Dataflow Management Platform for Smart Communities using an Edge Computing Environment

Shogo Shimahara
Graduate School of Science and Technology, Keio University
3-14-1 Hiyoshi, Kohoku, Yokohama, Kanagawa 223-8522, Japan
shimahara@west.sd.keio.ac.jp

Hiroaki Nishi
Department of System Design, Faculty of Science and Technology, Keio University
3-14-1 Hiyoshi, Kohoku, Yokohama, Kanagawa 223-8522, Japan
west@sd.keio.ac.jp

IECON 2021 - 47th Annual Conference of the IEEE Industrial Electronics Society | 978-1-6654-3554-3/21/$31.00 ©2021 IEEE | DOI: 10.1109/IECON48115.2021.9589430

***Abstract*—As various data services are provided to realize Society 5.0, the usage of personal data is estimated to increase along with the explosive increase in data traffic. It is important that the protection of privacy keeps pace with increases in the exchange of data containing personal information. Starting from the enforcement of the General Data Protection Regulation (GDPR), stricter privacy protection regulations are expanding to more countries. These restrictions require that personal data should be hidden or anonymized before they are propagated over the network. The secondary usage of data is assumed in smart communities, and systems that can protect privacy are required for secure network infrastructures. In this study, we propose an edge-based computing platform that manages the privacy of users on the network of a smart community. For the platform, we prepared three models: The Basic, Preceding Packet, and Piggyback models. These are considered OpenFlow models, and the network efficiency for each was evaluated.**

Keywords*—*Edge computing, information bank, network transparency, OpenFlow

I. Introduction

Following the penetration of user-oriented data services, personalized services and secondary-use-of-private-data services, there is an increasing need for preserving privacy. Privacy preservation functions are essential to the smart-community data infrastructure. They are in demand because of the changes in regulations for private data in some countries. The General Data Protection Regulation (GDPR) [1] is a legal instrument that came into effect on May 25, 2018 and applies to the European Union (EU). The GDPR states that the regulation applies to unique data that can identify an individual, such as IP addresses and cookies, which manage the user's authentication information. Fines of up to 4% of the global annual turnover, or up to 20 million euros, could be imposed on non-compliant companies that process personal data. If a company fails to comply with this regulation when providing services to EU member states, it could be fined in accordance with the GDPR. Companies providing services to EU member countries are also urged to comply with the GDPR. GDPR restrict the retention of data by service providers, which makes data management more complicated. Against the background of restriction and concerns, the data providers themselves should control data usage, and the smart community needs to adhere to these restrictions.

The concept of an information bank is attracting attention for its ability to support compliance with the regulations. For example, companies such as Google and Amazon retain the location information, purchase history, and search history of each user, and provide recommendation services to match the users' profiles. However, a survey conducted by the Japan Fair Trade Commission (JFTC) revealed that only 12.3% of the users agreed to the terms of use after perusing them. This indicates that the users who provide data do not understand how their data are being used by the service providers, which means that the data providers themselves do not control personal data. An information bank is a system based on the Personal Information Protection Law enforced in Japan, in which a data provider entrusts personal information to a company, within the scope of the data provider's responsibilities. Information banks manage data transactions centrally and act as intermediaries. The information banking system is an essential technological element for establishing a smart community that manages data related to personal privacy. However, in some stream-based instant services, it is also necessary to make peer-to-peer data exchanges between service users as data providers and data consumers as service providers. The system must consider privacy preservation for instant services. Edge computing [2] is a suitable network architecture for this scenario.

Edge computing is a network architecture that sets servers near users, to distribute workloads and decrease the bandwidth required by cloud servers. Compared with cloud processing, edge computing is often described as a low-latency platform; however, restrictions remain on resources for servers distributed near users. An application that requires high computational power, such as machine learning, should be processed by cloud servers with abundant resources to manage computational complexity. Architectures must be selected according to the type of application being used. Edge computing has the advantage that it can apply confidentiality to data before it reaches the cloud, which enables fine-grained access control of data. This type of fine-grained access control is required for users' private data transaction management in a smart community.

