Demo Abstract: LASK: A Distributed Service Discovery Platform on Edge Computing Environments

Yuuichi Teranishi, Takashi Kimata, Hiroaki Yamanaka, Eiji Kawai, Hiroaki Harai
National Institute of Information and Communications Technology, Tokyo, Japan
Email: {teranisi,kimata,hyamanaka,eiji-ka,harai}@nict.go.jp

***Abstract*—We present the LASK protocol and its platform implementation that supports distributed k-Nearest Service Discovery. LASK achieves scalable and locality-aware name-based service discovery and routing for the target nodes avoiding redundant lookup message exchanges across the edge networks.**

I. Introduction

The edge computing architecture has been considered as a next-generation network computing paradigm.The edge computing architecture enables applications to have a shorter response time, reduces the computation load on devices, and localizes the data processing. Such properties are suitable for the advanced Internet of Things (IoT) applications. In a recent forecast, the number of Multi-access Edge Computing (MEC) resources in 5G is expected to be over a million in the near future [1].

In such environments, the *service discovery* plays an important role for efficient executions of applications by finding appropriate service or resource to use under the situation of growth in the size and the diversity of the services and resources [2]. As a simple but flexible service discovery, we focus on the name-based service discovery. The name-based service discovery discovers an edge node (i.e., a server, a process) that has a specified name that describes a certain status or a running service. The service discovery needs to be scalable as the number of services is assumed to be a large number. There is a specific requirement for the edge computing environments that an application needs to discover a service or a resource on the computer physically located closely, in order to achieve the small latency. In addition, there might be a large number of edge nodes that register the same name because the tasks are offloaded. The offloaded task of a real-time IoT application often requires parallel executions to be finished within a short period. For such tasks, multiple edge computers must be discovered for the parallel executions. For such cases, a certain number of nodes (k nodes) that holds the specified name must be discovered to satisfy the required performance.

We demonstrate a service discovery protocol called LASK (Locality-Aware Service discovery protocol for K-nearest search) [3], which provides functions for the above purpose. LASK supports distributed *k-Nearest Service Discovery (kNSD)*, which finds k nearest services that hold the matched name in order of the closeness.

II. k-Nearest Service Discovery

An example of the edge computing network model is shown in Fig. 1. The edge computing environment on the Internet has natural hierarchical properties such as the access networks, the Internet service providers' networks, and the wide-area networks between countries and continents. The target nodes of the service discovery can belong to any leaf subnetwork in the hierarchy.

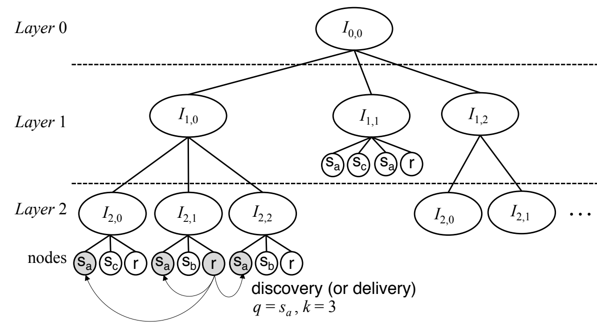

Fig. 1. An example of the edge computing network model and kNSD

Each subnetwork in each layer has a network *identifier*. The identifier is denoted as $I_{l,u}$, where l is the layer in the hierarchy and u is the unique value in the layer l, e.g., a sequence number in each layer. The network identifier of a node can be denoted as an ordered list of the identifiers. For instance, the leftmost subnetwork in the layer 2 in Fig. 1 is denoted as $\{I_{0,0}, I_{1,0}, I_{2,0}\}$. We define the order of network identifier as the lexicographical order. We also define a notion of *network distance*. The network distance $D(x, y)$ between two network identifiers x and y can be calculated as follows:

$$D(x, y) = max(|x.nid|, |y.nid|) - |p(x.nid, y.nid)|$$

where $x.nid$ denotes the network identifier of x and p is a function to pick up the common prefix in the two network identifiers. For example, $D(\{I_{0,0}, I_{1,1}\}, \{I_{0,0}, I_{1,1}\})$ is 0 and $D(\{I_{0,0}, I_{1,2}, I_{2,0}\}, \{I_{0,0}, I_{1,1}\})$ is 2.

By using the above definition, The kNSD can be defined formally as follows.

> Given a set of all nodes in the entire network N, a query for the name q, and a query requester p, kNSD finds a set of nodes S such that $|S| = k$ and for any $s \in S$ and $s' \in V - S$, $D(p.nid, s.nid) \leq D(p.nid, s'.nid))$, where $V = \{v | sv(v) = q, v \in N\}$ and $sv(v)$ is the the name of the service that the v provides.

Fig.1 shows a case where a requester node r requests $3(= k)$ services named s_a.

