Distributed Edge Computing for Cooperative Augmented Reality: Enhancing Mobile Sensing Capabilities

Cheng-Yu Cheng[b], Qi Zhao[a], Cheng-Ying Wu[b], Yuchen Yang[a], Muhammad A. Qureshi[c], Hang Liu[b], and Genshe Chen[a]

[a]Intelligent Fusion Technology, Inc, 20410 Century Blvd, Suite 230, Germantown, USA
[b]The Catholic University of America, 620 Michigan Ave NE, Washington D.C., USA
[c]US Army C5ISR Center, 6662 Gunner Circle, Aberdeen Proving Ground, USA

ABSTRACT

Cooperative Augmented Reality (AR) can provide real-time, immersive, and context-aware situational awareness while enhancing mobile sensing capabilities and benefiting various applications. Distributed edge computing has emerged as an essential paradigm to facilitate cooperative AR. We designed and implemented a distributed system to enable fast, reliable, and scalable cooperative AR. In this paper, we present a novel approach and architecture that integrates advanced sensing, communications, and processing techniques to create such a cooperative AR system, and demonstrate its capability with HoloLens and edge servers connected over a wireless network. Our research addresses the challenges of implementing a distributed cooperative AR system capable of capturing data from a multitude of sensors on HoloLens, performing fusion and accurate object recognition, and seamlessly projecting the reconstructed 3D model into the wearer's field of view.

The paper delves into the intricate architecture of the proposed cooperative AR system, detailing its distributed sensing and edge computing components, and the Apache Storm-integrated platform. The implementation encompasses data collection, aggregation, analysis, object recognition, and rendering of 3D models on the HoloLens, all in real-time. The proposed system enhances the AR experience while showcasing the vast potential of distributed edge computing. Our findings illustrate the feasibility and advantages of merging distributed cooperative sensing and edge computing to offer dynamic, immersive AR experiences, paving the way for new applications.

Keywords: Distributed edge computing, Cooperative sensing, Augmented reality, Mobile sensing, Apache Storm, HoloLens

1. INTRODUCTION

Augmented Reality (AR) has emerged as a transformative technology, enriching real-world environments with real-time, immersive, and context-sensitive digital overlays.[1] Cooperative AR further advances this innovation by harnessing the collective sensing capabilities of multiple devices[2] to enhance situational awareness and enrich user experiences. However, the comprehensive deployment of cooperative AR presents substantial challenges, including complexities in data processing, communication, and system scalability.

Distributed edge computing[3,4] has surfaced as a potent solution to these challenges. By leveraging the computational power and proximity of edge servers, this technology supports cooperative AR systems effectively. It facilitates the offloading of computationally intensive tasks from mobile devices, thereby enhancing processing speeds and reducing latency. This capability is crucial for enabling real-time data processing, accurate object detection, and seamless integration of virtual content with the physical environment, which is essential for immersive cooperative AR experiences.

Further author information: (Send correspondence to Qi Zhao or Genshe Chen)
Qi Zhao: E-mail: qi.zhao@intfusiontech.com
Genshe Chen: E-mail: gchen@intfusiontech.com

Moreover, the importance of mobile sensing[5] is escalating across various domains, becoming more critical as applications become increasingly distributed, dynamic, and large-scale. Mobile sensing faces significant challenges, such as accurately capturing and effectively coordinating information from an expanding global network of sensors. Distributed stream processing[1] addresses these challenges by enabling the real-time handling and analysis of continuous data streams across multiple interconnected nodes. This method is particularly relevant for applications that generate large volumes of data, including sensor networks, social media platforms, and Internet of Things (IoT) devices.

Edge computing complements distributed stream processing by bringing computational resources closer to where data is generated. This shift not only significantly reduces latency and bandwidth usage but also enhances privacy and security by allowing sensitive data to be processed locally, minimizing exposure to long-distance transmission risks. In dynamic environments such as autonomous vehicles, smart cities, and industrial IoT, real-time analytics and decision-making are crucial, making edge computing a cornerstone for modern mobile sensing applications.

In sum, the convergence of distributed edge computing and distributed stream processing forms a robust foundation for advancing mobile sensing technologies, particularly in the context of cooperative AR. This paper discusses how these technologies collectively enhance the capabilities of cooperative AR, paving the way for innovative applications that require dynamic, efficient, and secure data processing.

Specifically, we introduce an innovative framework and architecture that facilitates advanced sensing, communication, and processing technologies to develop a robust distributed cooperative AR system. Our approach addresses the operational challenges of collecting extensive sensor data, executing efficient data fusion, ensuring precise object recognition, and integrating interactive 3D models directly into the user's view. At the heart of our proposed system is an architecture that combines the capabilities of HoloLens—a state-of-the-art AR headset—with edge servers interconnected via a wireless network. The HoloLens serves as a comprehensive sensing unit, gathering inputs from a variety of sensors, including cameras, depth sensors, and Inertial Measurement Units (IMUs). This sensor data is transmitted to edge servers, where it is processed through data fusion, analyzed, and used for object detection, all within the distributed edge computing framework. Our system not only elevates the AR experience by providing timely, context-aware insights but also demonstrates the extensive capabilities of distributed edge computing in supporting cooperative AR applications. By delegating computationally demanding tasks to edge servers, our architecture significantly eases the load on mobile devices, facilitating fluid and interactive AR experiences.

