Regular Paper

Performance Evaluation of a CyReal Sensor System

Kei Hiroi[†], Akihito Kohiga[‡], and Yoichi Shinoda[‡]

[†]Disaster Prevention Research Institute, Kyoto University, Japan
[‡]Japan Advanced Institute of Science and Technology, Japan
hiroi@dimsis.dpri.kyoto-u.ac.jp

Abstract - IoT devices are expected to enable inexpensive and easy measurement and collection of a wide range of environmental information, especially in the field of disaster prevention. Whereas, preliminary verification is difficult, because of a functional design that assumes their distributed deployment, the cost of development itself. Therefore, we develop a sensor system emulator with CyReal concept, which integrates the virtual device with the actual device, and federated with other simulators. This paper presents a prototype of our sensor system, its feasible performance, and discusses a design for flexible integration, in order to figure out that our sensor system supports large-scale virtual sensor devices for verification of the development, update, debugging, and operation of sensor systems as a distribution network for disaster prevention information.

Keywords: sensor network, emulation system, sensor testbed

1 INTRODUCTION

IoT (Internet of Things) devices equipped with sensors and related technologies are expected to enable inexpensive and easy measurement and collection of a wide range of environmental information. In the field of disaster prevention, they are being utilized for various fields such as river observation and slope measurement, and are expected to be useful for collecting data at points where it has been difficult to figure out the condition of the environment.

Conventional environmental monitoring for disaster response is usually installed at a few vulnerable sites that require monitoring. The environmental observation had been carried out through a robust monitoring system using a leased network lines. However, due to the frequent and large-scale river floods in recent years, we face on pressing importance requiring a larger-scale monitoring network, even for small rivers and waterways that have not been monitored well in the past. An observation network using IoT has the potential to minimize the cost of implementation and operation. This implies that a large-scale observation system can be applied.

However, installation of a large number of sensor devices creates a new problem. IoT devices are expected to be used in urban areas and mountainous regions. Devices can be installed in large numbers to provide valuable measurements that have not been possible with conventional observation systems. Whereas, installing a large number of devices over a wide area makes it complicated to handle them and difficult to improve them by relocating them after installation. Especially in data measurement where the relationship between devices is meaningful, it is important to design functions and consider how to utilize them with distributed deployment. Preliminary verification in this paper indicates to test the operation of the entire sensor network before installation, to find bottlenecks, and to consider solutions based on this functional design. By conducting preliminary verification before installing sensor devices, we can clarify data from the unique characteristics of sensor devices and the relationship between data from multiple sensor devices, as well as the transmission characteristics of data due to terrain and communication infrastructure, and use these results to design measurements that satisfy the purpose of use. Nevertheless, since a large number of devices are required, it is difficult to verify the functions in advance by preparing the necessary number of actual devices, which increases the cost of development itself and improving such as relocation. Although various sensor emulators have been developed to enable such preliminary verification, they are limited to verification of some functions such as network performance and device performance after using virtual devices.

In this study, we develop a sensor emulator system to support research, development, and operation of disaster prevention information collection and operation, and to enable preliminary verification assuming a specific installation environment and operational configuration. A sensor emulator is a virtualization technology that emulates the computers performance involved in sensor devices and their functions. In this research, we develop an emulator system that can exchange the virtual device with the actual device and can be federated with other simulators in order to support the development, update, debugging, and operation of sensor systems as a distribution network for disaster prevention information. By separating the Node and Sensor, and incorporating a connection mechanism between the virtual and physical devices for each of them, we can verify the functions of a large-scale sensor device. By enabling connection and verification not only with the virtual device but also with the physical device, it is possible to assume functions and communication environments that cannot be demonstrated with the virtual device, and it also facilitates connection to the cloud, thereby reducing operating costs and enabling multiple use of resources based on the assumption of an actual operating environment. Federation with other simulators can be possible through data exchange using the IoT linkage infrastructure. Through collaboration, data that assumes actual operation can be incorporated into the sensor, and preliminary verification, including operational forms such as disaster response based on data collection, becomes possible.

2 RELATED WORKS

Sensor emulator is a powerful tool that helps researchers and users to consider the design of sensor networks. An enormous amount of effort has been invested in developing emulators for various technologies related to sensor networks; communication protocols, computer processes, application software. As the number of sensors increases with the spread of IoT, the need to handle a large number of sensors has led to the development of a number of sensor virtualization technologies. SenaaS [1] is IoT virtualization framework to support connected objects sensor event processing and reasoning. This framework provide an ontology design by a semantic overlay of underlying IoT cloud and a policy-based service access mechanism in terms of semantic rules. Bose et al. [2] have presented resource abstraction at the sensor level on Sensor-Cloud infrastructure with virtualization of sensors for developing applications in various fields. This is a design for virtual sensor on cloud station. SenseWrap [3] is a middleware architecture providing virtual sensors as representatives for any type of physical sensors. This midlleware supports sensor-hosted services and a standardized communication interface that applications can use without having to deal with sensor-specific details.

Wireless networks, an indispensable technology for sensor networks, are also subject to emulation. TOSSIM [4] focuses on simulating a wide range of network interactions. TOSSIM, which features a high fidelity and scalability, can capture network behavior while scaling to thousands of nodes. COOJA [5] is a sensor network emulator aimed at cross-level simulation, allowing simultaneous simulation at many levels of the system; sensor node platforms, operating system software, radio transceivers, and radio transmission models. Many other emulators for sensor network verification have been researched and developed, such as EmStar [6], Avrora [7], and J-Sim [8]. These sensor emulators have been developed on the premise of wireless sensor networks. Recently, based on recent developments in low power wide area networks, including the rise of Long Range (LoRa) technology, a LoRa Coverage Emulator [9] has also been developed. This LoRa emulator consists of a transmitter and receiver and provides a reliable network coverage estimation based on the LoRa network design framework.

