PROCEEDINGS OF SCIENCE

# Architecture of a Resource Manager for Software-Defined IT Infrastructure

**Yuki Matsui*[a], Yasuhiro Watashiba[b],Yoshiyuki Kido[b],Susumu Date[b] and Shinji Shimojo[b]**

[a]*Graduate School of Information Science and Technology, Osaka University, Japan*
[b]*Cybermedia Center, Osaka University, Japan*
*E-mail:* matsui.yuki@ais.cmc.osaka-u.ac.jp

The concept of Information-as-a-Service (InfaaS) is a critical concept for disaster management applications. In the disaster management, the flow of data and the synchronization of information between different sites must be maintained to facilitate the decision making process. InfaaS-compliant application need to satisfy clarity, synchronization and continuity, the mechanism of distributed visualization system is suitable as the application platform for developing and deploying those applications. The distributed visualization system provides clarity and synchronization. Additionally, for achieving the continuity in the distributed visualization system, the Software-Defined IT Infrastructure has been studied and developed. However, a mechanism to autonomously control components of Software-Defined IT Infrastructure is currently lacking. In this paper, we propose a resource manager that handles the behavior of these components comprehensively. We extracted the required functions for the Software-Defined IT Infrastructure and designed an inner structure of the proposed resource manager to introduce required functions.

*International Symposium on Grids & Clouds 2019, ISGC2019*
*31st March - 5th April, 2019*
*Academia Sinica, Taipei, Taiwan*

*Speaker.





PoS(ISGC2019)009

## 1. introduction

Disaster management is an important topic: some natural disaster, for example the Tohoku earthquake and the Hurricane Harvey brought serious human and material damage to the world. When such disasters have happened, actions for recovering from the disaster damage are required. Disaster management is the set of actions to recover from a disaster and the control thereof. For example, rescue activities and provision of relief supplies are part of the disaster management.

In contemporary disaster management, collaboration between different sites has become important. Especially, expectations for global collaborations are rising [1]. Figure 1 shows conceptual formation of collaborations in the disaster management. In the disaster management, multiple organizations participate from different sites or countries [2]. Within the each organization, decision makings are made on a disaster from different perspectives. To give an example, a group in a damaged site discusses about rescue activities such an evacuation, an announcement of evacuation order, and treatments of victims. In non-damaged neighboring sites, topics of decision makings are damage prediction caused by a disaster, provision of relief supplies and disaster relief operation. Since disaster management is made more efficient by taking into account the decisions of other sites, disaster management application to support the sharing of decisions made at each site are desired.

