XR Collaboration Architecture based on Decentralized Web

Seungyeon Huh
Korea Institute of Science and Technology
Seoul, Korea
huh@kist.re.kr

Shapna Muralidharan
Korea Institute of Science and Technology
Seoul, Korea
023870@kist.re.kr

Heedong Ko
Korea Institute of Science and Technology
Seoul, Korea
ko@kist.re.kr

Byounghyun Yoo
Korea Institute of Science and Technology
Seoul, Korea
yoo@byoo.net

ABSTRACT

The web has been an extremely effective collaboration platform, enabling services like Wikipedia article co-authoring, blogging, social messaging, video conferencing, and many others. However, the collaboration should ideally occur Peer to Peer (P2P) among the participants instead of going through a centralized server as in the current centralized web, which acts as a mediator as well as a repository of data, especially for face-to-face collaboration in 3D XR context. Most notable XR applications like MMORPG have been developed in a dedicated application platform with their own centralized game servers. Nowadays, the decentralized web is being promoted as the next web architecture in numerous fronts such as blockchain in cryptocurrency, reviving P2P storage, and networking technologies as the next web, Web 3.0. It would be beneficial if we could make an XR collaboration framework based on the recent developments of the decentralized web. This paper explores one possible amalgamation of the decentralized web technology stack toward a webized XR collaboration framework.

CCS CONCEPTS

• **Human-centered computing** → **Mixed / augmented reality**; **Virtual reality**; **Web-based interaction**; **Collaborative interaction**.

KEYWORDS

Webizing, collaboration, decentralized web, cross reality, extended reality, XR

ACM Reference Format:
Seungyeon Huh, Shapna Muralidharan, Heedong Ko, and Byounghyun Yoo. 2019. XR Collaboration Architecture based on Decentralized Web. In *Web3D '19: The 24th International Conference on 3D Web Technology (Web3D '19), July 26–28, 2019, Los Angeles, CA, USA.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3329714.3338137

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
Web3D '19, July 26–28, 2019, Los Angeles, CA, USA
© 2019 Association for Computing Machinery.
ACM ISBN 978-1-4503-6798-1/19/07…$15.00
https://doi.org/10.1145/3329714.3338137

1 INTRODUCTION

XR (Cross Reality or eXtended Reality) encompasses a wide spectrum of the reality continuum including virtual reality (VR), augmented reality (AR), and mixed reality (MR) [Somasegar and Lian 2017; Wikipedia 2019]. XR is needed for the uniform representation of the reality continuum and its run-time framework, which facilitates collaboration in immersive environments that are more suitable for users' work environments and equipment. Asymmetric collaboration, which enables two or more humans from a combination of the real world using AR and a virtual world using VR to share a workspace is a typical example that can be made more efficient by using a uniform representation of XR contents regardless of the user's context, i.e., whether the user is in VR or in AR [Morde 2007; Oda et al. 2015]. The lack of a mechanism that allows the sharing of a single scene with uniform representation across the whole reality continuum restricts the utilization of proper immersion in shared workspaces and is a crucial reason behind why we find only remote AR-based collaboration services such as Vuforia's Chalk AR [Vuforia 2018], Scope AR's Remote AR [Scope AR 2019], MAXST's Vivar [MAXST 2018], and Virnect's Remote AR [Virnect 2019] in the remote maintenance market. The concept of remote AR has been researched fairly enough for one-to-one collaboration during simple tasks. However, one-to-many collaboration is necessary for more general situations such as remote control and the management of complex environments. For instance, a subject-matter expert in a remote place like a control room collects information from multiple local users to make adequate decisions during emergencies, as well as for regular training in large manufacturing facilities or power plants. In this case, the proper type of immersion for each participant of an asymmetric collaboration improves the perception of information from local users, and tasks from the subject-matter expert. An architecture for VR in remote and in AR for local users through a shared workspace is necessary.

Our earlier work [Seo et al. 2018] focused on facilitating a shared workspace for asymmetric collaborators by webizing XR interaction spaces for the synchronization of XR content and interaction events among multiple users. However, it still lacks the necessary collaboration architecture for practical situations which, unlike specific lab environments, do not guarantee communication quality. Most smart factories and future mobility vehicles such as smart ships and military vessels do not provide enough network bandwidth and latency. Furthermore, the connection itself is often lost in the case of drastic situations involving underwater vessels, i.e., submarines. However, ironically, there is a stronger demand for

XR collaborations in environments exposed to harsh situations because of the necessity to reduce the number of crew members by automation.

To overcome the limitations of networking in practical environments, we propose an XR collaboration architecture based on the decentralized web (DWeb). The new architecture enables synchronization among local users and ensures that the application's core functionality still works in the absence of a reliable network connection with the offline first strategy. Eventual consistency of XR content between local and remote sites is secured with graceful degradation and recovery. The shared workspace using the proposed XR collaboration architecture is resilient to network latency and failure. The remainder of this paper is structured as follows. We present supporting related works on the DWeb and XR framework in Section 2. We introduce the proposed XR collaboration architecture in Section 3. Then, we explain the prototype implementation in Section 4, and our results in Section 5. Finally, we conclude with a summary and an outlook of future work in Section 6.

