Monitoring and Managing IoT Applications in Smart Cities Using Kubernetes

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

Gyuwon Song
Korea Institute of Science and Technology
Seoul, South Korea
Email: gyuwon@kist.re.kr

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

***Abstract*—With the rapid urbanization, cities are transforming to smart cities with core objectives to maintain a safe, healthy and livable environment for the people. The current landscape of smart cities are continuously evolving with unique challenges and gaining ground with new technology-based solutions on the Internet of Things (IoT) and cloud computing. The efficient integration of IoT and cloud computing can tackle the unprecedented growth of smart cities by supporting various smart services like healthcare, transportation systems, environment monitoring, smart grids, etc. Recent advances in cloud computing like containerization of applications are promising solutions to host, supervise and reform the diverse IoT applications in smart cities. In this paper, we have explored the possibilities to implement a secure, distributed and reliable cloud-based monitoring system for IoT applications for effective management of a smart city environment. We propose to build a container-based system with low latency, a reliable and secure communication among large scale deployment of IoT devices with a strong focus on horizontal interoperability among various IoT applications. Our experiment with Docker-based containerization techniques along with a Kubernetes container orchestration platform emphasizes an efficient way to manage and monitor the status and events in IoT applications in the scale of smart cities.**

Keywords- Smart city; Internet of Things (IoT); Cloud computing; Docker; Kubernetes; Interplanetary File System (IPFS)

I. INTRODUCTION

The current trend indicates that the urban areas are expanding massively, with predictions indicating 70% of the world's population in cities by 2020 [1]. Due to the anomalous increase in the urban population the standard quality of life is deteriorating. The concept of smart cities is put forth to improve the living standard in cities which is curbed by challenges like environmental issues, air pollution, traffic congestion, etc[2]. The countermeasures needed to tackle these issues are overwhelming owing not only to the scale of the smart cities but also the heterogeneous technologies, devices and the platforms involved in the development. Technologies like cloud computing and Internet of Things (IoT) envision the large-scale development of smart cities. Various applications like smart healthcare, intelligent transportation, smart grids, smart homes, etc., deploy many connected IoT devices which are distributed over a wide geographical area and perform various activities with a massive volume of data generated over a time period. IoT along with the cloud computing technologies has made an impact on analyzing and processing the diverse data collected from various applications to be useful to the end users [3].

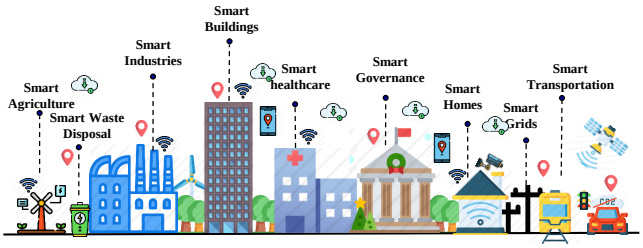

Fig. 1: A Smart City Scenario

Although IoT is the key technology to keep the smart city connected, the predictions for connected IoT devices in the future along with the traditional centralized cloud architecture might limit the horizontal development among the vertically integrated smart cities. Figure 1 shows a smart city scenario. The centralized cloud architecture is more vulnerable to increasing network loads, low latency requirements, energy issues and exposed to a single point of failure which does not suit delay-intolerant IoT applications [4][5]. To overcome these issues and to increase the efficiency of IoT applications proposals to distribute cloud architecture with edge and fog computing was introduced [6][7]. Further efforts to design frameworks based on the concept of software-defined networked systems by virtualizing IoT nodes and resources were initiated [8]. Though the distributed, hierarchical cloud architecture proved advantageous, it is challenging to implement a fully integrated approach in a hybrid smart city with its limitations.

To overcome the challenges faced by the smart city environment due to its scale and the heterogeneity of applications we need a hybrid cloud architecture managing the micro-level IoT applications. The current focus in cloud development is shifting towards containers an alternative to virtualization technique by changing the way the operations are carried out. To tackle the critical characteristics of IoT systems on its scale and data-centric nature containers make it easier to build, deploy and maintain IoT applications even when IoT devices have limited resources to support operating systems.

 ISBN: 978-1-61208-703-0

Containers packaged with all dependencies and software for IoT applications are portable, light and secure. Though the IoT applications are deployed at ease with containers, still the scale of the smart cities hosting multiple IoT applications makes it difficult to monitor and coordinate the containers running in different applications. A platform to orchestrate all these containers along with their varying workloads, computing, networking, and storage are in demand.

