Network Transparent Fog-based IoT Platform for Industrial IoT

Ryo Morishima
Graduate School of Science and Technology
Keio University
3-14-1 Hiyoshi, Kohoku, Yokohama, Kanagawa 223-8522, Japan
morishima@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

***Abstract*—There has been rapid growth in the number of Internet of Things (IoT) devices that produce a large volume of data such as location data, temperature data, and power usage data. These data are used in various types of services on IoT platforms. IoT services such as autonomous vehicles and factory automation system have the following three requirements. First, confidential data such as machine ID, power usage, and location information need to be locally anonymized before they are sent to a cloud server. Second, these applications require low latency response of less than 10 ms. Third, the bandwidth usage that an enormous number of IoT devices generate needs to be efficiently reduced to alleviate the burden on the cloud server. However, existing IoT platforms have limitations on the configuration of applications because most of them are cloud-based. In this study, we propose a fog-based IoT platform, where fog nodes achieve network transparency for the IoT devices and the cloud server. The network-transparent machine can be installed without modification of the existing network configuration and routing. Fog nodes transparently achieve the three requirements: anonymization of specific data in packets, real-time feedback, and reduction in the bandwidth usage.**

***Keywords*—IoT platform, fog computing, network transparency**

I. Introduction

In recent years, the number of Internet of Things (IoT) devices has increased drastically. Cisco white paper [1] predicted that the number of M2M connections would reach 14.6 billion by 2022, considering the growth of the Internet, and it would account for 51% of all the connections in 2022. This enormous number of IoT devices generate a large amount of network traffic to the IoT platform. This increase in the connections leads to the increase in computational resources to handle incoming requests from the networks in a cloud. To address this situation, an increasing number of computing and storage resources need to be allocated in the cloud to avoid overloading the cloud. However, it would be difficult for the cloud to process all the requests from the IoT devices. Therefore, the IoT platform using fog computing is proposed for offloading the processes in the cloud. Fog computing and IoT platform are explained in the following section.

Fog computing is a horizontal, system-level architecture that distributes computing, storage, control, and networking functions closer to the users along a cloud-to-thing continuum [2]. Fog nodes are located near the end devices; therefore, fog computing can provide lower latency compared to cloud computing. On the contrary, in fog computing, the usable resources are limited compared with a cloud-based application. In this context, fog computing and cloud provide a cooperative computing environment for the future.

The IoT platform facilitates the deployment of software applications for providing data-oriented services within a local area. It manages all resources for providing local services such as embedded computing resources, storage attached to the computing devices, communication resources, and allocation of software and hardware resources for maximizing the total efficiency of the service provisioning.

For constructing the IoT platform with fog computing environment, Service-oriented Router (SoR) was proposed [3]. SoR is a router that uses a special middleware: Deep packet inspectOr On Router (DooR) as its packet and stream handling function. SoR enables to capture and analyze all the packet stream contents and provide IoT services to end users and applications by using the contents. The details of SoR and its middleware DooR are described in section II.

IoT platform with fog computing environment must meet the following requirements to provide various IoT services.

Firstly, time-critical applications such as smart power grid and factory automation system require real-time packet processing. International Telecommunication Union (ITU) reported a draft [4] claiming that factory automation needs under 10 ms end-to-end delay.

Secondly, fog nodes need to reduce the bandwidth usage that IoT devices generate in the platform to alleviate the burden of traffic in the cloud server. At the same time, the size of machines acting as fog nodes must be small enough to be installed into the existing network. A typical server machine that is installed in a server room is too large to be installed in the existing network infrastructure, for example, 5G antenna modules, network router, switch devices. While considering installability for fog use, it is advisable to use Barebone PC-based Mini-STX motherboard matching and supporting 1 Gbps throughput after classification of network traffic.

Thirdly, the IoT platform must be able to transparently deploy additional applications and services without modifying the network configuration between IoT devices and cloud servers. To investigate such network transparency, an anonymization application that has an important function in the future IoT services is discussed.

978-1-7281-2927-3/19/$31.00 ©2019 IEEE

II. Background

A. Cloud-based IoT platform

Cloud platform provides the flexibility for users to scale services to fit their needs, customize applications, and access cloud services from anywhere via the Internet. Moreover, by using cloud platforms, enterprise users can bring applications to market quickly without worrying about underlying infrastructure costs or maintenance.

