PolyVerse: An Edge Computing-Empowered Metaverse with Physical-to-Virtual Projection

Yinfeng Cao, Jiannong Cao, *Fellow, IEEE,* Dongbin Bai, Zhiyuan Hu, Kaile Wang, Mingjin Zhang*
Department of Computing, The Hong Kong Polytechnic University

***Abstract*—Incorporating physical-world objects and information into the virtual world is a key aspect of enhancing immersiveness in the metaverse. However, existing solutions are neither efficient nor scalable when projecting enormous objects due to their reliance on expensive specialized equipment. To fill this gap, we propose PolyVerse, an edge computing-powered metaverse platform that supports practical Physical-to-Virtual (P2V) projection while preserving low costs. PolyVerse tackles two challenges in P2V. First, P2V inherently requires real-time projection of enormous physical objects, which is difficult to achieve without specialized equipment. To overcome this issue, we alternatively utilize edge computing-based methods to collect, process, and analyze large-scale physical-world data with cost-effective edge devices and AI models. Workloads are adaptively scheduled among edge devices through a heuristic algorithm to reduce latency. Second, metaverse users may encounter inconsistent states among multiple metaverse service providers, which can be caused by potential networking issues or malicious state tampering when projecting vast objects. To this end, we develop a blockchain-based metaverse management scheme among service providers to ensure a consistent view for users. States are initially constructed as a Metaverse State Tree that supports efficient accumulation, retrieval, and membership proof generation. The tree digests are further secured by blockchain consensus to ensure consistency. Finally, we develop a metaverse campus prototype where real-world pedestrians are projected into the virtual world in real-time. Evaluation shows that PolyVerse can project objects within an average latency of 250ms.**

***Index Terms*—Metaverse, Immersive Experience, Edge Computing, Blockchain**

I. Introduction

The metaverse is an interconnected platform enabling users to interact with a shared digital world, thus facilitating an immersive and engaging experience that merges the physical and virtual realms. This concept has attracted significant interest from industry and academia in recent years, resulting in the development of numerous metaverse platforms catering to various purposes such as gaming, social media, online collaboration, and dynamics simulation.

The Problem and Motivations. The core value of the metaverse lies in its immersiveness, allowing users to interact with virtual objects while experiencing a heightened sense of presence, engagement, and authenticity. To this end, existing metaverse platforms attempt to provide users with immersive experiences from different perspectives, such as finely-rendered building models and customizable avatar tools. Among them, **Physical-to-Virtual Projection (P2V)** is an important method for enhancing immersiveness, which refers to incorporating physical-world objects and information into the virtual world [1], as shown in Fig.1. P2V connects users with the physical world, allowing them to experience external activities without direct physical participation. For instance, some education metaverse platforms aim to offer normal university life experiences for disabled students. By including real-time scenes and activities occurring on campus, these students gain a greater sense of participation, even if they cannot attend them physically [2].

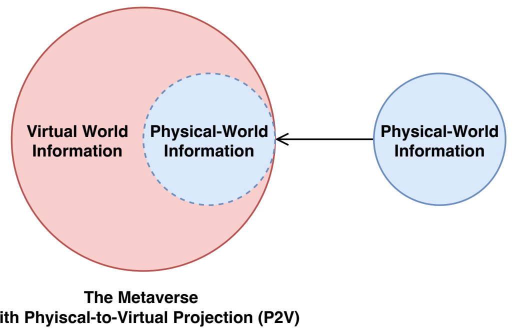

Virtual World Information
Physical-World Information
Physical-World Information
The Metaverse
with Phyiscal-to-Virtual Projection (P2V)

Fig. 1: A conceptional illustration of Physical-to-Virtual projection in metaverse. Physical-world information can significantly enhance the immersive experience of the metaverse

However, practical P2V solutions that offer significant efficiency and scalability are still lacking. Specifically, existing metaverse platforms either provide only user-generated 3D models [2] to simulate the real-world environments, or rely on expensive specialized Virtual Reality (VR) systems to incorporate physical objects into virtual worlds [3] [4]. These solutions are neither efficient nor scalable because most physical objects are vast, dynamic, and scattered in practice. For example, when there are a large number of pedestrians in a physical scene, it is infeasible to collect their attributes using VR equipment. As a result, most metaverse platforms cannot provide P2V, leading to reduced immersiveness and limited application scenarios.

Research Challenges. Enabling practical P2V in the metaverse poses several challenges. On the one hand, it implicitly requires real-time collection, processing, and analysis of large volumes of physical data to achieve immersive experiences [5]. These tasks demand significant computation, storage, and networking resources, which are difficult to obtain without high-end devices. On the other hand, state inconsistency may arise when projecting large volumes of physical objects [6] [7] [8]. Specifically, networking issues, such as packet dropping and transmission delays caused by high network bandwidth requirements for real-time projection, and malicious state