2 RELATED WORK

2.1 Need for Decentralized Web in XR Framework

The web architecture is in a state of constant evolution. Although the current centralized system works efficiently, the centralized point for data access might create issues when a myriad of new technologies and devices that demand ubiquitous connectivity join the network. New technological developments such as the Internet of Things (IoT), smart assistants, and augmented and virtual reality with WebXR are anticipated to elevate the threats exposed by the centralized client-server architecture. WebXR is a relatively new technology used to create multi-user immersive and interactive experiences on the web. The issues faced by XR frameworks in the client-server system can be ameliorated with the Peer-to-Peer (P2P) communication enabled by WebRTC and WebSockets in browsers [Marx et al. 2017]. In general, a majority of the browsers support the WebRTC, a web API that can facilitate message exchange in a P2P fashion.

Some of the main advantages of the DWeb include the elimination of central authority and the decrease in likelihood of a single point of failure. DWeb avoids data silos, reduces censorship, and creates data permanence and availability with consistency over the web [Knutsson et al. 2004]. They are open to new development platforms like XR, blockchain, etc. The main issues of a collaborative XR framework are the consistency of updates, availability, and reduced latency values. These demands of the XR framework can be settled by using a DWeb in a P2P fashion. First, the main issue with connectivity is resolved with decentralized solutions since they are serverless and avoid congestion, ensuring ubiquitous connectivity within the network. Better connectivity facilitates consistency, with quick data updates and data availability. Serverless architecture can expedite faster data updates, reducing the latency values in the XR framework. Next, DWeb addresses the most significant issue on security with the elimination of single point of failure and peer addressing. Furthermore, the problems regarding scalability, and the ability to handle more diverse devices are resolved with DWeb.

2.2 Decentralized Database in XR Framework

Applications like XR use databases, to which the publishers write the current state. To enable a collaborative XR model in DWeb, we have explored the option of using Decentralized Databases (DDB) for this XR framework. The benefits of the DDB are paramount to the decentralized vision of the XR scenario. DDBs allow for a user to be actively backed up on multiple peer instances without requiring the user to explicitly connect to any of them. By eliminating control from centralized servers, DDBs can reduce the service downtime, providing resiliency to XR. DDBs enhance security with their immutable feature, another added advantage. A few of the versatile widely adopted DDBs include GunDB [Nadal 2016] and OrbitDB [Haadcode 2016]. DDBs require reliable communication among peers. Most DDBs use WebSockets or WebRTC for communication. They also tend to use Conflict-free Replicated Data Type (CRDT) to synchronize data among its peers. For authentication, they either use public keys or password-based authentication. DDBs use HTTP servers to host their data or use the pubsub protocol to discover peers that are replicating the given DB instance. In this article, we use GunDB for storing shared states and their updates.

GunDB is an open-source, graph-based DDB that facilitates peer-to-peer application development and works even when their peers are offline. GunDB uses WebSockets or WebRTC technology for networking among peers, which makes it decentralized without relying on a single server. It uses a state-based CRDT for strong eventual consistency and communicates the state of the system at any given time among the peers. It defaults to the CAP theorem [Brewer 2012]. GunDB is not strongly consistent, but it is eventually consistent, which makes the DDB available to users even when they are offline. When the peers are back online, they synchronize data with low latency values, and all peers have data availability within a stipulated time frame without any extra coordination. GunDB ensures the stability of our XR Framework so it can synchronize simultaneously among peers without redundancy, ensuring data availability.

Further, GunDB has authentication among peers based on alias and password and can dynamically provide access to peers in the XR framework without centralized authorization. GunDB stores the data in HTTP servers, and it runs as an instance, replicating all the changes from its peers, thereby reducing the latency of our system. The main issues in establishing a collaborative XR framework like scalability, latency, consistent synchronization, and privacy can be minimized using a DDB like GunDB. In this work, we have adopted the features of the DDB GunDB for our collaboration among multiple users in our XR framework.

2.3 Distributed Virtual Environment (DVE) and the Web

Multiuser collaboration in DVE has been used with SIMNET in the 1960s, NPSNet in the 1990s, to more recently in MMORPG and Second life [Linden Lab 2003] for VR games and Chat programs [Gaylor and Joudrey 2017]. VR environments are being expanded with AR into XR with the recent advancement of mobile AR game like Pokémon GO [Niantic 2016]. These distributed 3D XR environments require a new representational framework to cover diverse interaction situations. Furthermore, if these interaction situations

are shared like in the web, they will fulfill the ultimate goals promoted by VR and AR technology in the media with the four fundamental factors mentioned by Dionisio: Immersive realism, ubiquity of access and identity, interoperability, and scalability [Dionisio et al. 2013].

Over the last 30 years, web technology has advanced to the extent that it is a new computing platform working towards universal interoperability for computing services as well as ubiquitous access and identity platform for everyone on earth. More recently, in addition to text, image, and video, 3D content descriptions like eXtensible 3D (X3D) [Web3D Consortium et al. 2004] is being brought into the web as a first-class citizen like A-Frame [Mozilla 2015] with ECS (Entity-Component Structure); here, component ecosystems afford community participation like AR.js [Etienne 2017] for AR scenes in addition to WebVR devices and scene descriptions, boosting XR content as a first-class citizen of the web. As a side benefit, a popular web framework like React can be used for VR scene as ReactVR [Facebook 2019], boosting the client-side productivity gain.

There have been numerous attempts to accommodate scalability and collaboration factors in a distributed virtual environment [Liu and Theodoropoulos 2014]. Unlike a collaboration framework which is not real-time like the Web, XR collaboration mandates real-time scalability. Sharing a frame scene using webRTC, as well as multiuser scene editing components have been developed [Puente 2017] [Lee 2017].