Although cloud computing is an efficient solution for processing content in distributed environments, it is less suited for applications that require low latency, data aggregation on middle nodes, and anonymization of privacy-sensitive data. For example, Google Cloud Platform (GCP) [5] provides a load balancing option to reduce end-to-end latency. However, it does not guarantee the latency requirement. Amazon Web Service (AWS) [6] has a configuration option that can reduce bandwidth usage by throttling. However, this approach also reduces the amount of data transferred to the cloud. Microsoft Azure [7] provides Azure stream analytics (ASA) [8]. ASA leverages IoT edge devices that respond with low latency, preprocessing of data, and anonymization. However, unlike fog computing, such IoT edge devices do not have hierarchical structure; therefore, they cannot deploy applications dynamically based on latency requirement. Other cloud-based IoT platforms such as Oracle [9], Kaa [10], and Aeris [11] are also not capable in this respect.

B. Fog-based IoT platform

Fog-based IoT platform distributes traditional cloud computing resources and services to the network edge, thereby bringing cloud resources nearer to the endpoints of the network (e.g., sensors and other IoT devices). Fog computing resources are located in the local network to avoid the need to send data to the servers in a remote cloud for processing and storage. Therefore, fog-based platforms can serve applications that are not well suited to (centralized) cloud computing.

OpenFog is the leading community of fog computing in the world. The OpenFog reference architecture that has been adopted by IEEE for fog computing was released in February 2017. However, there are no real-world implementations with the capabilities of anonymizing sensor data on the middle nodes, low-latency response, and efficiently reducing bandwidth usage of IoT devices. Other fog-based platforms such as OpenIoT [12] and Linksmart [13] provide the functionality of aggregating IoT data stream and simplify service deployment. However, they do not provide a function to anonymize IoT data.

These IoT platforms in addition to other platforms such as Foglets [14] and FogFlow [15] mainly focused on how to optimize task deployment over distributed devices in terms of saving bandwidth and reducing latency. However, they did not focus on the method to keep fog nodes transparent from IoT devices and cloud server.

C. Service-oriented Router (SoR)

SoR is a router comprising a packet analyzing processor and an ample in-network storage module [16]. SoR was proposed with the aim of providing content-based services by performing authorized stream content analysis (ASCA) to analyze packet streams and storing necessary packet stream information (e.g., the direction of the stream, 5-tuple) [3]. ASCA is a method of analyzing packet streams by considering both privacy and service affinity. For achieving privacy, it filters the streams to be analyzed according to the marker tags in the contents of the streams under the Opt-In manner as a prior confirmation. SoR can be implemented on general-purpose hardware. It supports hardware accelerators for improving the processing throughput.

Moreover, SoR can be used as transparent network hardware such as an L2 switch. SoR can be installed as a general router, a stream capturing device, or an application server without modifying the original network configurations and routing tables. Therefore, SoR can provide various types of IoT services when an appropriate IoT platform server program is executed as an application of SoR. The application can provide the function of software-defined resource allocation and management and provide more flexibility compared to hardware implementations. Next, we explain DooR, which is our research team's software implementation toward SoR.

1) Main Components of DooR: DooR's fundamentals comprise Intel Data Plane Development Kit (DPDK) [17]. Intel introduced a packet processing framework: DPDK to analyze packets and simultaneously maintain high throughput packet flow. The speed of Linux kernel networking is not sufficient for specialized workloads. Consequently, DPDK bypasses Linux kernel and processes packets in the Linux user space. To achieve packet analysis, SoR uses Intel Hyperscan [18]. Hyperscan is a high-performance multiple regular expression matching library. Intel showed that they could analyze packets by providing throughput up to 160 Gbps [19].

2) Network Transparency: DooR can be used as a transparent middlebox. The end hosts can ignore the existence of SoRs; therefore, they can communicate with each other without the need to know where SoRs are located (IP address). In case of an IoT platform, the location of a resource does not hold any significance to either the IoT application developers or the IoT devices. An additional benefit is the flexibility it provides. Applications can be migrated to a different computer anytime without changing the network setting of IoT devices.

III. Proposed platform

We propose an IoT platform comprising a three-layer hierarchical structure as shown in Fig 1: cloud node, fog nodes, and IoT devices. In our proposed architecture, fog nodes are network-transparent and perform ASCA.

