Acquisition Delay Time Evaluation of Cloud-based Load Distribution Model in ICSN

Eishin NAGAOKA†, ***Student Member***, **Ryohei BANNO**††, ***Member***, ***and*** **Osamu MIZUNO**†, ***Senior Member***

SUMMARY IoT services that allow sensor devices to autonomously collect data and send it to the cloud via gateways are expected for various services. In many cases, the service providers need to build their own sensor network and many sensors will place in a narrow area. To solve this problem, we have proposed the Information-Centric Network based wireless Sensor Network (ICSN) to provide multiple IoT services. Also, we have proposed the ICSN Platform (ICSNP) that provides the functions of executing IoT services in sink nodes. However, the number of requests increases due to the increasing number of services simultaneously at the sink node. It causes performance degradation due to rising processing load. Therefore, we have proposed the cloud-based load distribution model for applications to an environment of multiple IoT services. In this paper, we evaluated the acquisition delay time of sensor data by implementing the proposed model with actual hardware resources. As a result, the proposed method can reduce the acquisition delay time of sensor data compared to the conventional method even in an environment with many concurrently connected applications and short sensor data acquisition intervals.

keywords: *Information-Centric Networking (ICN), Internet of Things (IoT), Wireless Sensor Network (WSN), Cloud, Load Distribution*

1. Introduction

The IoT is spreading worldwide. In 2023, it is expected that about 34.1 billion IoT devices will be connected to the Internet [1]. Many IoT services collect data from sensor devices automatically and send it to the cloud via gateways. A sensor network is often built by an IoT service. As a result, when new services are introduced, construction and operating costs increase. Therefore, instead of building new sensor network for introducing IoT services, we proposed a sensor network platform. It can provide multiple IoT services on the same sensor network. Fig. 1 shows the overview. As shown in Fig. 1, the camera receives requests from two applications to share the same sensor data for each application. In this way, if sensor data can share using only one sensor network for each service, the construction and management costs will be reduced.

Based on this idea, we have proposed Information-Centric networking based wireless Sensor Network (ICSN), which is an application of Information-Centric Networking (ICN) [2] to sensor networks [3]. Also, we have proposed ICSN Platform (ICSNP) as a platform to provide the processing of user requests and the execution environment of IoT services

†The authors are with Graduate School of Electrical and Electronics Engineering, Kogakuin University, Nishi-Shinjuku, 163-8677, Shinjuku, Tokyo, Japan.

††The author is with Graduate School of Informatics, Kogakuin University, Nakano-machi, 192-0015, Hachioji, Tokyo, Japan.

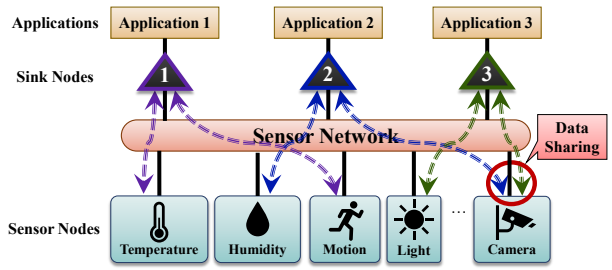

Fig. 1 A sensor network for realizing an environment with multiple IoT services

[4]. ICSNP enables an environment in which multiple IoT services exist in a single sensor network, enabling early deployment. However, the number of requests rises due to increasing number of services simultaneously at a sink node. As a result, congestion may occur at the sink node. To solve this problem, we proposed a cloud-based load balancing model in ICSN [5]. In the past, we have clarified the load characteristics of the proposed model in actual environments [6]. To realize the proposed model, it is necessary to evaluate the delay time during sensor data acquisition for clarifying the load characteristics of the link.

The structure of this paper is as follows. Section 2 introduces related work. Section 3 describes ICSN. The motivation for this research is also presented. Section 4 presents the proposed cloud-based load distribution model. Section 5 describes the experimental environment and results. Section 6 summarizes this paper and discusses future work.

2. Related Work

In this section, we introduce related research on ICN-IoT. Yokotani has proposed an architecture for a data exchange platform using ICN in the operation of IoT services [7]. A number of communication methods have been identified as candidates for the operation of IoT services, including HTTP, CoAP, and ICN. HTTP needs 3-Way Handshake in TCP and IP-based addressing. Therefore, the increase in the amount of traffic on the network results in a larger network load. CoAP, a lightweight protocol that is a simplified version of HTTP, does not require the 3-Way Handshake, and has lightweight headers. However, the CoAP is IP routing and depends on the physical location of the device. Therefore, communication paths depend on physical location and the device information. On the other hand, ICN has the advantage of being able to form a communication path

independent of the physical location and information of the device, since it uses the name of the data to communicate. Therefore, the author states that ICN is the most suitable communication for the operation of IoT services.