In this paper, we have created a smart city scenario and analyzed the possible options to containerize IoT applications with the help of Docker containers. Further, we have used Kubernetes an open-source platform to manage their workloads, coordinate the services providing effective monitoring and management environment. To be more precise the contributions of the paper are:

- Creating a Smart city scenario in our testbed
- Deploying IoT nodes with P2P pubsub communication model based on Interplanetary File System (IPFS)
- Containerizing IoT applications using docker containers
- Orchestrating various Docker containers in Kubernetes
- Evaluating the use of Docker containers and the Kubernetes service for IoT applications in smart cities

The remainder of this paper is structured as follows. Section II discusses the requirements of a smart city and existing related work. The enabling technologies for smart cities is illustrated in Section III. We explain the prototype implementation of our experimental framework in Section IV and results in Section V. Section VI annotates the conclusion.

II. Requirements in a Smart City & Related Work

In this section, we will describe the main requirements and challenges of an IoT-Cloud based smart city framework and the existing related works to address these issues.

A. Requirements in a Smart City

The convergence of the ubiquitous IoT technologies and the cloud resources to process, store and network data generated from IoT devices has led to the concept of a smart city. Several challenges and requirements arise from developing a smart city which include interoperability, providing efficient data management mechanisms and seamless integration of the infrastructure [9]. The essential features to develop IoT-Cloud based smart city include:

- *Reliability:* IoT devices present a range of sleep patterns and uncertainties in network connectivity can make sensitive data unavailable when needed. It is a foremost concern in safety-critical applications like healthcare.
- *Scalability:* Billions of connected devices are forecast, making it challenging to scale while ensuring its reliable data delivery.
- *Latency:* Managing latency values for delay-intolerant applications like healthcare, smart grids, demanding P2P scalability, avoiding the single point of failure by moving away from the centralized cloud-based framework.
- *Flexibility:* Providing flexibility by containerization making the IoT nodes available and inter-operate horizontally.
- *Monitoring:* Efficient monitoring is required to coordinate the IoT devices deployed in a distributed platform like smart cities.
- *Security:* Strong security measures are required to handle the data transactions among various applications.

The aforementioned challenges and requirements need to be addressed to facilitate an integrated smart city environment. The current shift in focus from virtualization to containerization can overlook and satisfy the challenges and the requirements in a smart city.

B. Related Work

The rapid development in the concept of smart cities is demanding an upgrade in a wide array of domains like IoT and cloud computing. The traditional centralized cloud-based architectures used by the IoT applications can cope with their varying storage and computing resource requirements. Existing works initially discussed the possibilities of virtualizing IoT resources with the help of Software Defined Networks (SDN) and development of an integrated IoT framework [10][11]. The solutions from virtualizing though looked promising limited the flexibility in deploying various IoT systems due to its heterogeneous nature with varying resource requirements in near real-time. To bridge this gap lightweight virtualization using containers is getting adopted. The last few years existing works have explored possibilities to use Docker containers in an IoT framework [12][13]. Container-based solutions are inherently optimized for running applications on IoT devices which have limited resources, and they are portable and lightweight, unlike virtual machines [14]. Other existing works on containerization focus their work on using containers for specific use cases even for smart cities, but there is no specific work exploring the need to orchestrate all these containers to maximize the benefits [15]-[17].

The existent works on smart cities based on IoT and cloud adopts containerization in some IoT use cases, but limited work related to the usage of containers for multiple use cases exists. Moreover, some of the issues in deploying and maintaining containers across various applications need a

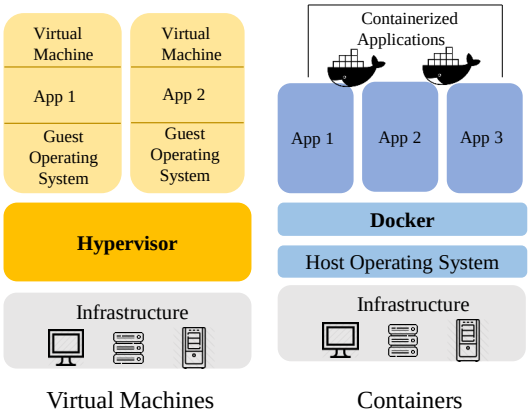

Fig. 2: Virtual Machines and Docker Container Model

monitoring platform like Kubernetes. In this paper, we have exploited the benefits of containerized IoT applications along with a monitoring platform based on Kubernetes to effectively maintain a smart city scale deployment.