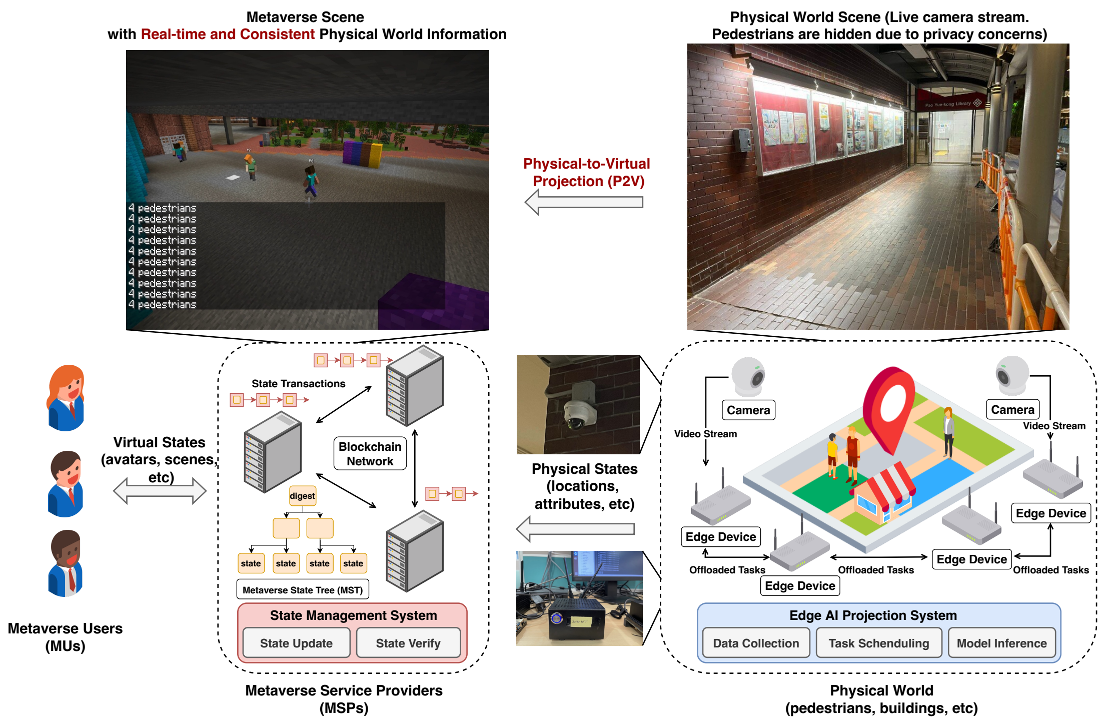


Fig. 2: PolyVerse Overview: the first edge computing-empowered metaverse platform that facilitates Physical-to-Virtual projection (P2V), providing immersive experiences. We deploy sensors (e.g., cameras) and edge devices to collect, process, and analyze physical world information with AI models in real-time and represent it in the virtual world. Besides, a blockchain-based state management system is also integrated to ensure the state consistency among MSPs.

modifications from Metaverse Service Providers (MSPs), may lead to inconsistent views of metaverse states among different users, which would negatively impact their experience. As far as we know, existing works and platforms fail to adequately address these issues while missing detailed solutions [9] [10].

Our Approach and Contributions. In this paper, we introduce PolyVerse, an edge computing-powered metaverse platform that facilitates practical Physical-to-Virtual (P2V) object projection. PolyVerse addresses the real-time and consistency challenges by incorporating two key techniques. First, we deploy an edge computing cluster in the physical world to collect, process, and analyze sensor data using AI models, enabling low-latency physical state updates in a cost-effective manner. Within the cluster, workloads are scheduled through a heuristic algorithm to optimize latency on resource-constrained edge devices. Second, we develop a blockchain-based metaverse management scheme to ensure consistency among multiple MSPs. States are initially constructed as a Metaverse State Tree (MST), which supports efficient state accumulation, retrieval, and membership proof generation. Then, the digests of the MST are uploaded to a blockchain network, collaboratively maintained by MSPs. Furthermore, we demonstrate that PolyVerse is extensible and can be easily adapted for projecting various physical objects into the metaverse, encompassing heterogeneous information sources such as sound and voice.

In summary, our contributions are listed as follows:

- **Metaverse with Real-time Edge AI Projection**. To the best of our knowledge, PolyVerse is the first metaverse platform that incorporates physical world information in an efficient and scalable manner to enhance immersiveness. To achieve this, we employ edge computing clusters with integrated AI models to support low-latency data collection and processing for enormous physical objects. We also propose a task scheduling algorithm to allocate workloads among edge devices and sensors, thereby improving overall efficiency. Additionally, PolyVerse can be easily extended to project other heterogeneous physical information.
- **Blockchain-based Metaverse State Management**. To ensure a consistent view of metaverse states even under potential malicious MSPs or networking issues, we propose a blockchain-based state management scheme. Specifically, we develop a new data structure named MST to organize states, enabling efficient accumulation, retrieval, and generation of membership proofs. The states are further secured by a decentralized Byzantine Fault Tolerant (BFT) consensus mechanism, which makes them resistant to malicious tampering and service unavailability.
- **Prototype and Evaluation**. We implement PolyVerse through a metaverse campus prototype. We demonstrate that this prototype can and efficiently project campus pedestrian states of physical-world pedestrians., e.g., locations and attributes, into the metaverse in real-time with low-end edge devices.

II. PolyVerse Design Goals

PolyVerse aims to offer efficient and scalable P2V ability to achieve high immersiveness in metaverse. Real-time projection with consistent state update are fundamental requirements to achieve such immersiveness.

Real-time Projection. It refers to the continuous and instantaneous mapping, representation, and rendering of virtual objects (e.g, avatars) into a virtual environment according to physical objects (e.g, pedestrians). This allows users to experience and interact with the dynamic external activities in virtual world. For example, if an avatar can not follow a real-world person's movement in real-time, we can hardly say such projection is immersive due to the unexpected hysteresis.

