IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received February 21, 2019, accepted March 31, 2019, date of publication April 16, 2019, date of current version May 1, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2910848

Device Identification Based on Communication Analysis for the Internet of Things

HIROFUMI NOGUCHI, MISAO KATAOKA, AND YOJI YAMATO, (Senior Member, IEEE)
NTT Network Service Systems Laboratories, NTT Corporation, Tokyo 180-8585, Japan
Corresponding author: Hirofumi Noguchi (hirofumi.noguchi.rs@hco.ntt.co.jp)

ABSTRACT As the Internet of Things (IoT) is rapidly expanding, a huge variety of devices is being connected to the Internet. Device management is becoming an important topic for IoT. Especially for using devices properly and securely, it is necessary to visualize what types of devices are in the network. However, most conventional device identification methods are not suitable for resource-constrained IoT devices. Therefore, we have developed a method of device identification that identifies the type and model of devices on the basis of general communication information. It determines the type and model of devices by calculating the similarity of features extracted from their network packets. The great merit of our proposed device identifier is that it can be applied to various IoT devices without special equipment. We conducted three experiments to evaluate its effectiveness. In the experiments, we focused on devices specialized for specific functions such as network cameras and factory-used devices because they are effective targets of our device identifier. In addition, we tried identifying models from devices of the same type. The first experiment revealed the relationship between the packet header information used for identification and the success of identification. The second experiment with 11 types of network cameras showed that the device identifier correctly identified nine of them. In addition, the third experiment in a simulated factory environment showed the device identifier correctly identified six types of factory-used devices. Thus, we have demonstrated the feasibility of the proposed device identifier in a real environment.

INDEX TERMS Internet of Things, device identification, network management.

I. INTRODUCTION

The Internet of Things (IoT) is rapidly expanding as a huge variety of devices is being connected to the Internet. McKinsey estimates that the potential economic impact of IoT applications will be US$11.1 trillion per year in 2025 [1]. More and more devices will be installed in various environments such as homes, factories, and streets [2]–[4]. Since many different types and huge numbers of devices will be connected to the network, network administrators will not be able to grasp what kinds of devices are connected to the network. For detecting illegally connected devices or managing assets of devices, it is necessary to visualize what types of devices are currently connected to the network. However, IoT device management presents several problems.

First, IoT devices are diverse. For example, IoT devices include sensors such as cameras and thermometers, small computers such as smartphones, and actuators such as speakers and displays. They have different computing resources and use different communication protocols. Therefore, it is difficult to determine the properties and states of various devices in a common way.

Second, many mobile devices will be connected to IoT, and their installation location and network connection state will change frequently. For example, in a factory, when a production line is refurbished, a device used on the line may be reused on a separate line. In a home, users move devices such as a laptop computer, headphones, or a speaker in accordance with their circumstances. In addition, not only the physical location but also the network location changes dynamically. Some mobile devices have multiple access interfaces such as Wi-Fi and 4G LTE and change access networks frequently in accordance with their location. Thus, devices in the network dynamically change. Therefore, conventional Internet protocol (IP)-based management is not suitable for IoT network management. Also the media access control (MAC) address cannot be used to identify devices because some operating systems now generate MAC addresses randomly for each network connection for security.

The associate editor coordinating the review of this manuscript and approving it for publication was Zhong Fan.

Third, the number of IoT devices will become huge. Even small offices will have more than 100 IoT devices at all times. Therefore, it is not realistic to manually manage a database of all devices and reflect frequent changes in a huge number of devices. Management must be automated.

To solve the above problems, we previously proposed a method to identify devices in the network [5]. In this paper, we will describe its implementation and evaluation in a device identification system. This device identifier automatically determines the type and model of the device by passive communication analysis. In particular, we focus on IoT devices that have one specific function, because conventional identification methods cannot be applied to most of these simple devices. We will describe the experiments with many models of devices of the same type and also describe the experiments with devices in a simulated factory environment.

The rest of this paper is organized as follows. Section II surveys the conventional device identification. Section III describes our approach for device identification and related research. Section IV presents the developed device identifier. Section V shows the results of experiments. Section VI describes the discussion and future tasks. Finally, Section VII concludes the paper.

II. CONVENTIONAL DEVICE IDENTIFICATION

There have been some prior research and technologies for identifying devices.

Universal Plug and Play (UPnP) [6] is a protocol for connecting devices to a network. It handles descriptions describing the product and manufacturer name of the device. Although it is effectively grasps the types of devices, UPnP is not applicable to all IoT devices. Also, details of descriptions such as a device name are manufacturer-specific. There is a method for managing resource-constrained devices using management protocols such as SNMP and NETCONF [7]. It utilizes IP-based protocols on devices with limited central processing units (CPUs) and memory. However, an operating system (OS) must be able to handle these protocols. There is technology to identify the user of a device and its hardware configuration from web fingerprints [8], [9]. It finds the characteristics of users and devices and identifies them with high accuracy by investigating the behaviors of the browser. However, it can only be applied to devices that can operate browsers. These are only applicable to a few types of IoT devices because many IoT devices are resource-constrained devices such as primitive sensors. Since we are aiming at managing a wide variety of devices in various environments for IoT, we have to take another way.

There are methods for identifying devices from hardware fingerprints such as radiometric fingerprints [10]. These methods are effective for identifying a Network Interface Card (NIC), but they need special equipment such as a radio sensor. Thus, they are not suitable for managing a large number of devices at low cost.

There is research to identify the OS and applications by analyzing network traffic. Matsunaka *et al.* [11] and Chang *et al.* [12] tried to identify an OS from a unique domain name and transmission cycle of domain name system (DNS) queries. They are general methods and are not dependent on the type of the device. However, these methods are intended only for identifying the OS, not the device type or model.