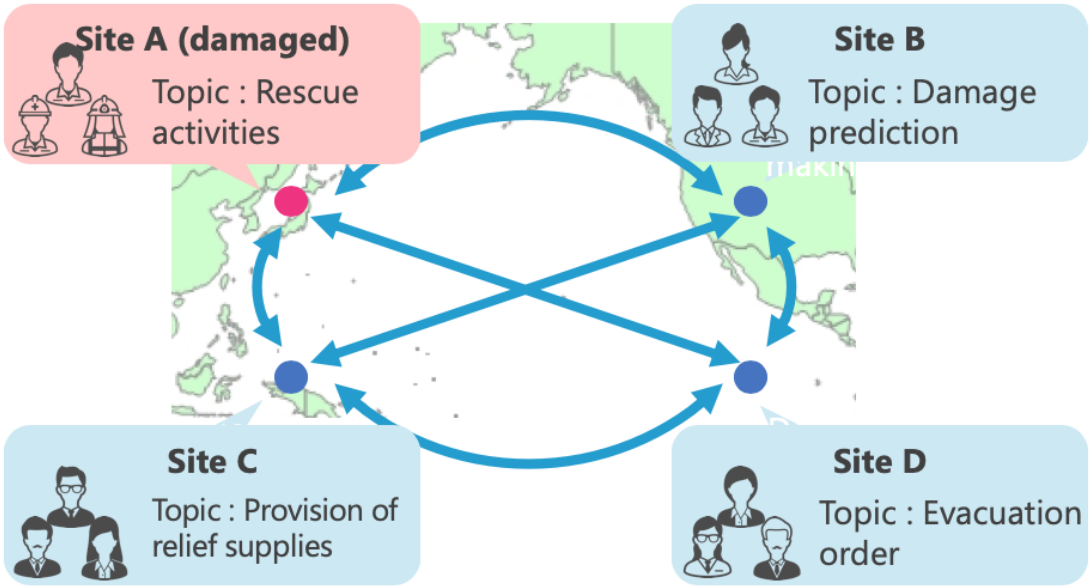



**Figure 1:** Collaborations in the disaster management.

For constructing the disaster management application, Information-as-a-Service (InfaaS) is a critical concept. InfaaS means that any information is provided in the proper format for receivers. The reason why InfaaS is required is that many people deal with diverse information in the disaster management application. For achieving the InfaaS-compliant disaster management application, there are three key factors: clarity, synchronization and continuity.

Clarity means to understand information easily and quickly. In the disaster management, opportunities to exchange various types of data are increasing. The data includes analysis results of data collected from IoT devices and simulation results of damage range caused by disaster and more. In general, visualization helps us to understand information easily and quickly. Previous studies have suggested that visualization enhanced clarity of information [3]. It is desirable that disaster management application visualizes any information.

Synchronization refers to share information among different sites. The disaster management is performed by various groups making decisions based on information about the disaster and sharing the conclusions of discussion. The conclusions drawn from different sources of information at different times may hinder collaboration between different sites. It is desirable for all sites to make decisions based on the same information.

Continuity means to keep any services offered by the application available even during a disaster. When a disaster damages sites running the application, the application is required to stay operational. Generally, for keeping service is to deliver information to harmless sites by bypassing the undamaged site. Such a resilient system increases the availability of the application.

The disaster management application requires a mechanism that can satisfy these three factors. However, since it is necessary to deal with the infrastructure layer, it is difficult to achieve the key factor for each individual application Thus, the InfaaS-capable application platform which is a development and deployment base of the disaster management application is important. By incorporating appropriate technology in to the application platform, the disaster management application generated on the application platform can satisfy the key factors.

## 2. Software-Defined IT Infrastructure

For achieving these key factors, the mechanism of distributed visualization system is suitable. The distributed visualization system is a form in which multiple remote sites and a site that owns a visualization device. The distributed visualization system has a function to deliver data to the visualization device.

Many of current distributed visualization systems satisfy clarity and synchronization. Nevertheless, continuity is not satisfied in it. Since continuity is a requirement that needs to be addressed at the hardware layer, a distributed visualization system on general IT infrastructure can not provide the resilience needed by the disaster management application. Hence, in order to achieve resilience, we have been studying and developing a novel IT infrastructure. The deliverable is named Software-Defined IT Infrastructure, which is able to be managed from software. From this point of view, the distributed visualization system with Software-Defined IT Infrastructure has been developing as the InfaaS-capable application platform. Figure 2 is an overview of the application platform. This Software-Defined IT Infrastructure consists of main four components shown below.

1. Tiled Display Wall (TDW)
2. Scalable Amplified Group Environment 2 (SAGE2)
3. Docker
4. Software-Defined Networking (SDN)