Here, we define the meaning of network transparency. Network-transparent machine refers to a machine that can be installed without modification of the existing network configuration and routing. In case of network-transparent fog nodes, IoT devices and cloud nodes are not required to manage the configuration of layer 3 (IP). Therefore, IoT devices and cloud nodes can communicate with each other without knowing the location of fog nodes.

A cloud node communicates with IoT devices via TCP to analyze and store sensor data into a database. When platform users deploy their applications, the cloud node receives the applications and their configuration files having the necessary information for deployment such as CPU resources, memory resources, disk size, and latency requirement. By considering these factors, the application is deployed on a suitable machine (e.g., the lower the latency requirement, the closer the deployment to end devices). The configuration file also describes the necessary information to invoke the right application to

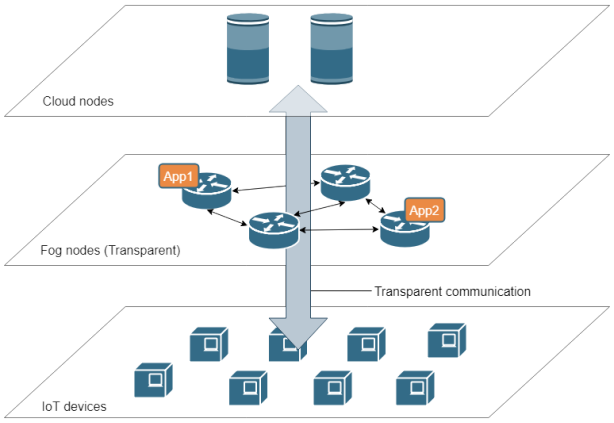

Fig. 1. Proposed system

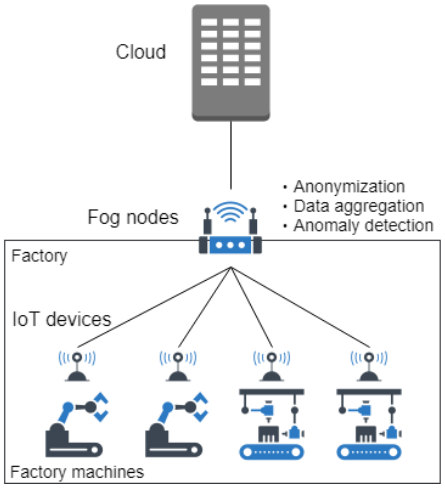

Fig. 2. Industrial IoT application

process sensor data. For example, when pattern matching is used for ASCA to identify packets with specific contents, the rule of the pattern matching should be described in the configuration file.

The fog nodes in the proposed system are network-transparent devices; therefore, the cloud node and the IoT devices do not have to know the IP addresses of fog nodes. In the network-transparent environment, the cloud node and the IoT devices are not required to change the communication partner irrespective of the application's location on the fog nodes. The cloud node and the IoT devices always point to the opponent's IP address without knowing which fog nodes process the application. Fog nodes perform ASCA according to the rule described in the configuration file received by the cloud node. The rule is used to execute user applications on specific sensor data. IoT platform must transparently achieve the three requirements: anonymization, low-latency response, and bandwidth reduction.

IV. Implementation

This section describes our scenario of IoT application and its implementation.

A. Target IoT application

We consider applications of industrial IoT. Fig 2 illustrates the three-tier architecture of the environment. IoT sensors installed in factory machines communicate with the cloud node via TCP/IP, where fog nodes transparently provide three functions: anonymization of confidential data, anomaly detection with low latency feedback, and data aggregation for bandwidth reduction.

B. Implementation overview

The implementation has a three-tier architecture: cloud node, fog nodes, and IoT devices.

- *Cloud node:* When a cloud node tries to deploy applications, it instructs the IoT devices to calculate round - trip time (RTT) to the fog nodes and the cloud node. The IoT devices send dummy data to identify a machine that satisfies the latency requirement of the application. In this implementation, the configuration file is written in YAML.
- *Fog nodes:* Fog nodes are machines using DooR. DooR transparently modifies the payload and generates acknowledge packets. These features are explained in section IV-C.
- *IoT devices:* IoT devices periodically send JSON formatted data to a cloud node. IoT devices always specify the cloud node's IP address.

