2019

A Microservices Oriented Edge Computing Framework for LVC Simulation in the IoT Era

Wenxin Sun
Beijing Institute of Technology

Yanlong Zhai
Beijing Institute of Technology, ylzhai@bit.edu.au

Tianhong Bao
Beijing Institute of Technology

Muhammad Mudassar
Beijing Institute of Technology

Jun Shen
University of Wollongong, jshen@uow.edu.au

See next page for additional authors

Publication Details

Sun, W., Zhai, Y., Bao, T., Mudassar, M., Shen, J. & Yang, K. (2019). A Microservices Oriented Edge Computing Framework for LVC Simulation in the IoT Era. ICCMS 2019: Proceedings of the 11th International Conference on Computer Modeling and Simulation (pp. 190-195). New York, United States: ACM.

A Microservices Oriented Edge Computing Framework for LVC Simulation in the IoT Era

Abstract
Modeling and simulation are essential methods to better understand a complex system in the real world. Many simulation systems have strong demand for involving physical devices or equipment in the system to improve the fidelity. But most of the current cloud based simulation systems are designed for equipment with adequate resources and suffering from high latency for their centralized communication. Edge computing is getting much attention as a new paradigm for distributed systems involving heterogeneous devices and requiring real-time communication. However, edge computing is still in its initial stage and there is no consensus on the edge framework and implementation. To the best of our knowledge, no detailed studies have explored the edge-side simulation and modeling. In this paper, we study and design an edge-side simulation framework, which exploits light-weight microservices architecture to build scalable and real-time LVC (Live, Virtual and Constructive) simulations. Specifically, the application model is designed to illustrate how LVC simulation integrates with edge computing and cloud computing. The autonomous negotiation service and time synchronization services are discussed in details. Though it is a preliminary edge computing framework for LVC simulation, no similar research has been conducted for modeling and simulation using edge computing.

Disciplines
Engineering | Science and Technology Studies

Publication Details
Sun, W., Zhai, Y., Bao, T., Mudassar, M., Shen, J. & Yang, K. (2019). A Microservices Oriented Edge Computing Framework for LVC Simulation in the IoT Era. ICCMS 2019: Proceedings of the 11th International Conference on Computer Modeling and Simulation (pp. 190-195). New York, United States: ACM.

Authors
Wenxin Sun, Yanlong Zhai, Tianhong Bao, Muhammad Mudassar, Jun Shen, and Kai Yang

A Microservices Oriented Edge Computing Framework for LVC Simulation in the IoT Era

Wenxin Sun, Yanlong Zhai, Tianhong Bao, Muhammad Mudassar
School of Computer Science, Beijing Institute of Technology
Beijing, China
ylzhai@bit.edu.cn

Jun Shen
School of Computing & Information Technology, University of Wollongong
Wollongong, Australia
jshen@uow.edu.au

Kai Yang
Science and Technology on Special, System Simulation Laboratory, Beijing Simulation Center
Beijing, China
yangkai101@163.com

ABSTRACT

Modeling and simulation are essential methods to better understand a complex system in the real world. Many simulation systems have strong demand for involving physical devices or equipment in the system to improve the fidelity. But most of the current cloud-based simulation systems are designed for equipment with adequate resources and suffering from high latency for their centralized communication. Edge computing is getting much attention as a new paradigm for distributed systems involving heterogeneous devices and requiring real-time communication. However, edge computing is still in its initial stage and there is no consensus on the edge framework and implementation. To the best of our knowledge, no detailed studies have explored the edge-side simulation and modeling. In this paper, we study and design an edge-side simulation framework, which exploits light-weight microservices architecture to build scalable and real-time LVC (Live, Virtual and Constructive) simulations. Specifically, the application model is designed to illustrate how LVC simulation integrates with edge computing and cloud computing. The autonomous negotiation service and time synchronization services are discussed in details. Though it is a preliminary edge computing framework for LVC simulation, no similar research has been conducted for modeling and simulation using edge computing.