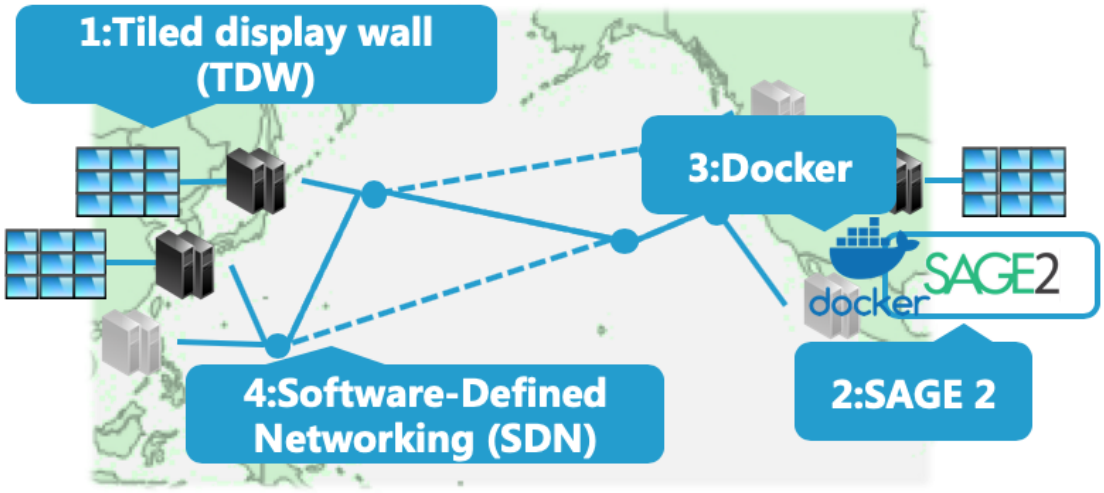



**Figure 2:** Architecture of application platform.

The TDW is one of the potential visualization devices. The TDW provides a large scale virtual screen by using multiple display. Since each display cooperates with the others, the TDW can provide a high resolution and scalable visualization environment for large data. Moreover, the TDW can offer a collaborative and sharable environment because visualized contents on the TDW can be seen by multiple people.