Thus, there is no device identification method suitable for various kinds of devices in various environments that does not require special measurement equipment. We propose a device identification method that meets these requirements.

III. APPROACH AND RELATED RESEARCH

Our device identification method uses network information as common information that does not depend on the type of devices and can be acquired without special measurement equipment. Naturally, all IoT devices are connected to the network and communicate differently depending on the type and usage of devices. For example, a camera continues to continuously transmit large video data at all times, whereas primitive miniature sensors such as temperature sensors periodically send sensor values with small data size. Furthermore, even with the same camera, transmitted data size and the information differ depending on models. In other words, we believe that by looking at the communication information of the device, the type of device and individual device can be specified.

In addition, since communication information can be obtained via a general network switch, it is unnecessary to set a special device into the existing environment or install software on the device. Therefore, our method is low cost and can be commonly used for various kinds of devices.

Our proposed method identifies the type of device and individual device by comparing the communication of the identification target device with the communication stored in the database. The database stores communication information of devices existing in the network from past to the present. By looking at the similarity of communications among these devices, we can determine which devices match. This method also features automatic database management. The communication data of the devices are always stored while the device is operating. As a result, communication information specific to each model and type is automatically accumulated, and identification performance improves over time.

There is related research taking a similar approach to specify the device type from the communication information of the device. Meidan *et al.* [13] proposed a method for identifying the device type with features extracted from transmission control protocol (TCP) packets. In their experiment, their method classified data from nine types of devices including a printer, television, and smart watch. However, it does not identify each model. Kawai *et al.* [14] proposed a method for identifying the device type and model with only two factors: packet size and packet inter-arrival time (IAT). Since it does not depend on a specific protocol, it is very effective for identifying various IoT devices. However, only experiments with a few identical models are mentioned, and further experiments with many devices of the same type are worth doing.

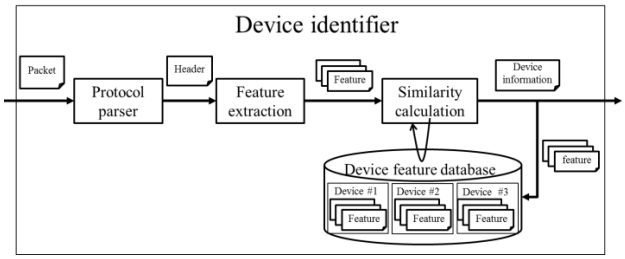

FIGURE 1. Components of device identifier.

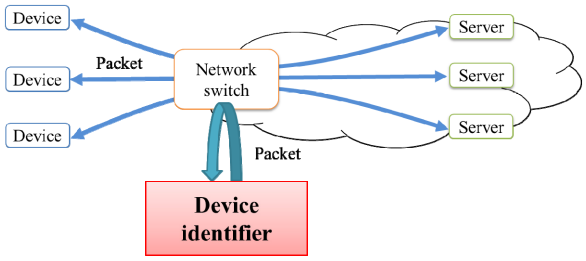

FIGURE 2. Network packet capture by device identifier.

Received time	Header #1	Header #2	Header #3	···	Header #n
09:00:02	40	80	128	···	2500
09:00:10	40	80	128	···	2500
09:00:32	1500	80	128	···	2500
09:00:56	40	80	128	···	2500
09:00:58	700	63000	64	···	8200

Feature	Header #1	Header #2	Header #3	···	Header #n
Maximum value	1500	63000	128	···	8200
Minimum value	40	80	64	···	2500
Average value	464	12664	115.2	···	3640
(Slope, Intercept)	(6.8, 248.1)	(630.3, -7254.9)	(-0.6, 135.5)	···	(57.1, 1835.5)

FIGURE 3. Procedure for extracting communication features.

Dalai and Jena [15] proposed a method for specifying a device type for identifying and isolating vulnerable device types for wireless devices. Their method calculates similarity of devices from communication information. However, the method deals with the probe request frames sent by the devices during network scanning for availability. It is only applicable to wireless devices, and the device identifier must be in the same WiFi area to capture data. Markus *et al.* [16] also adopted a similar approach to isolate vulnerable devices. They proposed a total system including a security gateway. Twenty-three kinds of features from 16 protocols are chosen. However, features for identification need to be refined, and the method needs to be made more independent of specific protocols.

Different from the above related research, we aim to identify models from many devices of the same type. Furthermore, we focus on devices specialized for specific functions such as factory-used devices and network cameras. Many such single-function devices will be in IoT, and they are effective targets of our proposed method. In the experiment section, we will analyze them in detail. Since the execution application software of these devices does not change like that of a smartphone, communication characteristics change little in accordance with the usage situation. Therefore, the experimental results of this paper are highly reproducible in different environments.

IV. IMPLEMENTATION

This section describes our system, a device identifier, which embodies the device identification method. Fig. 1 shows the overall image of the device identifier.

A. DEVICE IDENTIFICATION PROCEDURE

We designed a system in which the processing is divided into several steps in order for the device identifier to be extensible to various kinds of devices. Details will be described below in accordance with the process order.

1) ACQUISITION OF COMMUNICATION INFORMATION

The first step for determining the similarity of communication information is to extract header information from the communication packet. We consider the header information of the communication packet to be the minimum unit that expresses the nature of communication. The header information has attributes such as packet length and a TCP port number, which express characteristics of devices. The system stores this header information separately for each device in the network. This process does not affect the operation or service of the devices since the system acquires packets from the mirror port of the network switch. Fig. 2 shows how to apply the device identifier to the network. Communication from the same device is distinguished by a temporary unique header, which is arbitrarily chosen from headers that change less frequently.