There are many researches on emulators from Hardware / Software point of view as well as network. The Freemote Emulator [10] is an emulator for developing software for nodes. It provides developers with a system architecture in several layers: Physical, Data Link (MAC), Routing and Application. Similarly, ATEMU [11] is a well-known sensor network emulator with a lot of contributions. A unique feature of ATEMU, which can operate on different application/hardware platforms, is its ability to simulate a heterogeneous sensor network. ATEMU emulates the processor, radio interface, timers, LEDs and other devices. Kasprowicz et al. [12] focused on CCD sensors as devices and developed a hardware emulator to speed up and streamline post-assembling tests and debugging. Furthermore, research on emulators has also focused on commoditization, with an emphasis on lightweight and small IoT systems. Brady et al. [13] developed an emulator for an IoT environment using the popular QEMU system emulator to build a testbed of inter-connected, emulated Raspberry Pi devices. The effort to emulate functionality extends to applications that anticipate not only specific devices, but also power supply and utilization. Deda et al. [16] have designed a battery emulator/tester system to reduce development time for developing and testing of high-voltage power supply systems. Abrishambaf et al. [14] have developed a laboratory emulation model for energy scheduling in an agriculture system using real nodes.

The conventional emulators so far can be said to be emulators that specialize in a certain function of the computer. By using these technologies, hardware, software, and network can be emulated in an integrated manner. However, with the evolution of IoT, we have to treat power supply, heterogeneous sensor devices, and network devices. We need to provide an operator-friendly verification environment. The benefits of the IoT have increased the opportunities for sensing technologies to be more readily available to a wider user. And this advantage means that sensor networks can be built more widely and in more places than ever before. Verifying the required functionality with a few emulators may provide the required results. Althogh, for actual operators whose work is on the use of application services, it is very difficult to verify the functions in isolation. Nonetheless, emulators that require special technology are considered to have little affinity with this kind of operators. Also, for operators, when considering many things that could not be verified with previous emulators, such as geospatial and distributed installation for business efficiency, these requirements are not considered with previous emulators.

3 SENSOR SYSTEMS BY CYREAL EMULATOR

3.1 Overview

The sensor emulator system developed in this paper enables preliminary verification of IoT devices based on the assumption of their specific installation environment and operational configuration. Based on the concept of CyReal, this sensor emulator system is designed to support the development, update, debugging, and operation of sensor systems, so that the virtual machine and the actual machine can be exchanged and can be federated with other simulators.

3.2 System Requirement

We aim to develop a verification environment for sensor systems for actual operators whose work is on the use of application services. What is the most user-friendly verification environment for operators? Our definition is an emulator that does not require any special skills and produces output that is directly relevant to thier work. With the development of IoT, there is a need to master the power supply, different types of sensor devices, and network devices. What is important for operators who work with IoT-based applications and services is that the data from the IoT is stable and their applications and services can work smoothly.

In case of a disaster, there is a risk of failure of the sensor system, power supply to operate the sensor system and obtain data, and communication problems. Thus, it is necessary to verify that the operation of the system as a whole is stable, not only the operation of the IoT sensor system, but also the power supply and communication status. Further, in considering how to improve the efficiency of their work, it is essential for operators to verify this in light of utilization, such as device locations in geographical environments and distributed installations that address disaster vulnerabilities. Our primary focus in this verification environment is to capitalize on the IoT sensor devices currently being operated by operators. This is because using real installed and operating devices as part of the emulator not only enables preliminary verification including actual performance, but also aims to operate as a real-time emulator in the future. For these reasons, we develop a verification environment that is compatible with real systems and can handle the large scale of IoT devices.

We defined the requirements for a sensor emulator system to meet these requirements as follows.

- The sensor system emulator should be compatible with a real system.

- The sensor system emulator should be able to handle a large number of sensors.

To meet the above requirements, we develop an emulator that incorporates the concept of CyReal.

3.3 CyReal Approach

CyReal is intended to be an entity that plays an intermediate role in the concept of digital twin [15]. CyReal enables flexible replacement and integration of real and virtual entities, such as computer systems, people, and environments. The configuration based on CyReal strongly promotes the digital twin of disaster prevention IT systems. Digital twin refers to the creation of digital objects to be handled in the real world and computer systems. This is the concept of debugging various properties on the created twin, in the case of computers, and is an important concept in the AI world.

Namely, the configuration on the left in Fig. 1 is one in which all subsystems are configured by simulators, and the subsystems are connected via a federation platform. On the other hand, the various simulations and systems that configure this platform can all be replaced with real things, real systems. The configuration on the right side of Fig. 1 shows a situation in which the subsystems are all real and in actual operation, such as people, natural phenomena, and a real disaster prevention information system. These two are in a digital twin relationship. Then, we have further extended the digital twin concept to allow real systems and simulators to be integrateded in a subsystem (The center of Fig. 1). This concept is beginning to be called CyReal, as an integration of Cyber and Real.

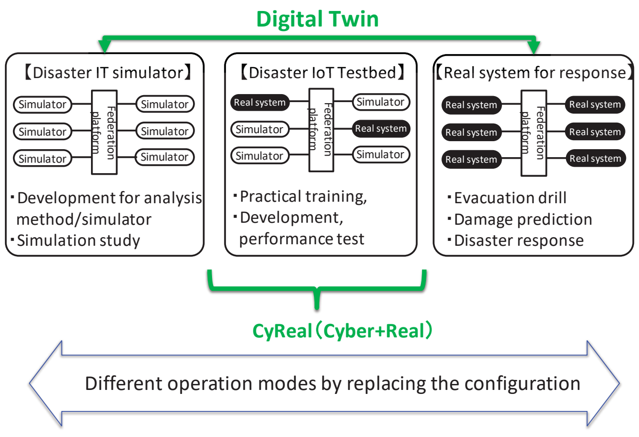


Figure 1: CyReal Approach for Sensor System

3.4 CyReal Sensor System

The sensor system we are developing is configured in accordance with the CyReal concept. The sensors are connected as subsystems in Fig. 1. The sensors can be replaced by real or virtual ones. We expect that this CyRealization allows the subsystems to work in various ways and the system to be used for various purposes. That is, depending on the system we replace, this configuration can be transformed into a system with various purposes.