III. Enabling Technologies

The paradigm focus shift from traditional virtualization to container-based virtualization solutions have gained great momentum in recent years because containers utilize kernel features to create an isolated environment of the running process. Further, they use the hardware of the host system and does not use the virtualized hardware like a hypervisor. The usage of host hardware makes the containers lightweight and able to start in milliseconds allowing it to perform well in large scale environments like in smart cities [18]. A comparison between hypervisor and docker is illustrated in Figure 2. The following explanations clearly state the reasons to choose docker to create our IoT based containers in this paper.

A. Docker

Docker is an open source project offering standardized solutions to enable the ease in implementing Linux applications inside portable containers [19]. There are a variety of system-level container services like OpenVZ and LXC available, but we chose docker since it is application oriented and it can work well with the micro-services environment like IoT [20]. Docker containers are built from base images, and they are the building blocks of docker. The images act as a template to create the containers and can be configured with the applications. Docker hub shares every change in the image with a team like a layer in git. Commands in Docker containers can be executed manually or automatically using Dockerfiles holding a series of instructions. Docker containers can be linked to each other to form a tiered application, and they can be started, stopped and terminated. There is a docker daemon that interacts with the containers through CLI or Representational State Transfer (REST) API'S. The lightweight virtualization technique is mainly used because of its features like rapid application deployment, portability, versioning of images in docker along with minimal overhead and ease in maintenance helps in building Platform as a Service(PaaS). Figure 2 shows a model of Docker container.

B. Container Orchestration

Containerization in docker expedites the feasibility to run applications that are containerized over multiple hosts in multiple clouds [21]. Cluster architecture in containers enables the need to operate multiple containers in different hosts and clouds which is inevitable in smart cities [22]. Different hosts holding same docker containers can be clustered and controlled. Further, typical applications residing in clusters are logically created from the same base images, making easier replication among various hosts. This feature of scaling the nodes can enable the vision in the scale of a smart city. The cluster-based containerization in docker creates a need to bridge the gap between the clusters and cluster management.

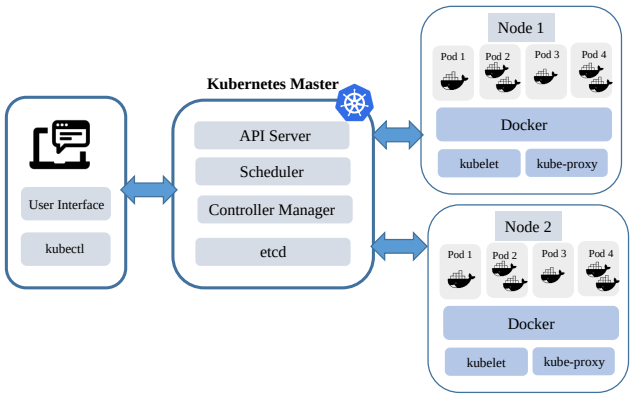

Fig. 3: Kubernetes Architecture

A cluster orchestration platform should be able to monitor the scaling, load balancing and the other services of containers residing across different hosts. It should support the scalable discovery and orchestration of the containers and provide communication in the clusters. Among various available orchestration platforms in this paper, we have used Kubernetes for monitoring and managing IoT applications.

C. Kubernetes

Kubernetes is a multihost container management platform, which uses a master to manage Docker containers over multiple hosts [23]. A sample of the Kubernetes architecture is shown in Figure 3. As mentioned before we need an orchestration platform for the clusters and Kubernetes can dynamically monitor the applications running in containers and can perform the resources provisioning along with auto-scaling support with its built-in features [24]. We have exploited this feature of Kubernetes to invigilate the nodes residing in various IoT application containers in a smart city based scenario. Kubernetes creates pods the basic deployment units, which holds one or more grouped containers. The Kubernetes master can assign each pod a virtual IP. A node agent called Kubelet monitors the pod, and it reports the pod status, its resource