A smart community network has a hierarchy based on regional levels. Therefore, appropriate locations for the services are required. Fig. 1 shows the hierarchical structure of the appropriate processing locations and the required technical specifications for each service. Anonymity is a technical requirement. When considering the actual deployment of smart community services, such as for IoT devices, they are closer to the personal area and are provided

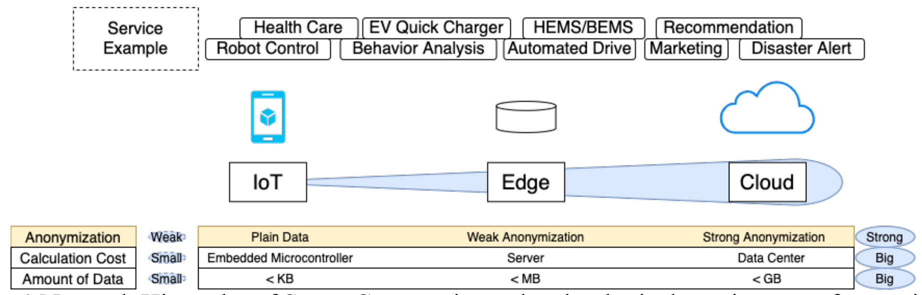

Fig. 1 Network Hierarchy of Smart Community and technological requirements for services

with data that are structured similar to plain text. In the higher network hierarchy, e.g., cloud servers, service is provided using strongly anonymized data. The required degree of anonymization should be increased as service provision moves up the hierarchy. It is important to consider the fact that privacy standards differ from region to region and the management of privacy becomes more complicated when considering privacy level differences.

By managing the data path of services in the network, measures against privacy risk are enabled. In a smart community, the privacy requirements of each region must be understood because privacy levels should be managed regionally. Even if data sharing among users in the same region is acceptable, problems may arise when data are disclosed in another region. In other words, the privacy level is unique to each region. To prevent inter-regional privacy issues, a platform must protect private data within a secure or permitted area.

In this paper, we propose an edge computing-based dataflow management system of private information for smart community services and reduces its required management traffic for improving the network efficiency.

II. Related Works

A. Architecture of Secure Smart Community

Secure platform of smart community needs to fulfill the restrict requirements of GDPR, and the architecture of the platform has been discussed in some papers [3] [4]. These papers only mention the importance of the follow management; however, they do not discuss about the detail of how the data will be managed and processed in the platform. Some papers propose detailed secure platform [5] [6], and they are based on blockchain-method. Our paper will take approach by integrating VCRM system with edge nodes in smart community network to control data access, where edge nodes decide how to process data.

B. Vendor and Consumer Relationship Management

Vendor and consumer relationship management (VCRM) [7] integrates VRM and CRM, and manages and

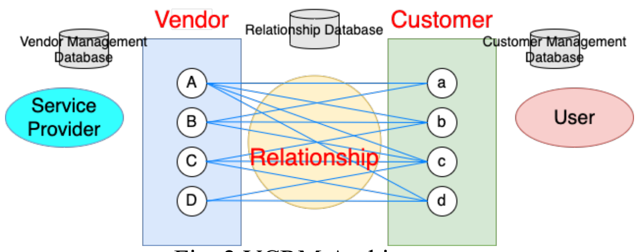

Fig. 2 VCRM Architecture

controls data usage according to customers' and vendors' respective requirements. The architecture of VCRM is illustrated in Fig. 2. The VCRM system defines "Relationship" as a unit of management. Relationships are automatically generated according to the registered attributes of each customer and vendor. In addition, the access to data is determined by the relationship and can be changed by customers. Services will be provided based on the level of access to data. By integrating VCRM with the edge computing-based network infrastructure we propose, edge nodes capture data on route to the cloud and anonymize it or provide services before reaching the cloud. Such integration enables regional access control of data at a network level. The functions of VCRM are those required by an information bank and are necessary for the network architecture to manage the data path.

C. OpenFlow

OpenFlow [8] is a protocol for realizing a software-defined network (SDN), the specifications of which were developed by the Open Network Foundation (ONF) [9].

OpenFlow consists of an OpenFlow controller (OFC) and OpenFlow switches (OFS). The OFC manages the packet forwarding tables on layer 3 switches by packet-matching rules and actions. Routing decisions are made by the controller and are deployed to the OFS, which is entrusted with the actual forwarding. Packets not matching the rules will be forwarded to a controller that decides how to process them.