Definition 1 Let V be the virtual environment in the metaverse, and $VO = \{vo_1, vo_2, ..., vo_n\}$ be a set of virtual object states, each element in VO corresponding to a physical object in set $P = \{po_1, po_2, ..., po_n\}$. A real-time projection in metaverse is a function $f : P \times T \to V$, where T represents time. For a given object state po_i and time instance $t \in T$, $f(po_i, t)$ projects the object state po_i into the virtual environment V at the time instance t, and update corresponding vo_i. The projection f is considered real-time if the time period Δ required for updating vo_i from $f(po_i, t_1)$ to $f(po_i, t_2) < \epsilon$, where ϵ is a given threshold.

State Consistency It is essential for MUs to have a consistent view of virtual world states, even under unstable or adversarial environments. In case of inconsistency, honest MUs should be able to actively detect it through cryptographic proofs. Inconsistency situations are likely to occur when implementing real-time projection. For instance, when a group of MSPs needs to obtain physical states from real-time data sources, some states may be occasionally ignored by certain MSPs due to high network traffic. This leads to metaverse users (MUs) having inconsistent views. Similarly, some MSPs might be malicious, motivated by the high-value items inside the metaverse platform, such as digital assets, and may attempt to tamper with some states.

Definition 2 Let M be a set of MSPs which jointly host V by maintaining VO and PO. V is considered consistent under adversarial environments if the following conditions are satisfied:

- If two honest MSPs m and m' project a po_i as vo_i and $vo_i^{'}$ using f, then $vo_i = vo_i^{'}$.
- For any malicious MSP $m \in M$ that does not honestly update the virtual object states, there exists a cryptographic proof π such that any honest MUs can actively detect the malicious behavior using π.

III. PolyVerse Overview

In this section, we provide a brief overview of PolyVerse, focusing on its system model and two essential modules. The architecture of PolyVerse is shown in Fig. 2,

System model. The system model of PolyVerse consists of the following entities:

- *Physical Objects*: These refer to the objects from the physical world that are to be projected into the metaverse. Distinct from MUs, these objects do not actively participate in the metaverse (e.g., pedestrians and buildings). Their primary purpose is to provide physical-world information to enhance immersiveness.
- *Edge Computing Cluster (ECC)*: It gathers information from the physical world and transforms it into physical states. These states are then transmitted to MSPs for further processing.
- *Metaverse Service Providers (MSPs)*: MSPs offer metaverse services by facilitating network communication and managing metaverse states. They are responsible for receiving physical states from the ECC and calculating the corresponding virtual states based on specific mapping rules. The states managed by multiple MSPs should be identical.
- *Metaverse Users (MUs)*: MUs access the metaverse by connecting to MSPs through their client software. They obtain the virtual states VO from the MSPs and render them locally using their client software.

To facilitate P2V within this system model, we develop two techniques:

Real-time projection with edge AI. We utilize edge computing-based methods for executing projection tasks to enable real-time P2V. Specifically, sensors are deployed to collect physical data, and nearby edge devices process and analyze this data using AI models to obtain physical states. For example, cameras are used to detect pedestrian physical locations with object detection models and send their locations to MSPs in our prototype. However, the inference of AI models is resource-greedy and computation-intensive, while the resources of individual edge devices are constrained. When faced with heavy workloads, such as a large number of pedestrians entering into the camera view, the models will suffer from significant inference delays. To address this issue, we design an efficient task scheduling algorithm that enables resource sharing among multiple edge devices and distributes tasks among these devices. inference tasks are accelerated reasonably, and the real-time P2V requirement is met successfully. The detailed design of the projection procedures is shown in Section IV.

Blockchain-based metaverse state management. In PolyVerse, states are managed by MSPs who jointly maintain a consortium blockchain network to ensure state consistency. Physical states from the edge computing cluster and corresponding virtual states are submitted to the blockchain network in the form of transactions for consensus. Consequently, MUs can verify the consistency of the virtual world view among MSPs. However, the high latency and low throughput natures of the consensus algorithm cannot meet real-time state update requirements. To address this issue, we develop a new authenticated data structure (ADS) called Metaverse State Tree, which is optimized for rapid state updates and verification. The MST recursively accumulates states and outputs a digest, allowing MUs to verify if the states are correctly included. States are first constructed as MSTs upon their arrival at MSPs. When a state verification is triggered by MUs, or a specific time/size epoch has passed, the digest and states are then submitted to the blockchain network. This optimization allows MSPs to add states without waiting for blockchain consensus while still guaranteeing the consistency, thus improving the overall performance of PolyVerse. The details of state update system is shown in Section V.

IV. Real-time Projection with Edge AI

We first formulate the real-time projection as a task scheduling problem with constrained resources and then propose a heuristic algorithm, named *PolyHeuristic*, to solve it. For clarity, we use the projection of physical-world pedestrians as an illustrative example. This technique can be generalized and extended to project other physical objects by simply deploying other types of sensors and AI models on edge devices.

A. Network Model

PolyVerse constructs an edge network to connect edge devices and cameras. The network is modeled as a graph $G = (V, E)$, where $V = i|1 \leq i \leq M$ is the set of edge devices and $E = l_{ij}|i, j \in V$ is the set of network links. Suppose there are K cameras, and each camera can stream the video to any edge device in the network. We use PS_i to represent the computation capacity of device i. Each device also has a limited resource of R^i_{max}, and the available resource is indicated by R^i_{avail}. Let e_{ij} represent the bandwidth between devices i and j. The devices and network links can be heterogeneous in computation and bandwidth capacities. The transmission rate between edge devices i and j is R_{ij}. The video transmission latency between camera k and edge device i is T_{ik}.