KEYWORDS

Edge-side simulation, microservices, LVC simulation, IoT

1 INTRODUCTION

The system simulation technology has been deeply applied in the fields like aerospace, aviation, manufacturing, weapon development, etc. In order to improve the credibility and fidelity, system simulation technology is constantly trying to design and build the simulation system close to the original real system. LVC (Live, Virtual and Constructive) simulation is an important simulation method of "integrating the virtual and real world". Real people and real devices are deployed in the simulation as the *live* part. *Virtual* refers to a real person operating a emulator or simulator. *Constructive* refers to a virtual person operating a computer-generated virtual device and system.[6]. In the era of IoT, various devices and quantities of sensors are used in LVC simulation. Traditional LVC solutions like TENA[11] are mostly designed for adopt a heavier middleware architecture, which is not suitable for the IoT environment with a large number of sensors and lightweight devices[8].

Edge computing and Fog computing are technologies to extend the Cloud to the edge of the network[10]. "Fog computing is a horizontal, system-level architecture that distributes computing, storage, control and networking functions closer to the users along a cloud-to-thing continuum"[1]. Edge computing pushes the localized processing farther to the data sources. Instead of doing the bulk processing in a fog node or IoT gateway, edge computing will handle the processing directly in the programmable automation controllers (PAC)[9]. Compared with the Cloud-based simulation, the edge-side simulation can significantly reduce the data transmission and improve the real-time performance by collaborating among edge nodes locally. We believe that in the IoT era, combining the Cloud-based simulation with the edge-side simulation is an inevitable trend to achieve real-time, high performance and large-scale complex system simulation. In this paper, we focus on the specific requirements of edge-side simulation, design and implement a microservices oriented edge-side simulation framework. This framework could integrate edge, fog, and cloud into a uniform platform for the LVC simulation to leverage the advantages of hierarchical computation. Notably, this is the first microservices based edge computing framework for simulation. The theoretical analysis and the primary experimental result indicates that the framework build on light-weight microservices architecture can be deployed on devices with limited resources. The collaboration among edge-side simulation models can achieve better performance than connecting to the centralized backend cloud. This framework is a valuable reference for the modeling and simulation field.

2 RELATED WORK

2.1 Edge/Fog Computing

Edge Computing is a recent paradigm, which moves computing application and services from centralized units into the logical extremes or at the closest locations to the source and provides data processing power there. Fog computing is an architecture that uses

edge devices to carry out a substantial amount of computation, storage, communication locally and routed over the internet backbone, and most definitively has input and output from the physical world[5]. Fog computing consists of Edge nodes directly performing physical input and output often to achieve sensor input, display output, or full closed loop process control, at the Edge or nearer to the Edge than centralized Clouds residing in very large data centers. In one hand, fog computing places edge nodes at the edge of the network, providing attractive computing capabilities, storage capacity, network services for edge devices, reducing application processing time for edge devices, and improving quality of service (QoS)[12]. In the other hand, the edge computing makes full use of the computing and storage capabilities of the edge network, which improves the degree of participation of edge network devices and sensors in computing, and improves the processing capabilities of cloud computing on the edge.

2.2 Microservices Architecture

These years, microservices have gained much attention in industry. Microservices shared similar concept with SOA (service oriented architecture) except more fine-grained independent services are encouraged, whereas SOA more focused on compositing services to build a complex business process and integrate different business system[2]. The central concept behind microservices is that functions of an application are broken down into smaller, composable component and expose the function as lightweight services using techniques like REST api[13]. This benefits the applications to a large extent in building and maintaining over the life of the application. Edge computing relies on the integration of many IoT devices from different providers and domains. It is considered to be a suitable solution to build IoT platforms based on microservices architecture[4]. Some research work has proposed several microservices based frameworks for IoT[3].