The key contribution of this paper can be summarized as:

1. Proposed a novel framework that integrates advanced sensing, communication over wireless networks, and distributed edge computing for enabling real-time, scalable, and reliable cooperative AR experiences.

2. Addressed the challenges of implementing a distributed cooperative AR system by capturing data from multiple sensors on HoloLens, performing data fusion and accurate object recognition on edge servers, and seamlessly projecting reconstructed 3D models into the user's field of view.

3. Demonstrated the feasibility of the proposed cooperative AR system through an implementation that leverages the sensing capabilities of HoloLens, the computation power of edge servers integrated with Apache Storm, and wireless communication, thereby showcasing the potential of distributed edge computing for immersive AR applications.

In subsequent sections, we explore the detailed architecture of our cooperative AR system, focusing on its distributed sensing elements, edge computing infrastructure, and integration with the Apache Storm platform. Those details are presented in Section 4. We outline the implementation processes, encompassing data collection, aggregation, analysis, object detection, and the reconstructing and rendering of 3D models on the HoloLens in Section 5. Additionally, we present our findings and discuss the practical benefits and potential of merging distributed cooperative sensing with edge computing to deliver dynamic, immersive AR experiences in Section 6. This integration opens avenues for novel applications across various fields. Moreover, we list the most related work in Section 2 and conclude our work in Section 7.

2. RELATED WORK

In our previous work,[6] we conducted a comprehensive study and evaluation of Coded Distributed Computing models to demonstrate their capability to enhance distributed computing. We also integrated Named Data Networking into the Apache Storm-based distributed computing environment for improving object classification and recognition tasks. However, this work only remained at the simulation level with virtual machines for building the testbed. The capability of supporting other applications or services was missing either. In this work, we aim at higher level framework design and implementation and demonstrate the framework capability by using the Microsoft HoloLens enhanced AR sensing as an example application. Moreover, we have done other work on edge computing in the medical field. In our most recent work,[7] we delved into the integration of 5G[8] connectivity, edge computing, and Medical Extended Reality (MXR)[9] in healthcare, exemplified by an MXR setup in an edge computing-enabled 5G network testbed. It assessed the effects of 5G network configurations on MXR by analyzing communication traffic, and providing insights into MXR application behavior and infrastructure. Notably, our work focused solely on client-server architecture, overlooking distributed computing setups.

Previous works such as the scalable distributed stream processing,[10] only focus on the specific systems Aurora[11] and Medusa,[12] which have several limitations. While Aurora assumes a single administrative domain for all nodes, Medusa attempts to address federated operation across administrative boundaries, but the proposed economic contract model and mechanisms for load sharing and availability may be overly complex and impractical in real-world scenarios. Additionally, the paper was written in 2003, and the landscape of distributed stream processing has evolved significantly since then, with the emergence of more modern systems and frameworks. Consequently, we conducted a meticulous comparison of various distributed stream processing frameworks as detailed in two survey papers[1,13] The frameworks evaluated include Apache Storm,[14] Apache Spark Streaming,[15] S4,[16] Amazon Kinesis,[17] and IBM Streams.[18] Our analysis encompassed several dimensions: the type of framework, implementation language, supported languages for application development, level of abstraction, data sources, computation or transformation models, persistence mechanisms, execution reliability, fault tolerance, latency, and vendor affiliation. After careful evaluation,[19] we ultimately selected Apache Storm as our distributed stream processing framework for several reasons. Firstly, Apache Storm offers robust support for real-time data processing with its low latency and fault-tolerant architecture. Secondly, its scalability and flexibility make it suitable for handling large volumes of data streams across distributed environments. Additionally, Apache Storm provides a wide range of programming languages for application development and integrates seamlessly with other big data tools and platforms. Overall, its comprehensive features and proven performance make Apache Storm the ideal choice for our cooperative augmented reality system.

3. BACKGROUND

3.1 Apache Storm

Apache Storm[20] is an open-source, distributed real-time computation system designed for processing unbounded streams of data. Developed by the Apache Software Foundation, Storm is widely utilized for real-time analytics, online machine learning, and continuous computation applications. Its primary strength lies in its ability to reliably process vast amounts of data in real-time, making it an ideal choice for building robust and scalable distributed systems.

At the core of Apache Storm is its streaming data model, which represents unbounded sequences of Tuples (key-value pairs). These Tuples are processed by Topology, which is composed of Spouts (data sources) and Bolts (computational units). Spouts ingest data from external sources, such as message queues or databases, and emit Tuples into the Topology. Bolts, on the other hand, consume Tuples, perform computations or transformations, and optionally emit new Tuples downstream. Apache Storm's architecture is designed to be horizontally scalable, fault-tolerant, and highly available. It employs a master-worker paradigm, where a central component called Nimbus manages the distribution of tasks across a cluster of worker nodes (Supervisors), ensuring efficient resource utilization and fault tolerance through automatic reassignment of tasks upon node failures. One of the key features of Apache Storm is its real-time processing capabilities, enabling low-latency computation and response times. This characteristic makes it well-suited for applications that require immediate analysis and decision-making, such as real-time monitoring, anomaly detection, and event processing. Our testbed is

built upon the Apache Storm framework, leveraging its distributed real-time computation capabilities to enable cooperative AR experiences. Apache Storm's ability to process unbounded streams of data in real-time makes it an ideal platform for handling the continuous flow of sensor data required for cooperative AR applications.