If the system is entirely composed of simulators, it will work as a disaster management IT simulator. For example, new analysis technologies and simulators can be connected to the disaster prevention IT simulator, and all simulators can be run based on data from past disasters. Since data is difficult to collect in disaster research, evaluating the performance of analysis technologies and simulators is a difficult and costly task. By creating this disaster prevention IT simulator, we have the prospect of providing an evaluation environment and facilitating development. Alternatively, if the subsystems are replaced with real ones (i.e., if the sensor system is replaced with a real one), it becomes a test bed that can be used for research and development and performance evaluation of the sensor system.

This configuration eliminates the limitation of devices, simulators, and systems that can be verified, which is a requirement of this paper. In other words, we can connect not only sensor devices, but also geospatial and operational simulations to verify the functionality and effectiveness of the system in actual operations. In order to develop a system that can flexibly switch between these three modes, we have embarked on a sensor system as one of the proofs of concept for the digital twin and CyReal that bridges the gap between Real and Virtual.

3.5 Significance of CyReal Sensor System

This sensor system is not a simple sensor virtualization. We have designed the system to anticipate the actual data utilization of the sensors. Conventional sensor virtualization has only simulated the data and functions involved in sensing. On the other hand, we not only simulate the data, but also develop the sensor device and its environment as an instance.

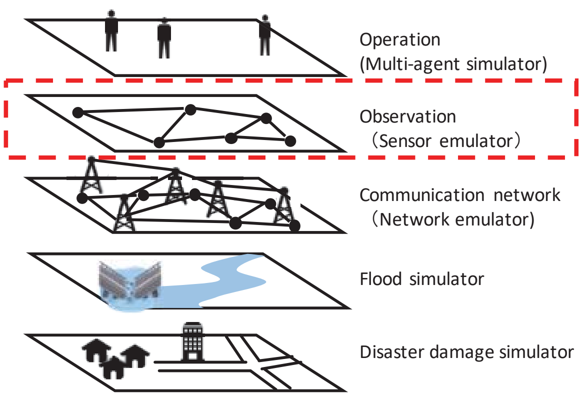


Figure 2: Federation with External Simulators

Our expected applications are as follows. Data missing is an unavoidable problem in large-scale sensor deployments. It is difficult for a system that only simulates data to represent the handling of such missing data. Our system design is able to produce data by incorporating not only the sensor device itself, but also the external environment, such as the wireless environment and the conditions of the location where the sensor is installed. This makes it possible to represent fluctuations in the data due to the influence of the external environment(Fig. 2). In addition, data diversity can be expressed by integrating with external simulators, such as simulators for operations and simulators for natural phenomena. This is the advantage of an emulator that can replace and integrate actual and virtual nodes and sensors. Such a sensor system has many uses in verifying operations, but it also has many challenges. This paper shows proof-of-concept for a sensor system according to this concept, investigates its performance when running virtual sensors on a large scale, and consider the challenges of implementation.

3.6 Structure of CyReal Sensor System

Based on the CyReal concept, we have developed a sensor emulator system that can connect and verify not only virtual devices but also real devices. A sensor emulator is a kind of virtualization technology that enables preliminaly verification of sensor-related systems, such as network performance, device performance, and computer processing performance. The sensor emulator is used for functional verification using virtual sensor devices. Conventional technologies and prior research to date have only supported virtual sensor devices, so that it is difficult to conduct verification based on actual operational environments. Recent IoT devices and related technologies are said to make it possible to distribute sensors over a wide area and to measure and collect data easily and inexpensively. However, in reality, it is difficult to design functions assuming distributed deployment and to prepare the necessary number of actual devices for functional verification in advance, and this has the disadvantage of leaving the decision to the user. Therefore, we have attempted to develop a CyReal sensor system to support research and development and operations related to the collection and distribution of disaster prevention information. In Fig. 3, Sensor refers to a measurement device and Node refers to a computer that processes and communicates measurement data. Here, both sensor and node are designed to be interchangeable between actual and virtual machines. Furthermore, the system can be communicated with other simulators. This is based on data from past disasters, and is intended to be extended to verification based on actual situations such as power outages, device failures, and network disconnection.

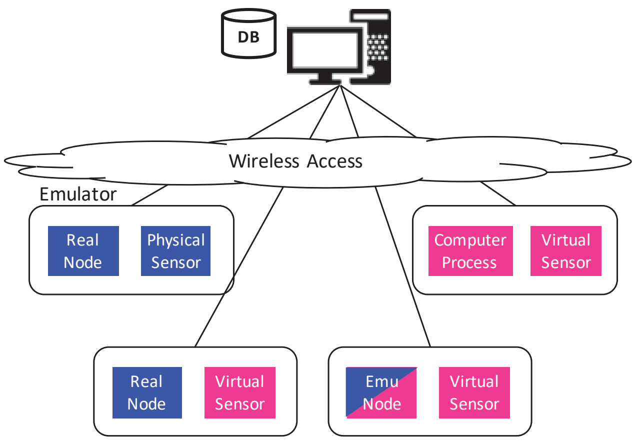


Figure 3: Structure of CyReal Sensor System

4 IMPLEMENTATION AND PERFORMANCE TEST

4.1 Overview

The implementation of this paper is based on two points. The first point is the use of existing IT technologies. The first point is that we use existing IT technology, using AWS, a commercial IoT cloud, to aggregate and process data without preparing any onpremise servers. This allows us to consolidate and process data without any servers. Since there are no servers, there is no need for maintenance. This is an attempt to minimize the cost of implementation and operation.