The client-server architecture of the current web is vulnerable to numerous attacks like DDOS and identity theft which are problematic in private collaborations among several participants using a real-time XR collaboration application. Furthermore, big data monopoly by FANG has opposed what the web was intended to be by Tim Berners-Lee, a universal collaboration framework by democratizing access and sharing of web content. Recently, Brewster Kahle and others have promoted DWeb to combat the monopoly [Kahle 2015]. Democratizing the identity and monopoly of currency using blockchain [Zyskind et al. 2015] is also being actively explored as a platform for the next web, Web 3.0. XR collaboration requires both the sharing of the scene, and its states. LiveCodingSpace [Nikolai 2019] explored collaboration in XR using GunDB.

3 DESIGN

In this paper, we have proposed an architecture to introduce a bridge between web-based A-Frames and DDBs to explore the possible benefits of a decentralized framework for XR collaboration. Before explaining the design of our proposal, we will elaborate on some of the commonly used terms in this paper. In XR collaboration, it is necessary to consider the perspective of each participant. Although all participants have an immersive experience in a common shared space, technically, each participant has an individual space that is called *Extension*. Figure 1 shows a common shared space with participants and. their *Extensions*, where they interact practically. Every single *Extension* has differences that are defined by various factors including shared data from DDB, their work environments, equipment, and real-time interactions. For example, in *Extension* 3 of Figure 1, the user in the VR environment is obviously distinguishable from other *Extensions* based on 3D and AR environments. Moreover, even in case of the same environment with the same equipment, for example, if there are five participants in a VR environment, depending on interactions like turning one's head, each participant has their own *Extension*. To cater to these unique features of XR collaboration, we propose an architecture that classifies data and synchronizes among various participants to make their *Extensions* available in a common shared space.

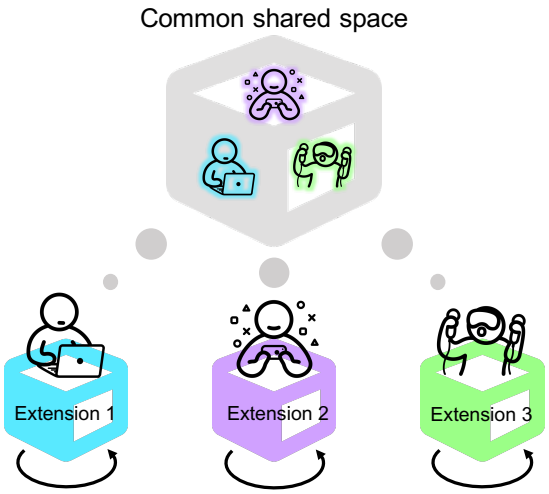


Figure 1: Extensions of common shared space

Figure 2 shows the data flow of the proposed architecture in detail. First, the*Data deliverer* and the *Data storer* bridge the A-Frame and the DDB. Their specific roles are mentioned in Subsections 3.1 and 3.2. In addition, there are the following two data categories in our framework.

- `Base data`: *Base data* is stored in the DDB and synchronized among all participants. *Base data* is common for all the participants in XR collaboration framework.
- `Derived data`: Each participant in the XR collaboration framework has its own data called *Derived data* which makes different *Extension* from a common shared space. *Derived data* need not be synchronized among the different participants. An *Extension manager* shown in Figure 2 gathers and combines data from DDB and A-Frame.

To deal with *Base data* and *Derived data* efficiently, we propose an architecture that has separate parts in a peer for the publish side and the subscribe side. We have explained the publish side with *Data storer*, the subscribe side with*Data deliverer*, and the *Extension manager* in detail in the following subsections.

3.1 Publish side

On the publish side, the *Data storer* stores *Base data* in a DDB from A-Frame as shown in Figure 2. Figure 3 concretely shows the data flow in a publish side. There are two states available in the A-Frame due to the mapping between A-Frame and three.js, and the *Data storer* has access to these two states. The two different states in A-Frame are as follows:

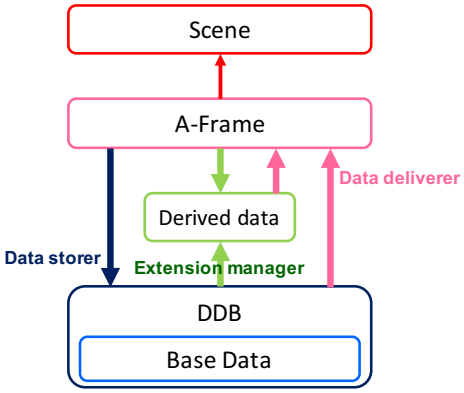


Figure 2: Data flow of the proposed architecture

- `Application state`: The object is represented in HTML as a DOM element according to the entity-component system (ECS).
- `Scene graph state`: The object is represented in three.js as a combination object3D [Cabello 2010]. It is updated from the *application state* by A-Frame.

After accessing these two states, the *Data storer* stores the *Base data* in the DDB, which triggers the DDB to publish the updated data to other peers. However, the network condition may face unexpected obstacles. In this case, the DDB waits for reconnection with only the local update. Finally, when the peer's network come back online, the DDB publishes it immediately so they have eventual consistency.