Owing to the OpenFlow design, switch management is centralized to a controller. The scalability of the number of switches that can be controlled by a controller is often discussed for OpenFlow. To reduce the overhead occupied by controller-directed inquires, there have been proposals on reducing the number of inquiries to the controller [10], and methods to distribute the loads to several controllers [11] [12]. In addition, the problem of the flow table overflowing with entries has also been discussed, and methods have been proposed to reduce the number of flow entries [13].

In this paper, we propose an OpenFlow-based network model to obtain the data path information from the VCRM system.

D. Network Transparency

Network transparency refers to the state in which services can be provided by edge devices in the middle of the network while exhibiting no changes between the cloud and end devices. This means that no functions are added to the end device. Fig. 3 describes the network hierarchy and device functions for each hierarchy. The network

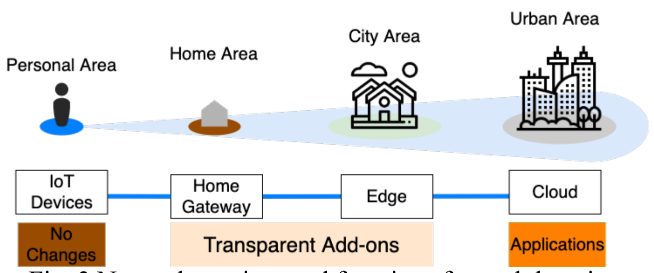

Fig. 3 Network services and functions for each location

architecture of a smart community should meet the conditions of network transparency and provide services to the edge nodes. This method of providing additional services at the edge computing node as an intermediate network node is termed a transparent add-on.

1) Authorized Stream Contents Analysis

Authorized stream content analysis (ASCA) [14] is a technology for reconstructing TCP streams and extracting or modifying information even from encrypted data communication, such as HTTPS used in web services and MQTT over SSL used in IoT applications. For SSL decryption, ASCA maintains a shared key at the edge, which, in advance, is shared from the cloud. In addition, ASCA has features to reconstruct TCP streams by decompressing data compressed by gzip, which is used to decrease data transfer loads in web services and decode the chunked encoding used to maintain HTTP communications. In addition, it extracts strings and rewrites information from the reconstructed TCP stream using regular expressions. This technology realizes protocol conversion [15], data modification, and security improvement transparently in the intermediate nodes between the cloud and end devices. ASCA enables transparent add-ons. Moreover, ASCA is used for transparent service migration [16].

2) Context Cache Management function

Context cache management (CCM) is a function provided in ASCA. The CCM function manages information of the stream entering the edge device via the hash table in the ASCA mechanism. The CCM function randomly monitors the arriving streams and caches the packet header and data processing information when a new packet is detected. The data are reused for processing continuous packets of the same stream. Considering the cache misses of stream information caused by the hash table's space limit, the CCM function has a timeout manager that removes stream information that has not been forwarded for a specified time. The CCM function can be

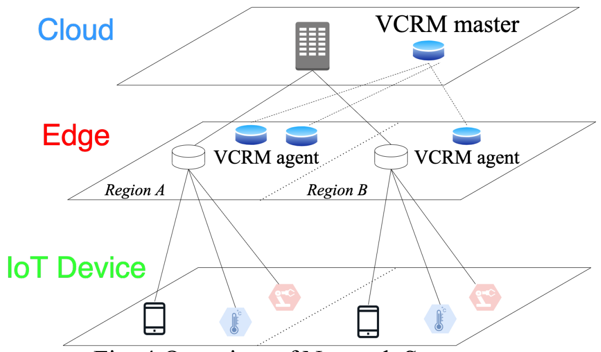

Fig. 4 Overview of Network Structure

leveraged to process privacy content transmitted in the network.

III. Proposed Platform

We propose a platform consisting of three layers: the cloud, edge, and IoT device regions. In our proposed architecture, the edge nodes detect streams sent from IoT devices and process data. The network structure is illustrated in Fig. 4.