The second point is API unification. By using SigFox and AWS, whose specifications are open to the public, we can take advantage of the functions that already exist. Also, as an open system, others can take advantage of this mechanism. By using the existing functions, we can simplify the development of the system, and finally, the internal structure is quite simple.

The data from the CyRealized sensor emulator system is currently only recorded in the slack as a logger. If this data is put into a DB, it can be displayed in various UIs and used in applications. In addition, we are planning and implementing data exchange with other simulations.

4.2 Implementation

Figure 4 shows a prototype configuration of the sensor emulator system we have implemented. From Real Node, we used Sigfox, an IoT network and cloud to aggregate the measurement data and put it into the AWS cloud. We used an Arduino Uno as the hardware for Real Node. The Arduino

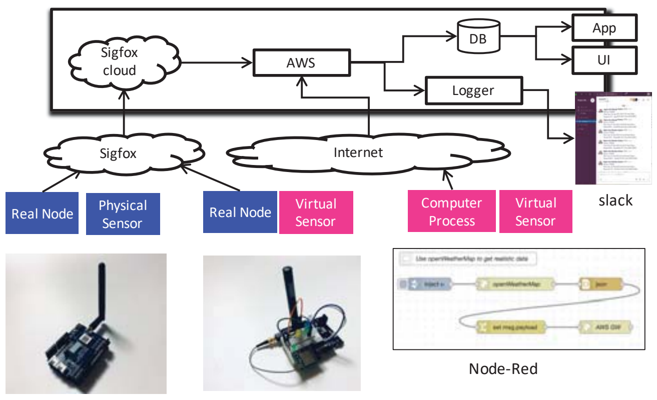


Figure 4: Prototype Configuration

Uno is equipped with a Physical Sensor that measures temperature, humidity, and air pressure. The Arduino Uno sends these data to the AWS API gateway via the Sigfox una shield. Although the hardware used in this paper is an Arduino Uno, it also has the commutativity to connect data obtained from other devices.

To show the commutativity, we use a Raspberry pi as another Real Node. The Raspberry pi collect the data from the Virtual Sensor. In this proof of concept, data is retrieved from Open Weather Map[1], which is a weather information API. We use Node-Red to retrieve the data and send it to the AWS API gateway using the flow shown in Fig. 4. We are using AWS as a Serverless service. The physical/virtual sensor data sent to the AWS API gateway is sent to the Slack webhook API via AWS Lambda. Figure 4 shows the data sent to Slack and recorded.

In addition, we created a combination of a computer process and a Virtual Sensor as a Virtual Node. The sensor data generated by the Virtual Sensor is sent to AWS via the computer process. Using this function, we can of course incorporate open weather information published on the Internet as sensor data. In addition, sensor data generated by rainfall simulation and inundation prediction simulation can be used as data for the entire system. In addition, sensor data generated by rainfall simulation and inundation prediction simulation can be used as data for the entire system. With this configuration, we are demonstrating the CyReal environment that can operate simultaneously heterogeneous computers, as a Real/Virtual Node and a Physical/Virtual Sensor.

4.3 Performance Test and Discussion

We have proposed and implemented a design using SigFox and AWS for a CyReal sensor system. In 4.3, we conduct a test to measure the performance of this sensor system. The performance test focuses on the sensor system connected with only virtual devices. Considering that preliminary verification of these devices can be conducted on this sensor system, it is necessary to investigate the performance of scalability to show how large an emulation can be performed using this sensor system. Therefore, this paper sets up virtual devices on our sensor systems and tests the following evaluation points using the number of virtual sensors as a parameter.

[1]Open Weather Map: https://openweathermap.org/

Table 1: Performance Test Scenario

Scenario	Duration(min)	Time interval(s)
A	100	1.0
B	30,60,120,180,240	0.5, 1.0
C	30	60.0

- Performance of a large scale installation of virtual sensors
- Limitations of devices, simulators, and systems, and discussion based on actual operations

4.3.1 Setup

For the performance test, we use a computer process and virtual sensors implemented in Python. As the Ministry of Land, Infrastructure, Transport and Tourism is working to install 3,000 water level sensor devices in small and medium-sized rivers, and in addition to the Japan Meteorological Agency, many companies have installed rainfall observation sensor devices, with each company operating about 100-400 sensor devices. As a result, nearly 4,000 sensor devices are expected to be in constant use in the field of disaster prevention. Thus we configure the virtual sensors on a scale of 2^n $(n = 0 - 16)$. Data is transmitted via MQTT.

For performance testing, we experimented and discussed two evaluation points. In the first test, we calculate CPU usage and processing time when emulating the CyReal sensor system on a PC. This test is to investigate that this sensor system can handle large scale virtual sensors. Secondly, we discuss whether the requirements in this paper have been implemented and further for future practical use. Both the emulated and virtualized environments are run on a PC with a 2.4 GHz Intel Core i9 (4 cores / 8 threads), 64 GB RAM, under MacOS 10.15.7 64-bit.

4.3.2 Performance Test Scenario

In the future, we aim to federate the sensor system with other simulators in order to obtain useful results for disaster response. The purpose of federating with simulators is to verify the operation of the entire disaster response system, including the operation of the IoT sensor system, power supply, and communication network status. Moreover, the sensor system is intended to be a verification environment for operators to consider how to improve the efficiency of their disaster response work. For the performance test, we prepared three scenarios, Scenario A)-C).