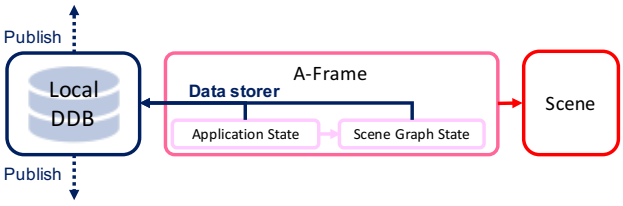


Figure 3: Data flow in publish side

To store *Base data* in a DDB, it is necessary to model a suitable data structure based on our XR collaboration. Figure 4 shows a representation of the gift object by HTML and Figure 5 shows a representation of the gift object by JSON in DDB. These representations are compatible based on key-value pairs as shown in Figure 6. The object in Figure 5 has all the data needed to create the HTML DOM element of Figure 5 including "id", "tagName", and "attributes" and also has "parent" to append to after creating the DOM element. In XR collaboration, when the participants interact with an object, data such as color, parent, or X position value is changed partially. These slight changes in a collaboration model need to be stored as shown in Figure 6. The proposed architecture uses a DDB to synchronize the changes using the data structures across the peers in a decentralized manner and reduces the network resources. It also reduces the latency for updates, enabling data consistency and availability in our XR framework.

```
<a-box id="gift" color="red"
       position="x:0; y:0; z:0"
       rotation="x:0; y:45; z:45"
       scale="x:1; y:1; z:1"
       transform-controls="activated: false">
</a-box>
```

Figure 4: Representation by HTML as DOM element

```
{
   id: 'gift',
   tagName: 'a-box',
   parent: 'a-scene',
   attributes:{
      position: { x: 0, y: 0, z: 0 },
      rotation: { x: 0, y: 45, z: 45 },
      scale: { x: 1, y: 1, z: 1 },
      color: 'red',
      'transform-controls' : { activated: false },
   }
};
```

Figure 5: Representation by JSON in DDB

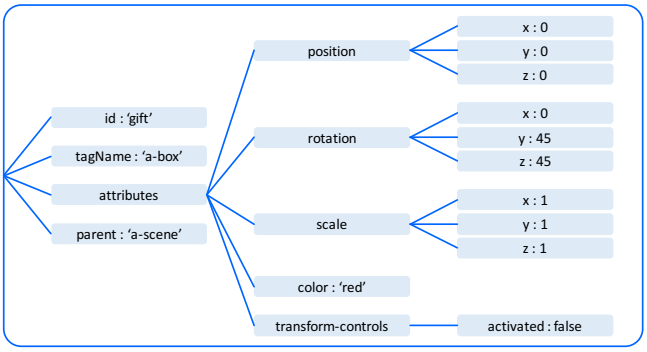


Figure 6: Data structure of an object

Further, objects should have relationships based on the DOM tree in the *application state* as shown in Figure 7. They have one-to-many relationship as parent-children, and the stored objects in a DDB also have the same hierarchy by linking with each other as shown in Figure 8. Even though an object of Figure 5 specifies the "parent", the *Data storer* links objects according to their relationship in the DDB. These relationships help to construct the DOM tree, and are especially used during *Extension* initialization.

3.2 Subscribe side

In the subscribe side, when there is an update, the peer receives data through the DDB from other peers and delivers the data to

```
<a-scene>
    <a-assets>
        <a-asset-item id="ball-obj" scr="ball.obj"></a-asset-item>
        <a-asset-item id="ball-mtl" scr="ball.mtl"></a-asset-item>
    </a-assets>

    <a-box id='gift' position="x:0; y:0; z:0" rotation="x:0; y:45; z:45" scale="x:1; y:1; z:1"
        color=red" transform-controls="activated: false"></a-box>
    <a-entity id="valve">
        <a-entity id="pipe">
            <a-cylinder id="pipe-up"></a-cylinder>
            <a-cylinder id="pipe-down"></a-cylinder>
        </a-entity>
    </a-entity>
    <a-entity id="ball" obj-model="obj: #ball-obj; mtl: #ball-mtl"></a-entity>
</a-scene>
```

Figure 7: Representation of an Extension as HTML

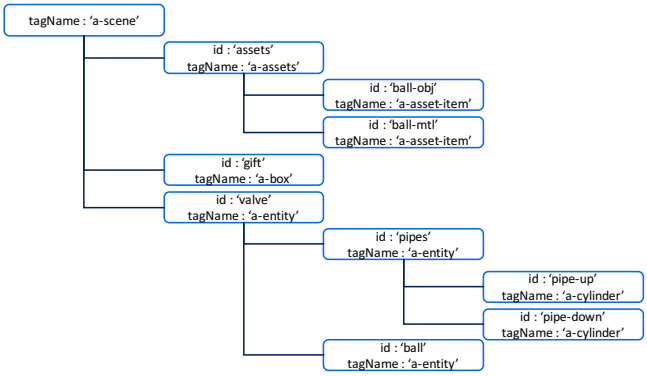


Figure 8: Data structure of an Extension

A-Frame to update the current *Extension* with the*Data deliverer*. Figure 9 shows the data flow with the*Data deliverer* in the subscribe side. The*Data deliverer* writes the updated data from the publish side to the application and *scene graph state*. While writing the update to the *application state*, A-Frame lets the browser reconstruct the DOM tree, and then changes the *scene graph state* to render the scene for the *Extension*. In contrast, writing to the *scene graph state* while leaving the *application state* shows better performance because it does not need to manipulate the DOM tree. However, the difference in both states can create issues in the XR framework. It is tough to determine the state when the data is created on the publish side and the state when the data is written. As shown in Figure 10, the data should be matched to the same state between the publish and subscribe side. The data created on the *application state* in the publish side should be written on the *application state* in the subscribe side to uniformly maintain the state of all participants. The DDB helps to maintain this consistency in data by synchronization among all participants.