3 MICROSERVICE ORIENTED EDGE COMPUTING FRAMEWORK

The proposed edge computing platform is designed following the SOA paradigm. Basic function is encapsulated into atomic microservice, which will be further composited into composite services according to some business logic and rules. Compared with SOA, Edge computing has some similar requirements and some special features also[2]. Such as the hardware environment for edge computing is very different from the SOA. The development of IoT brings a tremendous number of devices and different M2M (machine to machine) communication technologies.

3.1 Application Mode of Edge-side Simulation

According to the intentional analysis of the edge-side simulation described above, the edge-side simulation mode is designed as shown in Figure 1. The edge-side simulation architecture can be divided into three levels: the top-level cloud simulation environment, the middle-level fog simulation environment, and the bottom edge simulation environment. The cloud simulation environment has sufficient storage and computing resources and is suitable for running large-scale complex calculations, such as some computing models of computer-generated complex equipment (Constructive). The fog simulation environment has more resources and functions than common edge devices and can be used to handle calculation of the simulation model and simulation tasks for different simulation domains. The edge simulation environment is directly connected to sensors and various actual equipment or simulators (Live and Virtual), which can preliminary process and compute the data of sensors and devices to complete local simulation. If the edge devices cannot complete the simulation task, it can find the adjacent capable edge nodes or fog nodes through the workload offloading and discovery services, and then establish the cooperation mode through the negotiation mechanism. As shown in Figure 1, a local real-time simulation environment is established between the fog layer and edge layer. Edge nodes and fog nodes can autonomously discovery and negotiate to collaborate and distribute data using local network. This can gain better real-time performance of the simulation. Some computation intensive simulation models will be deployed in the cloud to leverage the computation power. The data generated in the edge layer is processed and filtered before sending to the fog and cloud layer. This significantly reduces the amount of data that must be transmitted to the cloud. After the calculation and analysis in the cloud center, the intelligent information and specific instructions are generated and transmitted back to the edge layer.

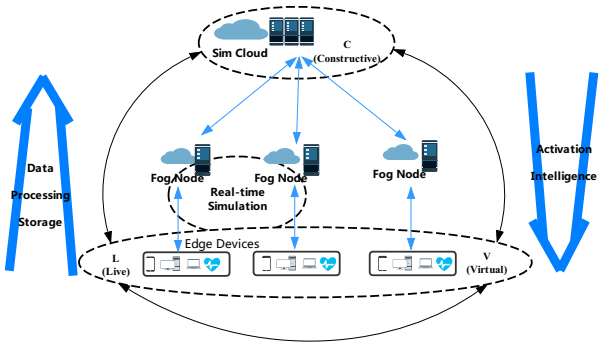

Figure 1: The Edge-side LVC Simulation Mode

3.2 Framework

In order to combine edge computing, fog computing, and cloud computing with the simulation technology to achieve the requirements of the edge-side simulation platform, a hierarchical edge-side simulation framework is designed. As shown in Figure 2, it is divided into application deploy, simulation services, microservice, communication, hardware and IoT layers. By using microservices architecture and supporting various M2M communication technologies, the framework can be efficiently deployed on Edge nodes, Fog nodes, PCs, and Cloud.

3.2.1 Application Deployment Layer. The application deployment layer provides tools and application interfaces for simulation system developers. Developers can use the workflow based tool to design the simulation logic according to the requirements of the simulation.

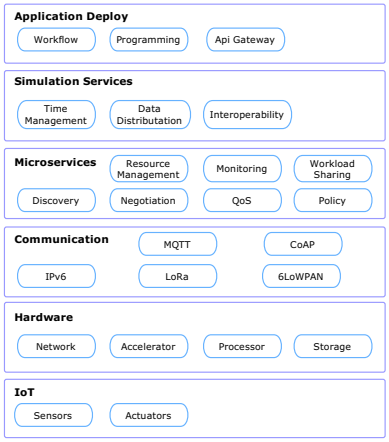

Figure 2: Microservice-based Edge-side Simulation Framework

The Api Gateway is also deployed in this layer for developers to utilize the microservices provided by the framework.

