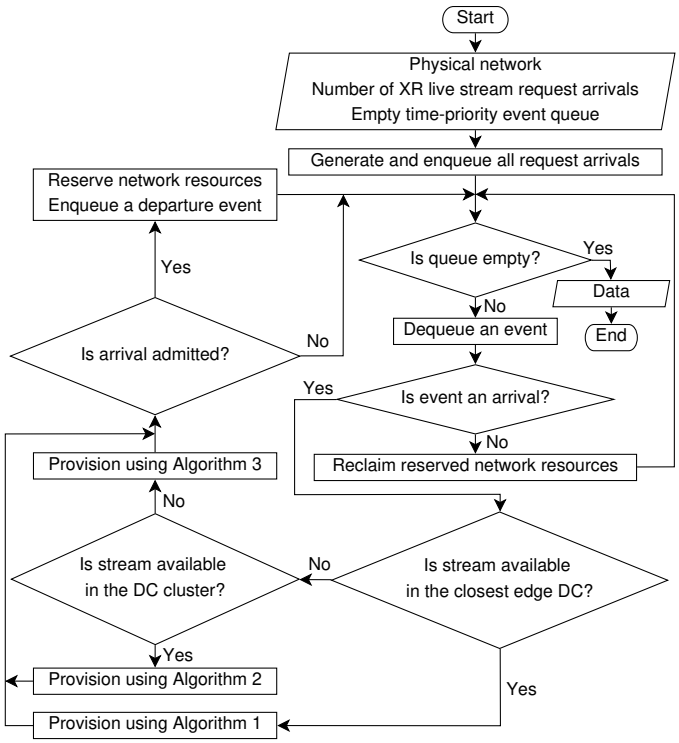

Fig. 4. Dynamic simulation framework.

[29] where data were collected in a 3-month period and contain over 70 million requests for 5,000 distinct URLs from clients in 200 different countries at Akamai Technologies. In detail, the remote DC, r, is selected from $\mathcal{R}$ (i.e., $r \in \mathcal{R}$). The XR live stream ID, s, is selected from the service provider catalog using its popularity (i.e., $s \in \mathcal{S}$). The XR live stream arrival time, t, is modeled following a discrete Poisson process. The requested bandwidth, u, is selected from four values, 0.1, 0.4, 0.7, and 1 Gbps corresponding to four levels of XR resolution: early-stage XR, entry-level XR, advanced XR, and extreme XR, respectively [22]. The degraded bandwidth in case the primary path is disrupted, v, is selected such that an active XR live stream can be downgraded to a lower level of XR resolution. The holding time of the XR live stream, w, is modeled using a truncated Pareto distribution with minimum and maximum duration of 15 and 120 minutes, respectively. The required latency, x, is set to 15 ms to avoid motion sickness [12]. Finally, the requesting node, y, is selected among those metro nodes which are not edge DCs (i.e, $y \notin \mathcal{D}$). To analyze end-to-end latency for this network, we adopt the values reported in [22] where T_{Rad}, T_{Proc}, and T_{Rend} (Eqn. 1) are 50, 100, and 500 μs, respectively. Each XR live stream from a remote DC to an edge DC requires a 1-Gbps connection and a buffering storage of 500 GB.

To obtain numerical results for BDD-MA, we use the dynamic simulation framework in Fig. 4. We first start an empty network and generate all XR live stream request arrivals and enqueue them in a time-priority queue. During simulation, each XR live stream request arrival is dequeued from the queue and processed using a) `Algorithm 1` if the requested XR live stream is available in the closest edge DC; b) `Algorithm 2` if the requested XR live stream is available in the DC cluster; or c) `Algorithm 3` if the requested XR live stream is only available in a remote DC. In case a request arrival is admitted, required network resources for the XR live stream including bandwidth, computing, storage, and node throughput are reserved; and a departure event with departure time equal to the arrival time plus the holding time is enqueued to the queue. If the dequeued event is a departure, reserved network resources are reclaimed. We found that the numerical results obtained for simulating 500,000 and 1,000,000 XR live stream requests are comparable. To reduce experiment time, below, we report the numerical results for simulating the network with 500,000 incoming XR live stream requests.

