IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received September 4, 2018, accepted October 1, 2018, date of publication October 15, 2018, date of current version November 8, 2018.

Digital Object Identifier 10.1109/ACCESS.2018.2875734

Distributed Search Architecture for Object Tracking in the Internet of Things

HIROFUMI NOGUCHI, TATSUYA DEMIZU, 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 Internet of Things (IoT) is rapidly expanding, which will enable many devices to be installed in various environments. However, current IoT services cannot maximally utilize devices because of their silo model. To solve this problem, we aim to realize Open IoT, in which services share devices. In this paper, we propose an architecture for an object tracking service, one of the main services of Open IoT. The architecture uses video data from shared devices, such as surveillance cameras or pedestrians' smartphones. An important research task is to discover the most appropriate devices for a service out of a huge number of devices connected to the Internet. We named real-time data generated by devices "live data" and are trying to use these data to discover appropriate devices. However, it is difficult to collect and handle all live data in the cloud because of the network band limit. Therefore, we propose a distributed search architecture. Generally, distributed architecture uses network and computing resources less efficiently than cloud architecture. Our proposed architecture overcomes this by deploying a search function dynamically and copes with arbitrary searches. We developed a system that embodies our proposed architecture and evaluated its feasibility. An experiment simulating a moving object tracking service with network cameras is shown that the architecture reduces the communication bandwidth of the core network to 1000th or less of that when cloud computing is used. In addition, another experiment is shown that the architecture search speed is sufficient for a walking-person tracking service.

INDEX TERMS Internet of Things, network architecture, distributed information systems, sensors, search.

I. INTRODUCTION

The Internet of Things (IoT) is rapidly expanding. Fifty billion devices will be connected to the Internet by 2020 [1], security cameras and temperature sensors will be installed in various environments such as homes and streets [2], [3], and portable devices such as smartphones and wearable devices will be seen everywhere. McKinsey estimates that the potential economic impact of IoT applications will be US$11.1 trillion per year in 2025 [4]. A large amount of data generated by IoT devices will be used for various applications, enabling innovative services to be created for a wide spectrum of domains [5].

However, current IoT services cannot maximally utilize those personal devices because of their silo model. In the silo model, service providers provide an integrated service by preparing a service application and dedicated devices. Therefore, many devices are necessary to provide services over a wide area. However, most service providers can only provide small-area services such as services in offices or towns.

To solve this problem, we are aiming to realize Open IoT, which will provide wide-coverage services at low cost by using various shared devices around us (Fig. 1). In Open IoT, we will be able to access various owners' devices connected to the Internet and use them for various services. For example, a store's security camera or a pedestrian's smartphone camera will be temporarily used to search for a lost child. In another example, public warnings will be displayed on digital signage during a disaster such as an earthquake. In Open IoT, service providers will not need to prepare dedicated devices. As long as there is the Internet, anywhere can be a service area.

To realize Open IoT, several requirements must be met. First are security and privacy, which are also important in the conventional silo model IoT. Atzori *et al.* [6] note the authentication and data integrity of the sensor node as concrete requirements of IoT security. Roman *et al.* [7] note identity and authentication, access control, protocol, and network security as specific challenges of IoT security. In addition, Qiu *et al.* [8] note that all types of communications may be

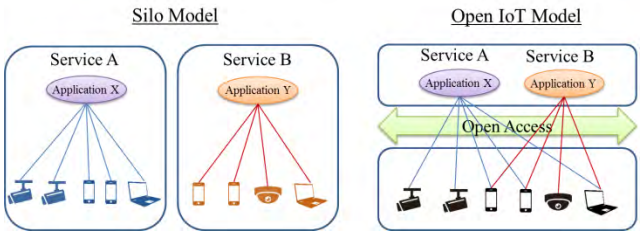

FIGURE 1. Silo IoT model and Open IoT model.

used in heterogeneous device environments such as the Open IoT environment and insist on the importance of security communication protocols in a survey.

Privacy protection is another important requirement in IoT. Many IoT services use a large amount of data generated from sensors around us, which may include privacy information. Therefore, only necessary information for a service must be extracted from data. For example, Roman *et al.* [7] show that as a core idea of privacy protection, distributed entities control the granularity of data they provide for each service. As another example, in a camera network for video surveillance, images of people can be blurred to protect their privacy [9].

In OpenIoT, services will use shared devices of multiple owners. Therefore, it is necessary not only to protect from external attackers and anonymize the data contents but also to securely connect the service provider and the device owner. We believe all these requirements can met by networking functions, which can mediate the service provider and the device owner. Accessing the devices through the network intermediation function enables data to be checked, data to be anonymized, unauthorized access to be prevented, and device owners to be anonymized. Monitoring the use of data by using the network intermediation function also enables the device owner to be provided with an appropriate incentive to allow his/her device to be used.