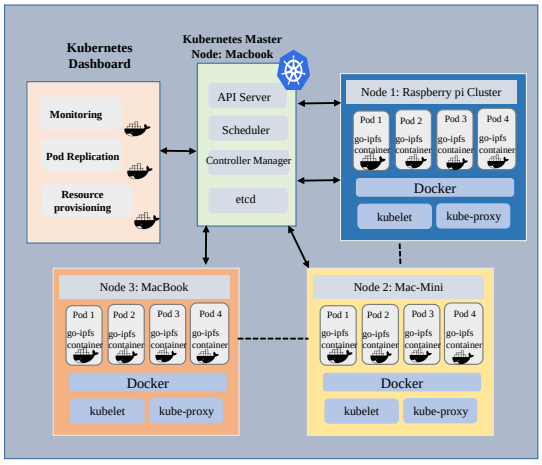

Fig. 4: Proposed Experimental Framework

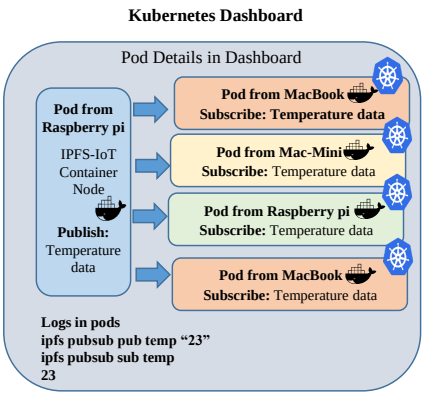

Fig. 5: A Model IoT application enabled among the pods

utilization, and events to the master. The Kubernetes master controls a scheduler, storage component, an API manager and the controller manager. Kubernetes provisions namespaces separately to enable each application to be partitioned and prevent them from affecting each other. In this paper, we have used the Kubernetes platform to monitor docker containers in the smart city scenario.

IV. Proposed Experimental Framework

- *Cluster Setup:* To replicate the smart city scenario hosting various applications, we deployed a similar prototype and evaluated the scenario experimentally. The setup consisted of a set of three different machines of different capacities hosting the docker images to imitate the different specifications of IoT nodes. The whole experimental setup is shown in Figure 4. We have used a set of five Raspberry Pi 3 nodes with Quad Core 1.2 GHz Broadcom BCM 2837, 64 bit CPU and 1 GB RAM, Mac Mini with processor i5 – 2410M, RAM 2GB 1333MHz, CPU 2.3 GHz and Mac book with processor i5 RAM 8GB for the experiments. To begin with, we have installed the docker base images of go-ipfs in all the three different sets of devices [25]. IPFS is a well-known P2P file system with inherent capabilities like clustering, pubsub model and distributed storage. We have used ipfs images so that it can emulate our IoT nodes enabled with the IPFS development Stack. So each device holds a set of containers holding go-ipfs based images packaged in it. We have mainly used the pubsub protocol in IPFS for data exchange among the IPFS-IoT nodes.
- *Cluster Orchestration:* After creating the docker images now to orchestrate these containers created we have installed Kubernetes 1.13 in all the machines and enabled a master node in the Macbook. The master node is the principal node controlling the rest of the machines which ran as container execution nodes. The IPFS daemon was initiated after enabling the IPFS based containers as pods in Kubernetes. The IPFS Daemon was initiated with the pubsub mode to enable communication among the different containers. Each container is perceived to perform a different IoT application like monitoring temperature, humidity, air quality, and many more. Each container hosting different IoT applications might need the data from another container running diverse applications in smart city scenario needing interoperability. The data exchange is enabled with the pubsub model with subscribers receiving data from publishers for a particular subscribed topic.
- *Monitoring:* To enable monitoring of the IoT applications clustered under one platform in Kubernetes we have used Heapster v.0.19.1. Heapster enables a web GUI-based Kubernetes Dashboard in the master node which helps in monitoring the system resource metrics. It can collect the resource utilization information of each node, and the gathered information can be viewed in the Kubernetes dashboard. Heapster queries the master for a list of nodes in the system using Kubernetes API, and it is possible to determine whether the node is still active or down due to some issues. Furthermore, we can visualize the information concerning the nodes, pods and the services