Since this system is based on the assumption that here is that a device has no persistent unique header, the correspondence between the value of this temporary unique header and each device is cleared in a fixed time. That is, even if the MAC or IP address is temporarily handled as a unique header and associated with the device, it is reset after a certain period of time and the communication including the same MAC or IP is re-identified. In this way, even if the correspondence between the device and MAC or IP changes in the network, it is possible to identify whether a device is the same device or a different device.

2) EXTRACTION OF COMMUNICATION FEATURES

The device identifier obtains the header information accumulated for each device at a constant cycle and extracts communication features. Fig. 3 illustrates the process of extraction. Communication features include fluctuation trends in packet length within a certain period, occurrence of burst, periodicity of change of used port number, and the like. By adding an element of time, single header information becomes more effective for device identification. The device identifier uses the communication features generated in these certain periods as a minimum unit for determining the similarity

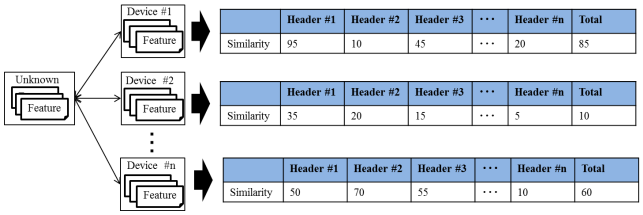

FIGURE 4. Procedure for calculating similarity of communication feature.

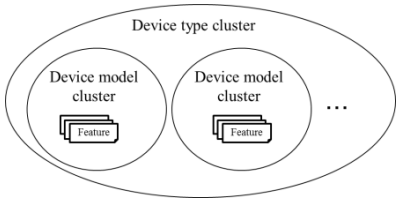

FIGURE 5. Hierarchy of feature database.

of communication. We digitized all the header information for extracting the features and calculating their similarity. We use several features: the maximum value, the minimum value, the average value within each certain period, and slope and intercept of the primary approximate curve within each certain period. The maximum, minimum, and average values express the existence or absence of the burst of the communication packet transmitted and received.

Also, the slope and intercept of the primary approximation curve express the fluctuation trend within a certain period.

3) CALCULATION OF SIMILARITY BY COMMUNICATION FEATURES

The device identifier compares the communication features of the identification target device and other devices that were connected to the network and determines whether they match or not. If a device has connected to the network at least once, its communication features are stored in the database. Fig. 4 illustrates the similarity calculation process. The device identifier calculates the similarity of the features of the target devices and features in the database. Then, if a type or model has similarity higher than a certain threshold, the device identifier determines that it is the same as the target device. If no type or model has similarity higher than the certain threshold, the device identifier determines that the target device is a new device in the network. The similarity is calculated by using the Euclidean distance of each feature and is normalized to a value range of 0 to 100. Equations 1 and 2 show the formulas of similarity calculation.

$$c_i = 1 - (\Delta x_i / \max(\Delta x)) . \tag{1}$$

$$C = \sum k_i c_i. \tag{2}$$

c is the degree of similarity for each feature, C is the comprehensive degree of similarity, and x is a digitized feature. c and C take a value in the range of 0 to 100. k is the weight of each feature.

We will describe the details of feature weight. Unique features in the network should be reflected more strongly for identification. For example, in an environment containing many HTTP client terminals, the destination port number of the transmit packet overlaps with many devices. If the destination port number is used to calculate similarity, all HTTP client terminals have uniformly high similarity, and a new device cannot be correctly identified. We solved this problem by setting an appropriate weight to each feature.

Additionally, although we are aiming to identify various kinds of devices in various IoT environments, it is difficult to set the weight manually in every environment. Therefore, the device identifier determines the uniqueness from the features in the environment and dynamically sets the appropriate weight. Equation 3 shows the formula for calculating the weight.

$$k_i = v_i / \sum v \tag{3}$$

v is a variance value of similarity for all features. k is calculated in accordance with the ratio of variance value. When comparing features, the device identifier also calculates the variance of all digitized features. Then the ratio of the sum of variance of all devices is taken as the weight. In other words, the sum of the weight of all the headers is 1, and the higher the variance, the greater weight given to the header with higher uniqueness.

B. ACCUMULATION METHOD OF COMMUNICATION FEATURES

The device identifier manages communication feature data of devices hierarchically to identify information of multiple devices. The hierarchy is shown in Fig. 5. When feature data are stored in the database, corresponding attributes are set. Attributes indicate to which clusters the data belong. In the device similarity calculation, feature data corresponding to identification target information are retrieved from the database using these attributes. Since clusters are not prepared for information of each identification target, the amount of data can be minimized.

1) DEVICE MODEL CLUSTER

A device model means information that specifies a device such as the device manufacturer or model number. By calculating similarity to communication features in this cluster, the device identifier identifies the device model of the target device. At least one sample feature needs to be prepared in the device model cluster. Alternatively, some devices can specify a device model from a combination of information such as user agent information and an MAC address contained in the packet without extracting features.

2) DEVICE TYPE CLUSTER

A device type means the classification of the devices in accordance with their functions, such as a camera, speaker, and smartphone. By calculating similarity to communication features in this cluster, the device identifier identifies the device type of the target device. At least one sample feature

TABLE 1. Specifications of network cameras.

Model	Resolution		
AXIS M1034-W EUR	320×180 (QVGA)	640×360 (VGA)	1280×720 (HD)
Canon VB-M720F	320×180 (QVGA)	640×360 (VGA)	1280×720 (HD)
VSTARCAM C7823WIP	320×180 (QVGA)	640×360 (VGA)	-
I-O DATA TS-WRLP	352×200	720×400	1280×720 (HD)

needs to be prepared in the device type cluster, but as identification is performed, features in the cluster are added.