3.2 Microsoft HoloLens

In our evaluation testbed and demonstration, we leveraged the capabilities of the Microsoft HoloLens,[21] a pioneering AR headset, to showcase the potential of our cooperative AR system. The HoloLens played a crucial role as both a sensing device and a visualization platform, enabling us to capture real-world data, process it through our Apache Storm-based distributed system, and render the augmented content back into the user's field of view. The HoloLens' advanced sensor suite, comprising multiple cameras, depth sensors, and IMUs, was instrumental in capturing rich environmental data. These sensors collaborated to generate a detailed spatial map of the surroundings, facilitating a precise understanding of the physical environment and accurate placement of virtual objects within it.

One of the key advantages of the HoloLens is its untethered, self-contained nature, which allowed for seamless mobility and freedom of movement during our evaluation. Without the constraints of external computing devices or cables, users can freely explore and interact with the augmented environment, providing a truly immersive and natural experience. The see-through display of the HoloLens played a pivotal role in visualizing the augmented content generated by our cooperative AR system. Leveraging the device's spatial awareness capabilities, we can render 3D virtual contents that appear to coexist with real-world objects, creating a seamless integration of virtual and physical elements. Interaction with these virtual objects was facilitated by the HoloLens' intuitive interface, which supports natural gestures, gaze tracking, and voice commands. Users could manipulate and explore the augmented content using hand gestures, providing a highly engaging and interactive experience.

By incorporating the Microsoft HoloLens into our evaluation testbed and demonstration, we showcase the full potential of our cooperative AR system, from distributed sensing and real-time data processing to immersive visualization and natural interaction. The HoloLens' unique capabilities enabled us to validate the feasibility and efficacy of our approach, paving the way for future advancements in cooperative AR applications across various domains.

4. FRAMEWORK ARCHITECTURE DESIGN

In this section, we outline our framework architecture, detailing the integration of the Apache Storm and the interactions between its components. As shown in Figure 1, our Service-Centric Distributed Resource-Aware (SCDRA) architecture aims to enhance edge computing platforms with distributed computing, content management, and advanced communication. In addition, our framework focuses on developing a platform that is not merely a data transit but an intelligent and proactive participant in data processing, management, and dissemination. The framework integrates various components into a coherent structure that simplifies complex operations while retaining the sophistication required for high-performance edge computing tasks. Moreover, the SCDRA architecture is composed of three primary layers: the Physical Layer, the Management Layer, and the Service Layer, which interact synergistically to support various edge computing applications.

4.1 Physical Layer

The Physical Layer constitutes the fundamental infrastructure of the SCDRA architecture. It is an ensemble of edge devices, each outfitted with a suite of sensors, computation resources, and communication interfaces. These devices form the backbone of the architecture, collecting data and executing tasks in concert with the overlaying layers. Devices in this realm include advanced AR headsets, such as the HoloLens, which provide immersive experiences through detailed environmental scanning and interaction capabilities. Complementing these are drones and other mobile platforms that extend the sensing and computational reach of the framework. Communication technologies interconnecting these devices range from conventional WiFi and wired networks to cutting-edge 5G infrastructures, ensuring rapid and robust data exchange. This layer is meticulously engineered to ensure seamless integration and interoperability, providing a reliable and responsive fabric for the complex workflows demanded by contemporary edge computing applications.