C. Network Transparency on TCP

To operate fog nodes as network-transparent machines and achieve anonymization, low-latency response, and bandwidth reduction on fog nodes, we implemented two additional functions on DooR.

1) Network-Transparent Payload Modification: When DooR modifies payload of a packet by using DPDK and Hyperscan, the total packet length changes. If DooR forwards the packet to the cloud node without updating the IP and TCP headers, the TCP connection between the IoT devices and the cloud node will be terminated because of the mismatch of checksums. DooR must update checksum, sequence number (seq), and acknowledge (ack) number after payload modification. This function enables DooR to recalculate checksum, seq number, and ack number of the header to achieve network transparency.

This function allows DooR to store the necessary information to manage multiple TCP connections. Once DooR modifies the payload of a packet in a TCP connection, it stores the number of modified bytes, the direction of modified packets (client to server or vice-versa), and the hash value of 5-tuple of the TCP connection. When DooR captures a packet, it detects how many bytes were modified in the direction from the time since the connection was established, and recalculates checksum, seq, and ack number of the header.

2) Network-Transparent Acknowledge Packet: As aforementioned, DooR can update a header by recalculating IP and TCP headers. Using this, we implemented a function that creates a transparent acknowledge packet that is compatible with an existing TCP connection. When DooR transparently sends a message to an IoT device, it generates an acknowledge packet by swapping the source and the destination, adding the message on payload, and recalculating checksum. Moreover,

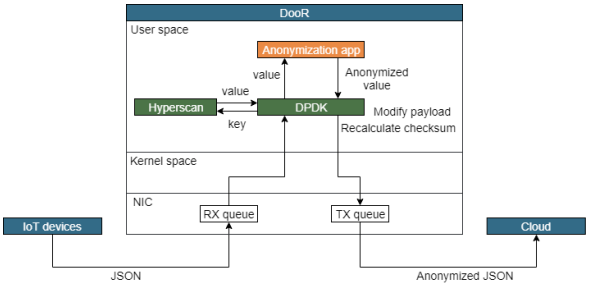

Fig. 3. Anonymization on DooR

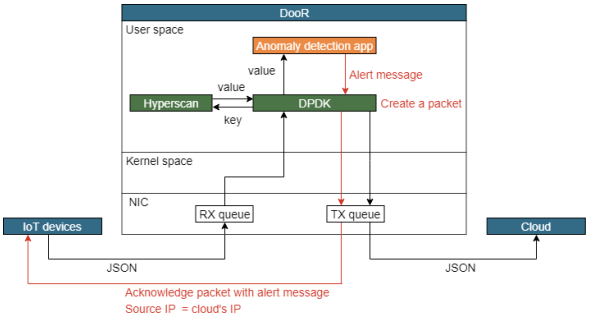

Fig. 4. Anomaly detection on DooR

DooR stores the length of the sent message (bytes) to recalculate the seq and ack numbers.

D. Application of network transparency

There are three applications that use the feature explained in section IV-C.

1) Anonymization application on DooR: Fig 3 shows the flow of anonymization on DooR. DooR can modify any string in a packet transparently. It uses Hyperscan to find a JSON key whose value is to be anonymized. Further, it sends that value to Python scripts (application on fog nodes). It then receives the result of the Python scripts and modifies the payload. Users can configure the desired JSON keys in a YAML configuration file when they deploy anonymization applications.

2) Anomaly detection and low-latency feedback: In addition to anonymization on fog nodes, DooR detects the configured keys in packets and sends their values to the anomaly detection application as shown in Fig 4. If the application does not detect any anomaly, DooR behaves like a usual router and forwards the packet to the cloud node. If the application notices that it needs to send the alert messages back to the IoT devices, DooR creates and sends transparent acknowledge packets.

3) Reduce the number of headers on DooR: The function that generates transparent acknowledge packets achieves efficient bandwidth reduction while ensuring low-latency communication. Fig 5 shows the overview of data aggregation on DooR. DooR aggregates the contents of a packet if the sum of the lengths of the contents does not exceed maximum transmission unit (MTU). Alternatively, this function reduces the number of transmitted headers (Ethernet + IP + TCP) by waiting for the packets and filling the payload. To avoid stacking on waiting, DooR sends a packet after a certain period elapses, even if the payload is not filled up.