Next, another important requirement is to discover the most appropriate devices for a service out of a huge number of devices connected to the Internet. In this paper, we will mainly focus on this requirement and propose architecture for a tracking service as an example of a wide-coverage service realized by Open IoT. The tracking service described in this paper finds tracking targets and constantly acquires and views images them. It can be used for finding and monitoring a lost child. In Open IoT, services use output data from shared devices such as a surveillance camera or pedestrian's smartphone in a town center. Unlike general device search, statically set metadata for the device are not useful to discover appropriate devices or data because when an object or a camera moves, the camera that is filming the object changes. Appropriate devices and data must be selected dynamically. Therefore, we take the approach of naming output data generated by a sensor "live data" and using them to discover devices and data [10]. Live data include position data and image and environmental information, such as temperature, humidity, and luminance. Live data search must meet two requirements. One is that it can gather and search data among a huge amount of live data quickly because live data become invalid after a certain period. For example, in the video tracking of moving objects, a camera image of the moving object is necessary, and an image has no value after the object passes. The other is that live data search can cope with arbitrary searches on unstructured data because search queries differ by service and the live data format also differs by resource. For example, in the tracking of moving object, not only image data but also video streaming data are useful. There are also variations in search queries. Someone may want to designate a person to be tracked by facial images; others may want to designate by clothing images.

Furthermore, there is a big problem in handling live data. As the number of IoT devices will increase in the future, the data size of live data will become enormous. It is difficult to collect and handle all live data in the cloud because of the network band limit. Today's main Internet exchange mainly uses 10- or 100-Gbps interfaces, and 400-Gbp Ethernet is under development. Thus, live data will need to be handled efficiently.

Therefore, we adopted a distributed search architecture. Generally, distributed architecture uses network and computing resources less efficiently than cloud architecture. Our proposed architecture overcomes this by deploying a search function dynamically and copes with arbitrary searches.

The rest of this paper is organized as follows. Section II surveys the related work. Section III describes the proposed architecture for a tracking service. Section IV compares the proposed architecture with cloud and edge computing. Section V presents the developed system and experimental results. Section VI discusses the future tasks. Finally, Section VII concludes the paper.

II. RELATED WORK

Leading global companies are giving special attention to and making significant effort in IoT for their industrial solutions in order to stay profitable. Most of their IoT solutions adopt managed resources and create silos. Therefore, standardization organizations, such as oneM2M [11], are aiming to interconnect these solutions. The basis of such an interconnected solution is to use managed resources, and a service program can determine the types, places, and number of resources beforehand. However, in Open IoT, the service program will never be able to determine the resources beforehand. To thus realize Open IoT, a new method and architecture using unmanaged resources are required.

The "resource directory" is a concept to discover an appropriate resource on the basis of the resource's metadata. The resource directory collects a resource's metadata that are connected in its domain and provides search functionality of the collected data. Currently, standardization organizations, such as CoRE [12], [13] are specifying such metadata and procedures of gathering and querying metadata. AllJoyn [14], which is one of the open source projects, enables multicast and multicast Domain Name System (DNS)-based [15]

discovery using metadata. The Web search system using Universal Description, Discovery and Integration (UDDI) [16] also discovers devices by using metadata. Though these technologies can discover and identify devices connected to the network, their purposes are different from ours. Our aim is to find the devices and data that have the target sensing data in a constantly changing situation. In other words, we want to search for contexts, not to find a specific individual device.

From the above, we believe that live data search is effective as a new method to discover IoT resources. For live data search, Elahi *et al.* [17] focused on predicting current live data of resources. Their prediction model helps us to find sensors matching a user query; however, it is not always accurate and can be applied only to devices that show periodic patterns. Perera *et al.* [18] focused on the context-property-based sensor search; however, the properties were predefined, which does not allow services to meet arbitrary search requirements.

Furthermore, network bandwidth to cope with live data must be considered. Because of the data transfer volume, it is difficult for live data search to adopt web search engine architecture, which crawls web page information at datacenters. There are various types of live data: some are strings, some are images, and some are streams. Therefore, the transfer of live data may cause a huge amount of network traffic. Web search engine architecture crawls the content of web pages at datacenters and creates a huge index using the collected data [19]–[22]. This means that the architecture collects all live data into a centralized datacenter and creates an index, which may cause communication congestion.