Scenario A) Scenario A) assumes 100 minutes of sensing time duration at 1 minute intervals, and all virtual sensors transmitted this data every second for emulation. This is a test to estimate the performance and processing time when federating an emulator that uses sensor system data with other simulaters for efficiency improvement of operation and damage prediction. Simulations for efficiency improvement and prediction can be repeated many times by changing the predicted values of sensor data and simulator parameters to calculate results for various situations. Obviously, the faster the processing time is, the greater the number of possible trials in a limited time period. For this reason, it is desirable to have a fast processing time, although it is necessary to achieve the utmost performance possible with a commodity computer in light of its utilization. The purpose of Scenario A) is to investigate the performance of a large-scale operation using a commodity computer with ordinary performance.

Scenario B) In Scenario B), we prepare several different sensing time durations as shown in Table 1, in order to investigate the performance for each of them. When the sensing time duration is 30 minutes, we assume a short simulation. For example, it can be used to analyze the prediction accuracy of flood simulations immediately before a disaster, or to simulate effective emergency measures in the operations department. With a sensing time duration of 120 minutes, it is assumed that the simulation will be used for predictions based on a flood simulation and emergency measures based on the results of the predictions, or for the simulation of emergency measures at the disaster site in case of sensor device or network failure. In the case of a relatively long sensing time duration, such as 300 minutes, we plan to simulate operations when long-term warning is required due to typhoons or long rains, and scenarios of sensor device and network failures and recovery. We will examine the performance of such short and long time simulations to find out how much processing time will be available for calcuration. For each sensing time duration, all virtual sensors transmit their data every 0.5 seconds or every 1 second. We assume 100 minutes of sensing scenario at 1 minute intervals, and all virtual sensors sent this data every second for emulation.

Scenario C) Then, in Scenario C, we estimate the performance of the sensor system in real time. In Scenarios A and B, data generated every minute was transmitted at one-second time interval. Our objective is to verify the system in a preliminary manner by federating with other simulators such as disaster response and damage prediction. Scenario C measures performance by connecting to real sensor devices and operating in a integrated environment with virtual sensors. That is, we verify whether our sensor system is capable of handling a large number of sensor devices in a integrated environment with real sensors. This is the scenario in which the CyReal sensor system shows its full potential. Here, data is transmitted every minute with a time duration of 30 minutes.

4.3.3 Result and Discussion

Scenario A) The Scenario A result obtained for the performance of the large scale installation of virtual sensors are shown in Fig. 5. This figure shows the maximum CPU usage for $n = 0 - 16$ with the number of virtual sensors as 2^n sensor devices. When the number of sensors was set to 2^0, the CPU usage was 6.8%, and the transmission time was about 90 seconds. Then, the number of sensors was set to 2^9, the CPU usage reached 97.64%. The processing time for transmission

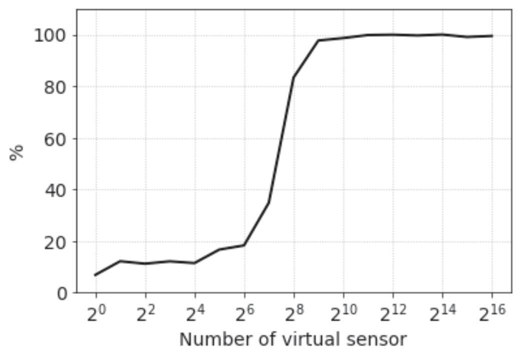


Figure 5: Scenario A) Performance (CPU usage)

in this case was 119 seconds. For $n = 10 - 16$, the CPU usage was 98.58-99.95%, almost 100%. The processing time was about 120 seconds up to the 2^{11} unit, gradually increased as the number of sensor devices increased. It reached 212 seconds with 2^{12} sensor devices, which is close to the number of 4,000 sensor devices that will be operated for disaster prevention in the near future. 1,071 seconds with 2^{14} devices, and 2,528 seconds with 2^{16} devices.

Considering the use of such a sensor system emulator during a disaster situation, two patterns can be considered: preliminaly verification, and scenarios that require real-time data verification. In the case of preliminaly verification, there is no time limitation, thus it is possible to use a processing speed of $n = 14$ or more. In contrast, in the case of real-time use, considering the sensor data missing during a disaster and related actions, it is appropriate to use about $n = 12$. It is certainly that this performance vary depending on the computer performance. However, in the case of disaster response, where it is difficult to use high performance computers, this result is useful to consider usecase scenarios.

Scenario B) Figure 6a-6d show the results of the performance test for five different time duration patterns. Figure 6a shows the processing time for transmitting sensor data with a time interval of 60 seconds, and Fig. 6b shows the CPU usage. Given that the sensing time duration is 30 minutes, the data that is sensing every minute is being transmitted at 60x speed. Therefore, if data is being transmitted and received successfully, all data transmission and receiving would be completed in about 30 seconds. Up to $n = 11$, transmission and receiving were performed in about 30 seconds, whereas as the number of sensors increased to $n = 12$ or more, the processing time required for transmission and receiving increased to 66 seconds. Similarly, in the case of 60 minutes of time duration, if processing proceeds smoothly, all data transmission and reception can be completed in about 60 seconds, while processing time is about doubled for the number of sensors above $n = 13$.

The probable reasons for this are saturation due to failure of virtual sensor process generation due to insufficient memory of the computer used for performance testing, CPU shortage, and overhead of process switching. For the number of sensors above $n = 12$ or $n = 13$, the CPU shortage and the overhead of process switching caused the increase in processing time.

Also, in the case of $n = 16$ with 120 minutes time duration, processing time decreased in spite of the increase in the number of sensors. It is considered that the process that actually did the work was less than the designated process because the transmission process of the virtual sensor was not generated due to insufficient memory. Then, with the number of sensors above $n = 12$ or $n = 13$, correct sensor transmission and receiving were not performed. This means that the commodity computer used for the performance test can be operated with this number of sensor devices or less.