3.2.2 Simulation Services Layer. The simulation layer is the logical implementation of the application deployment layer. It provides time management services, data distribution services, and interoperability services to implement basic simulation logic and shield the transmission delay and time of waiting for execution between edge nodes. The time management services are indispensable for the simulation system. Time management balances the relationship between simulation time and physical time and interoperates with multiple edge nodes to guarantee the time constraints. This is similar to the time management requirements in the HLA (High Level Architecture) distributed interactive simulation architecture. Therefore, the implementation of the time management services could reference the time management services in the HLA simulation system, including the message distribution sequence, the logical time promotion protocol, and the transmission service. The data distribution services preprocess data through a filtering mechanism and distribute the data involved in the calculation to only the corresponding edge nodes, thereby reducing the redundancy and repetition of sending data. Interoperate services include the simulation execution management services and the simulation member registration services. The execution management services include the basic running instructions such as the start and end of simulation tasks, while the registration services are the registration operation of multiple simulation models to join and exit the simulation.

3.2.3 Microservice Layer. The microservice layer provides basic services of fog computing and edge computing including resource management services, monitoring services, workload sharing services, discovery services, communication services, QoS services, and policy services. This layer aims to make full use of the computing resources and storage resources of the edge nodes to provide fundamental services of an edge computing framework. The resource management services manage the resources of the edge nodes. By setting the resource scheduling of the edge nodes, the resource is removed and loaded according to the heat of the resource requested by the mobile/edge devices, which can improve effectively the resource utilization and reduce transmission costs. In the context of system simulation, the resource management services will allocate computing resources and storage resources for simulation models and simulation tasks. The monitoring service monitors the application execution status. When the workload or some predefined parameters exceed the specified threshold, the discovery services are used to find the available edge nodes. The policy services and QoS services will find the most suitable edge node to transfer the task using the workload sharing services. Edge computing is different from the centralized cloud computing, edge nodes can interact with other edge nodes directly through M2M connection without communicating with the cloud center. The negotiation services provide methods for the autonomous interaction between different nodes to establish the interaction context and negotiate on some predefined topics.

3.2.4 Communication Layer. The communication layer provides communication interfaces for edge nodes to connect with edge nodes, fog nodes, and the cloud. An extensible architecture is required to support different M2M communication technologies. Some protocols such as LoRa and 6LoWPAN are designed for IoT applications. This framework focuses on supporting the application(session) protocols. CoAP is deployed to support request/response communications and MQTT is deployed to support Pub/Sub communications. Most of the microservices will expose their service interface as CoAP endpoints.

3.2.5 Hardware Layer. The hardware layer provides services based on the operating systems of the devices. The framework is integrated with the operating systems of the real devices, and the operating system of the device itself is used to provide basic operations, networking, storage, and acceleration functions. This layer shields the heterogeneity between devices uses different operating systems to provide the same services to the upper layers of the framework. In the meanwhile, the implementation of this layer also determines that the framework is a lightweight and easy-to-deploy framework.

3.2.6 IoT Layer. The Internet of Things layer refers to the connection interface between sensors and actuators, including the interfaces of virtual devices and real devices. IoT layer is designed to integrate various sensing and computing devices that are worn by personnel or embedded with equipment. The framework will integrate IoT technologies to drive these sensors and actuators and using these devices to connect with the physical world.