V. EXPERIMENTS

To evaluate the effectiveness of the proposed method, we conducted three kinds of experiments using the device identifier. The first used several types of network cameras to check which header information is useful for identification. The second used more network cameras to evaluate identification accuracy. The third used a simulated factory environment to assess the device identifier's feasibility in a real environment.

A. CAMERA IDENTIFICATION FOR HEADER INFORMATION ANALYSIS

We conducted experiments using four types of network cameras made by different manufacturers as shown in Table 1. All are connected by an Ethernet, and their resolutions can be set by the user. The video compression format is motion JPEG, and the transmit packet length of the network camera fluctuates in accordance with the recorded video. We turned these cameras on and accumulated communication packets and identified devices in parallel. The time period of the communication feature extraction and device identification was 30 seconds in all experiments. As described in the previous section, the device identifier calculates the similarity for each device in the network on the basis of the features extracted from the communication for a certain period. In the experiment, if the similarity of the correct device was calculated as the highest value, the identification was successful. Regarding the accumulation of device feature quantities, in the actual operation of this system, the storage destination cluster of the feature is determined on the basis of the identification result each time. In this experiment, however, the feature is always stored in the correct cluster irrespective of the identification result.

1) ANALYSIS OF HEADER INFORMATION FOR IDENTIFICATION

We selected the header information used for identification from the viewpoint of uniqueness and reproducibility. In these experiments, we used packet length and time to live (TTL) of the network layer, the TCP window size of the transport layer, and the HTTP header of the application layer. The HTTP header fields shown in Table 2 were used as the

TABLE 2. Experiment configuration headers.

Number	Header	Status
1	Date	Use
2	Pragma	Use
3	Cache-Control	Use
4	Connection	Use
5	Transfer-Encoding	-
6	Upgrade	-
7	Via	-
8	Trailer	-
9	Warning	-
10	Authorization	Use
11	From	-
12	If-Modified-Since	-
13	Refer	Use
14	User-Agent	Use
15	Accept	Use
16	Accept-Charset	-
17	Accept-Encoding	Use
18	Accept-Language	Use
19	Host	Use
20	If-Match	-
21	If-None-Match	-
22	If-Range	-
23	If-Unmodified-Since	-
24	Max-Forwards	-
25	Proxy-Authorization	-
26	Range	-
27	Except	-
28	TE	-
29	Location	-
30	Server	Use
31	WWW-Authenticate	-
32	Accept-Ranges	Use
33	Age	-
34	Proxy-Authenticate	-
35	Public	-
36	Retry-Ager	-
37	Vary	-
38	Warning	-
39	Alternates	-
40	Etag	-

TABLE 3. Experiment configuration.

Experiment	Header			
	IP length	IP TTL	TCP window	HTTP
1	Use	Use	-	-
2	Use	Use	Use	-
3	Use	Use	Use	Use

packets of the network camera. To analyze the header used for device identification for each network layer, we measured the success or failure of identification for the three experimental conditions shown in Table 3. One condition is using only the IP header, another is using the TCP header and a layer header lower than it, and the other is using the HTTP header and a layer header lower than it. In this experiment, the resolution of the network cameras was unified to the minimum size, and the cameras shot the same object. A packet to be transmitted from the device to the viewing client was identified as a

TABLE 4. Success or failure of identification s: success, F: failure, m: success but marked.

Exper-iment	AXIS		Canon		VSTARCAM		I-O DATA	
	TX	RX	TX	RX	TX	RX	TX	RX
1	F	S	S	S	S	F	S	S
2	S	S	S	S	M	F	S	S
3	S	S	M	S	S	F	S	S

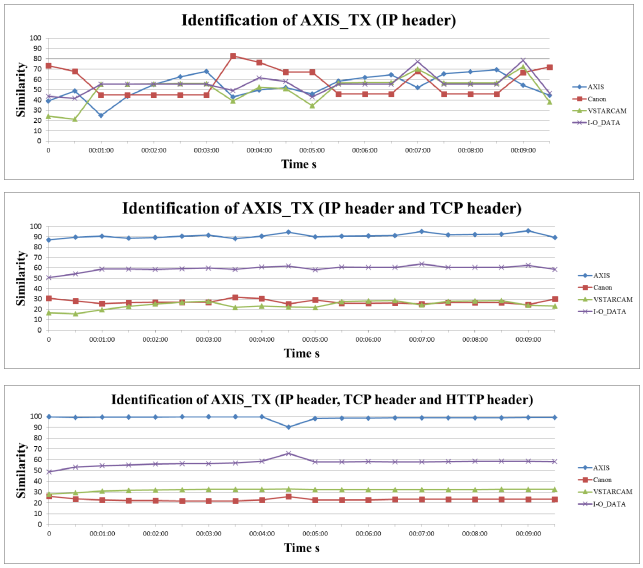

FIGURE 6. Similarity of transmitted packets of AXIS camera for each experiment.

transmitted packet (TX), and a signal transmitted from the viewing client to the device was identified as a received packet (RX). Table 4 shows the results.

We gathered the experimental results for identifying devices on the basis of packets they transmitted and received. In the case of using only the IP header, the identification success rate was 75%. However, in the case of using the TCP header and a layer header lower than it or using the HTTP header and a layer header lower than it, the identification success rate was 89%. This shows that the identification success rate was increased by increasing the amount of used features. Fig. 6 shows the similarity of the transmitted packets of the AXIS camera for each experiment. The vertical axis represents similarity, the horizontal axis represents elapsed time, and the plot shows similarity for each feature extracted in units of 30 seconds. With the IP header alone, similarity does not stabilize to the highest value and the device cannot be identified, but by using a header of TCP or higher, the similarity shows the highest value stably.