In our XR framework, a peer gets data from the DDB in two cases. First, when the participant enters the *Extension* of a common shared space, the peer replicates a database instance of the DDB and initializes the scene. Second, when another peer publishes data to which the other peer subscribes, the peer updates the scene with changed data by interactions of participants. Regarding the first case, it is significant to understand how A-Frame creates and renders the space in order. A-Frame has an asset management system which uses the "<a-assets>'' tag to allow us to place our assets in one place

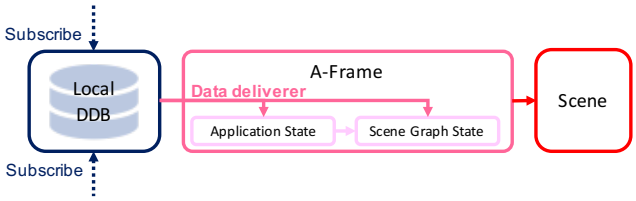


Figure 9: Data Deliverer with subscribed data

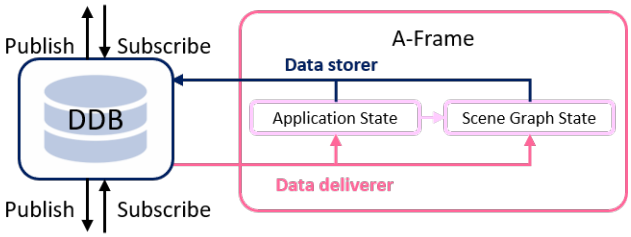


Figure 10: Maintain same state between publish side and subscribe side

and to preload the cache assets for better performance as shown in Figure 7. Hence asset DOM elements should be created before others using an asynchronous function. Figure 11 shows the code for the asynchronously separate function of assets and others. Moreover, the DOM tree is constructed depending on the stored structure in DDB.

```
let space = gun.get( spaceID );

async function initAScene(){
    await initAssets();
    await initEntities();
}
async function initAssets(){
    space.get('assets').map().once( createEl );
}
async function initEntities(){
    space.get('children').map().once( initEntity );
}
```

Figure 11: Initialize an Extension in order

After initializing the *Extension*, participants interact in a common shared space to collaborate with the other participants. The size of the data replicated in a common shared space among the participants is smaller than at the time of initialization. This is due to the fact that the subscribe side subscribes only to the *Extensions* which have a smaller data size, improving the performance of the system in our XR Framework.

For collaboration in the XR framework the data deliverer uses both *Base data* and *Derived data*. But the *Derived data* is not stored in the DDB and when the participant terminates the browser the

Derived data is lost. When the participant comes back online, *Extension manager* recreates *Derived data* by combining *Base data* and A-Frame with interactions as shown in Figure 12.

In the subscribe side, the job of the*Data deliverer* is to update A-Frame using *Base data* from the DDB as shown in Figure 12. For collaboration in the XR framework, the data deliverer uses both *Base data* and *Derived data*, but the *Derived data* is not stored in the DDB, and when the participant terminates the browser, the *Derived data* is lost. When the participant comes back online, the *Extension manager* recreates the *Derived data* by combining the *Base data* and A-Frame with interactions as shown in Figure 12.

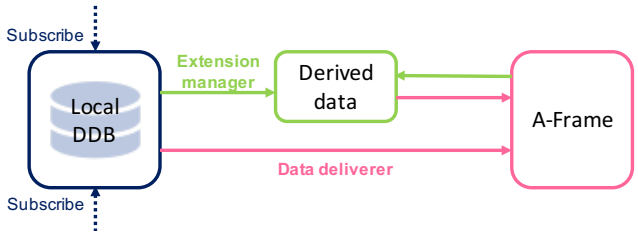


Figure 12: Data delivery of Base data from DDB and Derived data by Extension manager in subscribe side

4 IMPLEMENTATION

We implemented our proposed prototype using a DDB for our collaborative XR framework. To implement the framework, we used GunDB as the DDB and A-Frame as a web framework to render the scene on the web. Figure 13 shows a complete overview of the proposed system architecture that is used to share a common shared space and render their scene on the web. In the perspective of the whole system, each peer is connected by GunDB in a P2P manner with a conceptual scenario of a common shared space as shown in Figure 13. Our XR collaboration is similar to the BASE (Basically Available, Soft State, Eventual Consistency) framework [Pritchett 2008] [Brewer 2012] with webizing GunDB and A-Frame.

4.1 Extensions of each participant

Every participant has their own *Extension* of a common shared space where participants feel and experience interactions together. An *Extension* is constructed using the *Base data* and the *Derived data*. Participants basically share a common shared space by rendering it from *Base data* that GunDB synchronizes for remote updates from peers. It distinguishes between *Extensions* by rendering from *Derived data* updated locally. Figure 14 shows different *Extensions* by camera perspectives derived from interactions with each participant. The position of the camera's perspective is *Derived data*, and each of updates the local *Extension*. This allows each participant to visualize their space in their desired way.