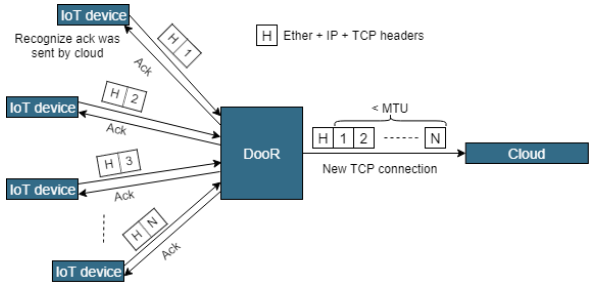

Fig. 5. Data aggregation on DooR

In the case when DooR reduces the number of headers, it no longer remains network transparent for the cloud node. This is because when DooR sends concatenated content to the cloud node, it must create a new TCP connection with the cloud node. However, DooR can ensure network transparency for IoT devices by creating and sending transparent acknowledge packets. Therefore, IoT devices can receive low-latency feedback while the data aggregation function is working on DooR.

V. Evaluation and Results

We evaluated the described functions in section IV. Barebone PC, Shuttle DH310 was used in the experiment. Shuttle DH310 is illustrated in Figs 6 and 7. Table I shows the specifications of the machine used for the evaluation. Table II shows the JSON formatted data created at IoT devices for evaluation. The data includes five keys: what type of sensor (measurement), the hostname (host), its location (region), timestamp (time), and value collected by the sensor (value).

Fig. 6. Shuttle DH310

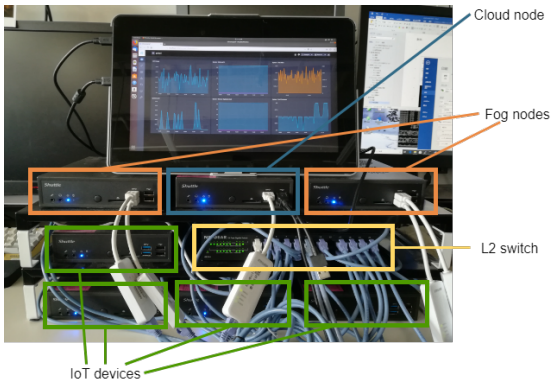

Fig. 7. Simulation environment using Shuttle DH310

TABLE I
SIMULATION ENVIRONMENT

Device	Shuttle DH310
OS	Ubuntu 18.04.1 LTS
CPU	Intel® Core^{TM} $i7-8700CPU@3.20GHz$
Memory	32 GB
Language	C, Python
Communication protocol	TCP
NIC	1 Gbps

TABLE II
JSON FORMATTED SENSOR DATA

Key	Value
measurement	”temperature”
host	”host00” - ”host99”
region	”region0” - ”region9”
time	”YYYY-MM-DD hh:mm:ss”
value	temperature value

TABLE III
THE RESULT OF ANONYMIZATION ON DOOR

Before processing (6 bytes)	After processing (8 bytes)
”host48”	”C5C0RQZ7”
”host23”	”BCYEQJZR”
”host56”	”ZCYPUPZ3”

DooR executed two applications: anonymization application and anomaly detection application between the IoT devices and the cloud node. We assumed that the anonymization application receives and modifies the value of the ”host” key in Table II. The key: ”host” was described in the configuration file. Similarly, the anomaly detection application judges if DooR needs to send feedback based on the value of temperature. Hence, key: ”temperature” was described in its configuration file. The latency requirement of anomaly detection app was 10 ms. Both the applications were written in Python. For simulating a real-world environment, we added latency in the simulation environment. The latency between IoT devices to DooR and IoT devices to cloud node was 7.5 ms and 13.2 ms, respectively. The latency between IoT devices to DooR was the measured latency from an end device to an ISP's router. The latency between IoT devices to cloud node was the latency from an end device to the AWS cloud server.

1) Anonymization application: Table III shows the result of the data modification on SoR. The raw data was modified into a hash value to prevent identification of the host machine. In this experiment, hashing was used to modify sensor data. However, there are several ways to modify data to achieve anonymization such as masking sensitive data and adding noise to sensor data. These data modification methods depend on user applications. Users can define how an application should anonymize data.

The data processing shown in Table III changed the length of the data from 6 to 8 bytes. In this case, as mentioned in point IV-C1, SoR stored the information about how many bytes were modified and recalculated seq, ack, and checksum. Consequently, SoR achieved transparent payload modification in the middle of IoT devices and cloud node.