In the case of received packets of VSTARCAM, the device could not be identified correctly under any experimental conditions. Analyzing the captured packet, we found that AXIS and VSTARCAM periodically receive similar HTTP response packets, but only VSTARCAM received the icslap packets, which are a protocol originally developed by Microsoft, at a time interval longer than the feature extraction

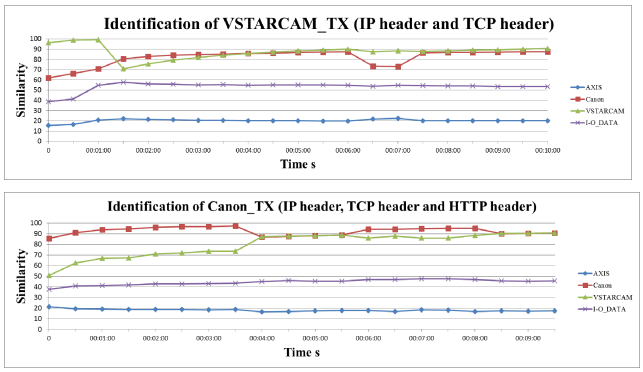

FIGURE 7. Results when similarity of correct device is not the highest in a small part.

time period. Therefore, this foreign feature was mixed in the database of VSTARCAM, so the similarity to the usual HTTP response packet became low and the similarity to accumulated features of AXIS became higher than for VSTARCAM. This caused erroneous identification.

In addition, some results of experiments showed that the similarity of the correct device is not the highest at a certain time when similarity is calculated in 30-second intervals. Fig. 7 shows the transition of the similarity of the corresponding experimental results.

This phenomenon did not occur in the case of using the VSTARCAM transmit packet and only the IP header or using Canon transit packets, a TCP header, and layer header lower than it.

Therefore, the results show that using low uniqueness and a low reproducibility parameter for identification negatively affected the calculation of similarity. In this way, simply increasing the number of features used for identification does not improve the accuracy of identification, and features need to be appropriately selected in terms of uniqueness and reproducibility.

2) EFFECTIVENESS OF DYNAMIC WEIGHTING FOR HEADER

We evaluated the effectiveness of dynamic weighting by measuring the similarity calculated with and without weight. In this experiment, the packet length of the IP header, TTL, window size of the TCP header, and HTTP header were used for calculating similarity. Fig. 8 shows the results. Weighting made the difference in similarity of each device clearer. The results demonstrate that dynamic weighting is effective to clarify the distinction between devices and determine the correct device even if there are many similar devices.

3) TRANSMITTED PACKET LENGTH RANGE

The packet length of network cameras varies in accordance with the color of the shot image even if the cameras are the same model and have the same settings. Therefore, if the captured image is changed by changing the usage environment, the feature of the packet length may change and then the camera cannot be correctly identified. We measured the value range of the transmitted packet length by variously changing

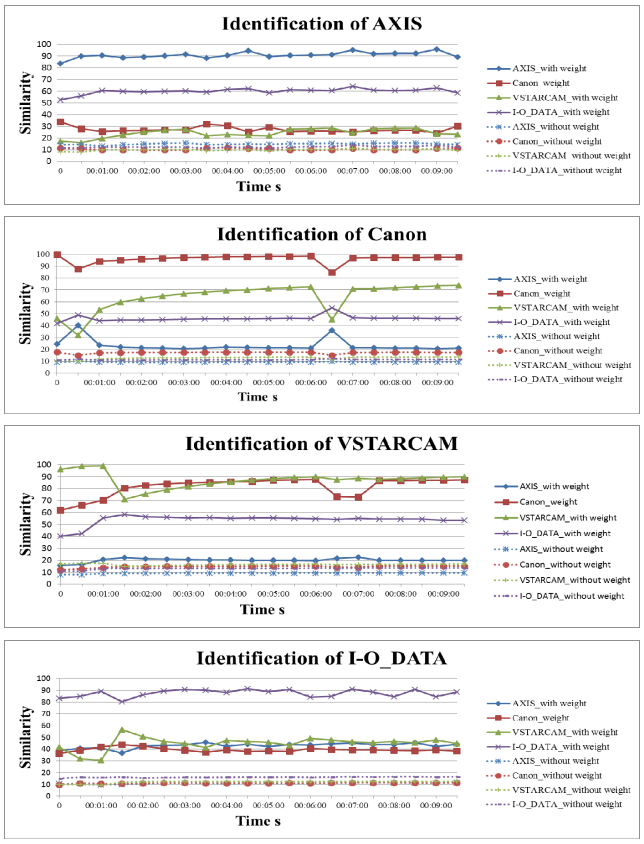

FIGURE 8. Similarity of devices with weight.

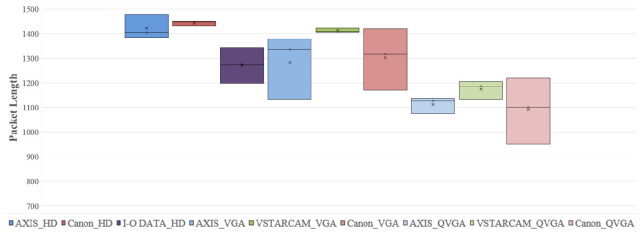

FIGURE 9. Value range of packet length in various settings.

the color of the photographed image with respect to the nine pattern conditions that consist of a combination of a model and resolution.