For the baseline service-provisioning schemes such as BSD, BDD, and BDD-M, numerical results can be obtained using the same dynamic simulation framework with some minor modifications. For BSD, in case the XR live stream is unavailable in the closest edge DC, the service-provisioning scheme first makes the desired XR live stream available in the closest edge DC and then delivers it to the requesting node using a pair of primary and backup paths (as in Fig. 1.a). In case the XR live stream is available in the closest edge DC, BSD immediately delivers the desired XR live stream to the requesting node using a pair of primary and backup paths. For BDD and BDD-M, for each incoming XR live stream request, the dynamic simulation framework must consider the following scenarios. If the requested XR live stream is available in the closest edge DC (and also in another edge DC), BDD and BDD-M deliver the desired XR live stream to the requesting node from the first and second closest edge DCs served as the primary and back up edge DCs. If the requested XR live stream is unavailable in the local DC cluster, BDD and BDD-M first make the desired XR live stream available in the first and second closest edge DCs and then deliver it to the requesting node. In case the requested XR live stream is unavailable in the closest edge DC but available in multiple locations in the DC cluster, BDD and BDD-M work differently. To make the desired XR live stream available in the closest edge DC, while BDD retrieves data from the remote DC close to the live event, BDD-M retrieves data from the closest node in the multicast tree.

In this work, we focus on the efficiency of our service-provisioning scheme and leave the choice of technologies in the physical layer as open topics. For instance, each reserved connection can be groomed with other signals and assigned to a lightpath in a WDM or an EO network [8]. We evaluate computing resources as the computing capability of DCs to process live streams (in Gbps) and neglect their physical configurations (e.g., number of CPUs). The throughput in each optical transit node is directly related to the implementation of optical components such as transceivers and transponders.

C. Numerical Results

In a dynamic network environment such as in this work, the acceptance ratio of incoming requests as a function of the arrival rates is an important metric. In an ideal network with abundant resources, the acceptance ratio, or η, is 100%. However, in practice, network resources are not infinite, and when XR live streams arrive at a very high rate, the network becomes congested, and some incoming requests may be

TABLE I
CONGESTION RATES FOR THE SERVICE-PROVISIONING SCHEMES.

Schemes	BSD	BDD	BDD-M	BDD-MA
C_0(requests/hour)	53	120	167	223

rejected (i.e., $\eta < 100\%$). We define the *congestion rate* in a dynamic network as the arrival rate at which the network starts to reject part of incoming XR stream requests, and use C_0 to denote this value. To run a network harder, with the same setting, a service-provisioning scheme with a higher C_0 is desirable. In a congested network, a service-provisioning scheme with a higher η is preferred since it can admit more XR live streams. Below, we use η and C_0 to evaluate our proposed service-provisioning scheme (i.e., BDD-MA, derived in `Algorithms` 1, 2, and 3) and compare its performance to that of the reference schemes (e.g., BSD, BDD, and BDD-M). Moreover, we compare the performance of the four schemes in terms of the utilization of network resources such as network bandwidth, computing and storage resources, and the throughput of nodes. In general, when the arrival rate increases, more XR streams are locally available in the DC cluster and an incoming request can be served from edge DCs. As a result, the average network resources reserved per admitted request decreases as the arrival rate increases (i.e., the curves are downward in Figs. 5-8). In the next sections, we elaborate on these results.

1) Acceptance ratio

In Table I, we report the congestion rate, C_0, for BSD, BDD, BDD-M, and BDD-MA. As expected, BDD-MA outperforms BSD, BDD, and BDD-M. While BSD, BDD, and BDD-M start to reject incoming requests around the arrival rates of 53, 120, and 167 requests/hour, respectively, BDD-MA can run the same network harder and only starts to reject incoming requests around the arrival rate of 223 requests/hour. In Table II, we report the acceptance ratios of BSD, BDD, BDD-M, and BDD-MA at different request arrival rates. We observe that, in a congested network (e.g., with arrival rate larger than 400 requests/hour), BDD-MA approximately accepts 20%, 8%, and 3% more incoming requests compared to BSD, BDD, and BDD-M, respectively. That is, for the same network setting, our proposed service-provisioning scheme can serve more users. Compared to the baseline service-provisioning schemes, the superiority of BDD-MA results from its better utilization of network resources. In the next subsections, we analyze the network resource utilization of the service-provisioning schemes in various aspects including network bandwidth, computing and storage resources, and throughput of nodes.

TABLE II
ACCEPTANCE RATIOS OF THE SERVICE-PROVISIONING SCHEMES.