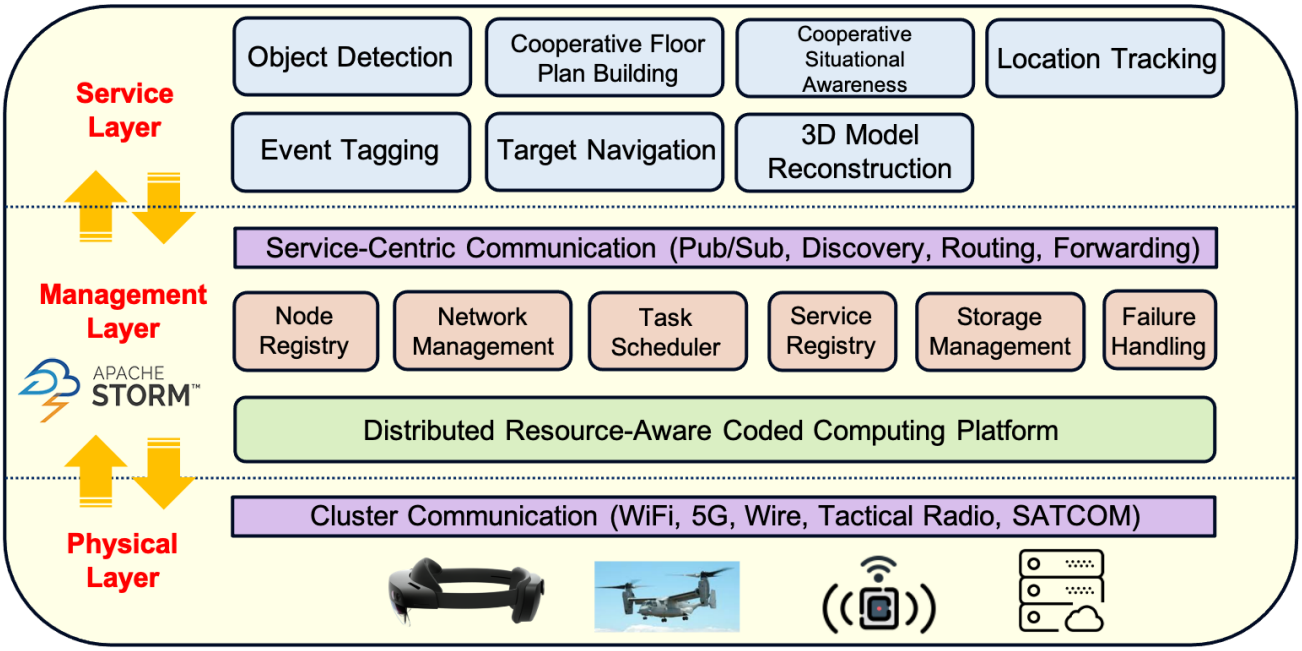

Figure 1. Service-centric distributed resource-aware architecture.

4.2 Management Layer

Central to the SCDRA architecture is the Management Layer which is entrusted with the coherent orchestration of tasks and the management of computational resources across the entire edge computing framework. In our architecture design, this layer was anchored by Apache Storm which brings resilience and elasticity to the framework. It is composed of multiple critical components, such as Node Registry, Network Management, Task Scheduler, Service Registry, Storage Management, and Failure Handling. To be concise, the management layer orchestrates the communications between the service layer and the physical layer, thereby enabling Apache Storm to seamlessly perform distributed computing across the network. This integration ensures efficient data flow and real-time processing capabilities, which are essential for the robust performance of our distributed architecture.

4.3 Service Layer

The Service Layer is the domain where specialized application services reside, offering a spectrum of capabilities that turn the raw resources into meaningful user experiences. In addition, with different service module designs, the service layer can act as a mediator which allows each module to convert complex data that it needs into accessible formats and provide the necessary interfaces for user interaction. For instance, the Object Detection module, as illustrated in Figure 1, is capable of collecting video from the HoloLens or any video-capturing edge devices and performing object detection on the footage. Any edge device that subscribes to this module will gain access to the processed video frames. This functionality not only enhances real-time data analysis but also enables devices across the edge network to benefit from advanced visual recognition capabilities, improving overall system responsiveness and enabling more informed decision-making processes. The Service Layer's versatility serves as its cornerstone, ensuring a fluid integration of a set of services that cater to a wide array of functionalities. These services range from Location Tracking, which offers real-time positioning information, to 3D Model Reconstruction, which converts spatial data into detailed three-dimensional representations. Each service is meticulously crafted to align with the specific preferences and objectives of the end-users. As a result, the layer becomes a dynamic ecosystem that not only responds to user demands but anticipates future needs, thereby fostering an environment of continuous innovation and service-centric development.

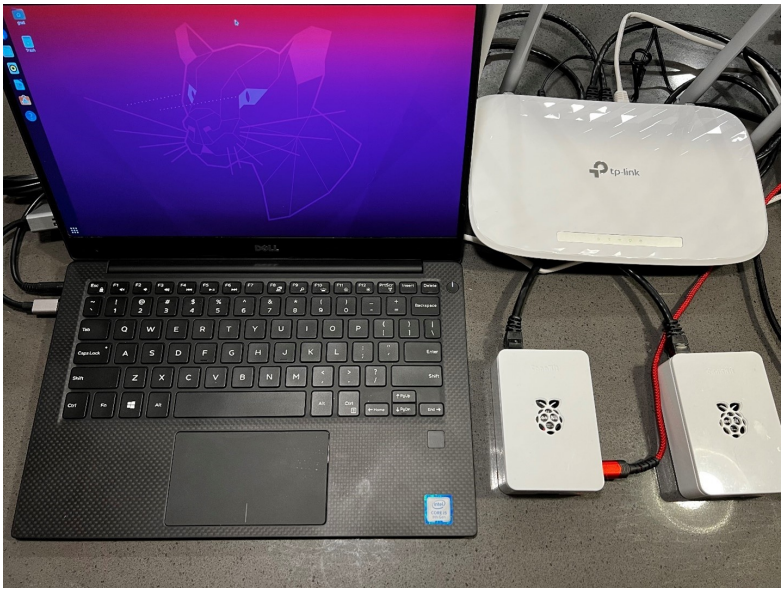

Figure 2. The small-scale evaluation testbed hardware implementation.

5. EVALUATION TESTBED IMPLEMENTATION

To conduct validation experiments and demonstrations, we developed a testbed based on our design introduced in Section 4, centering on a distributed computing environment utilizing Apache Storm framework. This section details the Apache Storm cluster constructed for the testbed, illustrated in Figure 2.