Fig. 9 shows the results. Comparing the value range of the packet length of the same model with different resolutions, duplication of range is seen in Canon's VGA and QVGA, but other combinations have almost no overlap. Therefore, even if a packet length changes in accordance with the shot image, the same model with different resolutions can be identified by using the packet length. Also, when comparing the value ranges of all nine patterns, even the combination with the most overlapping ranges is four patterns. Even if there is a change in the packet length in accordance with the shot image, the device can be narrowed down from the packet length by at least half. Thus, the packet length of the network camera is effective for identifying the device even if it is affected by the shot image.

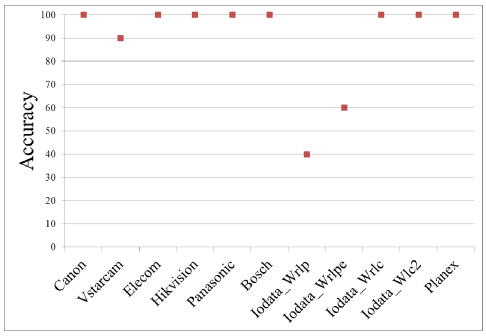

FIGURE 10. Ratio of correct identification for 11 types.

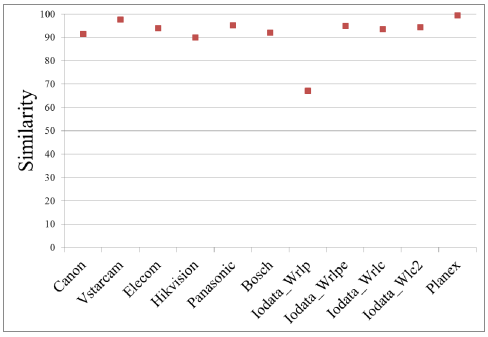

FIGURE 11. Average of similarity for 11 types.

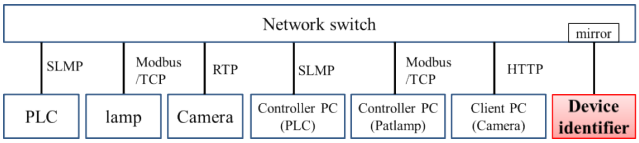

FIGURE 12. Devices and network configuration of experimental environment.

B. IDENTIFICATION ACCURACY FOR NETWORK CAMERAS

We measured the identification accuracy of the proposed system using 11 models of network cameras. As experimental conditions, only transmitted packets were used. The time period of the communication feature extraction and device identification was 30 seconds. Packet length, TTL of the IP header, and the window size of the TCP header are used for identification. The settings of devices were left in default. We prepared 20 training feature data and test feature data per model, and the device identifier identified test data one by one. Fig. 10 shows the percentage of test data classified as the correct device. We treated this value as identification accuracy. Fig. 11 shows the average of the similarity with the correct device only for the case of successful identification. Fig. 10 shows that 9 out of 11 devices were correctly identified with accuracy equal to and over 90%. If multiple test feature data are used to determine the same device, all these nine devices can be identified. On the other hand, the accuracy was poor for IO DATA_Wrlp and IO DATA_Wrlpe. According to the experimental data, they were sometimes mutually misidentified. Since they are of the same series from the same manufacturer, their specifications are presumed to be similar. When the device identifier is operated in a real environment, it will not be able to identify each model of the

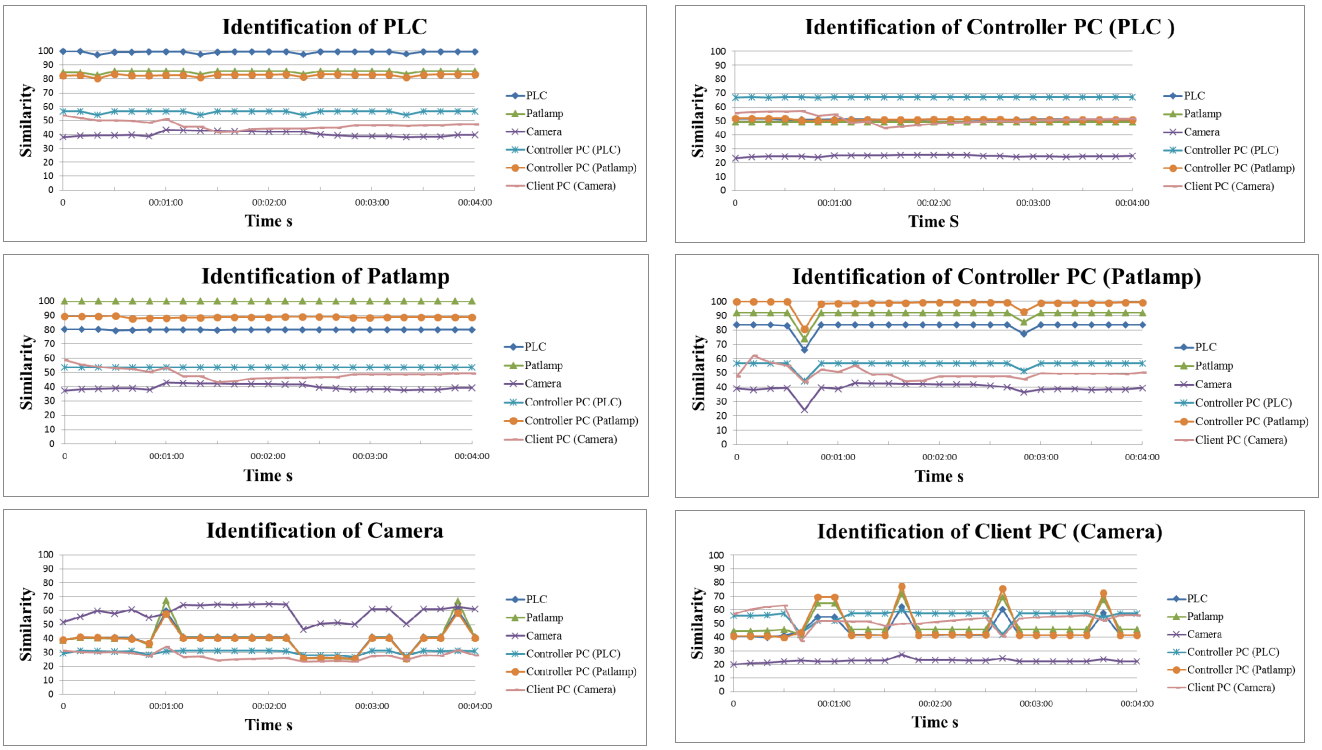

FIGURE 13. Similarity of devices based on extracted features per 10 seconds.

same series, but it may possibly be able to identify the series to which the model belongs.