Rates(requests/hour)	Acceptance ratio (%)			
	BSD	BDD	BDD-M	BDD-MA
100	97.1	100	100	100
400	79.8	88.1	91.8	94.7
700	72.2	81.9	87.3	90.4
1000	67.5	79.6	82.7	87.8
1300	63.8	76.4	79.2	84.1

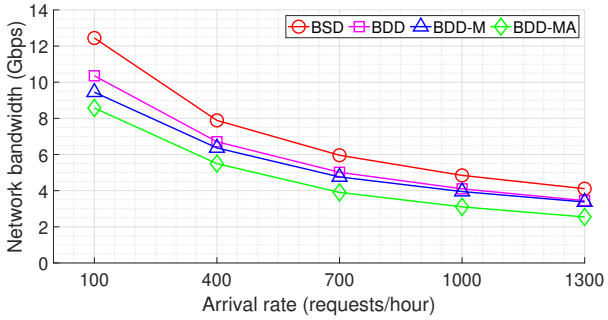

Fig. 5. Average network bandwidth per admitted XR live stream request.

2) Network bandwidth

In Fig. 5, we report the average reserved network bandwidth per admitted XR live stream (Gbps) (i.e., the sum of the bandwidth on the primary and backup paths weighted by the number of hops). The numerical results show that the average reserved network bandwidth per admitted XR live stream decreases from BSD, BDD, BDD-M, and to BDD-MA. For instance, at the arrival rate of 100 requests/hour, BDD-MA reserves 31%, 17%, and 9% less network bandwidth per admitted XR live stream compared to BSD, BDD, and BDD-M, respectively. At a higher arrival rate such as 1300 requests/hour, BDD-MA reserves 38%, 27%, and 24% less network bandwidth per admitted XR live stream compared to BSD, BDD, and BDD-M, respectively. This can be explained for BDD-MA as, in case the requested XR live stream is available in the local DC cluster, it takes advantage of the multicast functionality in the optical layer. Rather than establishing connections from the remote DC (hence, using more network bandwidth), BDD-MA utilizes the multicast functionality to make the requested XR live stream available in the closest edge DC from a local node in the multicast tree. Moreover, in case the end-to-end latency from the requesting node to serving edge DCs is less than the required latency, BDD-MA adaptively offers less bandwidth to the request while still ensuring user experience. At higher rates where more XR live streams are available in the local DC cluster, the superiority of BDD-MA becomes noticeable since its multicast functionality and adaptive bandwidth provisioning are frequently utilized.

3) Computing resources

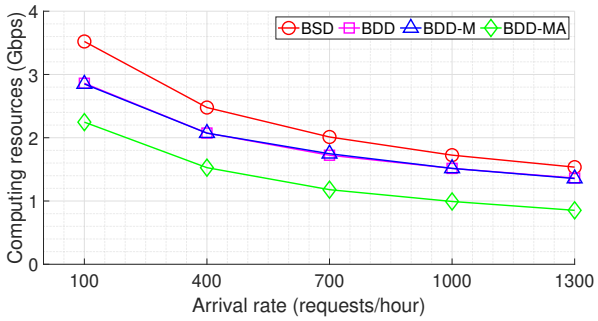

Fig. 6. Average computing resources per admitted XR live stream request.

In Fig. 6, we report the average reserved computing resources (in Gbps) per admitted XR live stream. The numerical results show that our service-provisioning scheme uses less computing resources. Specifically, BDD-MA approximately

reserves 36%, 22%, and 21% less computing resources to admit an incoming XR live stream compared to BSD, BDD, and BDD-M, respectively. The reason is that there are scenarios where the requested XR stream is unavailable in the closest edge DC but available in multiple locations in the DC cluster. In these scenarios, for BSD, it requires two (i.e., one for primary and one for backup) paths from a remote DC to enable the requested XR live stream in the closest edge DC. In contrast to BSD, in these scenarios, BDD and BDD-M require only one path (either from a remote DC or from a multicast DC) to enable the requested stream to the closest edge DC. Even though BDD and BDD-M retrieve data from different locations, the amount of computing resources BDD reserves in the closest and remote edge DCs is comparable with the amount of computing resources BDD-M reserves in the closest and multicast edge DCs. BDD-MA benefits from the multicast functionality and the adaptivity of offered bandwidth. Hence, it can further reduce the amount of computing resources to admit an incoming XR live stream request.