3. ICN-based Wireless Sensor Network (ICSN)

3.1 Overview and Operation

ICSN is a network that applies ICN to sensor networks. Sensor devices present in the sensor network are divided into several areas. Then, it selects one sensor device in each area as a Cluster Head (CH). Therefore, applying ICN to sensor networks can reduce the acquisition time of sensor data and the power consumption in sensor devices.

Also, we have proposed ICSNP as a platform to realize a processing and service execution environment for user requests in ICSN. The sink node has applications and databases. Users can request to acquire, register, and refer to sensor data from sensor devices through the application. Also, after setting the user's request information at the sink node, it converts the signal as an Interest for ICSN and sends it to the sensor device. Fig. 2 shows an example of ICSNP operation. As shown in Fig. 2, a user accesses the application on the sink node and sets up a request to acquire the temperature data in Area B at 4:00 p.m. on November 29. Then, the sink node converts the request to the data ID "*B/1129/1600/temperature*" and it sends Interest message including the data ID to the sensor device. After receiving the data containing the sensor data at the sink node, it registers the data in the database. Then, the results are notified to the user on the application.

3.2 Problems in the ICSNP

ICSNP enables processing of user requests and service execution environment. However, the number of requests increases as the number of concurrent IoT services increases, and the processing load on the sink node may rise. The sink node has an application that executes the service and a database that manages sensor data and sensor device information. Consequently, there is a risk of congestion as the rising number of services running at the same time. These problems may cause IoT services to stop due to the large number of requests, and delays in acquiring sensor data in each IoT service. Therefore, it is necessary to realize load-balancing function at the sink node to maintain availability as an IoT service. To address the issue, we have proposed a cloud-based load distribution model.

4. Cloud-based Load Distribution Model

In this chapter, we will explain the cloud-based load distribution model. Fig. 3 shows the cloud-based load distribution model. As shown in Fig. 3, we introduce a new

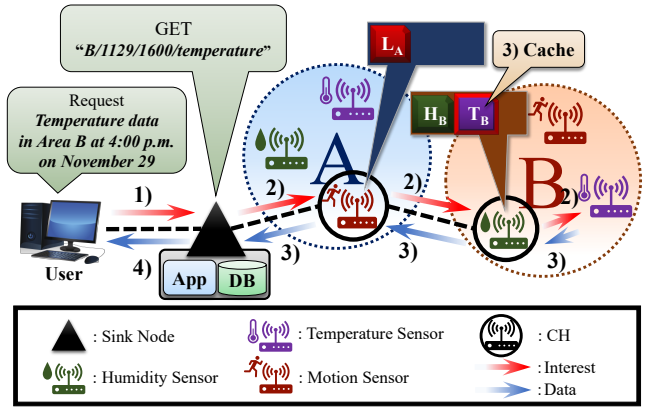

Fig. 2 Operation of sensor data acquisition in ICSNP

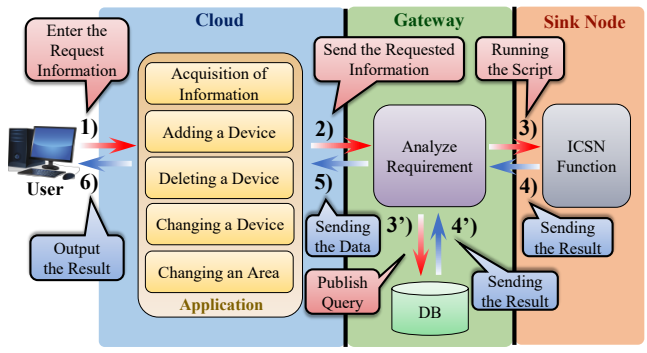

Fig. 3 Components in Cloud-based Load Distribution

gateway and cloud in ICSN. In their respective roles, the cloud is responsible for managing applications, the gateway is responsible for data management to manage sensor data and device information. In this model the sink node is responsible for data collection. The applications in the cloud were configured with five functions. In the gateway, we configured a database to manage sensor device information and sensor data, and a function to analyze requests. The request analysis function analyzes the user's request information received from the cloud. According to the result of the analysis, it distinguishes whether to process the acquisition of sensor data to the sink node or to the database. In the sink node, we configured ICSN function to send acquisition requests to sensor devices.