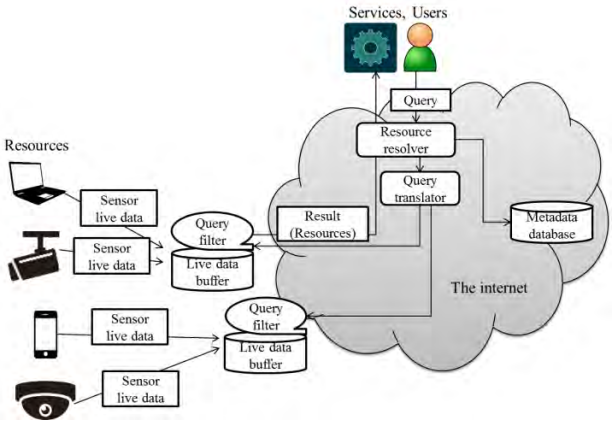

FIGURE 2. Overview of distributed search architecture.

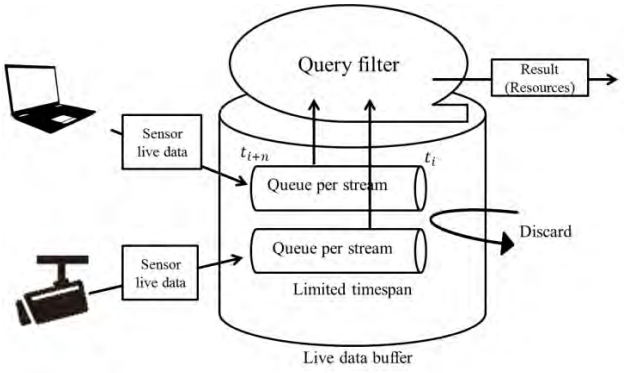

FIGURE 3. Structure of live data buffers.

III. PROPOSED ARCHITECTURE

We designed distributed search architecture for a tracking service. To realize live-data-based search, we propose two new concepts: a live data buffer and a resource resolver. The live data buffer gathers the live data of every device, and the resource resolver resolves the appropriate resource and data for a service at a certain moment. Fig. 2 gives a general overview of the architecture. Live data buffers are network nodes distributed in a wide-area network, and they gather a resource's live data belonging to the same network. When a service queries the resource resolver about the appropriate resources that meet the service requirements, the resource resolver then queries the local live data buffer about the condition, and then the local live data buffer checks the current live data of the resources. If there are appropriate resources, the local buffer returns the resource information to the service through the resource resolver.

To deal with arbitrary searches for unstructured live data, the live data buffer must have various data analysis functions, such as face recognition and color recognition. However, it is inefficient to have all functions in all live data buffers. Only the necessary function should be deployed in accordance with the query from a service. Therefore, the architecture dynamically distributes the search function named a query filter, which is software that analyzes live data and finds data corresponding to the query. When a user or service sends the query to the resource resolver, the query translator creates a query filter. When a new query filter is set to the live data buffer, the query filter starts to analyze live data. A query filter can be created from multiple search algorithms. Moreover, a query translator can distribute a query filter to live data buffers in a local network as a download application, virtual machine (VM) image file, or Linux Container. In addition, to narrow down the search area, a query translator determines the distributed destination live data buffer on the basis of the metadata. Metadata are static information of live data buffer and devices that include location, owner, etc.

Fig. 3 illustrates the details of live data buffers. A live data buffer is the gathering point of the live data of resources. Each resource pushes the live data to the nearest or a specified live data buffer, or the live data buffer actively pulls the resources' live data. Then the live data buffer collects each resource's live data in a period. It forms a certain time window, t_i to t_(i+n), (t is time), and the expired t_(i-1) data are discarded or transferred for other uses. The range of "n" should be set to reflect the dynamism of a resource's live data characteristics. Frequent queries permanently set corresponding filters to live data buffers. To search live data quickly, our architecture limits the time window to make the targeted data smaller. We will also explain how to implement the architecture on a commercial network. Edge computing [23] and fog

TABLE 1. Comparison of Architectures

	Cloud Computing	Edge Computing	Distributed search architecture
Network traffic	**Bad**	**Good**	**Good**
Query variation	**Good**	**Bad**	**Good**
Performance guarantee for multiple processing	**Sufficient** (Has plentiful computing resources)	**Sufficient** (Can estimate the maximum load)	**Difficult** (Uses shared and limited computer resources)
Performance guarantee for hardware dependent processing	**Sufficient** (Special computer)	**Sufficient** (Common computer)	**Difficult** (Common computer)
Other concern	**Nothing**	**Nothing**	Processing time and network traffic of query filter distribution

computing [24] are models that perform calculation processing at the network edge instead of cloud concentration.