To experience a common shared space with peers, the *Base data* should be synchronized. Figure 15 shows the synchronized position of gray cover that is moved by the participant shown in Figure 15a. When the participant in Figure 15a clicks the object in their scene, a gizmo that can move an object appears. As the participant of Figure 15a continues to interact with the gray cover using the gizmo, the

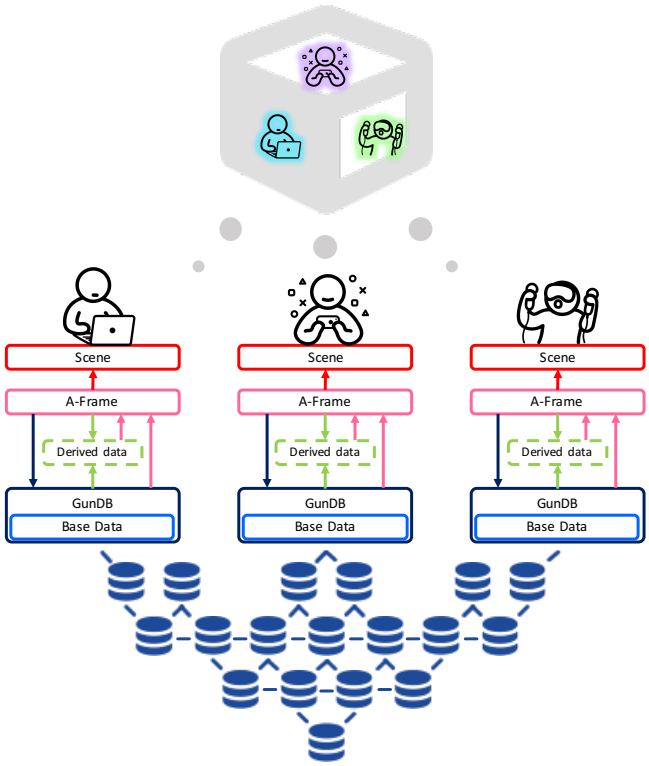
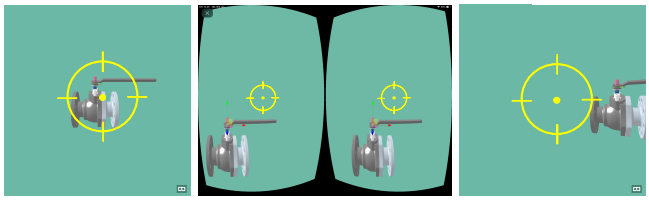


Figure 13: Overview of the proposed collaborative XR implementation



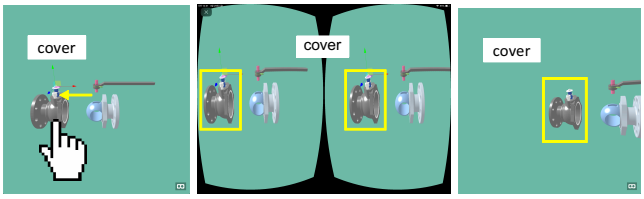
(a) Extension 1 **(b)** Extension 2 **(c)** Extension 3

Figure 14: Local updates from Derived data

position of the object is changed in the *scene graph state*. After storing this changed data to the GunDB, the participant in Figure 15a publishes the changed position data to the other peers, and the other peers receive this position data and update the position of their gray cover in the *scene graph state*, as shown in Figure 15b and Figure 15c. Given that this data was changed in the *scene graph state*, the *Data deliverer* writes it in the *scene graph state*. As a result of this process, participants have the same position of gray cover in each *Extension* synced across them, as shown in Figure 15.

4.2 Basically Available collaboration system

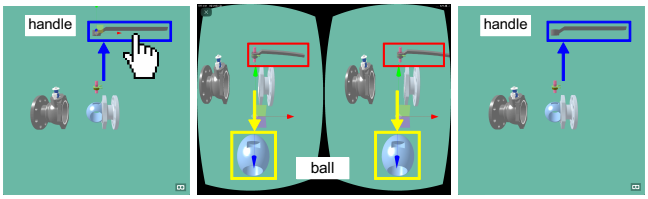
The proposed architecture in this paper applies DWeb and makes XR collaboration basically available even in harsh network environments with eventual consistency. Figure 16 and Figure 17 shows the collaboration between three participants. One of the participants is



(a) Extension 1 **(b)** Extension 2 **(c)** Extension 3

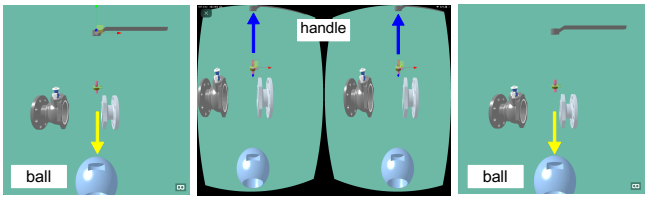
Figure 15: Synchronize Base data

in an offline state as shown in Figure 16b and Figure 17b, the other participants are in the online state. The objects of the participants in the online state are located correspondingly, along with changes of their handle, as shown in Figure 16a and Figure 16c. However, the participant in the offline state shown in Figure 16b does not have the same handle position, because the participant in Figure 16b cannot subscribe due to bad network conditions. Despite the fact that the offline participant cannot publish and subscribe, he/she still interacts with his/her own *Extension* and the updates are stored in Figure 16b, the offline participant manipulates the ball in the space, without publishing the position of the ball. However, when the offline participant gets online again as shown in Figure 17, Figure 17b subscribes to the position of the handle and publishes the position of the ball to other peers. The remaining peers as shown in Figure 17a and Figure 17c subscribe to position of the ball. After this process, the participants can experience a common shared space without any conflict using CRDT in GunDB.