4 FRAMEWORK IMPLEMENTATION

As showed in the layered architecture, we've implement the microservices oriented edge computing framework. The framework uses the Node-RED system for visual edge-side simulation process editing. The communication layer of the edge-side simulation fully uses for reference the technology of the Internet of Things communication layer. It uses the MQTT protocol to support the Pub/Sub data distribution model, and uses the CoAP protocol to implement

lightweight model interaction. We design and implement the framework by integrating MQTT, CoAP and Node-RED, etc. These build the basic IoT platform drive different sensor and support different M2M communication technologies. The microservices layer build the basic edge computing environment and enable the edge devices to collaborate. Because of limited space, we will not discuss the implementation of the whole framework. We only focus on the negotiation services and time synchronization services.

4.1 Autonomous negotiation service

Autonomous negotiation services provide edge computing the ability to automatically build the interoperation context. Edge nodes don't need to be hard coded to cooperate, they can autonomously discover new nearby edge nodes and start a negotiation. If the found peer node is capable and willing to provide the needed services, they set up the context and start to cooperate. Autonomous negotiation is the basis of many other microservice, such as workload sharing and simulation interoperability. WS-negotiation specification provides a reference for implementing the negotiation services[7]. WS-Negotiation defines the negotiation protocol and the interfaces between web services. It contains three parts: negotiation message, which describes the format for messages exchanged among negotiation parties; negotiation protocol, which describes the mechanism and rules that negotiation parties should follow; and negotiation decision making, which is an internal and private decision process based on a system model or other strategies. Figure 3 shows the activiy diagram of the proposed autonomous negotiation services. Similar concept and protocol is designed for edge nodes to establish the interoperation context and negotiate on an offer.

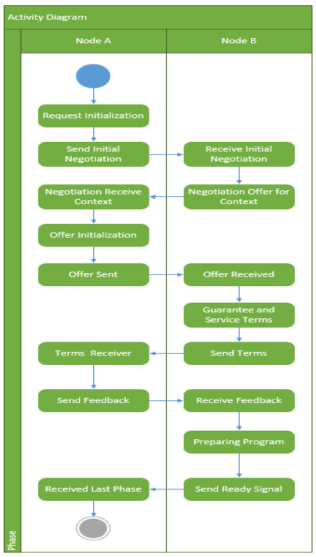

Figure 3: Autonomous Negotiation Activity Diagram

The designed autonomous negotiation mechanism is implemented by a group of Node-RED subflows exposed as CoAP microservices. Currently, we only implement a simple negotiation protocol and the decision making logic is implemented as a user-defined function. The content of the offer, service terms and feedbacks can only be processed by user functions, which makes the negotiation not completely autonomous. But it already capable of negotiating on joining or leaving the simulation, sharing resources like computation and storage.

4.2 TimeSync Services

Time synchronization algorithm is the core of system simulation, which determines the correctness and efficiency of the simulation. To ensure the simulation maintain the time causality, many synchronization algorithms have been proposed, which can be summarized into two categories: conservative synchronization algorithms and optimistic synchronization algorithms [7]. Conservative synchronization allows LPs to only process an event if it is guaranteed that no events with a smaller simulation time can arrive, whereas optimistic synchronization avoids blocked waiting by optimistically processing the events beyond the look-ahead window and provides a roll-back mechanism to recover from causality errors. LVC simulation in edge environment is quite different from typical distributed simulation in terms of simulation models may run in Cloud, PCs, Fog nodes, Edge devices or even sensors instead of running only in PCs or Cloud. Different simulation models will have different real-time constraints, some models have to update the status in microseconds whereas some others in seconds. It's apparent that optimistic synchronization algorithm cannot apply to some LVC simulations, for some hardware devices don't support roll-back. An enhanced null message synchronization algorithm is designed. When the difference in update cycle among models is relatively large, the enhanced algorithm ensures the correctness of the simulation results and greatly reduces the number of messages sent, comparing with the traditional CMB algorithm. Based on the traditional CMB algorithm, the enhanced algorithm optimizes the mechanism of creating and sending the message, which greatly reduces the number of messages; by calculating the time of downstream models' next time step to determine whether it is necessary to send simulation messages to the downstream LP (Logical Process), this significantly reduces the transmission of unnecessary simulation messages, saves hardware resources and improves the simulation efficiency. The implementation of the algorithm has to be omitted in this paper due to limited space. For details of the synchronization algorithm, please refer to our previous paper[14]. Table 1 shows some core time synchronization services in this framework.