Figure 6c shows the processing time for transmitting sensor data at a shortened to 120x speed, and Fig. 6d shows the CPU usage. In other words, in the case of 30 minutes of time duration, sensor data is transmitted and received every 0.5 seconds, and all data transmission is completed in 15 seconds. In the case of this transmission time interval, data was transmitted and received successfully when $n = 13$, while the processing time increased from $n = 14$. This is a typical method of estimating system performance, and depending on the time interval and time duration, as well as the computer that processes the virtual sensors and the computer of the sensor system, we can determine the number of sensors that are capable of transmission. It is essential to determine the number of sensors according to these verification environments when operating the sensor system in federation with simulation. In the commodity computer environment used for this performance test, the number of sensors that can be operated is about $n = 12$ or $n = 13$. Considering the number of sensors that are actually used in Japan, we found that this number is large enough for preliminary verification.

Scenario C) We measured the processing time of a sensor system that transmits data in real time, namely every minute, when the data is transmitted at 1x speed. Up to $n = 16$, no saturation occurred, and the data was successfully transmitted and received in 30 minutes, the same value as the time duration. This means that it is possible to operate a number of virtual sensor devices with $n = 16$ in this performance test environment. Scenario B), with 30 minutes time duration, was saturated with 2000 processes. Assuming that the overhead of process switching in particular dominates and leads to this result. Since the process switching time is 2000 processes per second, we can assume that it is about 500 [μs] per process. Considering the switching time of 500[μs/process] in a time interval of 60 seconds, the process can be estimated to saturate in $12{\times}10^4$ processes. Figure 7 shows the result of increasing the number of devices to $n = 17$ and transmitting/receiving data. At $n = 16$, the number of virtual sensor devices is 65,536, and even if all sensor devices transmit data simultaneously, the number of processes is less than $12{\times}10^4$ processes. In this case, saturation is not considered to occur. For $n = 17$, the number of virtual sensor devices is 131,072, and the number of processes exceeds $12{\times}10^4$ process. Thus, the process of the sensor system would saturate, and the processing time is expected to be more than 30 minutes.

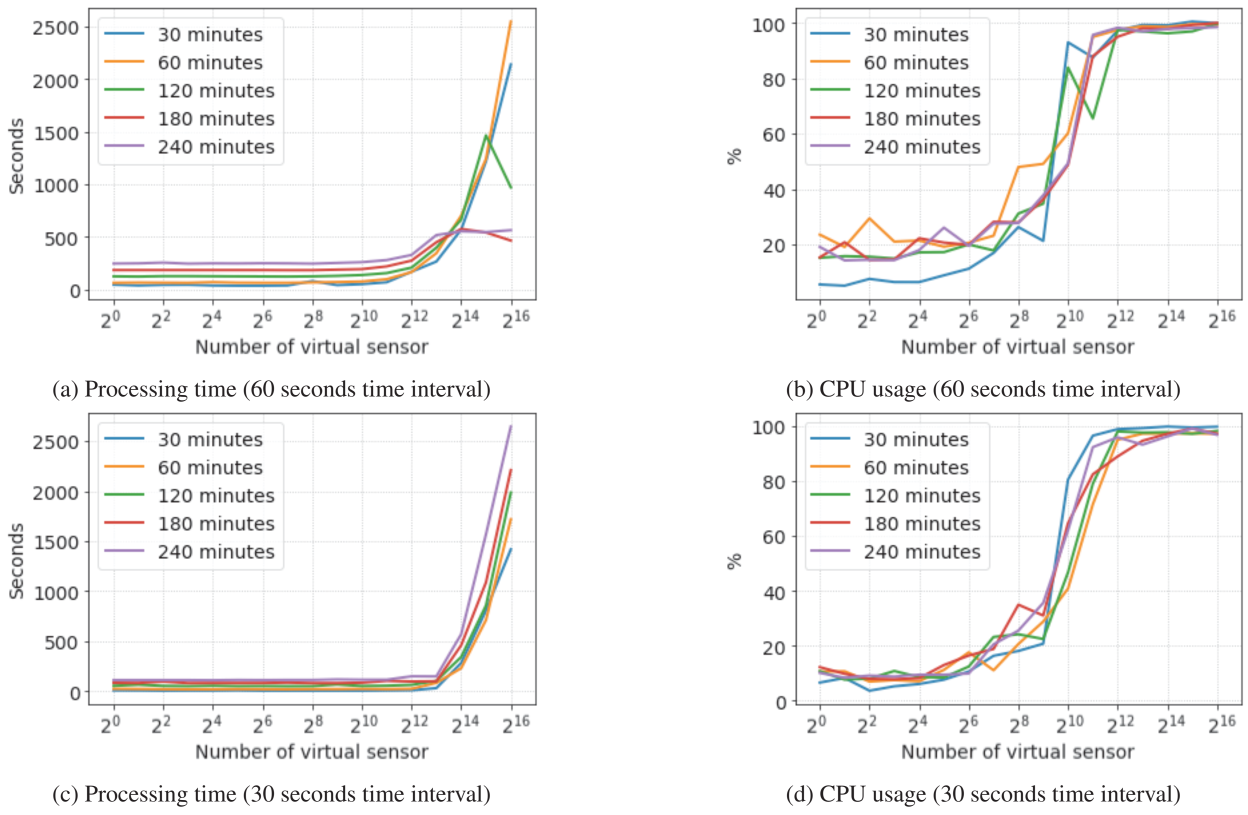
At $n = 16$, the processing time was 1,798 seconds, which is within 30 minutes. Meanwhile, at $n = 17$, the processing time was 2,043 seconds, which is considered to be saturated. This means that if the sensor system is operated in real time with a time dutation of 30 minutes and data transmission every minute, our sensor system supports 65,536 processes. On the other hand, 131,072 processes could not be handled, which is consistent with our estimate of $12{\times}10^4$ processes. This result shows that when the sensor system operates in real time under these environments, our sensor system can operate 2^{16} virtual sensor devices, considering the actual number of devices installed, this number is large enough.