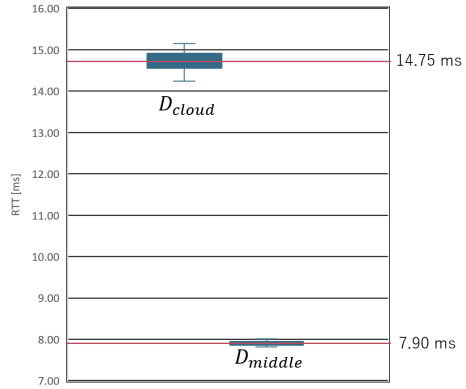

Fig. 8. RTT comparison between D_{cloud} and D_{fog}

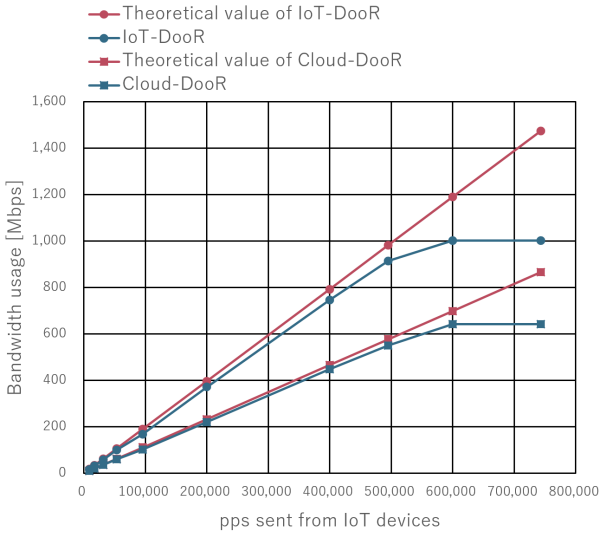

Fig. 9. Bandwidth reduction

2) Anomaly detection application: Fig 8 shows the result of the feedback from the anomaly detection application, where D_{cloud} represents RTT between IoT devices to cloud node and D_{middle} represents RTT between IoT devices to SoR. The result shows that the average of D_{middle} was 6.85 ms lower than the average of D_{cloud}, and the variance of D_{middle} was also lower. Therefore, when the latency requirement of the application was 10 ms, it was deployed on fog nodes to meet the requirement, and not on the cloud node.

3) Reduce the number of headers: Fig 9 shows the relationship between packets per second (PPS) from IoT devices to DooR, the bandwidth usage between IoT devices and DooR, and the bandwidth usage between DooR and the cloud node. In other words, Fig 9 shows how data aggregation function on DooR reduced the bandwidth usage. The total length of the packets that the IoT devices sent to the cloud node was 188 bytes (54 bytes header and 134 bytes payload). DooR concatenated ten units of payload, thereby causing the total packet length to be 1,397 bytes. Each packet needs an acknowledge packet of 60 bytes. The ratio of bandwidth reduction is obtained by equation 1.

$$1 - \frac{1397bytes + 60bytes}{(188bytes + 60bytes) \times 10transactions} = 0.41 \quad (1)$$

Fig 9 shows that DooR reduced the bandwidth usage between DooR and the cloud by 41% up to 1 Gbps throughput. The saturation throughput was 1 Gbps because the NIC used in the experiment was 1 Gbps.

When DooR concatenated the JSON formatted data from IoT devices, it preserved the JSON format rather than simply concatenating the data to assemble a single packet. Consequently, the cloud node could read the payload as valid JSON formatted data, which reduced the burden of parsing the payload.

An additional benefit was the anomaly detection function mentioned in subsection V-2 could run independently of this function because DooR could create and send transparent acknowledge packets while concatenating the payloads. Every time IoT devices sent sensor data, each IoT device received an acknowledge packet from DooR; the source of the packet was seemingly the cloud node. This means that the latency measured from IoT devices was always the same as D_{middle}, even if DooR was concatenating the payloads.