A. Data Structure of Relationship

To ensure privacy management and personal information protection in smart community networks, we propose a structure to manage data paths. We consider that the content managed in a smart community is related, as shown in Fig. 5.

The service provided in a smart community consists of relationships among service providers and users. For each relationship, the services that users receive are determined and managed based on the one-provider-to-many-users structure for each relationship. In addition, services are managed based on a concept consisting of a combination of streams identified by five tuples and Layer 7 content information held by the stream. This study terms this dataset as "dataflow" and manages the data path using it as a service identifier. To determine the dataflow, the contents of the application layer must be considered.

B. VCRM master / VCRM agent

A distributed VCRM system to manage the relationships of the smart community was deployed on the proposed platform. The VCRM master is defined as the parent server, which is placed on the cloud, and the VCRM agents as child servers, which were placed in the edge region. The VCRM agents are distributed in the edge region, and the processing loads on the parent server are balanced by the agents. The edge nodes inquire from the VCRM master/VCRM agent servers to confirm whether the data contents are managed.

C. Platform Overview

When assuming the actual operation of the dataflow management platform in the edge region, it is not feasible to deploy the edge nodes to inquire from the VCRM agents every time new dataflow information is detected. Excessive loads will concentrate on the VCRM agents because of the inquiries to the VCRM sent from all edge nodes. In addition, the communication delay caused by edge nodes' inquiries to the VCRM server accumulates, rendering the VCRM mechanism unfeasible. As described in the OpenFlow mechanism in Section II, the edge nodes store the dataflow information obtained by inquiring from the VCRM agent once, similar to the relationship between OFC and OFS. When the same dataflows are detected, they should not be

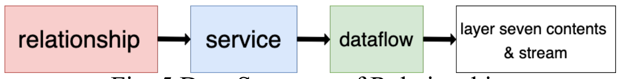

Fig. 5 Data Structure of Relationship

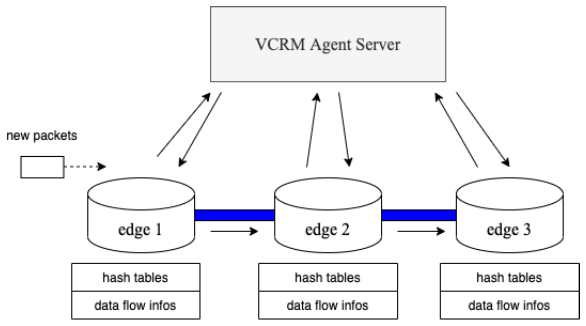

Fig. 6 Basic Model

inquired about from the VCRM servers. Therefore, the edge node forwards the packet to the next node without a query being sent to the VCRM servers. By processing many packets in the edge region, dataflows close to the user are detected, and this functions as a data filter that quickly determines whether the data are in the management ambit. The VCRM master / VCRM agent knows which edge nodes hold the information.

IV. IMPLEMENTATION

This section describes the technical requirements and models implemented for the proposed platform.

A. DPDK

DPDK [17] is a set of libraries and networking tools developed by the Linux Foundation to achieve high-speed packet processing. DPDK uses kernel bypass technology to realize high-speed packet processing. Kernel bypass technology accelerates packet processing by enabling direct control of hardware such as NICs and CPUs by user space rather than kernel space. This eliminates the context-switching process and accelerates packet processing.

B. Libpcap

Libpcap [18] is a packet sniffer API that captures packets flowing on a network. It is implemented in C/C++, and the APIs are provided as libpcap, which was developed by the Tcpdump development team for the UNIX OS. A zero-copy function is implemented to improve the efficiency of packet processing, enabling high-speed packet processing.

C. Kyotocabinet

The Kyotocabinet [19] is a key-value store and is a typical NoSQL implementation. We use the kyotocabinet as a table of edge nodes to store dataflow information. Unlike RDBMS, NoSQL omits transaction management, and this feature of NoSQL enables high-speed data processing. It has a simple dictionary-type key value data structure that enables many records to be retained without degrading performance. In this implementation, five tuples are used as the key of the kyotocabinet in use, and the value is set to route information. Originally, the key had to be an identifier determined by five-tuple and Layer 7 contents; however, as a proof-of-concept implementation, the details of dataflows are not managed in this study. The dataflow information was provided by the VCRM master. In this study, the implementation assumed that all five tuples are independent dataflows.