B. Application Model

Without loss of generality, we assume the application as AI model based personal attributes recognition, which aims to recognize attributes of a person, such as gender, clothing color, and hairstyle. This information will be updated in the metaverse to render the virtual objects. The application model for personal attribute recognition is considered as sequence of two dependent tasks, i.e., pedestrian detection and attribute recognition. The computation workload for detection and attribute recognition tasks corresponding to camera stream k is T^k_d and T^k_r, respectively. The dependent data between the two tasks is $D^k_{d,r}$. Resource requests of the detection model and attribute recognition model are R^{det}_{req} and R^{reg}_{req}, respectively.

C. Problem Formulation

There are two decision variables for mapping camera streams and scheduling DL tasks. The first decision variable x_{ik} is binary, which equals 1 if the camera video stream k is scheduled to device i. It also indicates that the detection task is deployed on device i. Another decision variable y_{ik} is binary, which equals 1 if the attribute recognition task corresponding to camera video stream k is scheduled to device i. For each camera stream, the detection and attribute recognition tasks can be locally executed or offloaded to another device. The time for executing the models on each device depends on the overall workload and the computation capacity of the device.

The total resource request on device i, notated as R_{req}, can be calculated as follows:

$$R_{req} = \sum_{k=1}^{K} (x_{ik} \cdot T^k_d + y_{ik} \cdot T^k_r) \tag{1}$$

The overall processing time for camera stream k, i.e., L_k, can be calculated as:

$$\begin{aligned} L_k = & \sum_{i=1}^{M} x_{ik} \cdot T_{ik} + \sum_{i=1}^{M} x_{ik} \cdot \frac{T^k_d}{PS_i} + \sum_{j=1}^{M} y_{jk} \cdot \frac{T^k_r}{PS_j} \\ & + \sum_{i=1}^{M} \sum_{j=1}^{M} x_{ik} \cdot y_{jk} \cdot \frac{D^k_{d,r}}{R_{i,j}} \cdot \sigma(i-j) \end{aligned} \tag{2}$$

where $\sigma(\cdot)$ is an indicator function. When $\cdot$ is zero, $\sigma(\cdot)$ equals to 1, otherwise $\sigma(\cdot)$ equals to 0.

The objective function is to minimize the sum of completion time for all applications from camera streams.

$$\min_{x_{ik}, y_{ik}} \sum_{k=1}^{K} L_k \tag{3}$$

$$R_{req} \leq R^i_{max}, \quad \forall i \in V \tag{4}$$

$$\sum_{i=1}^{M} x_{ik} = 1, \quad \forall j \in V, \quad k \in \{1, 2, \cdots, K\} \tag{5}$$

$$\sum_{j=1}^{M} y_{ik} = 1, \quad \forall i \in V, \quad k \in \{1, 2, \cdots, K\} \tag{6}$$

$$x_{ik}, y_{ik} \in \{0, 1\}, \quad \forall i, j \in V, \quad k \in \{1, 2, \cdots, K\} \tag{7}$$

Eq. 4 indicates that the resource request on an edge device cannot exceed its maximum resource. Eq. 5 and Eq. 6 show that the detection model and the attribute recognition model of a stream can only be deployed on one edge device.

D. Proposed Algorithm

The formulated optimization problem is a nonlinear integer programming problem, which can be reduced to a generalized assignment problem, proven to be NP-hard in literature. Consequently, we propose a heuristic algorithm (PolyHeuristic) to solve the problem. The algorithm is developed with two fundamental principles:

- *High workload first.* Generally, handling video streams with high workloads results in much larger latency than those with moderate workloads. If we allocate edge resources to video streams with moderate workloads first, high workload streams may suffer from prolonged latency when the remaining edge resources are inadequate.
- *Reusing attribute recognition model.* In the person attribute recognition application, the attribute recognition

model should not be frequently invoked because an individual's attributes do not change within a short period and a person will be continuously tracked once their attributes are determined. Hence, we consider the dependent deployment of the two models. When available resources are constrained, the attribute recognition model can be reused to handle multiple video streams, adapting to constrained edge resources.

Based on these principles, we solve the problem in two stages, as shown in Algo. 1. The first stage schedules the video stream and the detection task, while the second stage schedules the attribute recognition task. We first determine the priority of stream scheduling by sorting the video streams according to their workloads. Next, we filter candidate edge devices with abundant resources for the detection model. The video stream is allocated to the edge device with the minimum execution time, considering both the raw video transmission time and the inference time of the detection model.

$$\min_i (x_{ik} \cdot \frac{D^k_{d,r}}{R_{i,j}} + \frac{T_{r,k}}{PS_i}), \quad i \in M_{deployed}, k \in V_{rest} \tag{8}$$

After determining x_{ik}, we then allocate the attribute recognition models in stage 2. We use a similar greedy approach to allocate these models. To determine where to deploy the attribute recognition model, we allocate the models for streams with high workloads. When the available resources cannot support the deployment of new attribute recognition models, we reuse the existing models. The rest of the streams V_{rest} will be allocated to the edge device, which satisfies Eq. 8. By solving Eq. 8, the rest of the streams will be scheduled to the deployed attribute recognition model that can provide the least intermediate data transmission and inference time.