5 FRAMEWORK EVALUATION

We deployed the edge-side simulation framework to three lightweight edge nodes and built a simple simulation environment to verify the validity of the framework, as shown in Figure 4. In this simulation environment, different sensors including cameras were deployed with the Cubie Board 2 (ARM board), Orange Pi Pc Plus (ARM board) and Intel Compute Stick (x86 board). The edge network connects to Alibaba Cloud server with 1Mbps connection. The simulation tasks can be deployed at any edge node. With LVC simulation technology, the edge devices can be used as real IoT nodes that participate in the simulation and also run a constructive simulation model and complete it by computing. In this deployment,

Table 1: TimeSync Microservices

Verb	URI	Description
GET	coap://localhost/requestLookahead	Retrieve LP's lookahead time, send the lookahead value to the client.
GET	coap://localhost/requestLBTS	Retrieve LP's LBTS (lower bound time stamp)
GET	coap://localhost/requestNullwithLBTS	Retrieve LP's null message, pass client LP's LBTS as a parameter.
GET	coap://localhost/requestSimMsg	Retrieve simulation message.
POST	coap://localhost/receiveMsg	Receive any message from other LPs.

we simulate a simple C2 (command and control) scenario with one monitoring node(using a camera to collect visual information), one intelligence processing node (using an Intel Movidius neural compute stick to analyze the video) and one assessing node. Before we have the edge-side simulation framework, images need to be sent to the cloud for processing and assessing, which will take a long time for the transmission and have a high latency. Here all the tasks will be handled on the edge, only some processed key information need to be sent to the cloud.

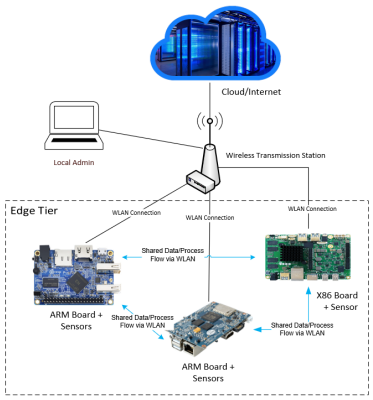

Figure 4: Deploy the Framework on Edge Nodes

As shown in the experiment, we collect 1162MB of video to do the simulation and compared the performance of edge-side simulation with the Cloud-based simulation. The Orange Pi collect the video and send to Intel Compute Stick for analysis in the neural network. The generated result will be sent to Cubie Board for assessing. Whereas in the Cloud-based simulation, videos will be transferred to the Cloud from the Orange Pi directly. The average execution time of Cloud-based simulation is 338 seconds and the average execution time of edge-side simulation is 112 seconds.

6 CONCLUSIONS

A microservices oriented edge-side simulation framework is proposed for building simulation systems on the edge in the age of IoT. By integrating edge, fog, and cloud computing, the light-weight framework is particularly suitable for LVC simulations which usually comprised of hardware models and software models. The most essential feature of this framework is combining the real-time local computing with the high-performance backend computing. The demo deployment and the primary results showed that the framework can work on some constrained edge nodes and improve the performance effectively. For future work, the framework still needs to be enhanced and more microservices like autonomous discovery services, workload offloading services, etc. will be designed and implemented.

ACKNOWLEDGMENT

This work is supported by the National Nature Science Foundation of China (Grant No. 61602037) and Equipment Pre-Research Field Foundation "The New Simulation Approach based on Edge Computing" (Grant No. 61400010104).

REFERENCES

[1] 2017. *OpenFog reference architecture for fog computing*. Technical Report. OpenFog Consortium Architecture Working Group.