Discussion As for limitations of devices, simulators, and systems, and discussion based on actual operations, we address two functional requirements for the sensor system emulator; The emulator should be compatible with a real system and be able to handle a large number of sensors. Performance tests have shown that it is possible to emulate a large number of sensors in a typical usage environment and the expected number of sensors. The compatibility with real devices already in operation is achieved by implementing the prototype in the configuration shown in Fig. 4. This configuration integrated three types of real/virtual systems: Real Node / Physical Sensor, Real Node / Virtual Sensor, and Computer Process / Virtual Sensor. We have achieved the integration requirement, then we plan to expand the types of sensor devices and communication ports. As a sensor device, we are also planning to incorporate an Emulator Node/Virtual Sensor. By emulating the Node, we will be able to test a wider variety of devices.

The key issues are CyReal-ness and software transparency. For the CyReal-ness, we are incorporating virtual sensors.Then we need to consider the degree of virtuality, and CyReal integration. We will provide more specialized virtualization and integration of hardware, algorithms, data, and its utilization (missing data and heterogeneous sensor data integration), rather than per node or sensor. We believe that this will enable more realistic emulation in large-scale IoT environments. The other is software transparency. The number of nodes and sensors that can be connected to this sensor system is planned to increase. By considering software that can be used in common regardless of the nodes, devices, and systems connected, the sensor system emulator will be able to easily support a large number of sensors. In addition, we will develop other simulators for specific use cases; for example, functions for optimize sensor installation and development support through fedelating a flood simulator, an evacuation simulator, a sensor system emulator for river level sensors and precipitation sensors, or a multi-agent simulator for disaster response, and a network emulator and sensor system emulator.

5 Conclusion

It is very difficult to prepare properly preliminaly verification for a large-scale IoT environment, in consequence, we have not benefits from IoT technologies. To address this problem, this paper proposes a sensor system emulation environment based on the CyReal concept that integrates real and virtual systems. We aim to support research, development, and operation of disaster prevention information collection



(a) Processing time (60 seconds time interval)

(b) CPU usage (60 seconds time interval)

(c) Processing time (30 seconds time interval)

(d) CPU usage (30 seconds time interval)

Figure 6: Scenario B) Performance

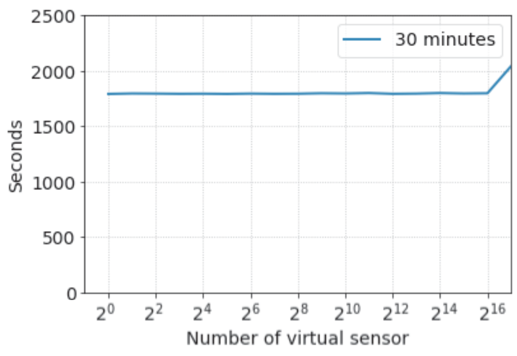


Figure 7: Performance of Sensor System (CPU usage)

and operation, and to enable preliminary verification assuming a specific installation environment and operational configuration. We expect to further develop the sensor system emulator in this paper and for verification by federating with other simulators in order to improve the efficiency of disaster response.

This paper developed a prototype of a sensor system that integrates real and virtual systems. The result of the experiments on the prototype showed the number of devices that can be used as a guide for larger scale in a general use environment, which is sufficient to operate the number of sensors currently used for disaster response. We plan to define CyReal-ness and improve the system to allow selective use of Real / Virtual at several layers: hardware, algorithms, and data. We expect preliminaly verifivcation for IoT systems from both functional and utilization aspects, including more realistic verification such as data missing.

Our objective is to develop a sensor emulator that enables preliminary verification of the behavior of the entire sensor network before installation to identify bottlenecks and determine how to resolve them, based on the premise of functional design. It is intended to be used to design measurements that satisfy the purpose of use by clarifying data due to the unique characteristics of sensor devices, relationships between data from multiple sensor devices, and data transmission characteristics due to terrain and communication infrastructure. In this paper, we have shown that Physical/Virtual Sensor/Nodes can be integrated in a CyReal Sensor System, and the performance of virtual sensors in a typical computational environment. In order to use the CyReal Sensor System for functional design in the future, it is necessary to develop an evaluation environment with large-scale physical devices, and to add functions for various conditions such as network conditions, power consumption, and environmental exposure due to the installation location, and then integrate these functions with the physical and virtual conditions. Even if we consider only the network, the variety of infrastructures available makes it necessary to consider a complex configuration to build these functions on the sensor device. Therefore, we are trying to solve this problem by developing a simulation layer that enables CyReal of network and power on a separate layer from the sensor device, and federating these simulations. Moreover, we plan to develop a verification environment for distributed installation from the perspective of geographical characteristics of sensor installation and utilization through MQTT-based federation with other simulators for external environment.

Acknowledgement

This work was supported by JST, PRESTO Grant Number JPMJPR2036, Japan.

REFERENCES

[1] A. Sarfraz, M. MR Chowdhury, J. Noll, "Senaas: An Event-driven Sensor Virtualization Approach for Internet of Things Cloud", In 2010 IEEE International Conference on Networked Embedded Systems for Enterprise Applications, pp. 1–6 (2010).

[2] S. Bose, A. Gupta, S. Adhikary, N. Mukherjee, "Towards a Sensor-cloud Infrastructure with Sensor Virtualization", In the Second Workshop on Mobile Sensing, Computing and Communication, pp. 25–30 (2015).

[3] P. Evensen, M. Hein, "SenseWrap: A Service Oriented Middleware with Sensor Virtualization and Self-configuration", In 2009 IEEE International Conference on Intelligent Sensors, Sensor Networks and Information Processing (ISSNIP), pp. 261–266 (2009).