VI. Conclusion

In this paper, fog-based IoT platform was proposed. It enables fog nodes to anonymize sensitive data, respond to a message with low latency, and reduce the required bandwidth by aggregating multiple data transactions. DooR was used for achieving network transparency of fog nodes. Network-transparent payload modification and network-transparent acknowledge packet generation as functions of DooR were implemented, and the correct operations of these functions were verified. Results showed that 1) user-defined application on DooR transparently modified data at any place in a packet, 2) DooR sent acknowledge packet to IoT devices transparently with latency less than 10 ms, and 3) DooR reduced the bandwidth usage by up to 41% by concatenating the contents whenever the sum of the lengths of the contents did not exceed MTU. DooR achieved low-latency feedback regardless of the data aggregation function by creating and sending transparent acknowledge packets.

Acknowledgment

This work was supported by grants from the Project of the Bio-oriented Technology Research Advancement Institution, NARO (the research project for the future agricultural production utilizing artificial intelligence), and MEXT/JSPS KAKENHI Grant (B) Number JP16H04455 and JP17H01739, respectively.

References

[1] Cisco visual networking index: Forecast and trends, 2017-2022, [Online]. Available: https://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/white-paper-c11-741490.html (visited on 02/09/2019).

[2] Openfog reference architecture for fog computing, [Online]. Available: https://www.openfogconsortium.org/wp-content/uploads/OpenFog_Reference_Architecture_2_09_17-FINAL.pdf (visited on 02/09/2019).

[3] J. Wijekoon, S. Ishida, E. Harahap, and H. Nishi, "Service-oriented router-based cdn system: An sor-based cdn infrastructure implementation on a real network environment," in *2013 IEEE 37th Annual Computer Software and Applications Conference Workshops*, Jul. 2013, pp. 742–747. DOI: 10.1109/COMPSACW.2013.130.

[4] Factory automation, [Online]. Available: https://www.itu.int/dms_pub/itu-r/md/15/wp5a/c/R15-WP5A-C-0469!N36!MSW-E.docx (visited on 02/09/2019).

[5] Google cloud platform, [Online]. Available: https://cloud.google.com/ (visited on 02/15/2019).

[6] Amazon web service, [Online]. Available: https://aws.amazon.com/ (visited on 02/15/2019).

[7] Microsoft azure, [Online]. Available: https://azure.microsoft.com/en-us/ (visited on 02/15/2019).

[8] Azure stream analytics, [Online]. Available: https://azure.microsoft.com/en-us/services/stream-analytics/ (visited on 02/15/2019).

[9] Oracle, [Online]. Available: https://www.oracle.com/index.html (visited on 02/20/2019).

[10] Kaa, [Online]. Available: https://www.kaaproject.org/ (visited on 02/20/2019).

[11] Aeris, [Online]. Available: https://www.aeris.com/ (visited on 02/20/2019).

[12] Openiot, [Online]. Available: http://www.openiot.eu (visited on 02/20/2019).

[13] Linksmart, [Online]. Available: https://linksmart.eu/#01-intro (visited on 02/20/2019).

[14] E. Saurez, K. Hong, D. Lillethun, U. Ramachandran, and B. Ottenwälder, "Incremental deployment and migration of geo-distributed situation awareness applications in the fog," Jun. 2016, pp. 258–269. DOI: 10.1145/2933267.2933317.

[15] B. Cheng, G. Solmaz, F. Cirillo, E. Kovacs, K. Terasawa, and A. Kitazawa, "Fogflow: Easy programming of iot services over cloud and edges for smart cities," *IEEE Internet of Things Journal*, vol. 5, no. 2, pp. 696–707, Apr. 2018, ISSN: 2327-4662. DOI: 10.1109/JIOT.2017.2747214.

[16] J. Wijekoon, E. Harahap, K. Takagiwa, R. Tennekoon, and H. Nishi, "Effectiveness of a service-oriented router in future content delivery networks," in *2015 Seventh International Conference on Ubiquitous and Future Networks*, Jul. 2015, pp. 444–449. DOI: 10.1109/ICUFN.2015.7182583.

[17] Data plane development kit, [Online]. Available: https://www.dpdk.org/ (visited on 02/09/2019).

[18] Hyperscan, [Online]. Available: https://01.org/hyperscan (visited on 02/15/2019).

[19] Delivering 160gbps dpi performance on the intel xeon processor e5-2600 series using hyperscan, [Online]. Available: https://www.intel.com/content/dam/www/public/us/en/documents/white-papers/160gbps-dpi-performance-using-intel-architecture-paper.pdf (visited on 02/15/2019).