We explain the operation of the proposed model using Fig. 3. First, the user accesses the application in the cloud and enters the requested information. The inputted request information is sent to the gateway as Interest. When the gateway receives an Interest packet, it uses the request analysis function to distinguish whether the request is for a database or a sink node. In the case of "*for a database*", it publishes a query, and in the case of "*for a sink node*", it forwards the Interest packet as a request to acquire sensor data. After that, it receives the requested reference results and sensor data as Data from the database or sink node, and forwards it to the cloud, and outputs the results to the user on the application.

5. Performance Evaluation

5.1 Overview and Experimental Environment

In this chapter, we explain the evaluation of sensor data acquisition delay time when the proposed method is operated. The evaluation index is the Round Trip Time (RTT) of the sink node, gateway, and cloud. The RTT is defined as the time from when Interest is sent as a sensor data acquisition request or reference request until the result is reflected. The comparison target is the RTT for data acquisition in ICSNP, which is a conventional method.

Table 1 shows the experimental environment and Fig. 4 shows the experimental network topology. We performed this experiment on the 22nd and 23rd floors of Kogakuin University's Shinjuku campus. As a connection type, the sink node, gateway, and cloud are connected by a wired connection. Sensor nodes and sink nodes are connected by wireless connection as 2.4 GHz Wi-Fi ad hoc mode. Moreover, all the nodes were implemented using Cefore-0.8.1 [8], which is provided by NICT as an OSS that realizes the ICN node. Furthermore, the application set up in the cloud was implemented using XAMPP [9], and a database in the gateway was implemented using MariaDB [10]. The experimental arrangement was made according to Fig. 5.

Table 2 shows the setting parameters. The experiment was conducted with a measurement time of 300 [sec], three patterns of sensor data acquisition intervals of 60, 30, and 15 [sec/req], and four patterns of 1, 2, 4, and 8 applications, with 10 trials for each pattern.

5.2 Results and Discussions

Fig. 6, 7, 8, and 9 show the results for the number of applications N=1, N=2, N=4, and N=8, respectively. For each of the RTTs, note that the average RTT takes about 1.614 to 1.615 [sec] when only Cefore is used. The results show that the RTT of the sink node in the conventional method is about 1.722 to 1.749 [sec], on the other hand, the RTT of the cloud in the proposed method is about 1.618 to 1.629 [sec], and a gateway's RTT is about 1.614 to 1.617 [sec]. Therefore, The RTT of the cloud and gateway in the proposed method could be reduced by about 0. 108 to 0. 132 [sec] compared to the RTT of the sink node in the conventional method. Also, there was almost no change in RTT with an increase in the number of applications or a shortening of the interval between sensor data acquisitions for both methods.

These results show that the proposed method can reduce the RTT and link load more than the conventional method. In the conventional method, the sink node executes the application and manages the database. As a result, two functions work when receiving request information and sensor data, which may have caused the processing delay. On the other hand, the proposed method distributed the functions of application execution and database management the cloud and the gateway. Therefore, by distributing the function to each node, the processing delay is also distributed, and the RTT may be shortened. In the case of application N=8, the RTT of some applications is longer. As the number of applications running

Table 1 Experiment environment

Node	Cloud	Gateway	Sink Node	Sensor Node
Product	DELL Precision T1700	Fujitsu CELJ01009	HP ProBook 430 G1	Raspberry Pi Model 3B+
OS	Ubuntu 16.04	Ubuntu 16.04	Ubuntu 16.04	Raspbian 9
Middleware	Cefore, XAMPP	Cefore, MariaDB	Cefore	Cefore
Memory	32 [GB]	4 [GB]	4 [GB]	1 [GB]
Core	4	4	2	4
Processor	Intel Xeon CPU E3-1240 v3 3.40GHz	Intel Xeon CPU E3-1275 v3 3.50GHz	Intel Core i5-4200U CPU 1.60GHz	Broadcom BCM2837B0 Cortex-A53 (ARMv8) 1.4 GHz

Node	User	(Virtual Machine)
Product	Oracle VirtualBox 6.1 version: 6.1.26	HP Spectre x360 convertible 13-ac0XX
OS	Ubuntu 16.04	Windows 10 Home
Memory	1 [GB]	16 [GB]
Core	1	2
Processor	Virtual Machine	Intel Core™ i7-7500U 2.90GHz