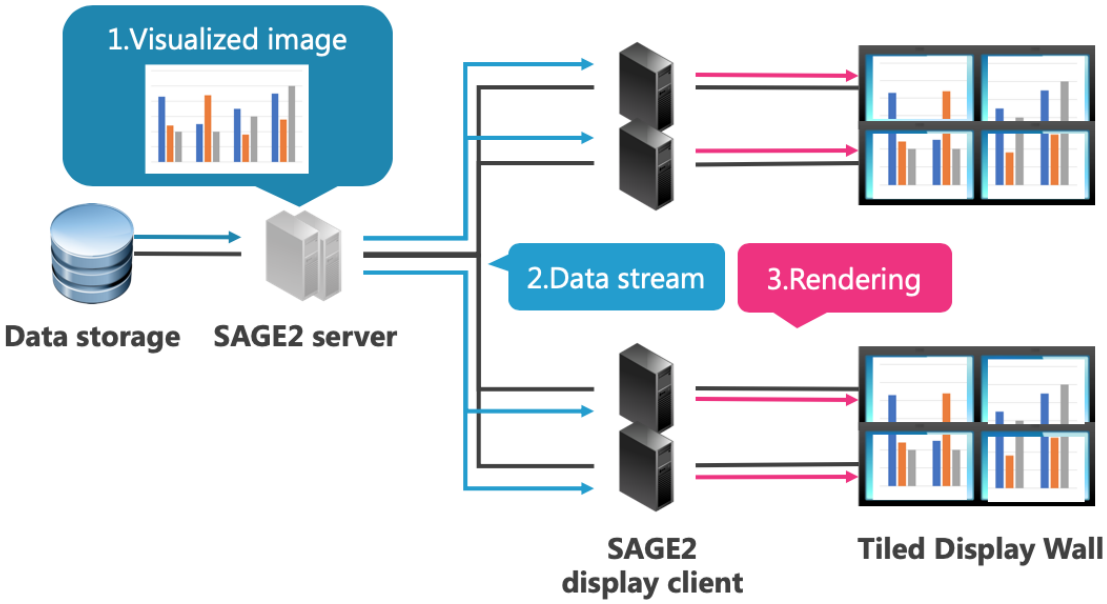


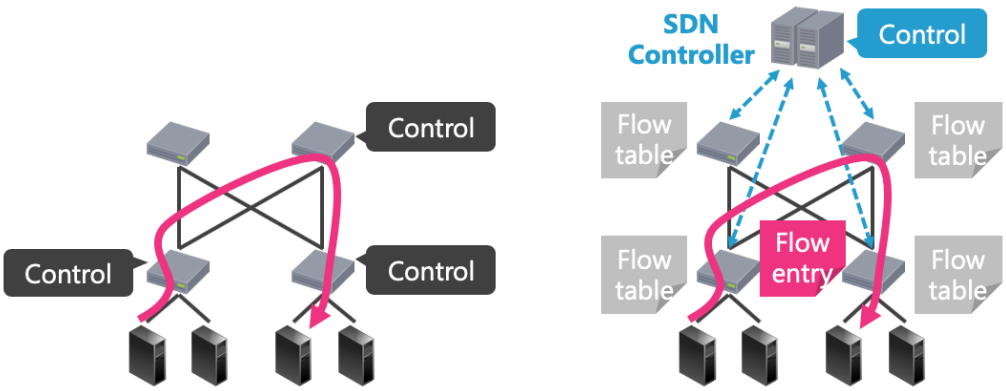

**Figure 3:** Diagram of SAGE2.

SAGE2 [4] is a middleware for supporting synchronization among different TDWs in application platform. Figure 3 shows a diagram of the SAGE2 architecture. There are two types of components running on SAGE2. The source of data is at the SAGE2 server. The other component

is a receiver of data, called SAGE2 display client. The SAGE2 display clients are deployed on computing nodes adjacent to the TDW. In the SAGE2 data flow, the SAGE2 server receives a visualized image from a data storage (Figure 3-1). Next, SAGE2 display clients receive the visualized image by querying to SAGE2 server (Figure 3-2). After that, by means of SAGE2 display clients render-processing, the visualized image is displayed on the all TDWs (Figure 3-3).

Docker [5] is a container technology that allows to package an application. In the application platform, the SAGE2 components run in a Docker container deployed on a computing node. When the computing node executing a SAGE2 component stops, the Docker container running the SAGE2 component is redeployed on the alternative resources. The SAGE2 component can continue to provide its services by moving the Docker container to any remaining available computing nodes.

SDN [6] enables dynamic control of the network by software. Figure 4 shows the difference between network control in traditional networking and the SDN. Before the concept of SDN emerged, each network switch independently controlled only the next hop for packets(Figure 4-a). On the other hand, if SDN is used for networking, all data flows are controlled collectively by dedicated software, called an SDN controller (Figure 4-b). The SDN controller configures a flow entry. The flow entry contains rules about the flow of data. Each network switch receives the flow entries from the SDN controller as a flow table. The controller centrally manages the flow entries and distributes them to the network switches, thereby easing management of the network.




a. Traditional networking. b. Networking using SDN.

**Figure 4:** Differences in the network control.

These four components have been deployed on the PRAGMA-ENT [7]. For assessing the behavior of Software-Defined IT Infrastructure in a wide-area distributed environment. The PRAGMA-ENT is an international SDN test bed across different countries. Every researcher can receive the necessary networking support for SDN from the PRAGMA-ENT.

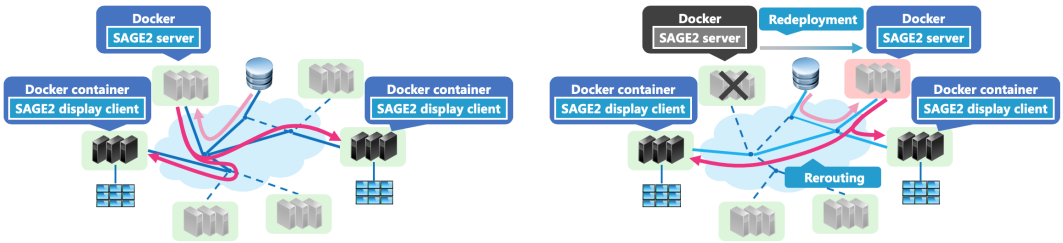
In our research to date, the foundation of service continuity utilizing the four components has been established. In the prototype on the PRAGMA-ENT, by operating each component individually according to the assumed scenario, service recovery can be realized. However, there is lack of mechanism for autonomously managing each component according to the situation.