(a) Extension 1 in online **(b)** Extension 2 in offline with only local update **(c)** Extension 3 in online

Figure 16: With an Offline Participant

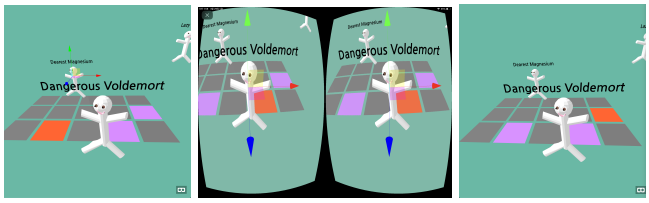


(a) Extension 1 in online **(b)** Extension 2 in online again **(c)** Extension 3 in online

Figure 17: After reconnecting of offline Participant

4.3 XR collaboration with multiple users

Next, we discuss the implementation of XR collaboration with multiple users. Figure 18 shows an example of XR collaboration based on DWeb with three participants in each of *Extensions*. While interacting with their *Extensions*, the participants share a common shared space. When a user clicks one of the gray seats in his/her *Extension*, the color of seat turns to orange and an avatar with a gizmo and ownership that can move appears. Even if two or more participants click the same seat at the same time, CRDT deduces a result for which participant occupies the seat. Other participants recognize that purple means the seat is unoccupied. In this case, ownership of each seat and the position of the avatar are synchronized by the *Base data*, and the color of seats is *Derived data*. Each *Extension* of Figure 18 has designated orange seat from *Derived data* by local updates and shared avatars from *Base data* by GunDB's synchronization. In this manner, additional participants can join this collaboration space without any conflicts, decreasing the scalability issues in the XR framework.



(a) Extension 1 **(b)** Extension 2 **(c)** Extension 3

Figure 18: XR Collaboration

5 RESULTS

Table 1: Experimental result videos

Video	Description	URL
Video 1	Data synchronization	https://youtu.be/IybLP3yCTds
Video 2	AR interaction	https://youtu.be/1iC--xMVjsI
Video 3	VR interaction	https://youtu.be/GSV-c7CYasM
Video 4	Mesoscale collaboration	https://youtu.be/wEHYlPkEVKY

In this section, we discuss the results from the XR collaborative framework. We have tabulated the link to the videos containing the XR implementation in Table 1. Video 1 shows the synchronization of data with five *Extensions*. Four of them are in the basic 3D environment on different browsers like Chrome and Safari. The other participant uses Galaxy Tab in the 3D environment. Depending on a participant's dragging gray cover in his/her *Extension*, all *Extensions* have same location of gray cover by updating its position through GunDB, which publishes data using the put( ) method and subscribes to data using the on( ) method. Although the participant using Galaxy Tab had lousy network connectivity, he/she had eventual consistency of data among all participants. The proposed decentralized framework works well in both VR and AR environments.

Video 2 shows an example with the AR environment implemented by AR.js. The Objects in the *Extensions* are augmented on the marker for the AR participant, using which they can collaborate in real time. Video 3 shows collaboration including a VR participant. Although participants interact using different ways such as a mouse or a VR controller, each end device need not be managed individually because data is published and subscribed by the proposed model. In Video 3, when the left user clicks one of the seats, the seat turns to orange in the left *Extension* and purple in the right *Extension*. The *Extension manager* in the subscribe side combines the owner of the seat from the DDB and the participant's name to get the value and determine to whom the seat belongs. If the seat is mine, the color is orange as *Derived data*. Although the right participant is in a VR environment that faces latency issues, after grabbing the VR controller, they have eventual consistency.

Lastly, Video 4 shows a practical experiment with twelve participants. The participants join the common shared apace in 3D or VR environments using their laptops or mobile devices. One of participants recorded this video which includes his *Extension* and another *Extension* on the screen. When he is dragging his avatar, it moves in his *Extension* and the screen *Extension* as well. We can see the movement of the avatars when the other participants interact. The experiment was performed at a conference room with anonymous participants attending an academic conference, and we did not provide any specific network environment for the experiment. This means that each participant's *Extension* is connected through heterogeneous networks such as Wi-Fi facility of the conference room and commercial LTE and 3G from multiple service providers. This video illustrates our collaborative XR framework using DWeb and the DDB efficiently syncing data among participants reducing the latency and ensuring data availability and consistency in a real heterogeneous network environment.

6 CONCLUSION

In this paper, we propose an XR collaboration architecture based on the DWeb for practical collaboration environments in the absence of a reliable network connection. The XR application's core functionality still works, and synchronization of the XR content and interaction is guaranteed among local users when the network connectivity between local and remote sites is slow or non-existent by the offline-first strategy of GunDB. The consistency of XR content among remote sites is gracefully degraded when the network connectivity gets worse, and recovers when the connectivity gets better. The XR content of each participant, which is an *Extension* of the common shared space, is eventually consistent in this manner and strongly resilient to network latency and failure. Accordingly, our proposed XR collaboration architecture works even in drastically challenging network environments. Its main difference from existing prior work on a DWeb in XR frameworks is that they have focused only on scalability [Hu et al. 2017]. We provide a clue for the impending problem of network connectivity while utilizing XR collaboration in a practical environment. We expect that the prototype implementation will be extended for practically low latency environments such as education and training for submarine crews because they need immersive collaboration between local crews and subject-matter experts on a land base.

ACKNOWLEDGMENTS

This work was supported by the National Research Council of Science and Technology (NST) grant by the Korea government (MSIT) (No. CMP-16-01-KIST) and the Korea Institute of Science and Technology (KIST) under the Institutional Program (Grant No. 2E29450).