These models are highly adaptable to the live data buffer. One way to implement the distributed search architecture is to deploy the query translator in the core network and utilize the edge server in the local network as the live data buffer. Edge side communication equipment, such as a radio base station that passes live data of IoT devices, is a candidate for the equipment on which to install the live data buffer. There is a model called Dew computing [25] that uses a device closer to the user than the network edge. It is also adaptable to the live data buffer. In fact, we are planning to implement the data analysis function of the live data buffer on abundant local computers.

IV. COMARISON OF ARCHITECTURES

We compared the network function deployment model of the proposed distributed search architecture with cloud computing and edge computing. Table 1 shows the comparison. First, the proposed architecture has smaller network traffic for discovering data and devices than cloud computing, as intended. The proposed architecture has the same network traffic as edge computing using local machine resources. In addition, the proposed architecture can handle many more kinds of requests than the simple edge computing, in which a specific program is installed and used beforehand in a local limited computing resource. Cloud computing with abundant computing resources can handle as many kinds of requests as the proposed architecture. However, there are several tradeoffs between features in the proposed architecture.

The first feature is that the proposed architecture does not use dedicated computers but shared computers in the local network as live data buffers. Computers are not occupied by one query filter. The live data buffer simultaneously executes various query filtering processes in accordance with the search query. Therefore, it is difficult to design appropriate computing resources by verifying the processing load in advance. This is different from both edge and cloud computing. In contrast, simple edge computing executes only preinstalled applications, so the maximum load can be estimated in advance. Cloud computing is originally suitable for utilizing shared computing resources. All data from multiple locations are efficiently processed by plentiful computing resources in the cloud. To solve this potential problem of the proposed architecture, we are planning to apply resource management tools such as OpenStack [26] to the system. For example, OpenStack can specify the size of central processing unit (CPU) cores and memory used by the VM by the catalog called flavor. In addition, OpenStack can be linked to the CPU pinning function of KVM (Kernel Virtual Machine) [27], which allocates the VM to the specific CPU core. When there are no free resources, these functions can detect and notify that. Thus, the system will be able to handle multiple query filter executions while guaranteeing the performance.

The second feature is that the proposed architecture does not use special computers but common computers. Since the live data buffer must handle various live data and query filters, preparing tuned computers for specific applications is not suitable. Therefore, the performance may be lower than when using special computers. For example, a graphics processing unit (GPU) is generally suitable for video data analysis and processing. As described above, in cloud computing, it is only necessary to prepare a certain number of GPU servers in the cloud. However, in the proposed architecture and simple edge computing, preparing the GPU servers in all local environments is inefficient. In the future, we are planning to prepare multiple types of computer resources that include GPU and large storage at the appropriate network aggregation points and dynamically allocate corresponding query filters.

The third feature is the distribution cost of the query filter. In the proposed architecture, since the query filters have to be distributed after receiving the search request, the total search time is longer than in the case of merely executing a preinstalled application. In addition, there is communication traffic for query filter distribution. Only the proposed architecture presents this concern. We will show experimental results on the degree of these costs in the next section.

V. IMPLEMENTATION AND EXPERIMENTATION

We evaluated the feasibility of the proposed architecture in experiments. We evaluated the effect of reducing network traffic on searches by analyzing live data locally. This is the main purpose of the architecture. In addition, we also evaluated the search speed because distributing the query filter affects response time.

For the experiment, we developed a distributed search system and application software that embodies the proposed architecture [28]. Fig. 4 shows the system configuration, and Fig. 5 shows a picture of the experiment environment. It simulates a tracking service across two networks. Smartphone cameras are arranged as shared devices. Within the

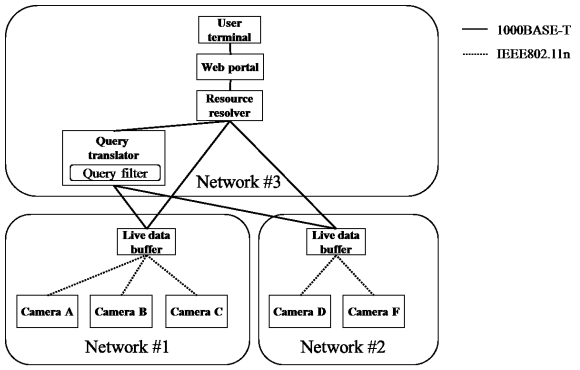

FIGURE 4. System configuration for experiments.