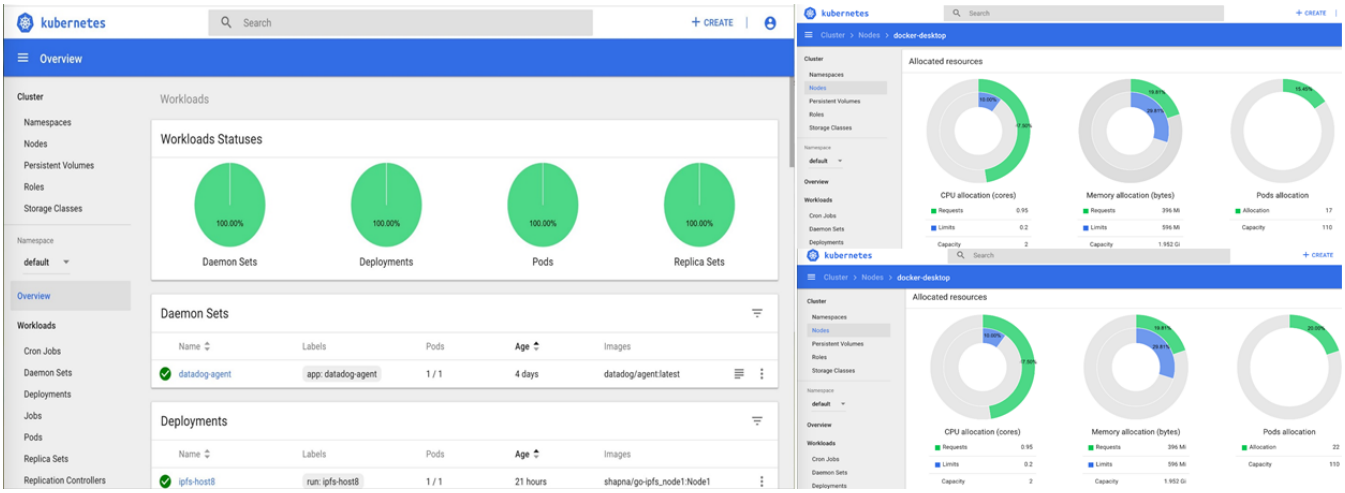

Fig. 6: Monitoring Pods in Kubernetes dashboard

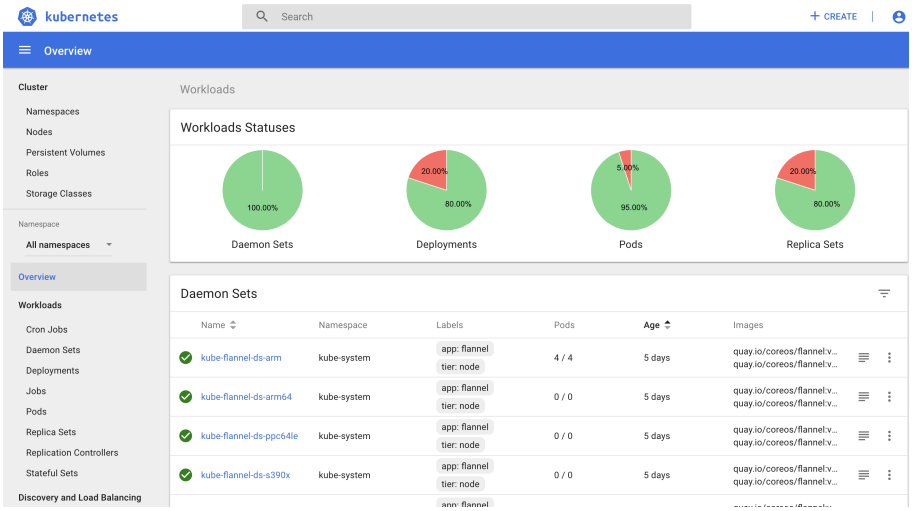

Fig. 7: Event log and Affected Pod Detection

and the data related to the deployments from the dashboard. In this paper, we have tried to establish the smart city scenario with diverse IoT applications and evaluate the docker containerized solutions along with Kubernetes orchestration platform for practical monitoring purposes. We have limited the number of containers in the pod to one to understand the Kubernetes healing action when the pod fails.