Algorithm 1: PolyVerse Joint Stream and Task Scheduling Heuristic (PolyHeuristic)

```
Input: Video stream $V$, computation capacity $\{PS_i\}_{i=1}^M$ and available resource $\{R^j_{avail}\}_{j=1}^M$
Output: Video and task allocation policy $x_{ik}$ and $y_{ik}$
1  Create stream priority $I$ in descending workload;
2  for $t \leftarrow 1$ to $I$ do // Stage 1
3      for each device $i \leftarrow 1$ to $M$ do
4          if $R^j_{avail} > R^{det}_{req}$ then
5              Calculate $t^i_{exec} = T_{ik} + \frac{T_{d,k}}{PS_i}$;
6          end
7      end
8      Select device $i^*$ with the shortest execution time $i^* = min_i\{t^j_{exec}\}$;
9      $x_{i^*,k} \leftarrow 1$, update $R^{i^*}_{avail}$ for device $i^*$;
10 end
11 for $t \leftarrow 1$ to $I$ do // Stage 2
12     for each device $i \leftarrow 1$ to $M$ do
13         if $R^j_{avail} > R^{reg}_{req}$ then
14             Calculate $t^i_{exec} = x_{ik} \cdot \frac{D^k_{d,r}}{R_{i,j}} + \frac{T_{r,k}}{PS_i}$;
15         end
16     end
17     Select device $i^*$ with the shortest execution time $i^* = min_i\{t^j_{exec}\}$;
18     Add $i$ to candidate list $M_{deployed}$;
19     if $V_{rest} \neq \emptyset$ then
20         Schedule remaining streams by solving Eq. 8;
21     end
22     $y_{i^*,k} \leftarrow 1$, update $R^{i^*}_{avail}$ for device $i^*$;
23 end
24 return $x_{ik}$, $y_{ik}$
```

V. Blockchain-based Metaverse State Management

This section presents the blockchain-based metaverse state management scheme in PolyVerse. We first review the used data models with involved entities, and then we discuss the threat model. At last, we give the details of state update and verification operations that jointly ensure the state consistency in PolyVerse.

A. Data Model

We define the following data model in PolyVerse:

- *State* that contains physical states PO describing the information of physical objects, and corresponding virtual states VO calculated by particular mapping rules MR. A sid is associated with each po by ECC, which indexes the states.
- *Mapping rule* that indicates the relationship between VO and PO, which also aligns the virtual world and physical world. For example, the coordinate correspondence between a physical scene and a virtual scene. Because of the uniqueness of such relationship, a mapping rule can also somehow represent a fix scene in the PolyVerse. Mapping rules are coded as binary strings.
- *Auxiliary data.* aux includes all supplement data for the states, such as timestamps, MSP identity information, etc.

B. Threat Model

We assume that MSPs could be malicious (BFT adversarial) in PolyVerse. Specifically, when states are posted by ECC, MSPs attempt to synchronize them with each other through broadcasting. Their goal is to provide MUs with a consistent view of metaverse states. Malicious MSPs may try to modify the states, deliberately remain silent to prevent other honest MSPs from obtaining complete states, or engage in other arbitrary malicious behaviors to obtain illegal benefits. Given this scenario, we require that at least 2/3 of the MSPs are honest, which is a common security boundary for blockchain consensus [11]. We also assume that the MSPs possess limited probabilistic polynomial-time (PPT) computational power, which prevents them from breaking secure hash functions or signature schemes.

C. Metaverse State Tree

MST is a Merkle Tree-like ADS that accumulates states in PolyVerse [12] [13]. As shown in Fig. 3, we define three type of nodes in MST:

Leaf nodes contain state and auxiliary data (po, vo, aux) at depth D (tree depth). F_{leaf} and F_{empty} are flag constants to distinguish leaf nodes and empty nodes.

$$N_{leaf} = Hash(index_i || D || (po_i, vo_i, aux) || F_{leaf})$$

The $index$ is calculated by concatenating mapping rule code MR as prefix, and the sid of its po. For example, when a MR is coded as 00 and with sid is 1, the $index$ is:

$$index_i = \underbrace{00}_{MR} || \underbrace{1}_{sid}$$

Empty nodes do not contain state or auxiliary data, but only contain MR as prefix at at depth D

$$N_{empty} = Hash(MR || D || F_{empty})$$

Interior nodes are computed by two child nodes with index 0 and 1. The top interior node is the root node, which accumulates the entire tree content into a digest.

$$N_{interior} = Hash(child_0 || child_1)$$

The MST serves for three main purposes: 1) generating states' membership proofs for MUs to verify if states are correctly included by MSPs. 2) improving the throughput of receiving states without awaiting them to be confirmed by blockchain consensus. 3) providing high states retrieval efficiency by indexing states' MR.

D. State Operation

To address the issue of potential inconsistency among MSPs, we leverage the BFT capability of blockchain consensus. By submitting the MST root hash (digest) to the blockchain network, divergent views created by MSPs within the security boundary are prevented through verifiable digest. In the followings, we define two operations built on the MST and blockchain network. These operations are executed by MSPs and MUs, respectively. Then we introduce optimization techniques to enhance throughput and reduce the latency of the two operations.