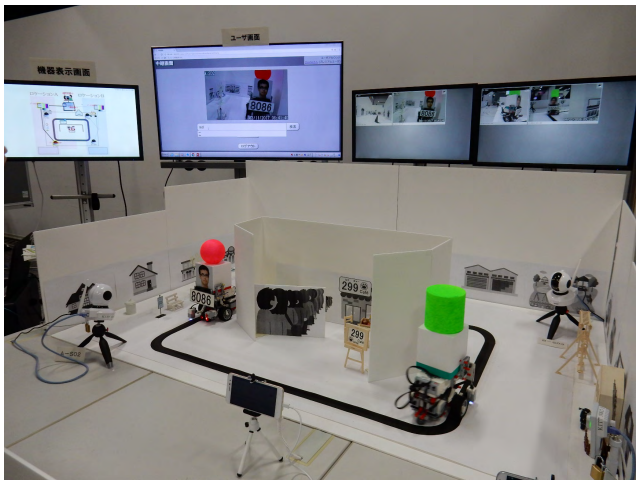

FIGURE 5. Experiment environment.

FIGURE 6. Service portal's screen.

TABLE 2. Machine specifications

Module		CPU (Virtual Machine)	Memory (Virtual Machine)
Web portal		Intel Core i5-6200U CPU @2.30GHz × 2 (1 core)	(2 GB)
Resource resolver		Intel Core i7-6700 @3.40GHz × 8 (2 core)	(12 GB)
Query translator		Intel Core i7-6700 @3.40GHz × 8 (1 core)	(4 GB)
Live data buffer	Query filter	Intel Xeon CPU E5-2620 v3 @2.40GHz × 12	32GB
	Other	Intel Core i7-6700 CPU @3.40GHz × 8	16 GB

viewing range of the smartphone camera, movable objects are arranged that have several features of appearance. The resource resolver and service portal are in different networks from the devices.

Fig. 6 shows the service portal's screen that accepts search queries from users. The query through the search portal is notified to the resource resolver. Then, a corresponding query filter application is selected by the query translator and distributed to the live data buffer on the two local networks. Application software for face recognition, color recognition, character recognition, and shape recognition are implemented as the query filter, and total file size is 115 MB. The live data buffer acquires the captured image (file size is 9.4 kByte) at a fixed cycle (200 ms) from smartphones of the same network domain. The live data buffer analyzes the video data, finds the video data matching the query, and reports the data source device and the score of the analysis to the resource resolver.

Video data of the search result are output to the service portal. When multiple cameras are displaying the object, the clearest one (i.e., has the highest algorithmic score for the query filter) is selected from the analysis result. The search process is always performed until the user makes a termination request, and the search result dynamically changes in accordance with the movement of the object. That is, video of tracking targets is always present in the service portal's screen as a result of the search. The specifications of each functional part and details of the query filter are shown in Table 2. In the experiment, we use two computers for a live data buffer: one for executing query filtering, and the other for the other processing. Functions other than the live data buffer are implemented as VMs.

A. NETWORK TRAFFIC MEASUREMENT

We measured the network traffic between the devices and live data buffer and between the live data buffer and resource resolver. In this environment, the live data buffer acquired 9.4 kByte of video data at a cycle of 200 ms per device, that is, network traffic on average between the devices and live data buffer is 47 kByte / s. If we aggregate and analyze all the image data in the cloud, this amount of network traffic will go through the core network. On the other hand, the network traffic between the live data buffer and resource resolver is 7.6 Byte / s on average. In the proposed system, only this amount of traffic will go through the core network. By using our architecture, the communication bandwidth of the core network can be reduced to 1000th or less of that when using the cloud model.

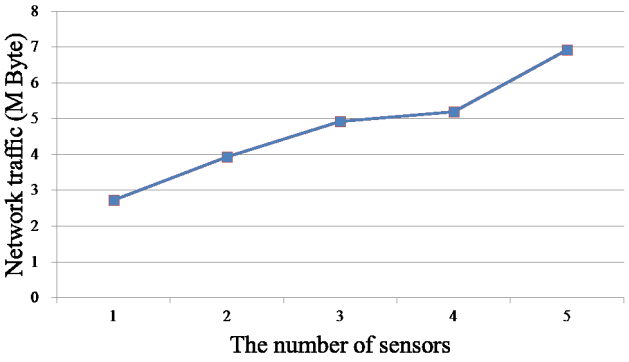

FIGURE 7. Integration of resource resolver notification data size with respect to number of sensors.

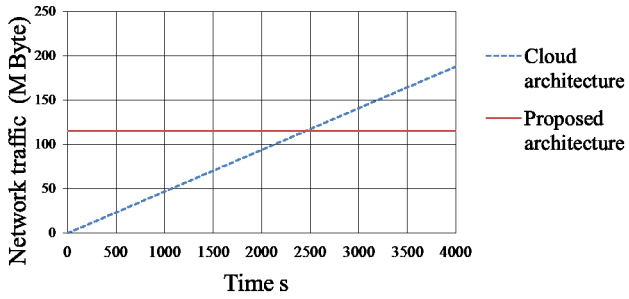

FIGURE 8. Time buildup of traffic volume of core network for cloud architecture and proposed architecture.