C. IDENTIFICATION IN SIMULATED FACTORY ENVIRONMENT

To assess the feasibility of the proposed method in a real environment, we conducted an experiment in a simulated factory environment. Fig. 12 shows the network configuration of an experimental environment simulating a production line of a factory. The network contains a programmable logic controller (PLC), lamp, network camera (AXIS M1034-W EUR), a personal computer (PC) for controlling the PLC, a PC for controlling the lamp, and camera viewing client PC. These devices were operated in the normal operation state on the production line, and the device identifier accumulated communication packets and identified devices in parallel. In this experiment, we treated each PC as a different device because it has only one specific function in its factory use.

The device identifier acquired transmitted packets of each device from the mirror port of the network switch. In this experiment, we used the packet length, TTL, and the TCP window size for identification. The time period of the communication feature extraction was set to 10 seconds. Fig. 13 shows the results. The vertical axis represents the similarity, the horizontal axis represents the elapsed time, and the plot shows the similarity for each feature extracted in units of 10 seconds. The title of each graph shows the identification target device. The results show that the devices other than the camera client PC have the highest similarity as the correct device. However, as several noticeable phenomena were discovered, we analyzed them.

The similarity decreased at some points in the identification of the PC for controlling the lamp. By checking the communication packets at the corresponding time, Internet Group Management Protocol (IGMP) packets were found to be included only during this period. To deal with such singular occurrences, it is effective to calculate similarity multiple times and to determine the correct device from their sum or average.

The camera client PC could not be correctly identified since its similarity to different devices was always high. This is due to the camera client PC sending two kinds of packets at the same low frequency: RTSP and HTTP. Therefore, only one is always included in the 10-second period and does not match with nearly half of the accumulated features. To deal with such low frequency packets, it is effective to sufficiently lengthen the period of feature extraction and make the feature amount homogeneous.

VI. DISUCUSSION

The results of experiments show that the proposed device identifier can distinguish the devices from the communication information and determine the same model of devices correctly. The devices and environment of this experiment are one of the use cases, and further improvement is necessary to apply the proposed device identifier to various devices in real environments.

One task is setting an appropriate feature extraction period. Cameras were constantly transmitting video data. However, some types of devices do not communicate frequently or they transmit or receive specific packets such as monitoring at low frequency. In the experiment, since the period of feature extraction is uniformly fixed, there was a difference in the distribution of the communication packets contained in the extraction period, so the identification accuracy was affected. To apply the device identifier to various devices other than cameras, the feature quantity needs to be extracted in an appropriate time period in accordance with each environment and device. In the future, we plan to consider a method of dynamically adjusting the feature extraction period in accordance with the communication characteristics during the identification.

Another important task is selecting appropriate information for identification. As the experimental results show, unique and reproducible header information needs to be selected and used for identification. To support various protocols and usage environments of IoT devices, useful headers need to be automatically selected. The proposed device identifier weights the headers dynamically, and the effect was evaluated in the experiment. We think that the frequency of transmission and reception for each packet type will also be effective for identification and plan to incorporate it in the future. For example, even if some packets such as monitoring packets are similar on several devices, if the transmission frequency differs from device to device, it can be used for identification. To deal with packet transmission and reception frequency, we plan to apply machine learning such as a naive Bayes classifier or random forest [17]. We will apply these algorithms to identification of types and models of various devices.

Furthermore, for practical use, a device feature database also needs to be designed. In the experimental software, the data size was about 2.2 kB per feature data extracted in 30 seconds. If feature data are accumulated for one hour, the data size is about 264 kB per device, and even if 10,000 devices are managed, the storage size is only 2.6 TB. In a local environment where limited kinds of devices are connected, a smaller amount of storage is sufficient.

VII. CONCLUSION

Device management is going to be important for safe and proper handling of an enormous number of IoT devices in the future. In this paper, we proposed a device identifier that identifies types and models of devices by analyzing communication information. We also showed the device identifier's effectiveness in experiments. The experiment with four types of network cameras revealed the relationship between the packet header information used for identification and the success of identification. The experiment with 11 types of network cameras showed the device identifier correctly identified 9 of them. In addition, the experiment in a simulated factory environment showed the device identifier correctly identified six types of factory-used devices. We plan to conduct experiments with various types of devices in various network environments and improve the method of communication feature extraction in the future.

We also want to mention that the developed device identifier can be utilized not only for visualizing devices but also various services. Security is one of the candidate fields. The device identifier will be able to find newly connected devices and alert the relevant people. Also, it can detect strangely behaving devices such as malware infected devices. We plan to address such security applications in the future. Furthermore, we are considering the automatic construction of IoT services as a prospective application of the device identifier. We believe that devices in the living environment will be shared for various purposes [18]. Under these circumstances, services must find the necessary shared device from the network every time the device is used. Device types such as cameras and speakers are effective information for matching services and devices. We will expand the device identifier to determine the function, capability, and performance of the devices connected to the network and use this information to match devices and services.