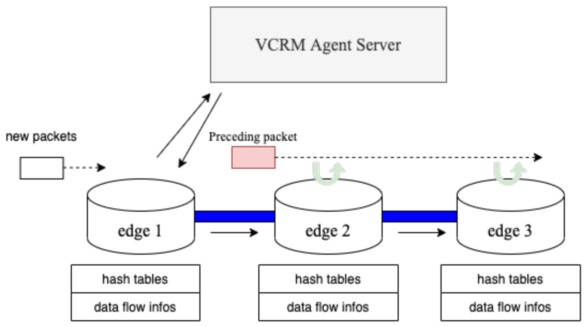

Fig. 7 Preceding Packet Model

D. Models of Platform

To evaluate the latency and the difference in data traffic caused by the interaction between the edge and the VCRM agents, three models were implemented and compared in this study. The edge nodes share a common context-cache management function.

1) Basic Model

The basic model was implemented using C++ and the packet sniffing library, libpcap. Model is described as Fig. 6. When a packet arrives, it checks if there is a key value in the hash table identical to its own 5-tuple. If there is an identical key value, it passes the packet. Otherwise, it obtains the dataflow information by inquiring from the VCRM agent, adds a record to the hash table, and then forwards the packet.

2) Preceding Packet Model

The preceding packet model was implemented using C++ and the packet sniffing library libpcap. Model is described as Fig. 7. When a packet arrives at an edge node, the hash table of the node checks whether a key with the same five tuple exists in the table. If it does, the packet is forwarded to the next node. If not, it obtains the dataflow information by inquiring from the VCRM agent and sends a UDP packet to the following edge node as a dataflow information transfer packet. The UDP packet sent by an edge node for dataflow information transfer contains five tuples and routing information as a message. A subsequent edge node that detects the UDP packet stores the message in its own hash table. To prevent dataflow information transfer packets from being detected as new dataflows, we implemented a dedicated socket interface in the edge node to receive these UDP packets. The arriving packets are temporarily buffered until a dataflow information transfer packet is sent.

3) Piggyback Model

The piggyback model was implemented using the C programming language and the DPDK packet processing library. When a packet arrives, it detects whether dataflow

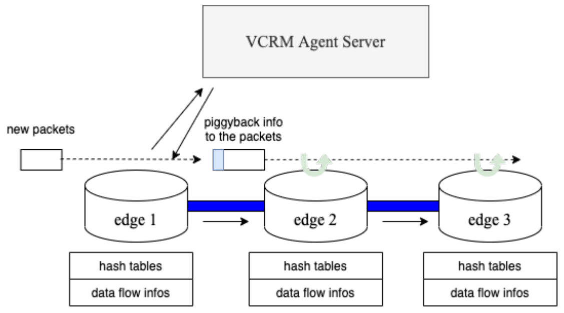

Fig. 8 Piggyback Model

TABLE I. EXPERIMENTAL ENVIRONMENT

	Intel NUC	Shuttle DH310
OS	Ubuntu 18.04.5 LTS	Ubuntu 18.04.5 LTS
CPU	Intel(R) Core™ i5-8259U @ 2.30 GHz	Intel(R) Core™ i7-8700 @ 3.20 GHz
Memory	8 GB	32 GB
Size	102 x 102 x 28 mm	190 x 165 x 43 mm

TABLE II. TEST TRAFFIC

Number of packets	14261 [packets]
Number of Applications	28
Traffic duration	5 [min.]

information is appended to the end of the packet. As shown in Fig. 8, the piggyback data are detected by subtracting the length of the payload from the length of the packet, including the piggyback. If not, an edge node appends dataflow information to the packet. If appended, the edge node only forwards the packet. The formula for detecting the piggyback of a packet is given by (1), using the original packet length $L_{original}$ and the length with the piggyback $L_{piggyback}$.

$$\begin{cases} L_{piggyback} - L_{original} > 0 \ (no\ piggyback) \\ L_{piggyback} - L_{original} = 0 \ (with\ piggyback) \end{cases} \quad (1)$$

V. EVALUATION AND RESULT

We measured and evaluated the amount of data traffic generated by inquiries to the VCRM agent and the processing delay caused by the inquiries for each of the three models, which is a concern in the proposed platform design. TABLE I shows the experimental environment to evaluate the three models implemented. The test traffic described in TABLE II is sent to evaluate the performance of the model, beginning from edge node 1.