The cluster comprises three nodes: a master node and two slave nodes. The master node, also known as the Nimbus node, is facilitated by a Dell laptop equipped with a 2.5GHz Intel i5 CPU and 8GB RAM. The slave nodes, referred to as Supervisor nodes, are implemented using two Raspberry Pi 4 model B devices. Each Raspberry Pi features a Broadcom BCM2711, Quad-core Cortex-A72 (ARM v8) 64-bit SoC CPU operating at 1.5GHz, and 8GB LPDDR4-3200 SDRAM. Additionally, these devices are equipped with an IEEE 802.11ac wireless interface supporting both 2.4 GHz and 5.0 GHz frequencies, and a BLE Gigabit Ethernet port. All nodes run a Linux operating system, which has been appropriately configured. The nodes are interconnected within a Local Area Network (LAN) via a TP-LINK router, as depicted in Figure 2. For optimal network connectivity, we utilized a wired connection between each node and the router, avoiding the potential variability of wireless connections.

Following the hardware setup and network configuration, we proceeded to install and configure the Apache Storm software across the three nodes. The installation process involved several critical steps to ensure a robust deployment of Apache Storm suitable for our testing and experimental needs. Initially, we installed the prerequisite packages required by Apache Storm on all nodes to prepare them for the subsequent software deployments. Next, we focused on the Dell laptop, which serves as the Nimbus node within our Apache Storm cluster. On this node, we installed ZooKeeper[22] to manage coordination and provide essential services such as configuration management, synchronization, and naming registry to both the Nimbus and Supervisor nodes. For preliminary testing, we opted for a simplified setup using a single-node ZooKeeper cluster. This decision was driven by the need to streamline the initial testing phase while planning to expand the ZooKeeper cluster for enhanced fault tolerance and scalability in future iterations and during the final demonstration.

Following the setup of ZooKeeper, we proceeded to install the Apache Storm package on all three nodes. Each node's configuration parameters were meticulously adjusted to optimize performance and ensure seamless integration within the cluster. After the cluster is launched, we can also activate the Apache Storm UI, which provides a graphical interface for monitoring and managing the cluster's operations. The successful setup and operational status of the cluster are depicted in Figure 3, illustrating the interaction between the nodes and the overall health of the system. The screenshot indicates that we are utilizing Apache Storm version 2.4.0

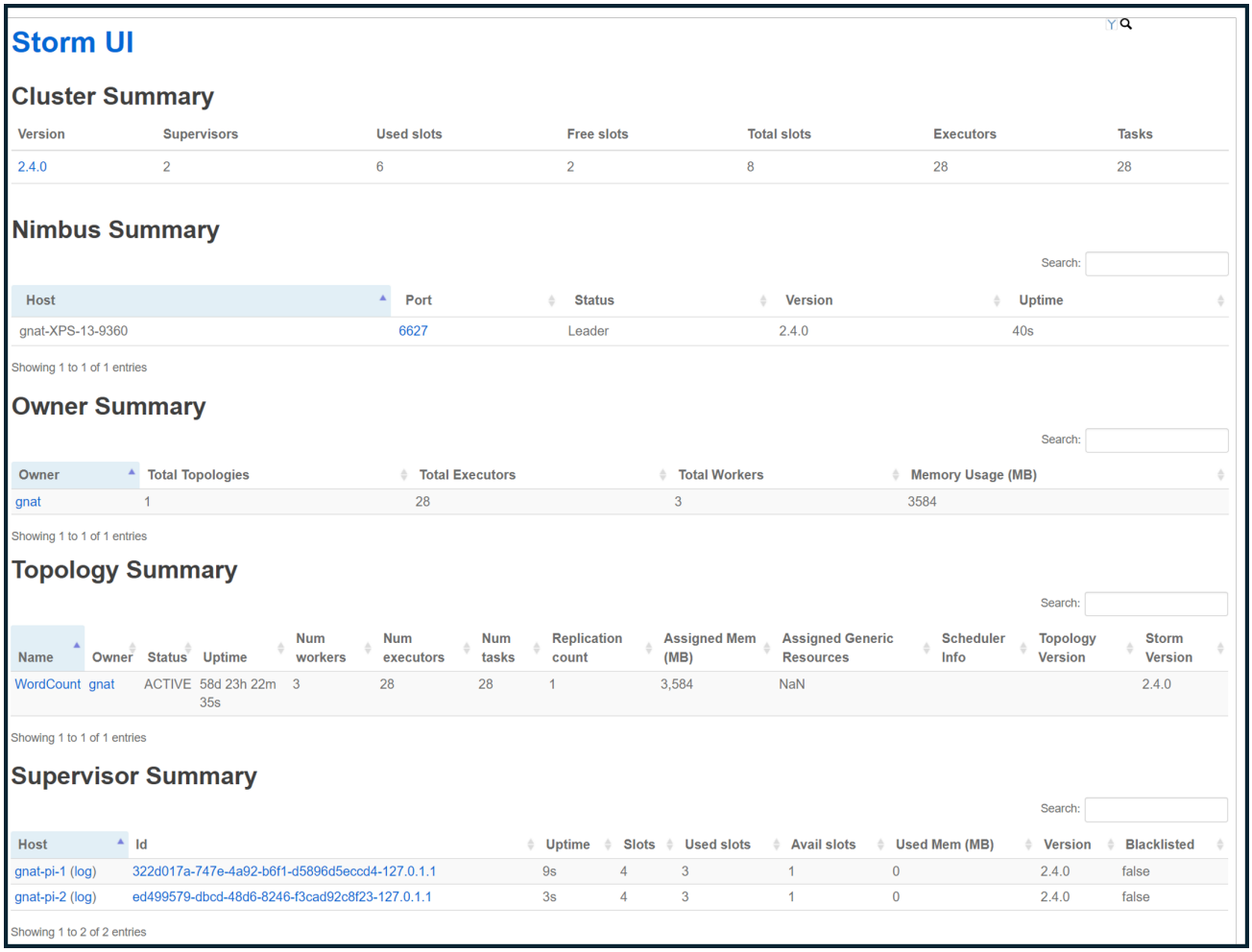

Figure 3. The screenshot of the Apache Storm UI webpage of the testbed.