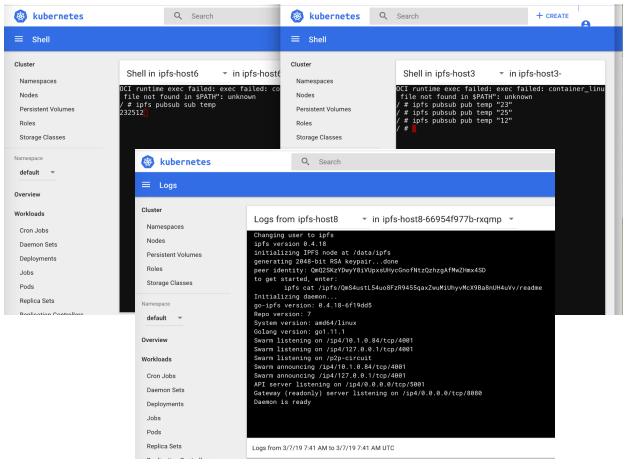

Fig. 8: Data Exchange among pods using IPFS Pubsub Mechanism

V. Evaluation of Experimental Framework

We have utilized the setup as mentioned earlier to evaluate the proposed smart city scenario. Each pod had a different schedule for publishing its data similar to an IoT application and the subscribers subscribed to a particular topic were immediately notified when there is a new update based on the IPFS pubsub model. A model setup for recording temperature data using a temperature sensor and pods subscribing for it is shown in Figure 5.

- *Monitoring:* Over this experimental setup to begin with the containers were started and the starting time was recorded in the time of milliseconds suiting IoT applications. The CPU and memory usage for the whole deployment as well as individual pod details from the time of creation is enlisted in the dashboard enabling complete details of the running pods. Then we were able to to see each log and event executed in the pods. Message logs in one dashboard logging each pubsub event happening across nodes residing in different machines made pod monitoring easier. This monitoring system hugely helps applications in large scale like IoT deployed in smart cities to locate the affected pods. The start of the entire deployment along with the pods CPU usage shown in the dashboard is illustrated in Figure 6.
- *Pod Failures:* The Kubernetes dashboard can efficiently show the pod failures by indicating the infected pods in red and the working pods in green. To try detecting this pod failure scenario, we manually terminated one of the nodes in one of the machines using the commands in Command-Line Interface (CLI). When one of the pods goes down, Kubernetes has 30 seconds to create a replication of the same pod, and this is one of the reasons we have tried to limit the number of containers in the pod to one in number to visualize the reaction of Kubernetes engine when one of the pod goes down. Figure 7 shows the self-healing capability in Kubernetes to increase the reliability and flexibility in the smart city ecosystem along with event logging.
- *Resource Scheduling:* Another important experimentation feature is the resource scheduler in Kubernetes engine. The trials are done to keep the CPU utilization of the pod within its limit. We have tried to flood many messages at a particular time from a pod to mimic this scenario. When the CPU limits cross a particular threshold, then the application is considered to be in heavy load and the Kubernetes engine can autoscale its pod to increase by one. This resource scheduling can efficiently manage real-time applications like IoT, where some of the applications

 ISBN: 978-1-61208-703-0

are event-driven and can considerably increase the CPU and memory usage. Efficient data exchange for a temperature sensor using the IPFS pubsub model among pods is shown in Figure 8.

From the experimental results discussed above, we can see that the docker containers enabled with Kubernetes orchestration can prove to be a comprehensive monitoring mechanism and has an ease in deployment and is flexible. This experimental setup to validate smart city scenario with containerizing IoT application proved to be advantageous.

VI. Conclusion

In this paper, we provide a container-based IoT application in a smart city scenario for efficient monitoring and management. The experiment showed efficient data exchange among the pods. Moreover, the active deployment of the application is monitored using the state of the pods. The Kubernetes dashboard helps in reviewing the system resource usage as well as the event logging in the pods which can satisfy the scalability issues in smart cities. We have also reviewed the self-healing nature of Kubernetes platform, an essential factor to ensure the reliability of the model. For further experimentation, we are trying this scenario in real life deployment at an elderly-care facility with 320 elderly in Seoul. From this work, we expect to demonstrate combining IoT applications in containers with a cloud management platform like Kubernetes would be indispensable in IoT deployment in a smart city.

ACKNOWLEDGEMENT

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

References

[1] "United Nations, Population Division", 2017, Available: http://www.un.org/en/development/desa/population/ [Accessed: 2019-02-25]

[2] T. Nam and T. A. Pardo, "Conceptualizing smart city with dimensions of technology, people, and institutions", in *Proc. ACM dg.o'11, College Park, Maryland, USA*, pp. 282–291, Dec. 2017.

[3] C. Zhu, V. C. M. Leung, L. Shu, and E. C.-H. Ngai, "Green Internet of Things for smart world," *IEEE Access*, vol. 3, pp. 2151–2162, 2015.

[4] S. Sarkar, S. Chatterjee, and S. Misra, "Assessment of the suitability of fog computing in the context of Internet of Things", *IEEE Trans. Cloud Computing*, 46-59, 2018.

[5] D. Bouley, "Estimating a data center's electrical carbon footprint," *Schneider Elec., USA, White Paper* 66, 2015.