State Update is responsible for receiving states from edge computing clusters, calculating MST, and commit the MST digest into blockchain networks, which is run by MSPs. Specifically, it contains following steps:

- `putPO(ECC)` $\rightarrow$ `id, PO, aux`: This function allows a MSP to obtain real-time physical states PO with auxiliary data aux from the ECC (for simplicity, we assume there is a specific edge device within the ECC that provides the APIs for obtaining PO). In our prototype, PO is a stream containing pedestrian ID, location coordinates, and other attributes.
- `calculateVO(PO,MR)` $\rightarrow$ `VO`: This function allows an MSP to calculate VO based on PO and a mapping rule MR. As mentioned before, MR maps and aligns the PO to VO in the metaverse.
- `generateADS(PO,VO,sid,aux,N)` $\rightarrow$ `digest`: This function allows an MSP to construct state tuples (VO, PO, aux) and sid as an MST and output its digest. N is the maximum size of state tuples that an MST can accumulate. D and F can be chosen offline by MSPs.
- `submitTX(PO,VO,aux,digest)` $\rightarrow$ `receipt`: This function allows an MSP to submit transactions that contain multiple (VO, PO, aux) sets and their $digest$, and returns the receipts of these transactions, such as the included block height.

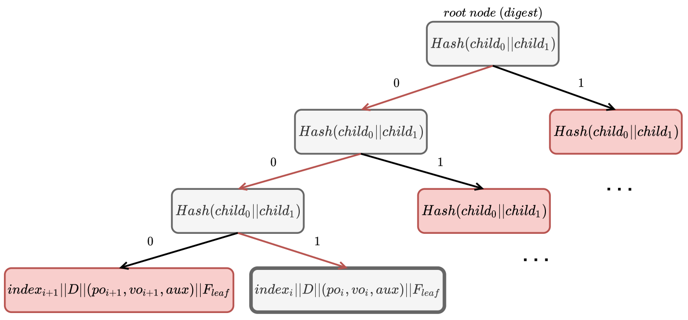
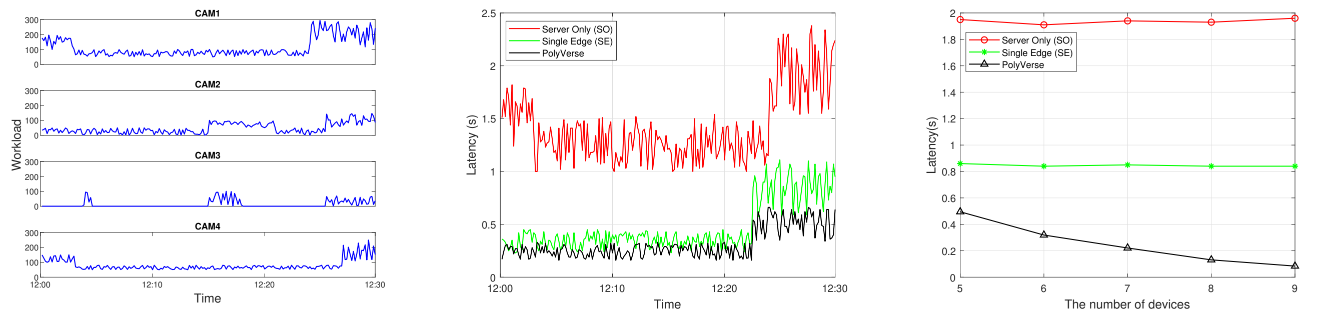


Fig. 3: Metaverse State Tree (MST) in PolyVerse: the $index_i$ of states (po_i, vo_i, aux) is 001, whose membership proofs are the red leaf nodes. Following the red arrow, MSP can efficiently retrieve the states.

State Verify is performed by MUs to request states and verify the consistency of them among multiple MSPs.

- `getVO(MSP)` $\rightarrow$ `VO`: MUs retrieve VO from an MSP. Client software used by MUs renders these VO into virtual objects.
- `reqVerify(VO)` $\rightarrow$ `(digest, proofs, receipt)`: MUs request the membership proofs and the blockchain receipt from MSPs for further local verification. The proofs are the authentication path in the MST. As shown in Fig. 3, the red leaf nodes jointly form a path to the root node.
- `verifyVO(VO, digest, proofs, receipt)` $\rightarrow$ `bool`: MUs verify if the VO is consistent among MSPs by checking if it satisfies the following conditions: (1) VO and $proofs$ can form the identical $digest$. (2) $receipt$ is valid for the blockchain network, i.e., it shows that the transaction is confirmed.

Throughput Optimization. In the aforementioned operations, if `submitTX` is triggered for each state update, the throughput would be limited by the blockchain consensus speed. New state updates would have to wait for previous pending updates to be confirmed. To address this issue, we implement a lazy trigger rule. Specifically, states constructed as MST can be accessed by MUs before the digest is confirmed. This means MUs can obtain states (`getVO`) as soon as MSPs complete the virtual state calculation (`calculateVO`) and output the ADS (`generateADS`), whose performance



(a) Workload dynamics (b) Latency vs. dynamic workload (c) Latency vs. number of devices

Fig. 4: (a) Workload dynamics of four cameras in real-world deployment. We can observe that the workload is not evenly distributed among different cameras. (b) In the case of a heavy workload, PolyVerse achieves much lower latency. (c) PolyVerse achieve better performance with increasing edge devices.

is determined by hardware rather than blockchain consensus algorithms. Consequently, `submitTX` can be triggered after reaching a maximum number of unconfirmed states, or a state verification request from MUs (`verifyVO`) that verifies certain unconfirmed states. Although this approach may occasionally return stale states, these issues will be resolved within a defined boundary due to the trigger rule outlined in the blockchain consensus [14]. In other words, stale states will not grow indefinitely, as there will be a maximal threshold or a limit on verification requests, depending on which is reached earlier.