A. Data Traffic of inquiry to VCRM

As shown in Table III, the amount of data traffic is the highest in the basic model. The VCRM agent is inquired independently in the basic model because the edge nodes do not share notifications about the acquisition of dataflow information among other edge nodes. The basic model is considered to be inefficient because of the redundant inquiry to the VCRM agent without considering any processing, which can be effected only by a single edge node. The preceding packet model and piggyback model reduce the amount of data communication to 32% compared to the basic model. We prepared three edge nodes in the experimental environment. In an actual smart community network, packets are likely to pass more than three edge nodes. When the number of edge nodes is n_{edge} and the number of dataflows is n_{DF}, the general formula for record reduction can be replaced by (1). This indicates that the edge nodes reduce redundant data traffic by cooperating.

$$\frac{n_{DF}}{n_{DF} * n_{edge}} \risingdotseq \frac{1}{n_{edge}} \quad (1)$$

TABLE III. DATA TRAFFIC MEASURED IN EACH MODEL

Model	Data Traffic [Kbytes]
Basic	9115
Preceding packet	2934
Piggyback	2926

TABLE IV. PROCESSING DELAY

Process	Latency [ms]
Hash table reference	0.06
VCRM inquiry	13.36

TABLE V. RECORDS MEASURED IN EACH MODEL

Model	Edge node 1	Edge node 2	Edge node 3
Basic	547	547	547
Preceding packet	547	547	547
Piggyback	547	0	0

B. Process Delay

We measured the processing delay caused by an edge node inquiring from the VCRM agent. We verified the process of referring the hash table and inquiring to the VCRM server 100 times each and calculated the average values. The results are shown in Table IV, and the ratio of the processing delay is shown in (2).

$$\frac{13.36}{0.06} = 222.7 \quad (2)$$

There is at least 200 time difference in the processing delay. Therefore, the delay caused by the hash table reference process is small compared to the VCRM master/VCRM agent inquiry process. This indicates that the basic model method, in which the generated processing delay accumulates in the number of nodes, also leads to a decrease in QoS.

C. Number of Records in the hash table

The number of records kept in the hash table by the edge nodes is listed in Table V for each implementation. The same number of records was retained for each edge in the has tables of the basic and preceding packet models. For the piggyback model, when a packet is detected to have piggyback data, it is only passed by the subsequent edge node, and thus only edge node 1 holds the data. This means that the data is held only at the edge node where the packet is first detected and dataflow information can be distributed among the edge nodes. From the results, we can see that the piggyback model reduces the number of records in the hash table by 33% compared to the basic and preceding packet models. Similar to the data traffic efficiency, when we define the number of edge nodes as n_{edge} and the number of dataflows as n_{DF}, the general formula for record reduction can be replaced by (3).

$$\frac{n_{DF}}{n_{DF} * n_{edge}} = \frac{1}{n_{edge}} \quad (3)$$

Using the piggyback method, we can see that the data retention efficiency of hash tables in the edge region can be increased as the number of edge nodes increases.

VI. CONCLUSION

In this paper, we proposed an edge computing-based data management platform for smart community and implemented three different models to make management efficient. In contrast to the basic model, which does not cooperate with edge nodes, the preceding packet and piggyback models successfully reduced the amount of data traffic between the VCRM master/VCRM agent and the edge nodes to 32%. Moreover, the piggyback model also reduced the number of records retained in the hash tables in the edge region to 33%. These results indicate that a mechanism to collaborate with edge nodes is necessary to realize an efficient dataflow management mechanism.

Acknowledgment

This work was supported by JST CREST Grant Number JPMJCR19K1, and the commissioned research by National Institute of Information and Communications Technology (NICT, Grant Number 22004), JAPAN.

References

[1] W. S. Blackmer, "EU general data protection regulation," 2018.