In addition, our query filter aggregates data analysis results for multiple sensors and notifies the resource resolver of them, so even if the number of sensors increases, the traffic volume between a resource resolver and live data buffers does not simply double. Fig. 7 shows the integration of the resource resolver notification data size with respect to the number of sensors. The data in this figure show the integrated value of communication when data of the device are processed for three minutes with two live data buffers. As the number of sensors increases, the amount of information to be reported increases but only very slightly.

Finally, we also measured the effect of distributing a query filter as an application file because it may increase network traffic temporarily. Fig. 8 shows the time buildup of the traffic volume of the core network in cases of the cloud architecture and the proposed architecture for a single smartphone camera. As a result of the experiment, in the case of one camera, the total traffic volume was smaller in the proposed architecture than in the cloud intensive model if the analysis request lasted over 40 minutes (2400 s). Also, in the case of two cameras, the traffic volume of the cloud architecture doubles, but as shown in Fig. 7, the traffic volume of the proposed architecture only increases very slightly.

Therefore, boundaries of service duration where the proposed architecture is beneficial are about 20 minutes for 2 cameras. The boundaries of service duration become smaller in direct proportion to the number of cameras.

TABLE 3. Results for processing time in each sequence

sequence	Time (s)	
	1 live data buffer	2 live data buffers
Receive query and distribute query filter	5.317	6.357
Execute query filter	1.044	1.044
Returns the address of the device	0.463	0.463
Total	6.824	7.864

TABLE 4. Time of query filter distribution.

Sequence	Time (s)			
	Download		Pre-download	
	1 buffer	2 buffers	1 buffer	2 buffers
Receive query and distribute query filter	5.317	6.357	0.923	1.046

B. SEARCH SPEED MEASUREMENT

We measured the time from when the resource resolver received the service request to when it returned the IP address and video data of the device. We have two experiment conditions: one network and two networks. In this experiment, each network has one live data buffer and one smartphone camera. We obtained the median value of the time by carrying out the test five times under the same conditions. Table 3 shows the results.

The maximum response time is 7.864 seconds when there are two live data buffers. If the average walking speed of a person is 80 m / min. (1.333 m / s), a person will move 10 m during the search. Although it depends on the viewing angle of the camera, this search speed is sufficiently fast for a walking-person tracking service. Also, the time taken to deliver the query filter is about 80% of the total time. This time can be expected to be shortened by optimizing the application size to be distributed.

The difference in response time due to the number of live data buffers appears in the delivery time of the query filter. To determine whether this difference is due to network load, we conducted experiments in which the query filter was delivered to the live data buffer beforehand. Table 4 shows the results. Even without delivery, the time is slightly longer when there are two live data buffers but is still much shorter than when there is a download. From the above, the reason that the time becomes longer when there are two live data buffers is that the network load is increased by the distribution amount of query filters. We also want to mention that subsequent requests that use the same query filter take pre-download time in Table 4. The total time required for discovering the same data and device is greatly shortened. When there are two live data buffers, the time is about 2.553 seconds. In this case, it is possible to track runners and cyclists estimated to be about three times as fast as pedestrians.

VI. DISCUSSION

In this paper, we described the distributed search architecture for a tracking service using shared devices and evaluated its effectiveness, but further examination is required.

First, query filter application software needs to be distributed more efficiently. When applied to networks with a large number of network domains such as the Internet, distributing a personal query filter to all live data buffers generates a large download cost. The live data buffer's computing cost is also greatly wasted. The distribution-destination live data buffer of the query filter needs to be narrowed down beforehand. For one approach, we believe that static metadata can be effectively used. For example, for a service that tracks a specific individual, the country or region to be searched may be narrowed down in advance. By registering existing areas as static metadata of the device and providing an index-based search like a pre-existing web service as a pre-filter, devices for live data search can be narrowed down. However, it is necessary to consider what should be managed as static metadata together with the service. Also, another approach is to improve query filters. By handling common queries collectively, the processing of the local live data buffer will be able to be made more efficient. For example, there is a concept of a gradual query filter. Initially, a filter that detects moving objects is distributed and commonly used, and only the personal conditions such as the identification of the face of a specific person are distributed later. This reduces the distribution cost and computation cost of query filters and also reduces search time by using preinstalled common filters.