[6] F. Bonomi, R. Milito, J. Zhu, and S. Addepalli, "Fog computing and its role in the Internet of Things," in *Proc. 1st Ed. MCC Workshop Mobile Cloud Computing, New York, NY, USA*, pp. 13–16, 2012.

[7] H. Chang, A. Hari, S. Mukherjee, and T. Lakshman, "Bringing the cloud to the edge," in Proc. IEEE Conf. Comput. Commun. Workshops (INFOCOM WKSHPS), pp. 346–351, May 2014.

[8] S. Nastic, S. Sehic, D. H. Le, H. L. Truong, and S. Dustdar, "Provisioning software-defined IoT cloud systems," in *Proc. Int. Conf. Future Internet Things Cloud (FiCloud)*, pp. 288–295, Aug. 2014.

[9] M.Vögler, J.M. Schleicher, C. Inzinger, S. Dustdar, and R. Ranjan, "Migrating smart city applications to the cloud", *IEEE Cloud Computing*, 3(2), pp.72-79, 2016.

[10] C. Buratti et al., "Testing protocols for the Internet of Things on the EuWIn platform," *IEEE Internet Things J.*, vol. 3, no. 1, pp. 124–133, Feb. 2016.

[11] T. Taleb, A. Ksentini, and R. Jantti, "'Anything as a service' for 5G mobile systems," *IEEE Netw.*, vol. 30, no. 6, pp. 84–91, Dec. 2016.

[12] T. Renner, M. Meldau, and A. Kliem, "Towards container-based resource management for the Internet of Things," in *Proc. Int. Conference Software Networking (ICSN)*, pp. 1–5, 2016.

[13] R. Morabito, "A performance evaluation of container technologies on Internet of Things devices," in *Proc. IEEE INFOCOM Demo San Francisco, CA, USA*, pp. 999–1000, 2016.

[14] C. Pahl and B. Lee, "Containers and clusters for edge cloud architectures–a technology review," in *Future Internet of Things and Cloud (FiCloud), 2015 3rd International Conference on. IEEE*, pp. 379–386, 2015.

[15] J. Rufino, M. Alam, J. Ferreira, A. Rehman, and K. F. Tsang, "Orchestration of containerized microservices for IIoT using Docker," in *Industrial Technology (ICIT), 2017 IEEE International Conference on IEEE*, pp. 1532–1536, 2017.

[16] R.G. Chesov, V. N. Solovyev, M. A. Khlamov, and A. V. Prokofyev, "Containerized cloud based technology for smart cities applications," *Journal of Fundamental and Applied Sciences*, vol. 8, no. 3S, pp. 2638–2646, 2016.

[17] M. Kovatsch, M. Lanter, and S. Duquennoy, "Actinium: A RESTful runtime container for scriptable Internet of Things applications," in *Internet of Things (IOT), 3rd Intl. Conf. on the IEEE*, pp. 135–142, 2012.

[18] M.Eder, "Hypervisor-vs. container-based virtualization." *Future Internet (FI) and Innovative Internet Technologies and Mobile Communications (IITM)*, 2016.

[19] "Docker", https://www.docker.com/ [Accessed: 2019-02-20]

[20] A. Sill, "The design and architecture of microservices," *IEEE Cloud Comput.*, vol. 3, no. 5, pp. 76–80, Sep 2016.

[21] B. Satzger, W. Hummer, C. Inzinger, P. Leitner, & S. Dustdar, "Winds of Change: From Vendor Lock-In to the Meta Cloud," *IEEE Internet Computing*, vol. 17, no. 1, pp. 69–73, 2013.

[22] V. Koukis, C. Venetsanopoulos, and N. Koziris, okeanos: "Building a Cloud, Cluster by Cluster," *IEEE Internet Computing*, vol. 17, no. 3, pp. 67–71, 2013.

[23] "Kubernetes", https://kubernetes.io/ [Accessed: 2019-02-01]

[24] C. C. Chang, S. R. Yang, E. H. Yeh, P. Lin, and J. Y. Jeng. "A kubernetes-based monitoring platform for dynamic cloud resource provisioning", *IEEE Global Communications Conference, pp. 1-6. IEEE*, 2017.

[25] "Docker-IPFS", https://hub.docker.com/r/ipfs/go-ipfs/ [Accessed: 2019-02-10]

 ISBN: 978-1-61208-703-0