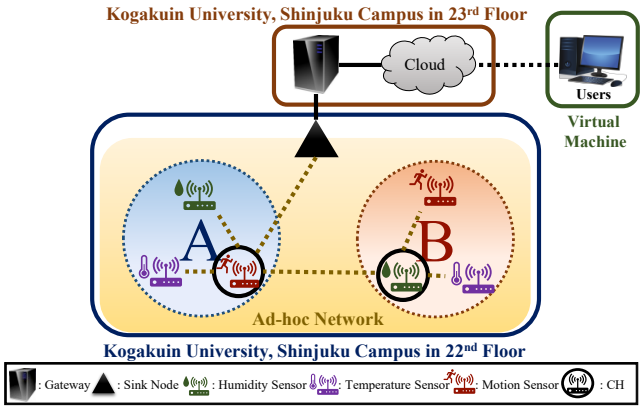

Fig. 4 Experimental topology

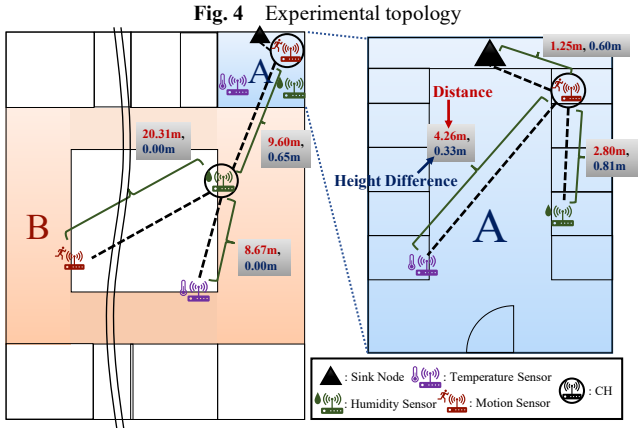

Fig. 5 Layout in the experimental environment

Table 2 Parameters

Parameters		Values
Number of Nodes	Sensor Node	6
	Sink Node	1
	Gateway	1
	Cloud	1
Memory Capacity	Sensor Node	1 [GB]
	Sink Node	4 [GB]
	Gateway	4 [GB]
	Cloud	32 [GB]
Number of Areas		2
Measurement Time		300 [sec]
Acquisition Interval		60, 30, 15 [sec/req]
Number of Applications		1,2,4,8
Number of Trials		10

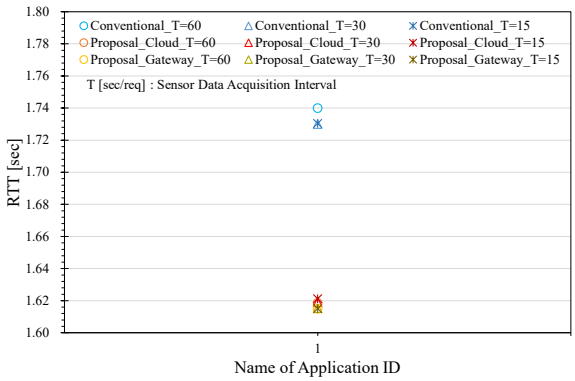

Fig. 6 Measurement result of RTT (N=1)

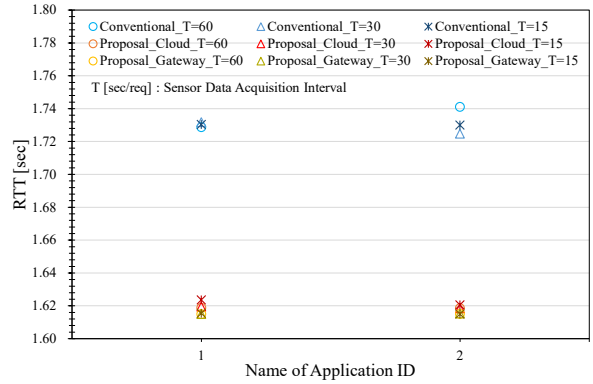

Fig. 7 Measurement result of RTT (N=2)