VI. Implementation and Evaluation

A. Prototype

We develop metaverse campus prototype to demonstrate PolyVerse, where pedestrians are projected in real-time. We implement it with a workstation, 5 edge devices, and 9 cameras. We use 3 Jetson Xavier NX (384-core Volta GPU with 8GB RAM) and 2 Jetson TX2 (256-core GPU with 8GB RAM) as the edge devices. A workstation with an Intel Core i9-11900K and 64GB RAM is used to to run a Minecraft server for hosting a virtual campus scene [15].

Human attribute recognition. We use MobileNet-V2 as the backbone network for pedestrian detection. MobileNet-V2 is a lightweight deep convolutional network suitable for resource-constrained edge devices. A Kalman filter-based tracker is used to track the pedestrians. For the attribute recognition model, we use ResNet-50 as the backbone network. The resolution of the video is 1920x1080 with 30fps.

Blockchain network. We utilize Ethereum in a private network setting through Geth client, which employs a Proof of Authority (POA) consensus [16].

B. Experimental Evaluation

Latency is a crucial factor for immersiveness in P2V projection. To showcase the effectiveness of implementing edge computing techniques in PolyVerse, we conduct a comparative analysis of the prototype's end-to-end latency against multiple established baselines, which are similar with settings of existing VR systems.

- *Server Only (SO)*: the video streams are sent to the server, where all the computation tasks are executed.
- *Single Edge (SE)*: the video streams are allocated to the edge devices with the shortest video transmission time. Computation tasks are executed locally without resource sharing among edge devices.

We test the prototype under various workloads with varying numbers of pedestrians to demonstrate the low-latency feature of PolyVerse. Fig. 4(a) shows the workloads of 4 cameras between 12:00 and 12:30 a.m.. The workload of each video stream varies with the number of pedestrians in a video stream and is dynamic as the content captured by each camera changes over time. We can also see that the workloads are different among cameras. In particular, camera 1 and camera 4 have a higher average workload than other two cameras as they are nearer to the library and canteen.

As shown in Fig.4(b), the latency of SE and PolyVerse is much lower than that of SO. This is because they process the application workloads, i.e., model inference of human attribute recognition, on local edge devices, which can respond in real-time. SO requires raw videos to be transmitted and processed on remote server, suffering from unexpected network latency. Moreover, we also observe that PolyVerse outperforms SE, especially when there is a high workload. From 12:05 to 12:25, PolyVerse and SE tend to have similar performance, as the data transmission latency is the main factor that dominates the end-to-end latency. As the workload increases, the inference latency becomes the dominant factor. In this case, PolyVerse shows apparent superiority, as it schedules the workloads among collaborative edge devices to achieve optimal latency, while SE does not enable resource sharing among edge devices. A similar finding is also shown in Fig.4(c), where the overall latency decreases with the increment of the number of edge devices for PolyVerse, demonstrating its ability to efficiently utilize the collaborative edge resources.

VII. Related Works

Recently, several representative industrial metaverse platforms have emerged. Sandbox Games like [17] offer immersive experiences through digital asset trading and avatar interaction in virtual 3D environments. Simulation Platforms like NVIDIA's Omniverse [18] provide real-time 3D simulations and visualizations for industrial applications. Collaboration Tools, such as Meta Horizon Workrooms [19], facilitate productive and collaborative VR experiences for enterprise teams.

Recent research efforts have also explored various aspects of the metaverse. Duan et al. built a blockchain-driven virtual campus and discussed its benefits for social goods [2]. Lam et al. proposed a human-avatar framework with full-body motion capture for metaverse [20]. Dhelim et al. proposed hybrid Fog-Edge computing architectures for metaverse tasks like 3D simulation [21]. Wang et al. designed a framework for Metaverse classrooms to achieve real-time synchronization of a large number of participants through VR equipment and sensors [3]. Shen et al. introduced the cyber-physical-social system (CPSS) paradigm to enhance lecturers' space immersion through sparse consumer-grade RGBD cameras [4]. Cai and Karunarathna et al. discussed networking optimization for efficient metaverse systems [22] [23].

However, existing works do not provide complete and detailed solutions for real-time projection of enormous physical objects into the metaverse in a scalable manner to enhance immersiveness.

VIII. Conclusion

In this paper, we introduce PolyVerse, an edge computing-powered metaverse platform that facilitates Physical-to-Virtual (P2V) object projection to provide an enhanced immersive experience. By leveraging edge computing and blockchain technology, PolyVerse efficiently collects, processes, and analyze sensor data, enabling the projection of physical-world objects into the virtual world while ensuring state consistency.

In the future, we will explore how to implement PolyVerse in full edge computing environments through edge blockchains and distributed rendering to further improve the security and performance [24] [25]. In addition, potential privacy issues such as attribute leakage during P2V processes are also important to discuss and address.

IX. Acknowledgement

This work was supported by the Research Institute for Artificial Intelligence of Things, The Hong Kong Polytechnic University, HK RGC Collaborative Research Fund (CRF) No. C2004-21GF, and Innovation and Technology Fund - Mainland-Hong Kong Joint Funding Scheme (ITF-MHKJFS) No. MHP/013/21.

References

[1] INTERFACING CYBER AND PHYSICAL WORLD WORKING GROUP, "IEEE 2888 standards," https://sagroups.ieee.org/2888/, accessed 2023, accessed on March 9, 2023.

[2] H. Duan, J. Li, S. Fan, Z. Lin, X. Wu, and W. Cai, "Metaverse for social good: A university campus prototype," in *Proceedings of the 29th ACM international conference on multimedia*, 2021, pp. 153–161.