REFERENCES

Eric Brewer. 2012. CAP twelve years later: How the "rules" have changed. *Computer* 2 (2012), 23–29. https://www.infoq.com/articles/cap-twelve-years-later-how-the-rules-have-changed

Ricardo Cabello. 2010. Object3D of Three.js. Retrieved April 1, 2019 from https://threejs.org/docs/#api/en/core/Object3D

John David N. Dionisio, William G. Burns III, and Richard Gilbert. 2013. 3D Virtual Worlds and the Metaverse: Current Status and Future Possibilities. *ACM Comput. Surv.* 45, 3, Article 34 (July 2013), 38 pages. https://doi.org/10.1145/2480741.2480751

Jerome Etienne. 2017. AR.js. Retrieved March 18, 2019 from https://github.com/jeromeetienne/AR.js

Facebook. 2019. React360. Retrieved March 18, 2019 from https://facebook.github.io/react-360

Graham Gaylor and Jesse Joudrey. 2017. VRChat. Retrieved March 18, 2019 from https://www.vrchat.net

Haadcode. 2016. OrbitDB. Retrieved March 18, 2019 from https://github.com/orbitdb/orbit-db

Yonghao Hu, Zhaohui Chen, Xiaojun Liu, Fei Huang, and Jinyuan Jia. 2017. WebTorrent Based Fine-grained P2P Transmission of Large-scale WebVR Indoor Scenes. In *Proceedings of the 22Nd International Conference on 3D Web Technology (Web3D '17)*. ACM, New York, NY, USA, Article 7, 8 pages. https://doi.org/10.1145/3055624.3075944

Brewster Kahle. 2015. Locking the Web Open: Rethinking the World Wide Web. Retrieved April 1, 2019 from https://thenextweb.com/insider/2015/04/10/locking-the-web-open-why-we-need-to-rethink-the-world-wide-web/

Bjorn Knutsson, Honghui Lu, Wei Xu, and Bryan Hopkins. 2004. Peer-to-peer support for massively multiplayer games. In *IEEE INFOCOM 2004*, Vol. 1. IEEE.

Hayden Lee. 2017. Networked-Aframe. Retrieved March 18, 2019 from https://github.com/networked-aframe/networked-aframe

Linden Lab. 2003. Second Life. Retrieved March 18, 2019 from https://secondlife.com

Elvis S. Liu and Georgios K. Theodoropoulos. 2014. Interest Management for Distributed Virtual Environments: A Survey. *ACM Comput. Surv.* 46, 4, Article 51 (March 2014), 42 pages. https://doi.org/10.1145/2535417

Robin Marx, Sander Vanhove, Wouter Vanmontfort, Peter Quax, and Wim Lamotte. 2017. DOM2AFRAME: Putting the web back in WebVR. In *2017 International Conference on 3D Immersion (IC3D)*. IEEE, 1–8.

MAXST. 2018. Vivar. Retrieved March 30, 2019 from https://vivar.me

Ashutosh Morde. 2007. *Asymmetric Collaboration: Enabling a Shared Workspace Through Augmented and Virtual Realities*. Rutgers The State University of New Jersey - New Brunswick. https://books.google.co.kr/books?id=_IoKcgAACAAJ

Mozilla. 2015. A-Frame. Retrieved March 18, 2019 from https://aframe.io

Mark Nadal. 2016. GunDB. Retrieved March 18, 2019 from https://gun.eco

Niantic. 2016. Pokémon GO. Retrieved March 18, 2019 from https://www.pokemongo.com

Suslov Nikolai. 2019. LiveCoding. space: Towards P2P Collaborative Live Programming Environment for WebXR. (2019).

Ohan Oda, Carmine Elvezio, Mengu Sukan, Steven Feiner, and Barbara Tversky. 2015. Virtual Replicas for Remote Assistance in Virtual and Augmented Reality. In *Proceedings of the 28th Annual ACM Symposium on User Interface Software and Technology (UIST '15)*. ACM, New York, NY, USA, 405–415. https://doi.org/10.1145/2807442.2807497

Dan Pritchett. 2008. BASE: An Acid Alternative. *Queue* 6, 3 (May 2008), 48–55. https://doi.org/10.1145/1394127.1394128

Salva de la Puente. 2017. Multi-user experiences with A-Frame. Retrieved March 18, 2019 from https://hacks.mozilla.org/2017/10/multi-user-experiences-with-a-frame

Scope AR. 2019. Remote AR. Retrieved March 30, 2019 from https://www.scopear.com/products/remote-ar

Daeil Seo, Byounghyun Yoo, and Heedong Ko. 2018. Webizing Collaborative Interaction Space for Cross Reality with Various Human Interface Devices. In *Proceedings of the 23rd International ACM Conference on 3D Web Technology (Web3D '18)*. ACM, New York, NY, USA, Article 12, 8 pages. https://doi.org/10.1145/3208806.3208808

Sivaramakirshnan Somasegar and Linda Lian. 2017. XR is a new way to consider the reality continuum. Retrieved March 29, 2019 from https://techcrunch.com/2017/05/02/xr-a-new-way-to-consider-the-reality-continuum

Virnect. 2019. Remote AR. Retrieved March 30, 2019 from https://remotear.io

Vuforia. 2018. Chalk AR. Retrieved March 30, 2019 from https://chalk.vuforia.com

Web3D Consortium et al. 2004. Extensible 3D (X3D)-ISO. *IEC FDIS (Final Draft International Standard)* 19775 (2004), 200x.

Wikipedia. 2019. X Reality (XR). Retrieved March 29, 2019 from https://en.wikipedia.org/wiki/X_Reality_(XR)

Guy Zyskind, Oz Nathan, et al. 2015. Decentralizing privacy: Using blockchain to protect personal data. In *2015 IEEE Security and Privacy Workshops*. IEEE, 180–184.