as our software foundation. Currently, there are two Supervisor nodes operational within the cluster. Despite no active workers, executors, or tasks currently running, the system's Topology remains active and its uptime continues to accrue. A distinctive feature of Storm Topology is that they remain active until manually deactivated or terminated by an administrator. The Storm UI serves as a crucial tool for administrators to manage and interact with the system. Through this interface, administrators can execute several critical operations. The first one is Activate. This function restores a previously deactivated Topology to an active state, allowing it to resume processing data. The second one is Deactivate. This option sets the Topology's status to inactive, temporarily halting its execution. However, it does not affect the Topology's recorded uptime or require redeployment. The third one is Rebalance. This powerful feature enables dynamic adjustment of the number of worker processes and executors assigned to a Topology. Remarkably, this can be accomplished without the need to restart the cluster or the Topology itself, ensuring uninterrupted operations. The last one is Kill. This command terminates the Topology entirely, removing it from Apache Storm and erasing it from the Storm UI. To run this Topology again, an administrator must redeploy the application from scratch. Administrators can initiate these actions by navigating to the Topology Summary section of the Storm UI, selecting the desired Topology, and accessing its Topology summary page. This level of interactivity and control enables efficient and flexible management of Storm Topology, crucial for maintaining robust and reliable system operations.

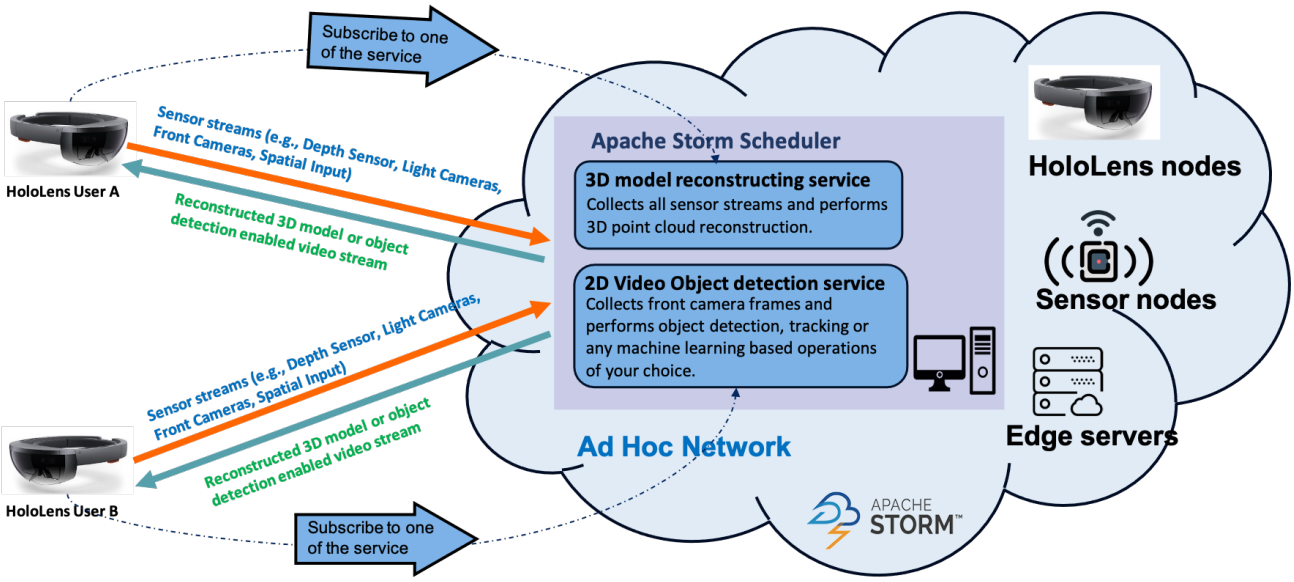

Figure 4. Service modules and system architecture demonstration.