Therefore, toward the acquisition of continuity in the application platform, we propose a resource manager to handle the behavior of these components comprehensively. The proposed re-

source manager acts as a central controlling component for all of the Software-Defined IT Infrastructure within its scope. The proposed resource manager allows Software-Defined IT Infrastructure to avoid the stop of service.

## 3. Analysis of required functionalities

For designing the proposed resource manager, a scenario is assumed. Figure 5 illustrates the scenario before and after a disaster. In this scenario, the SAGE2 server is distributing data from a site. The Docker packages up the SAGE2 components in containers.




a. Before the disaster. b. After the disaster.

**Figure 5:** A scenario for designing the proposed resource manager.

Before a disaster occurs (Figure 5-a), Docker containers are deployed on certain computing nodes. The SAGE2 server distributes data to the TDW sites. In this scenario, assume that the site where the SAGE2 server is running is damaged and the service is stopped. The storage and other compute nodes are not damaged.

After the interruption of service (Figure 5-b), the Docker container needs to be redeployed to other available computing node. By accessing to the same source of information, the SAGE2 server can keep to distribute data even after the redeployment of the Docker container. Additionally, for keeping the connection between the new SAGE2 server and TDWs, the network paths needs to be rerouted. By reconfiguring the network paths in accordance with the Docker container relocation, TDW sites can continue to receive information even if some sites were damaged. Based on the scenario, we extract five functionalities necessary for the proposed resource manager.

1. Collection of component's statuses
2. Detection of service down
3. Selection of alternative resources
4. Reroute of network
5. Migration of Docker container

The first function is to collect the statuses of all components. This is because it is necessary to constantly monitor resource's statuses for detecting a service interruption and selecting alternative resources. Various statuses are collected, for example, the CPU usage rate on computing nodes, reachability of the network endpoint and execution status of the Docker containers are raised. Based on the retrieved statuses, the second function detects the down of service.

After detecting the stop of service, the proposed resource manager needs to recover the service. Hence, the proposed resource manager selects alternative resources, that are network paths for maintaining data flow and available computing nodes for deploying the Docker container.

Based on the selection results, the proposed resource manager reroutes the network by using the SDN. In this research, rerouting refers to rearranging the network paths assigned in layer 2. By reconfiguring the network paths by the SDN, the TDWs are ready to receive data from SAGE2 server. Additionally, it also helps to remove the network paths that was connected to unavailable sites.

Finally, the proposed resource manager redeploys the Docker container with the SAGE2 server to the alternative available resources. This will restart the service of SAGE2. Additionally, this scenario assumes that the storage will not be corrupted, the SAGE2 server can configured to access the same data as before the redeployment. By creating the save file on the storage using the function of SAGE2, SAGE2 can provide the same contents as before the outage.

## 4. Design

Based on the analysis described in section 3, we made a functional design of the proposed resource manager to introduce required functions. As shown in figure 6, the proposed resource manager consists of seven modules.