Furthermore, to use various shared devices, it is necessary to abstract unique interface types and access protocols for each device. With Open IoT targeting indefinite devices, it is not realistic for a service program to implement multiple control logic for various resources in advance. To promote Open IoT, we plan to develop a mediation function for the architecture. It will enable service programs to easily use various resources made by different manufacturers.

Finally, security and privacy are important for Open IoT. The proposed architecture can also apply security and privacy measures. For example, the experiment system prohibits direct access to the device, so that it can acquire the data of the device only through the resource resolver. In addition, security applications such as data anonymization, encryption, and malware detection can be delivered to the appropriate live data buffers using query filters. One of the great merits of the proposed architecture is it delivers the security and privacy protection application suitable for each type of device and service.

VII. CONCLUSION

The Internet of Things (IoT) is rapidly expanding, and we are aiming to realize a wide service coverage and low cost service by Open IoT, where multiple services share devices in an environment. One of the main services of Open IoT will be a tracking service that will use real-time information generated by devices such as video data and detected values. We named such data "live data" and use them to search for appropriate devices for the service. This requirement is not in conventional resource search technologies using static data; it is a new requirement for realizing Open IoT that utilizes a variety of shared devices. However, there is a big problem in handling live data: as the number of IoT devices increases in the future, the data size of live data will become enormous. It is difficult to collect and handle all live data in the cloud because of the network band limit. Live data must thus be handled efficiently.

In this paper, we proposed a distributed search architecture for a tracking service in Open IoT. The architecture forms pairs of search queries and data analysis applications and distributes applications dynamically in accordance with search requests. It is a new method that extends edge computing. It solves the problem of network band for real-time and flexible device search and also has extensibility to accommodate various devices and service requirements.

We also developed a system that embodies the proposed architecture and evaluated its feasibility. An experiment showed that the architecture reduces the communication bandwidth of the core network to 1000th or less of that when using the cloud model. In addition, another experiment showed the search speed of the architecture is sufficiently fast for a walking-person tracking service. The resource discovery is a common function as the core of the Open IoT platform, and it is a technology that will greatly enhance the advancement, flexibility, and scale of Open IoT services. In the future, we plan to extend the architecture to make it applicable to various devices and services.

REFERENCES

[1] D. Evans, "The Internet of Things—How the next evolution of the Internet is changing everything," Cisco Internet Business Solutions Group, San Jose, CA, USA, White Paper, Apr. 2011. [Online]. Available: https://www.cisco.com/c/dam/en_us/about/ac79/docs/innov/IoT_IBSG_0411FINAL.pdf

[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)*, Ningbo, China, Sep. 2011, pp. 2944–2947.

[3] 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.

[4] J. Manyika *et al.*, "The Internet of Things: Mapping the value beyond the hype," McKinsey Global Inst., San Francisco, CA, USA, Tech. Rep., Jun. 2015.

[5] E. Siow, T. Tiropanis, and W. Hall, "Analytics for the Internet of Things: A survey," *ACM Comput. Surv.*, vol. 51, no. 4, p. 74, 2018.

[6] L. Atzori, A. Iera, and G. Morabito, "The Internet of Things: A survey," *Comput. Netw.*, vol. 54, no. 15, pp. 2787–2805, Oct. 2010.

[7] R. Roman, J. Zhou, and J. Lopez, "On the features and challenges of security and privacy in distributed Internet of Things," *Comput. Netw.*, vol. 57, no. 10, pp. 2266–2279, 2013.

[8] T. Qiu, N. Chen, K. Li, M. Atiquzzaman, and W. Zhao, "How can heterogeneous Internet of Things build our future: A survey," *IEEE Commun. Surveys Tuts.*, vol. 20, no. 3, pp. 2011–2027, 3rd Quart., 2018.

[9] J. Wickramasuriya, M. Datt, S. Mehrotra, and N. Venkatasubramanian, "Privacy protecting data collection in media spaces," in *Proc. ACM Int. Conf. Multimedia*, New York, NY, USA, Oct. 2004, pp. 48–55.