4) Storage resources

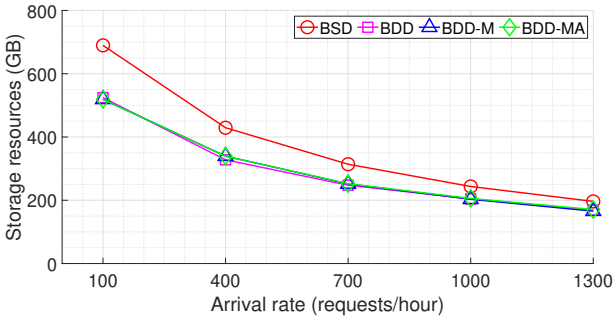

Fig. 7. Average storage resources per admitted XR live stream request.

In Fig. 7, we show the average reserved storage resources (in GB) per admitted XR live stream. These buffering storage resources are required to transfer data from an active stream to an edge DC. In general, BDD, BDD-M, and BDD-MA require less storage resources to admit an incoming XR live stream. On average, BDD-MA reserves approximately 20% less storage resources compared to BSD. This result can be explained as there are scenarios where the requested stream is unavailable in the closest edge DC but available at multiple locations in the DC cluster. In these scenarios, while BSD requires storage resources to buffer two active streams, BDD, BDD-M, and BDD-MA require storage resources to buffer only one additional stream (either from a remote DC such as in BDD or from a local multicast node such as in BDD-M and BDD-MA). We also observe that, on average, BDD, BDD-M, and BDD-MA require the same amount of storage resources to admit an incoming request. The reason is that, in each of the three scenarios in framework in Fig. 4, they require to add the same number of active streams (hence, the same amount of storage resources) to make the requested XR live stream available in the closest edge DC. Specifically, in case the requested XR live stream is unavailable in the closest edge DC but available in the DC cluster, they require storage resources to buffer for one additional active stream. In case the requested XR live stream is unavailable in the DC cluster

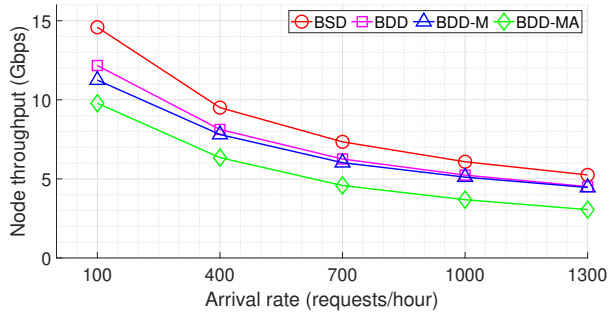

Fig. 8. Average node throughput per admitted XR live stream request.

but available in a remote DC, they require storage resources to buffer for two active streams.

5) Node throughput

In Fig. 8, we report the average reserved node throughput (Gbps) per admitted XR live stream. Overall, compared to BSD, BDD, and BDD-M, BDD-MA requires less node throughput to admit an incoming XR live stream. To elaborate, at the arrival rate of 100 requests/hour, BDD-MA reserves 33%, 20%, and 12% less node throughput compared to BSD, BDD, and BDD-M, respectively. At a higher arrival rate, e.g., 1300 requests/hour, BDD-MA noticeably outperforms the baseline service-provisioning schemes such that it can further reduce the total node throughput reserved for each admitted XR live stream. At this higher arrival rate, BDD-MA requires 42%, 33%, and 32% less node throughput compared to BSD, BDD, and BDD-M, respectively. We conclude that the higher the arrival rate is, the more the node throughput can be saved by BDD-MA. The reason is that, at a higher arrival rate, the probability a requested XR stream is active in the DC cluster is higher. As a result, BDD-MA has a higher chance to activate its multicast and adaptive bandwidth provisioning to save more node throughput. BDD-M with its multicast has a slight advantage over BDD only in the scenario where the requested XR live stream is unavailable in the closest edge DC but available in the DC cluster. As shown in Fig. 8, the average amount of reserved node throughput for BDD and BDD-M to admit an incoming XR live stream is comparable.

6) Number of requests served from local nodes

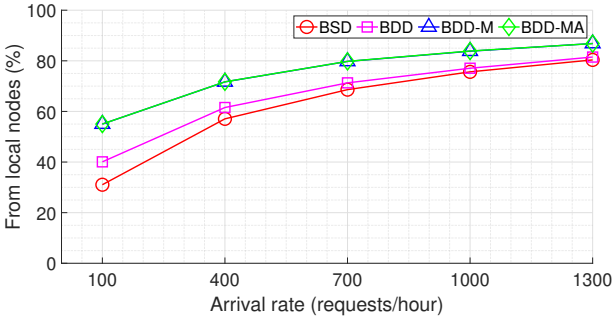

Fig. 9. Percentage of requests served by local nodes.