[4] P. Levis, N. Lee ,M. Welsh, D. Culler, "Tossim: Accurate and Scalable Simulation of Entire Tinyos Applications, In Computer Communications and Networks", International Conference on Embedded networked sensor systems, pp. 126–137 (2003).

[5] F. Osterlind, A. Dunkels, J. Eriksson, N. Finne, T. Voigt, "Cross-level Sensor Network Simulation with Cooja", In 31st IEEE conference on local computer networks, pp. 641–648 (2006).

[6] L. Girod, N. Ramanathan, J. Elson, T. Stathopoulos, M. Lukac, D. Estrin, "Emstar: A Software Environment for Developing and Deploying Heterogeneous Sensor-actuator Networks", In ACM Transactions on Sensor Networks (TOSN), Vol.3, No.3, pp. 1–34 (2007).

[7] B. Titzer, D. Lee, J. Palsberg, "Avrora: Scalable Sensor Network Simulation with Precise Timing", In IEEE Fourth International Conference on Information Processing in Sensor Networks(IPSN' 05), pp. 477–482 (2005).

[8] A. Sobeih, J.C. Hou, L. Kung, N. Li, H. Zhang, W. Chen, H. Tyan, H. Lim, "J-Sim: A Simulation and Emulation Environment for Wireless Sensor Networks", IEEE Wireless Communications, Vol.13, No.4, pp. 104–119 (2006).

[9] B. Al Homssi, k.Dakic, S. Maselli, H. Wolf, S. Kandeepan, A. Al-Hourani, "IoT Network Design Using Open-Source LoRa Coverage Emulator", IEEE Access, No.9, pp. 53636–53646 (2021).

[10] T. Maret, R. Kummer, P. Kropf, J. F. Wagen, "Freemote Emulator: A Lightweight and Visual Java Emulator for WSN", In International Conference on Wired/Wireless Internet Communications, pp. 92–103 (2008).

[11] J. Polley, D. Blazakis, J. McGee, D. Rusk, J. Baras, "Atemu: a Fine-grained Sensor Network Simulator, In Sensor and Ad Hoc Communications and Networks", pp. 145–152 (2004).

[12] G. Kasprowicz, L. Mankiewicz, K. T. Pozniak, R. S. Romaniuk, S. Stankiewicz, G. Wrochna, "Hardware Emulator of the High-resolution CCD Sensor for the Pi of the Sky Experiment", In Photonics Applications in Astronomy, Communications, Industry, and High-Energy Physics Experiments 2007, Vol.6937, pp. 693–708 (2007).

[13] S. Brady, A. Hava, P. Perry, J. Murphy, D. Magoni, A. O. Portillo-Dominguez, "Towards an Emulated IoT Test Environment for Anomaly Detection using NEMU", In 2017 Global Internet of Things Summit (GIoTS), pp. 1–6 (2017).

[14] O. Abrishambaf, P. Faria, Z. Vale, "Laboratory Emulation of Energy Scheduling in an Agriculture System. In 2020 IEEE/PES Transmission and Distribution Conference and Exposition (T&D)", pp. 1–5 (2020).

[15] S. Boschert, R. Roland, "Digital Twin—The Simulation Aspect, Mechatronic futures, Springer", pp. 59–74 (2016).

[16] S. Deda, A. Eder, V. Mhetre, A. Kuchling, R. Greul, O. Koenig, "Designing a Battery Emulator/Tester from Scratch to Prototyping to Automated Testing within a HIL Digital Twin Environment", In International Exhibition and Conference for Power Electronics, Intelligent Motion, Renewable Energy and Energy Management, pp. 1–8 (2020).

[17] Z. Ye, F. Hu, L. Zhang, Z. Chu, Z. O'Neill, "A Low-Cost Experimental Testbed for Energy-Saving HVAC Control Based on Human Behavior Monitoring", International Journal of Cyber-Physical Systems (IJCPS), Vol.2, No.1, pp. 33–55 (2020).

[18] P. Evensen, M. Hein, Sensor Virtualization with Self-configuration and Flexible Interactions, In the 3rd ACM International Workshop on Context-Awareness for Self-Managing Systems", pp. 31–38 (2009).

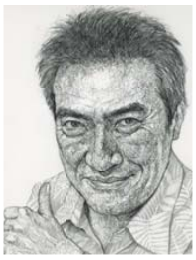
[19] H. Debnath, N. Gehani, X. Ding, R. Curtmola, C. Borcea, "Sentio: Distributed Sensor Virtualization for Mobile Apps", In 2018 IEEE International Conference on Pervasive Computing and Communications (PerCom), pp. 1–9 (2018).

[20] Z. Wang, M. Liu, S. Zhang, M. Qiu, "Sensor Virtualization for Underwater Event Detection", Journal of Systems Architecture Vol.60, No.8, pp. 619–629 (2014).

(Received: October 30, 2021)
(Accepted: July 19, 2022)

Kei Hiroi received her Master of Media Design and Ph.D. in Media Design in 2011 and 2014, respectively from Keio University. She has been an assistant professor in the department of Information and Communication Engineering, Graduate School of Engineering, Nagoya University. She is currently an associate professor in Disaster Prevention Research Institute, Kyoto University. Her research interests include disaster simulation, and crisis computing.

Akihito Kohiga is a Project Researcher of Information Sciences at Japan Advanced Institute of Science and Technology. He received Ph.D (Information Science) in 2020. His research interests lie in cloud computing, massive distributed simulation, modeling and architecture, especially in flood and evacuation fields.



Yoichi Shinoda is currently a professor of Japan Advanced Institute of Science and Technology. His research interests include Impact of digital technologies on human activities, Parallel and Distributed Systems, Networking Protocols and Systems, and Information Handlinf Systems.