[10] T. Ikebe, H. Noguchi, and N. Hoshikawa, "Distributed live data search architecture for resource discovery on Internet of Things," in *Proc. IEEE World Forum Internet Things*, Reston, VA, USA, Dec. 2016, pp. 591–596.
[11] *OneM2M*. Accessed: Oct. 15, 2018. [Online]. Available: http://www.onem2m.org/
[12] Z. Shelby, *Constrained RESTful Environments (CoRE) Link Format*, document RFC 6690, Aug. 2012.
[13] Z. Shelby, M. Koster, C. Bormann, and P. van der Stok, *CoRE Resource Directory*, document draft-ietf-core-resource-directory-09, Internet-Draft, Oct. 2016.
[14] *AllJoyn*. Accessed: Oct. 15, 2018. [Online]. Available: https://events.static.linuxfound.org/sites/events/files/slides/AllJoynThinLibraryTargetPorting_PeterKrystad_MathewMartineau.pdf
[15] S. Cheshire and M. Krochmal, *Multicast DNS*, document RFC 6762, Feb. 2013.
[16] K. Tamilarasi and M. Ramakrishnan, "Design of an intelligent search engine-based UDDI for Web service discovery," in *Proc. Int. Conf. IEEE Recent Trends Inf. Technol. (ICRTIT)*, Apr. 2012, pp. 520–525.
[17] B. M. Elahi, K. Römer, B. Ostermaier, M. Fahrmair, and W. Kellerer, "Sensor ranking: A primitive for efficient content-based sensor search," in *Proc. IPSN*, San Francisco, CA, USA, 2009, pp. 217–228.
[18] C. Perera, A. Zaslavsky, C. H. Liu, M. Compton, P. Christen, and D. Georgakopoulos, "Sensor search techniques for sensing as a service architecture for the Internet of Things," *IEEE Sensors J.*, vol. 4, no. 2, pp. 406–420, Feb. 2014.
[19] J. Dean and S. Ghemawat, "MapReduce: Simplified data processing on large clusters," *Commun. ACM*, vol. 51, no. 1, pp. 107–113, 2008.
[20] J. Dean and S. Ghemawat, "MapReduce: Simplified data processing on large clusters," in *Proc. 6th Conf. Symp. Oper. Syst. Design Implement.*, San Francisco, CA, USA, Dec. 2004, pp. 137–150.
[21] S. Ghemawat, H. Gobioff, and S. Leung, "The Google file system," in *Proc. 19th ACM Symp. Oper. Syst. Principles*, Bolton Landing, NY, USA, Oct. 2003, pp. 29–43.
[22] A. Toshniwal *et al.*, "Storm@twitter," in *Proc. ACM SIGMOD Int. Conf. Manage. Data*, Snowbird, UT, USA, Jun. 2014, pp. 147–156.
[23] Y. Jaraweh, A. Doulat, O. AlQudah, E. Ahmed, M. Al-Ayyoub, and E. Benkhelifa, "The future of mobile cloud computing: Integrating cloudlets and mobile edge computing," in *Proc. 23rd Int. Conf. Telecommun. (ICT)*, Thessaloniki, Greece, May 2016, pp. 1–5.
[24] F. Bonomi, R. Milito, J. Zhu, and S. Addepalli, "Fog computing and its role in the Internet of Things," in *Proc. 1st Ed. MCC Workshop Mobile Cloud Comput.*, Helsinki, Finland, Aug. 2012, pp. 13–16.
[25] P. P. Ray, "An introduction to dew computing: Definition, concept and implications," *IEEE Access*, vol. 6, pp. 723–737, 2017.
[26] *OpenStak*. Accessed: Oct. 15, 2018. [Online]. Available: https://www.openstack.org/
[27] *KVM*. Accessed: Oct. 15, 2018. [Online]. Available: https://www.linux-kvm.org/page/Main_Page
[28] M. Kataoka, N. Hosikawa, H. Noguchi, T. Demizu, and Y. Yamato, "Tacit computing and its application for open IoT era," in *Proc. IEEE Consum. Commun. Netw. Conf. (CCNC)*, Las Vegas, NV, Jan. 2018, pp. 12–16.

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 involved in the developmental research of server virtualization and Internet of Things. He is currently a Researcher with NTT Network Service Systems Laboratories.

TATSUYA DEMIZU was born in Osaka, Japan, in 1988. He received the B.S. degree in engineering from Osaka University, Japan, in 2011, and the M.S. degree in engineering from The University of Tokyo, Japan, in 2013. In 2013, he joined NTT Corporation, Japan, where he is currently a Researcher with NTT Network Service Systems Laboratories. His research interests include network architecture of a career network accepting multiple service providers with various requirements and global communication mechanisms for Internet of Things devices. He is a member of IEICE.

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 involved in simplifying networks. Since 2016, she has been involved in the developmental research of distributed processing platforms and Internet of Things platforms at NTT Laboratories. She is currently a Research Engineer with NTT Network Service Systems Laboratories.

YOJI YAMATO (SM'16) 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 involved in the developmental research of cloud computing platform, peer-to-peer computing, and Internet of Things. He is currently a Senior Research Engineer with NTT Network Service Systems Laboratories. He is a Senior Member of IEICE and a member of IPSJ.