[2] Nik Bessis, Xiaojun Zhai, and Stelios Sotiriadis. 2018. Service-Oriented System Engineering. *Future Generation Computer Systems* 80 (March 2018), 211–214. https://doi.org/10.1016/j.future.2017.11.025

[3] B. Butzin, F. Golatowski, and D. Timmermann. 2016. Microservices approach for the internet of things. In *2016 IEEE 21st International Conference on Emerging Technologies and Factory Automation (ETFA)*. 1–6. https://doi.org/10.1109/ETFA.2016.7733707

[4] Lei Chen and Cristofer Englund. 2018. Every Second Counts: Integrating Edge Computing and Service Oriented Architecture for Automatic Emergency Management. https://doi.org/10.1155/2018/7592926

[5] N. Chen, Y. Yang, J. Li, and T. Zhang. 2017. A Fog-based service enablement architecture for cross-domain IoT applications. In *2017 IEEE Fog World Congress (FWC)*. 1–6. https://doi.org/10.1109/FWC.2017.8368533

[6] Douglas D. Hodson and Raymond R. Hill. 2014. The art and science of live, virtual, and constructive simulation for test and analysis. *The Journal of Defense Modeling and Simulation* 11, 2 (April 2014), 77–89. https://doi.org/10.1177/1548512913506620

[7] P. C. K. Hung, Haifei Li, and Jun-Jang Jeng. 2004. WS-Negotiation: an overview of research issues. In *37th Annual Hawaii International Conference on System Sciences, 2004. Proceedings of the.* 10 pp.–. https://doi.org/10.1109/HICSS.2004.1265100

[8] G. Kecskemeti, G. Casale, D. N. Jha, J. Lyon, and R. Ranjan. 2017. Modelling and Simulation Challenges in Internet of Things. *IEEE Cloud Computing* 4, 1 (Jan. 2017), 62–69. https://doi.org/10.1109/MCC.2017.18

[9] Chao Li, Yushu Xue, Jing Wang, Weigong Zhang, and Tao Li. 2018. Edge-Oriented Computing Paradigms: A Survey on Architecture Design and System Management. *ACM Comput. Surv.* 51, 2 (April 2018), 39:1–39:34. https://doi.org/10.1145/3154815

[10] C. Mouradian, D. Naboulsi, S. Yangui, R. H. Glitho, M. J. Morrow, and P. A. Polakos. 2018. A Comprehensive Survey on Fog Computing: State-of-the-Art and Research Challenges. *IEEE Communications Surveys Tutorials* 20, 1 (2018), 416–464. https://doi.org/10.1109/COMST.2017.2771153

[11] J. R. Noseworthy. 2008. The Test and Training Enabling Architecture (TENA) Supporting the Decentralized Development of Distributed Applications and LVC Simulations. In *2008 12th IEEE/ACM International Symposium on Distributed Simulation and Real-Time Applications*. 259–268. https://doi.org/10.1109/DS-RT.2008.35

[12] Cássio Prazeres, Jurandir Barbosa, Leandro Andrade, and Martin Serrano. 2017. Design and Implementation of a Message-service Oriented Middleware for Fog of Things Platforms. In *Proceedings of the Symposium on Applied Computing (SAC '17)*. ACM, New York, NY, USA, 1814–1819. https://doi.org/10.1145/3019612.3019820

[13] L. Sun, Y. Li, and R. A. Memon. 2017. An open IoT framework based on microservices architecture. *China Communications* 14, 2 (Feb. 2017), 154–162. https://doi.org/10.1109/CC.2017.7868163

[14] B. Wang, Y. Zhai, Z. Wang, H. Zhang, and D. Qing. 2016. Enhanced null message algorithm for hybrid parallel simulation systems with large disparity in time step. In *2016 IEEE/ACM 20th International Symposium on Distributed Simulation and Real Time Applications (DS-RT)*. 61–68. https://doi.org/10.1109/DS-RT.2016.31