6. DEMONSTRATION

In this section, we present two service modules developed to harness the capabilities of the SCDRA architecture, which together facilitate cooperative augmented reality, thereby enhancing mobile sensing. This serves as a proof of concept to demonstrate the practical applications and effectiveness of our architectural design. These modules not only illustrate the potential for augmented collaboration but also set the stage for future innovations in mobile sensory augmentation. The design of these two modules harnesses the sensory capabilities of HoloLens devices, leveraging its advanced sensor suite to capture rich environmental data crucial for AR applications. In our module design for distributed sensing using Hololens, we use Microsoft HoloLens 2 Research Mode API[2321] to capture sophisticated sensors' streams such as Recurrent Modulation (RM) Depth Long Throw sensors, RM IMUs, Spatial Input, Scene Understanding, and RGB front camera, enabling precise depth perception, spatial mapping, and high-definition video capture. This distributed sensing capabilities of HoloLens ensures the availability of real-time environmental context for enhanced AR experiences, providing users with immersive and interactive overlays seamlessly integrated into their physical surroundings. For our proposed service modules, it is crucial to manage the data streams from HoloLens sensors, facilitating real-time computation and dynamic response. The proposed SCDRA architecture excels in fault tolerance, scalability, and the reliable processing of unbounded data streams, making it an ideal system for high-demand AR applications that require immediate processing and minimal latency. Our system also uses a network of Apache Storm Supervisors, each responsible for executing portions of stream processing tasks. The Supervisors are organized in a cluster managed by a Nimbus server, which distributes tasks, monitors performance, and reallocates resources as necessary to ensure efficient data handling. This configuration not only enhances the system's resilience against node failures but also improves load balancing across the computing nodes. Figure 4 demonstrates the data flow as well as how the service can be performed on the SCDRA system. The stream processing in our system is designed to handle various data streams transmitted from the HoloLens sensors, which can further be used by our service modules. This includes depth information from RM Depth Long Throw sensors, motion data from IMUs, and visual inputs from the front cameras. The Apache Storm Topology is configured to ingest these streams and orchestrate the processing workflows required for AR rendering. In our proposed architecture, two service modules have been developed for cooperative sensing which will be described in the following subsections.

6.1 3D Model Reconstruction Service

We've developed a 3D model reconstructing application in the Apache Storm to provide a 3D cooperative perception mobile sensing service. In this proposed service, the Apache Storm Topology receives data from

the HoloLens users through the HoloLens Research Mode API[23],[21] which provides raw sensor streams. Then, the service processes these streams to calculate positional data relative to the user's head position, capturing the surrounding environment, and calibrating the RGB front camera data with RM Depth Long Throw sensor information to generate a point cloud[24] output which is crucial for accurate 3D model placement in the user's field of view. In this 3D reconstructing service module, the 3D models are constructed using the Open3D toolkit and are then ready to stream to another HoloLens user for display. Whichever HoloLens subscribes to the 3D model reconstruction service can select another user's 3D model, enabling the user to interact with the environment augmented by accurate and real-time 3D content.

6.2 Video Object Detection Service

We've also developed a video object detection service module within our framework that utilizes the advanced capabilities of the HoloLens devices to create a video cooperative perception mobile sensing service. In this service module, the Apache Storm Topology captures video streams from the front cameras of the HoloLens, facilitated by the HoloLens Research Mode API, which offers access to raw sensor data. These streams are immediately processed using the YOLOv8[25] algorithm, implemented via OpenCV,[26] to perform real-time object detection. This detection is pivotal for interactive AR applications, enhancing user engagement by identifying and annotating objects within the user's environment. Furthermore, the processed frames are shared among users through the SCDRA system, fostering a cooperative AR experience. As shown in Figure 4, any user of the end devices that subscribes to this service module will have access to the processed video frames. This enables multiple users to view each other's perspectives with object detection enabled, allowing them to see the same object from different angles. This enriches the interactive experience and enhances the perception of the environment.

7. CONCLUSION

In this paper, we have presented a novel framework and approach for enabling cooperative AR through the integration of advanced sensing, wireless communication, and distributed edge computing. By leveraging the capabilities of the Microsoft HoloLens AR headset and edge servers connected over a wireless network, our system addresses the significant challenges involved in implementing a scalable and reliable distributed cooperative AR solution. Our implementation, built upon the Apache Storm platform, demonstrates the feasibility of this approach and illustrates the potential of distributed edge computing for delivering immersive and context-aware AR experiences. By effectively merging distributed cooperative sensing with edge computing resources, our system enables the seamless projection of reconstructed 3D models directly into the user's field of view, offering a truly dynamic and interactive AR experience. We have showcased the viability of combining cutting-edge technologies, such as the HoloLens and edge computing, to create innovative solutions that push the boundaries of what is possible in the realm of AR. Additionally, our findings pave the way for future advancements and applications in various domains, where the fusion of virtual and physical environments can provide significant benefits.

As we move forward, the integration of cooperative AR with distributed edge computing will continue to evolve, enabling more sophisticated and robust systems capable of handling increasingly complex scenarios. The lessons learned from our research will serve as a foundation for further exploration and development in this exciting field, ultimately driving the creation of more immersive, intuitive, and transformative AR experiences.

Acknowledgment

The work was supported under the Army contract W15P7T-21-C-0001. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the United States Army.

REFERENCES

[1] Azuma, R. T., "A survey of augmented reality," *Presence: teleoperators & virtual environments* **6**(4), 355–385 (1997).

[2] Kim, S.-W., Qin, B., Chong, Z. J., Shen, X., Liu, W., Ang, M. H., Frazzoli, E., and Rus, D., "Multivehicle cooperative driving using cooperative perception: Design and experimental validation," *IEEE Transactions on Intelligent Transportation Systems* **16**(2), 663–680 (2014).

[3] de Assuncao, M. D., da Silva Veith, A., and Buyya, R., "Distributed data stream processing and edge computing: A survey on resource elasticity and future directions," *Journal of Network and Computer Applications* **103**, 1–17 (2018).