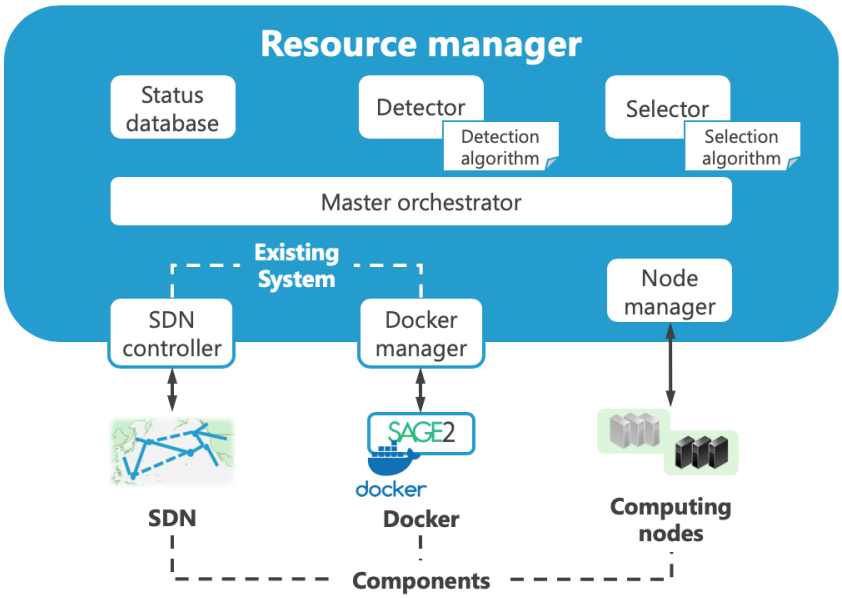



**Figure 6:** Design of the proposed resource manager.

The *Master orchestrator* is the core module of the proposed resource manager. The role of *Master orchestrator* is to instruct other modules according to the situation. The network status is provided by the functions of the *SDN controller*. All operations related to Docker are served by the *Docker manager*. Since the *SDN controller* and *Docker manager* are already sophisticated and existing modules [8], the proposed resource manager takes in the *SDN controller* and the *Docker*

*manager* as modules. The *Node manager* is responsible for managing computing resources. Based on the instructions from the *Master orchestrator*, the *Node manager* directly controls computing nodes by command operations.

The *Status database* stores data collected by the *SDN controller*, the *Docker manager* and the *Node manager*. The reason for installing the *Status database* is intended to store data according to time series. These stored data is an important input for the proposed resource manager.

The *Detector* periodically accesses the *Status database* to check for service interruptions. The *Detector* analyzes the stored data based on a *Detection algorithm* and then informs the *Master orchestrator* if any service interruption is detected. After receiving the announcement about failure from the *Detector*, the *Master orchestrator* throws a request to the *Selector*. The *Selector* seeks alternative resources, which are new network paths and/or alternative computing nodes to deploy the Docker container. This selection is performed by analyzing the statuses in *Status database* based on *Selection algorithm*. The *Detection algorithm* and *Selection algorithm* are characterized as pluggable for flexibility. These algorithms have attracted researchers for a long time, and new algorithms have been proposed in recent years [9, 10, 11]. For supporting the pluggable method is a critical solution for adapting new algorithms.