REFERENCES

[1] J. Manyika *et al.*, "The Internet of Things: Mapping the value beyond the hype," McKinsey Global Inst. New York, NY, USA, Tech. Rep., Jun. 2015. [Online]. Available: https://www.mckinsey.com/~/media/McKinsey/Business%20Functions/McKinsey%20Digital/Our%20Insights/The%20Internet%20of%20Things%20The%20value%20of%20digitizing%20the%20physical%20world/Unlocking_the_potential_of_the_Internet_of_Things_Executive_summary.ashx

[2] G. Chong, L. Zhihao, and Y. Yifeng, "The research and implement of smart home system based on Internet of Things," in *Proc. Int. Conf. Electron., Commun. Control (ICECC)*, Sep. 2011, pp. 2944–2947.

[3] J. Lee, "Smart factory systems," *Informatik-Spektrum*, vol. 38, no. 3, pp. 230–235, Jun. 2015.

[4] A. Gaur, B. Scotney, G. Parr, and S. McClean, "Smart city architecture and its applications based on IoT," *Procedia Comput. Sci.*, vol. 52, pp. 1089–1094, Jan. 2015.

[5] H. Noguchi, T. Demizu, M. Kataoka, and Y. Yamato, "Autonomous device identification architecture for Internet of Things," in *Proc. IEEE 4th World Forum Internet Things (WF-IoT)*, Feb. 2018, pp. 407–411.

[6] *UPnP*. Accessed: Apr. 22, 2019. [Online]. Available: https://openconnectivity.org/developer/specifications/upnp-resources/upnp#architectural

[7] A. Sehgal, V. Perelman, S. Kuryla, and J. Schönwälder, "Management of resource constrained devices in the Internet of Things," *IEEE Commun. Mag.*, vol. 50, no. 12, pp. 144–149, Dec. 2012.

[8] G. Acar, C. Eubank, S. Englehardt, M. Juarez, A. Narayanan, and C. Diaz, "The Web never forgets: Persistent tracking mechanisms in the wild," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur. Secur.*, Nov. 2014, pp. 674–689.

[9] G. Acar *et al.*, "FPDetective: Dusting the web for fingerprinters," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, Nov. 2013, pp. 1129–1140.

[10] V. Brik, S. Banerjee, M. Gruteser, and S. Oh, "Wireless device identification with radiometric signatures," in *Proc. 14th ACM Int. Conf. Mobile Comput. Netw.*, Sep. 2008, pp. 116–127.

[11] T. Matsunaka, A. Yamada, and A. Kubota, "Passive OS fingerprinting by DNS traffic analysis," in *Proc. IEEE 27th Int. Conf. Adv. Inf. Netw. Appl. (AINA)*, Mar. 2013, pp. 243–250.

[12] D. Chang, Q. Zhang, and X. Li, "Study on OS fingerprinting and nat/tethering based on DNS log analysis," in *Proc. IRTF ISOC Workshop Res. Appl. Internet Meas. (RAIM)*, Yokohama, Japan, Oct. 2015. [Online]. Available: https://irtf.org/raim-2015

[13] Y. Meidan *et al.*, "ProfilIoT: A machine learning approach for IoT device identification based on network traffic analysis," in *Proc. ACM Symp. Appl. Comput.*, Apr. 2017, pp. 506–509.

[14] H. Kawai, S. Ata, N. Nakamura, and I. Oka, "Identification of communication devices from analysis of traffic patterns," in *Proc. 13th Int. Conf. Netw. Service Manage.*, Tokyo, Japan, Nov. 2017, pp. 1–5.
[15] A. K. Dalai and S. K. Jena, "WDTF: A technique for wireless device type fingerprinting," *Wireless Pers. Commun.*, vol. 97, no. 2, pp. 1911–1928, Nov. 2017.
[16] M. Markus, S. Marchal, I. Hafeez, N. Asokan, A.-R. Sadeghi, and S. Tarkoma, "IoT sentinel: Automated device-type identification for security enforcement in IoT," in *Proc. IEEE 37th Int. Conf. Distrib. Comput. Syst. (ICDCS)*, Jun. 2017, pp. 2177–2184.
[17] L. Breiman, "Random forests," *Mach. Learn.*, vol. 45, no. 1, pp. 5–32, Oct. 2001.
[18] H. Noguchi, T. Demizu, M. Kataoka, and Y. Yamato, "Distributed search architecture for object tracking in the Internet of Things, " *IEEE Access*, vol. 6, pp. 60152–60159, 2018.

HIROFUMI NOGUCHI received the B.S. and M.S. degrees in mechanical engineering from Waseda University, Japan, in 2010 and 2012, respectively. He joined NTT Corporation, Japan, in 2012, where he has been engaged in the developmental research of server virtualization and the IoT. He is currently a Researcher with NTT Network Service Systems Laboratories.

MISAO KATAOKA received the B.S. and M.S. degrees in informatics from the University of Kyoto, Japan, in 2012 and 2014, respectively. She joined NTT East in 2014, where she was engaged in simplifying networks. Since 2016, she has been engaging in the developmental research of distributed processing platforms and the IoT platforms at NTT Laboratories. She is currently a Research Engineer with NTT Network Service Systems Laboratories.

YOJI YAMATO received the B.S. and M.S. degrees in physics and the Ph.D. degree in general systems studies from The University of Tokyo, Japan, in 2000, 2002, and 2009, respectively. He joined NTT Corporation, Japan, in 2002, where he has been engaged in the developmental research of cloud computing platform, peer-to-peer computing, and the IoT. He is currently a Distinguished Researcher with NTT Network Service Systems Laboratories. He is a Senior Member of IEICE and a member of IPSJ.