[4] Zhao, Q., Li, Y., Mingjie, F., Li, L., and Chen, G., "Edge network computing system with deep reinforcement learning based task scheduling," (May 18 2023). US Patent App. 17/490,861.

[5] Macias, E., Suarez, A., and Lloret, J., "Mobile sensing systems," *Sensors* **13**(12), 17292–17321 (2013).

[6] Zhao, Q., Li, Y., Dang, H. N., Liu, H., Tian, X., and Chen, G., "Resilient mobile distributed computing framework: a coded computing and named data networking approach," in [*Sensors and Systems for Space Applications XVI*], Chen, G. and Pham, K. D., eds., **12546**, 1254608, International Society for Optics and Photonics, SPIE (2023).

[7] Cheng, C.-Y., Liu, Y., Johnson, M., Beams, R., and Al Kalaa, M. O., "Assessing 5g connectivity in medical extended reality (mxr) applications: A testbed approach," in [*2023 IEEE International Conference on E-health Networking, Application & Services (Healthcom)*], 33–38, IEEE (2023).

[8] Gupta, A. and Jha, R. K., "A survey of 5g network: Architecture and emerging technologies," *IEEE access* **3**, 1206–1232 (2015).

[9] Beams, R., Brown, E., Cheng, W.-C., Joyner, J. S., Kim, A. S., Kontson, K., Amiras, D., Baeuerle, T., Greenleaf, W., Grossmann, R. J., et al., "Evaluation challenges for the application of extended reality devices in medicine," *Journal of Digital Imaging* **35**(5), 1409–1418 (2022).

[10] Cherniack, M., Balakrishnan, H., Balazinska, M., Carney, D., Cetintemel, U., Xing, Y., and Zdonik, S. B., "Scalable distributed stream processing.," in [*CIDR*], **3**, 257–268, Citeseer (2003).

[11] Abadi, D., Carney, D., Cetintemel, U., Cherniack, M., Convey, C., Erwin, C., Galvez, E., Hatoun, M., Maskey, A., Rasin, A., et al., "Aurora: a data stream management system," in [*Proceedings of the 2003 ACM SIGMOD international conference on Management of data*], 666–666 (2003).

[12] Ousterhout, J. K., Scelza, D. A., and Sindhu, P. S., "Medusa: An experiment in distributed operating system structure," *Communications of the ACM* **23**(2), 92–105 (1980).

[13] Isah, H., Abughofa, T., Mahfuz, S., Ajerla, D., Zulkernine, F., and Khan, S., "A survey of distributed data stream processing frameworks," *IEEE Access* **7**, 154300–154316 (2019).

[14] Iqbal, M. H., Soomro, T. R., et al., "Big data analysis: Apache storm perspective," *International journal of computer trends and technology* **19**(1), 9–14 (2015).

[15] Salloum, S., Dautov, R., Chen, X., Peng, P. X., and Huang, J. Z., "Big data analytics on apache spark," *International Journal of Data Science and Analytics* **1**, 145–164 (2016).

[16] Neumeyer, L., Robbins, B., Nair, A., and Kesari, A., "S4: Distributed stream computing platform," in [*2010 IEEE International Conference on Data Mining Workshops*], 170–177, IEEE (2010).

[17] Bhartia, R., "Amazon kinesis and apache storm," *Amazon Web Service* (2014).

[18] Hirzel, M., Andrade, H., Gedik, B., Jacques-Silva, G., Khandekar, R., Kumar, V., Mendell, M., Nasgaard, H., Schneider, S., Soulé, R., et al., "Ibm streams processing language: Analyzing big data in motion," *IBM Journal of Research and Development* **57**(3/4), 7–1 (2013).

[19] Lopez, M. A., Lobato, A. G. P., and Duarte, O. C. M., "A performance comparison of open-source stream processing platforms," in [*2016 IEEE Global Communications Conference (GLOBECOM)*], 1–6, IEEE (2016).

[20] Evans, R., "Apache storm, a hands on tutorial," in [*2015 IEEE International Conference on Cloud Engineering*], 2–2, IEEE (2015).

[21] Dibene, J. C. and Dunn, E., "Hololens 2 sensor streaming," *arXiv preprint arXiv:2211.02648* (2022).

[22] Hunt, P., Konar, M., Junqueira, F. P., and Reed, B., "{ZooKeeper}: Wait-free coordination for internet-scale systems," in [*2010 USENIX Annual Technical Conference (USENIX ATC 10)*], (2010).

[23] Ungureanu, D., Bogo, F., Galliani, S., Sama, P., Duan, X., Meekhof, C., St{"uhmer, J., Cashman, T. J., Tekin, B., Sch{"onberger, J. L., Tekin, B., Olszta, P., and Pollefeys, M., "HoloLens 2 Research Mode as a Tool for Computer Vision Research," *arXiv:2008.11239* (2020).

[24] Bui, M., Chang, L.-C., Liu, H., Zhao, Q., and Chen, G., "Comparative study of 3d point cloud compression methods," in [*2021 IEEE International Conference on Big Data (Big Data)*], 5859–5861 (2021).

[25] Jocher, G., Chaurasia, A., and Qiu, J., "Ultralytics YOLO," (Jan. 2023).

[26] Bradski, G., "The opencv library.," *Dr. Dobb's Journal: Software Tools for the Professional Programmer* **25**(11), 120–123 (2000).