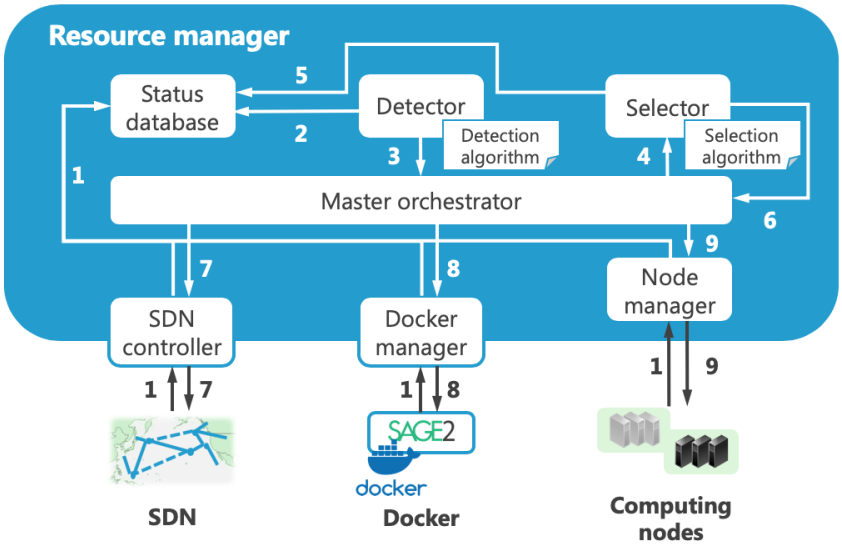



**Figure 7:** Flow of access in the proposed resource manager.

These algorithms may need to be modified to the environment to be introduced. The pluggable method is a critical solution in these situations. Figure 7 shows the process flow in the proposed resource manager. The five features considered necessary in Section 3 correspond as follows: Collection of component's statuses is represented by arrow 1. Arrows 2 and 3 indicate the interaction required to detect a service down by the *Detector*. The behavior of *Selector* required for alternative resource selection is summarized in arrows 4 - 6. Instructions for rerouting the network are linked to arrow 7. Arrows 8 and 9 show the operations required for docker container migration.

## 5. Conclusion

Toward the acquisition of continuity in the InfaaS-capable application platform, we designed the resource manager that handles the Software-Defined IT Infrastructure comprehensively. This design is based on the scenario where the components that make up the Software-Defined IT Infrastructure are down. The proposed resource manager will allow the Software-Defined IT Infrastructure to recover autonomously from situations where some components are down. In the next step, we will complete the prototype of resource manger and validate that it is applicable a practical use in the local environment of the laboratory. After that, we plan to deploy the Software-Defined IT Infrastructure with proposed resource manager on PRAGMA-ENT.

## References

[1] A. Aitsi-Selmi, S. Egawa, H. Sasaki, C. Wannous and V. Murray, *The sendai framework for disaster risk reduction: Renewing the global commitment to people ' s resilience, health, and well-being*, *International Journal of Disaster Risk Science* **6** (2015) 164.

[2] A. Sagun, D. Bouchlaghem and C. J. Anumba, *A scenario-based study on information flow and collaboration patterns in disaster management*, *Disasters* **33** (2009) 214.

[3] T. Munzner, *Visualization*, 2009.

[4] T. Marrinan, J. Aurisano, A. Nishimoto, K. Bharadwaj, V. Mateevitsi, L. Renambot et al., *Sage2: A new approach for data intensive collaboration using scalable resolution shared displays*, in *10th IEEE International Conference on Collaborative Computing: Networking, Applications and Worksharing*, pp. 177–186, Oct, 2014, DOI.

[5] "Docker: Enterprise Application Container Platform." `https://www.docker.com`, Apr., 2019.

[6] D. Kreutz, F. M. Ramos, P. Verissimo, C. E. Rothenberg, S. Azodolmolky and S. Uhlig, *Software-defined networking: A comprehensive survey*, *Proceedings of the IEEE* **103** (2015) 14.

[7] K. Ichikawa, P. U-Chupala, C. Huang, C. Nakasan, T.-L. Liu, J.-Y. Chang et al., *Pragma-ent: An international sdn testbed for cyberinfrastructure in the pacific rim*, *Concurrency and Computation: Practice and Experience* **29** (2017) e4138.

[8] "Docker docs: Swarm mode overview." `https://docs.docker.com/engine/swarm/`, Apr., 2019.

[9] C. Maru, M. Enoki, A. Nakano, S. Yamamoto, S. Yamaguchi and M. Oguchi, *Information Detection on Twitter for Network System Control in Large-Scale Disasters*, *DEIM2015* **3** (2015) .

[10] V. Pekar, J. Binner, H. Najafi, C. Hale and V. Schmidt, *Early detection of heterogeneous disaster events using social media*, *Journal of the Association for Information Science and Technology* (2019) .

[11] M. Nawir, A. Amir, N. Yaakob and O. B. Lynn, *Effective and efficient network anomaly detection system using machine learning algorithm*, *Bulletin of Electrical Engineering and Informatics* **8** (2019) 46.