[3] Y. Wang, L.-H. Lee, T. Braud, and P. Hui, "Re-shaping post-covid-19 teaching and learning: A blueprint of virtual-physical blended classrooms in the metaverse era," in *2022 IEEE 42nd International Conference on Distributed Computing Systems Workshops (ICDCSW)*. IEEE, 2022, pp. 241–247.

[4] T. Shen, S.-S. Huang, D. Li, Z. Lu, F.-Y. Wang, and H. Huang, "Virtualclassroom: A lecturer-centered consumer-grade immersive teaching system in cyber–physical–social space," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 2022.

[5] M. Xu, D. Niyato, B. Wright, H. Zhang, J. Kang, Z. Xiong, S. Mao, and Z. Han, "Epvisa: Efficient auction design for real-time physical-virtual synchronization in the metaverse," *arXiv preprint arXiv:2211.06838*, 2022.

[6] Y. Wang, Z. Su, N. Zhang, R. Xing, D. Liu, T. H. Luan, and X. Shen, "A survey on metaverse: Fundamentals, security, and privacy," *IEEE Communications Surveys & Tutorials*, 2022.

[7] C. Gehrmann and M. Gunnarsson, "A digital twin based industrial automation and control system security architecture," *IEEE Transactions on Industrial Informatics*, vol. 16, no. 1, pp. 669–680, 2019.

[8] O. Hashash, C. Chaccour, W. Saad, K. Sakaguchi, and T. Yu, "Towards a decentralized metaverse: Synchronized orchestration of digital twins and sub-metaverses," *arXiv preprint arXiv:2211.14686*, 2022.

[9] M. Xu, W. C. Ng, W. Y. B. Lim, J. Kang, Z. Xiong, D. Niyato, Q. Yang, X. S. Shen, and C. Miao, "A full dive into realizing the edge-enabled metaverse: Visions, enabling technologies, and challenges," *IEEE Communications Surveys & Tutorials*, 2022.

[10] H. Ning, H. Wang, Y. Lin, W. Wang, S. Dhelim, F. Farha, J. Ding, and M. Daneshmand, "A survey on metaverse: the state-of-the-art, technologies, applications, and challenges," *arXiv preprint arXiv:2111.09673*, 2021.

[11] H. Moniz, "The istanbul bft consensus algorithm," *arXiv preprint arXiv:2002.03613*, 2020.

[12] Y. Peng, M. Du, F. Li, R. Cheng, and D. Song, "Falcondb: Blockchain-based collaborative database," in *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data*, 2020, pp. 637–652.

[13] M. S. Melara, A. Blankstein, J. Bonneau, E. W. Felten, and M. J. Freedman, "{CONIKS}: Bringing key transparency to end users," in *24th {USENIX} Security Symposium ({USENIX} Security 15)*, 2015, pp. 383–398.

[14] D. Terry, "Replicated data consistency explained through baseball," *Commun. ACM*, vol. 56, no. 12, p. 82–89, dec 2013. [Online]. Available: https://doi.org/10.1145/2500500

[15] "Minecraft," https://www.minecraft.net, accessed: March 10, 2023.

[16] Ethereum, "go-ethereum," https://github.com/ethereum/go-ethereum, 2023, accessed: March 10, 2023.

[17] "The sandbox," https://www.sandbox.game/en/, accessed: March 10, 2023.

[18] M. Hummel and K. van Kooten, "Leveraging nvidia omniverse for in situ visualization," in *High Performance Computing: ISC High Performance 2019 International Workshops, Frankfurt, Germany, June 16-20, 2019, Revised Selected Papers 34*. Springer, 2019, pp. 634–642.

[19] "Meta horizon workrooms — virtual workroom — work with meta," https://www.meta.com/work/workrooms/?utm_content=95648, Mar 2023, accessed: March 10, 2023.

[20] K. Y. Lam, L. Yang, A. Alhilal, L.-H. Lee, G. Tyson, and P. Hui, "Human-avatar interaction in metaverse: Framework for full-body interaction," in *Proceedings of the 4th ACM International Conference on Multimedia in Asia*, 2022, pp. 1–7.

[21] S. Dhelim, T. Kechadi, L. Chen, N. Aung, H. Ning, and L. Atzori, "Edge-enabled metaverse: The convergence of metaverse and mobile edge computing," *arXiv preprint arXiv:2205.02764*, 2022.

[22] Y. Cai, J. Llorca, A. M. Tulino, and A. F. Molisch, "Compute-and data-intensive networks: The key to the metaverse," in *2022 1st International Conference on 6G Networking (6GNet)*. IEEE, 2022, pp. 1–8.

[23] S. Karunarathna, S. Wijethilaka, P. Ranaweera, K. T. Hemachandra, T. Samarasinghe, and M. Liyanage, "The role of network slicing and edge computing in the metaverse realization," *IEEE Access*, vol. 11, pp. 25 502–25 530, 2023.

[24] Y. CAO, J. CAO, Y. WANG, K. WANG, and X. LIU, "Security in edge blockchains security in edge blockchains: Attacks and countermeasures," *ZTE COMMUNICATIONS*, vol. 20, no. 4, 2022.

[25] S. Jiang, J. Cao, J. Zhu, and Y. Cao, "Polychain: a generic blockchain as a service platform," in *Blockchain and Trustworthy Systems: Third International Conference, BlockSys 2021, Guangzhou, China, August 5–6, 2021, Revised Selected Papers 3*. Springer, 2021, pp. 459–472.