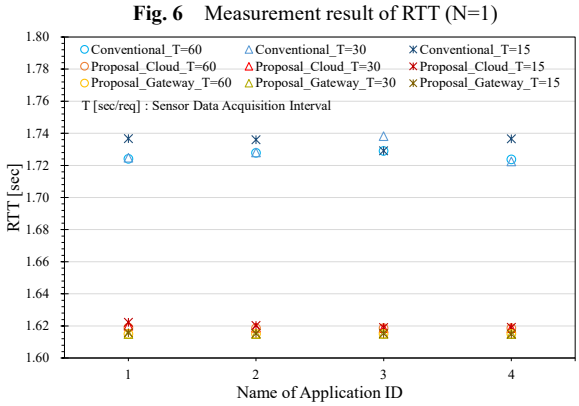

Fig. 8 Measurement result of RTT (N=4)

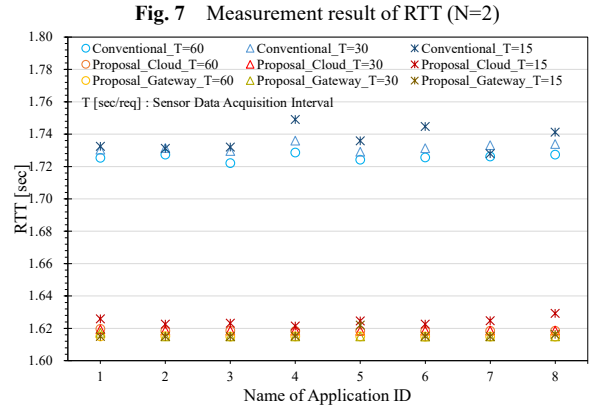

Fig. 9 Measurement result of RTT (N=8)

simultaneously increases, the number of requests and sensor data received also increases. Therefore, there may be a temporary wait process in the application execution process or database processing. To reduce the RTT further in the proposed method, it is necessary to apply a push-based communication method that allows multiple sensor data to be acquired autonomously with a single request. By applying this method, the sensor data will be obtained in a more timely manner, and the availability of the IoT service are further study.

6. Conclusion

In this paper, we have proposed a cloud-based load distribution model to address congestion at the sink node. To realize the proposed model, it is necessary to clarify the load characteristics of the actual operation. Therefore, we implemented and evaluated the proposed method using actual hardware resources. As a result, the proposed method reduces the RTT more than the conventional method, and the delay time for sensor data acquisition in the proposed method is clarified.

In the future, we will study and evaluate a push-based communication method that can acquire multiple sensor data routinely and autonomously with only one request.

Acknowledgments

Part of this research was funded by Japan Society for the Promotion of Science (JSPS) KAKENHI Grant Number JP22K12014.

References

[1] Ministry of Internal Affairsand Communications, Information and communications white paper 2021 (in Japanese), July 2021.

[2] B.Ahlgren, C.Dannewitz, C.Imbrenda, D.Kutscher, and B.Ohlman, “Asurvey of information-centric networking,” IEEE Commun. Mag., vol. 50, no. 7, pp. 26– 36, July 2012.

[3] T.Kaida, M.Koike and O.Mizuno, “The Data Cache Method on Sensor Networks Applied Information Centric Network”, IEICE Trans. Commun. (in Japanese), Vol.J100-B, No.2, pp48-58, Feb. 2017.

[4] K.Kimura and O.Mizuno, “Performance requirements evaluation of network functions in information-centric networking based wireless sensor network,” *IEEJ Trans. Information and Systems* (in Japanese), vol. 140, no. 6, pp. 583–584, June 2020.

[5] E.Nagaoka, M.Yoshii, R.Banno and O.Mizuno, “Evaluation for Cloud-based Load Distribution Model in ICSN”, IEICE Communications Express, Vol.X11-B, No.8, pp.509-514, Aug. 2022. DOI : https://doi.org/10.1587/comex.2022TCL0017

[6] E.Nagaoka, R.Banno and O.Mizuno, “Implementation and Node Load Evaluation of Cloud-Based Load Distribution Model in ICSN”, 2022 23rd Asia-Pacific Network Operations and Management Symposium (APNOMS), Sep. 2022, (to appear)

[7] T. Yokotani, “IoT Use Cases Analysis and Possibility of Adopting ICN Technologies for These loT Use Cases,” 2018 IEEE World Symposium on Communication Engineering (WSCE), Dec. 2018.

[8] Cefore, “https://cefore.net/”, (accessed July 4. 2022)

[9] Xampp, “https://www.apachefriends.org/jp/index.html”, (accessed July 4. 2022)

[10] MariaDB, “https://mariadb.org/”, (accessed July 4. 2022)