We now study how the multicast functionality in BDD-M and BDD-MA can offload computing and storage tasks from remote DCs, serve incoming requests from local nodes, and reduce reserved network bandwidth. In Fig. 9, we report the percentage of incoming requests which are served from

local nodes (over the total number of admitted requests). It is worth mentioning that an incoming request can be served locally if the XR live stream is available in the closest edge DC or can be made available in closest edge DCs from local nodes (i.e., not being the root node using multicast). As expected, BDD-M and BDD-MA with their multicast functionality serve most incoming requests from local nodes. To elaborate, at the arrival rate of 100 requests/hour, while BSD and BDD serve 31% and 40% of incoming requests from local edge DCs, BDD-M and BDD-MA can serve up to 55% of incoming requests from local edge DCs. At a higher arrival rate, e.g., 1300 requests/hour, since the probability of the requested XR live stream being available in the closest edge DC is higher, the multicast functionality is less frequently utilized. Consequently, the multicast functionality in BDD-M and BDD-MA is less efficient. However, on average, BDD-M and BDD-MA with their multicast functionality can still serve about 10% more incoming requests from local edge DCs compared to BSD and BDD.

V. Conclusion

In this work, we proposed a reliable service-provisioning scheme, named backup from different data centers with multicast and adaptive bandwidth provisioning, for low-latency and high-bandwidth extended reality live streams. We considered backup from different data centers to provide protection not only against link failures in the optical network but also against computing and storage failures in data centers. We investigated the multicast functionality in the optical layer and the unique trade-off between latency and bandwidth for extended reality live streams to save network resources while ensuring the strict requirements on reliability, latency, bandwidth, and thus user experience. We developed scalable algorithms for the proposed service-provisioning scheme and evaluated their performances on various complex network instances in a dynamic environment. Numerical data show that, compared to conventional service-provisioning schemes such as backup from the same data center, our proposed service-provisioning scheme efficiently utilizes network resources such that it provides services to more users, requires less network bandwidth, reserves less computing and storage resources, and serves most of the incoming requests from local edge data centers; therefore, it is very suitable for extended reality live streams.

Acknowledgments

We gratefully acknowledge the editor and anonymous reviewers for their constructive criticism which improved the paper. This work is supported by the NSF, Japan-US Network Opportunity (JUNO2/3 - Grant Nos. 1818972 and 2210384).

References

[1] Qualcomm, "Augmented and virtual reality: The first wave of 5G killer apps," *White Paper*, 2017.

[2] J. Park, S. Samarakoon, H. Shiri, and M. K. Abdel-Aziz, "Extreme ultra-reliable and low-latency communication," *Nature Electronics*, vol. 5, no. 3, 2022.

[3] L. Kontothanassis, R. Sitaraman, J. Wein, D. Hong, R. Kleinberg, B. Mancuso, D. Shaw, and D. Stodolsky, "A transport layer for live streaming in a content delivery network," *Proc. of the IEEE*, vol. 92, no. 9, 2004.

[4] G. Le, S. Ferdousi, A. Marotta, S. Xu, Y. Hirota, Y. Awaji, M. Tornatore, and B. Mukherjee, "Survivable virtual network mapping with content connectivity against multiple link failures in optical metro networks," *IEEE J. Opt. Commun. Netw.*, vol. 12, no. 11, 2020.

[5] J. Zhang, K. Zhu, and B. Mukherjee, "Backup reprovisioning to remedy the effect of multiple link failures in WDM mesh networks," *IEEE J. Sel. Areas Commun.*, vol. 24, no. 8, 2006.

[6] P. Gill, N. Jain, and N. Nagappan, "Understanding network failures in data centers: Measurement, analysis, and implications," in *Proc. of the ACM SIGCOMM Conference*, 2011.

[7] M. Tornatore, "The challenges of end-to-end network resilience," in *Proc. of the European Conference on Optical Communications*, 2021.

[8] B. Mukherjee, I. Tomkos, M. Tornatore, P. Winzer, and Y. Zhao, *Handbook of Optical Networks*. Springer, 2020.