[2] "What is edge computing?," Cloudflare, [Online]. Available: https://www.cloudflare.com/learning/serverless/glossary/what-is-edge-computing/. [Accessed 13 4 2021].

[3] C. Metallidou, K. E. Psannis and E. Alexandropoulou-Egyptiadou, "An Efficient IoT System Respecting the GDPR," *2020 3rd World Symposium on Communication Engineering (WSCE),* pp. 79-83, 2020.

[4] M. Ati and T. Basmaji, "Framework for managing smart cities security and privacy applications," *2018 IEEE Symposium on Computer Applications & Industrial Electronics (ISCAIE),* pp. 191-194, 2018.

[5] N. B. Truong, G. M. L. Kai Sun and Y. Guo, "GDPR-Compliant Personal Data Management: A Blockchain-Based Solution," *IEEE Transactions on Information Forensics and Security,* vol. 15, pp. 1746-1761, 2020.

[6] Y. Gao, Y. Chen, H. Lin and J. J. P. C. Rodrigues, "Blockchain based secure IoT data sharing framework for SDN-enabled smart communities," *IEEE INFOCOM 2020 - IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS),* pp. 514-519, 2019.

[7] A. Niwa and H. Nishi, "An Information Platform for Smart Communities Realizing Data Usage Authentication and Secure Data Sharing," *Proceedings - 2017 5th International Symposium on Computing and Networking, CANDAR 2017,* Vols. 2018-January, pp. 119-125, 2018.

[8] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker and J. Turner, "OpenFlow: Enabling innovation in campus networks," *ACM SIGCOMM Computer Communication Review,* vol. 28, no. 2, pp. 69-74, 2008.

[9] "Open Networking Foundation," [Online]. Available: https://opennetworking.org/. [Accessed 8 4 2021].

[10] E. D. Kim, S. I. Lee, Y. Choi, M. K. Shin and H. J. Kim, "A flow entry management scheme for reducing controller overhead," *International Conference on Advanced Communication Technology, ICACT,* pp. 754-757, 2014.

[11] H. G. Ahmed and R. Ramalakshmi, "Performance Analysis of Centralized and Distributed SDN Controllers for Load Balancing Application," *2018 2nd International Conference on Trends in Electronics and Informatics (ICOEI),* pp. 758-764, 2018.

[12] M. Nkosi, A. Lysko, L. Ravhuanzwo, T. Nandeni and A. Engelberencht, "Classification of SDN distributed controller approaches: a brief overview," *Proceedings - 2016 3rd International Conference on Advances in Computing, Communication and Engineering, ICACCE 2016,* pp. 342-344, 2017.

[13] A. Iyer, V. S. Mann and N. R. Samineni, "SwitchReduce: Reducing switch state and controller involvement in OpenFlow networks," *2013 IFIP Networking Conference, IFIP Networking 2013,* pp. 1-9, 2013.

[14] W. A. A. P. Shanaka, J. Wijekoon, R. L. Tennekoon and N. Hiroaki, "Software-accelerated Service-oriented Router for Edge and Fog Service Enhancement Using Advanced Stream Content Analysis," *IEEJ Transactions on Electronics, Information and Systems,* vol. 139, no. 8, pp. 891-899, 2019.

[15] K. Saito and H. Nishi, "Application Protocol Conversion Corresponding to Various IoT Protocols," *IECON Proceedings (Industrial Electronics Conference),* Vols. 2020-Octob, pp. 5219-5225, 2020.

[16] T. Miura, J. Wijekoon, S. Prageeth and H. Nishi, "Novel infrastructure with common API using docker for scaling the degree of platforms for smart community services," *Proceedings - 2017 IEEE 15th International Conference on Industrial Informatics, INDIN 2017,* pp. 474-479, 2017.

[17] "Data Plane Development Kit," [Online]. Available: https://www.dpdk.org/. [Accessed 8 4 2021].

[18] "TCPDUMP/LIBPCAP," [Online]. Available: https://www.tcpdump.org/. [Accessed 8 4 2021].

[19] "Kyoto Cabinet: a straightforward implementation of DBM," [Online]. Available: https://dbmx.net/kyotocabinet/. [Accessed 8 4 2021].