[9] G. Le, S. Ferdousi, A. Marotta, S. Xu, Y. Hirota, Y. Awaji, S. Savas, M. Tornatore, and B. Mukherjee, "Reliable provisioning with degraded service using multipath routing from multiple data centers in optical metro networks," *IEEE Trans. Netw. Service Manag.*, vol. 20, no. 3, 2023.

[10] M. Schiano, A. Percelsi, and M. Quagliotti, "Flexible node architectures for metro networks," *IEEE J. Opt. Commun. Netw.*, vol. 7, no. 12, 2015.

[11] M. Nooruzzaman and E. Halima, "Low-cost hybrid ROADM architectures for scalable C/DWDM metro networks," *IEEE Commun. Mag.*, vol. 54, no. 8, 2016.

[12] Qualcomm, "VR and AR pushing connectivity limits," *White Paper*, 2018.

[13] M. Zink, R. Sitaraman, and K. Nahrstedt, "Scalable 360° video stream delivery: Challenges, solutions, and opportunities," *Proc. of the IEEE*, vol. 107, no. 4, 2019.

[14] A. Yaqoob, T. Bi, and G.-M. Muntean, "A survey on adaptive 360° video streaming: Solutions, challenges and opportunities," *IEEE Commun. Surveys Tuts.*, vol. 22, no. 4, 2020.

[15] N. Singhal and B. Mukherjee, "Protecting multicast sessions in WDM optical mesh networks," *J. Lightw. Technol.*, vol. 21, no. 4, 2003.

[16] H. Luo, L. Li, H. Yu, and S. Wang, "Achieving shared protection for dynamic multicast sessions in survivable mesh WDM networks," *IEEE J. Sel. Areas Commun.*, vol. 25, no. 9, 2007.

[17] L. Long and A. E. Kamal, "Protecting multicast services in optical internet backbones," *Computer Networks*, vol. 57, no. 1, 2013.

[18] A. Cai, J. Guo, R. Lin, G. Shen, and M. Zukerman, "Multicast routing and distance-adaptive spectrum allocation in elastic optical networks with shared protection," *J. Lightw. Technol.*, vol. 34, no. 17, 2016.

[19] X. Corbillon, G. Simon, A. Devlic, and J. Chakareski, "Viewport-adaptive navigable 360-degree video delivery," in *Proc. of the IEEE International Conference on Communications*, 2017.

[20] A. T. Nasrabadi, A. Mahzari, J. D. Beshay, and R. Prakash, "Adaptive 360-degree video streaming using scalable video coding," in *Proc. of the ACM International Conference on Multimedia*, 2017.

[21] J. V. d. Hooft, M. T. Vega, S. Petrangeli, T. Wauters, and F. D. Turck, "Tile-based adaptive streaming for virtual reality video," *ACM Trans. Multimedia Comput. Commun. Appl.*, vol. 15, no. 4, 2019.

[22] Metro-Haul, "Functional architecture specifications and functional definition," *Deliverable D 2.2*, 2018.

[23] E. W. Dijkstra, "A note on two problems in connexion with graphs," *Numerische Mathematik*, vol. 1, no. 1, 1959.

[24] R. Bhandari, "Optimal physical diversity algorithms and survivable networks," in *Proc. of the IEEE Symposium on Computer and Communications*, 1997.

[25] E. Modiano and A. Narula-Tam, "Survivable lightpath routing: a new approach to the design of WDM-based networks," *IEEE J. Sel. Areas Commun.*, vol. 20, no. 4, 2002.

[26] T. Tachibana, Y. Hirota, K. Suzuki, T. Tsuritani, and H. Hasegawa, "Metropolitan area network model design using regional railways information for beyond 5G research," *IEICE Trans. Commun.*, vol. E106-B, no. 4, 2023.

[27] S. Arakawa, T. Sakano, Y. Tsukishima, H. Hasegawa, T. Tsuritani, Y. Hirota, and H. Tode, "Topological characteristic of Japan photonic network model," *IEICE Technical Report*, 2013.

[28] K. Sripanidkulchai, B. Maggs, and H. Zhang, "An analysis of live streaming workloads on the Internet," in *Proc. of the ACM SIGCOMM Internet Measurement Conference*, 2004.

[29] Z. Duanmu, A. Rehman, and Z. Wang, "A quality-of-experience database for adaptive video streaming," *IEEE Trans. Broadcast.*, vol. 64, no. 2, 2018.