978-1-7281-2700-2/19/$31.00 2019 © IEEE

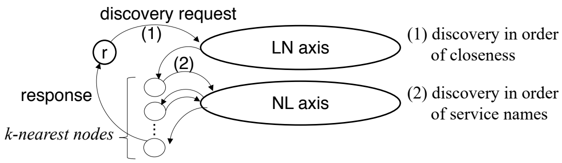

Fig. 2. Discovery sequence of LASK

TABLE I
THE KNSD REST API (PARTIAL LIST)

method	path	description
`PUT`	/netid/$\{name\}$	register nid
`PUT`	/name/$\{name\}$	register $name$
`DELETE`	/name/$\{name\}$	unregister $name$
`PUT`	/name/$\{name1\}$/$\{name2\}$	change $name1$ to $name2$
`GET`	/endpoints/$\{name\}$?k=$\{k\}$	find k services for $name$

III. LASK OVERVIEW

We have proposed a distributed kNSD protocol called LASK. LASK assumes Key-order Preserving Structured Overlay Network (KOPSON) as a base algorithm of the distributed discovery. In our LASK platform implementation, we used a bi-directional KOPSON algorithm called Suzaku [4], which we have proposed.

The principle idea of LASK is to use two KOPSONs for two-dimensional routing to the nearest service and the rest of the k services. By using KOPSON, the communication loads are balanced among nodes, and the node failures are recovered in a self-organized manner.

The one dimension is used for the location-name (LN) axis and the other dimention for the name-location (NL) axis. For the LN and NL axis, each node has keys represented as $\{nid$, $name$,$unique\_id\}$ and $\{name$, nid,$unique\_id\}$, respectively. The $unique\_id$ is the randomly assigned identifier for each node. The LN axis is used for lookup the nearest node that matches to the query. The NL axis is used for lookup the rest-of k nodes that match to the query in order of the closeness. LASK can perform scalable discovery because it uses the routing algorithm of KOPSON. The number of lookup hops logarithmically increase against the number of nodes. In addition, the locality of the routing is preserved by the network distance because the nodes in both dimensions are sorted by nid. Fig. 2 shows the interaction between a requester node r and both axises for a k service discovery. Please refer our original paper [3] for more in detail.

IV. LASK PLATFORM IMPLEMENTATION

We implemented a LASK platform using Suzaku as a KOPSON. Suzaku is implemented on an open source overlay network platform PIAX [5]. Fig. 3 shows the module structure of our LASK platform implementation.

As an application interface for the service discovery, we provide a REST API for easy access and simple implementations. Each edge node runs a process of HTTP(S) server and accepts operation requests for kNSD from the application processes. Some REST APIs provided in our implementation are listed in Table I. These REST APIs are intuitively designed. The `GET` method for the path beginning from "/endpoints" discovers the IP address of the edge node that matches to the request.

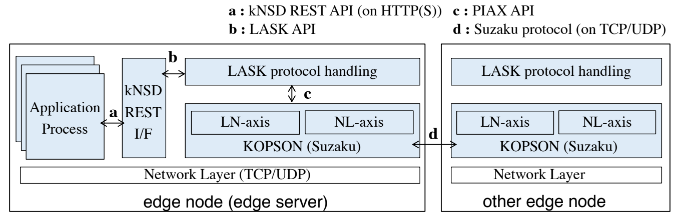

Fig. 3. The module structure of our LASK platform implementation

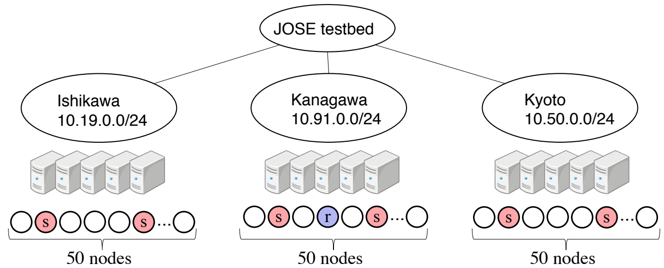

Fig. 4. The demonstration environment on JOSE testbed

The accepted request is forwarded to the LASK protocol handling module. The LASK protocol handling module registers/unregisters the name and discovers the endpoints for the specified name via PIAX API. The LASK-implemented edge nodes exchange messages to maintain the overlay network structure based on the Suzaku protocol. As the underlay network protocol, TCP or UDP can be used. The protocols for the underlay network are implemented as plug-ins.

In the demonstration, we present the system implementation of LASK platform running on 150 servers (edge nodes) in JOSE [6] testbed (Fig. 4). There are three edge networks (data centers) on the testbed, and each network contains several services that matches to the requested name s. By issuing kNSD REST request from a node r to the LASK platform, a discovery request is issued, forwarded, and matched by the nearest services in order of the closeness. The response is returned promptly by the scalable lookup feature of the distributed KOPSON protocol.

REFERENCES

[1] iGillottResearch Inc., "The Business Case for MEC in Retail: A TCO Analysis and its Implications in the 5G Era," 2017.
[2] E. Meshkova, J. Riihijärvi, M. Petrova, and P. Mähönen, "A survey on resource discovery mechanisms, peer-to-peer and service discovery frameworks," *Computer networks*, vol. 52, no. 11, pp. 2097–2128, 2008.
[3] Y. Teranishi, T. Kimata, H. Yamanaka, E. Kawai, and H. Harai, "Supporting k-nearest service discoveries for large-scale edge computing environments," in *Proc. of GLOBECOM 2018*, 2018, pp. 1–7.
[4] K. Abe and Y. Teranishi, "Suzaku: a churn resilient and lookup-efficient key-order preserving structured overlay network," *IEICE Transactions on Communications*, 2019.
[5] Y. Teranishi, "PIAX: Toward a Framework for Sensor Overlay Network," in *Proc. of CCNC'09 (Workshops)*, 2009, pp. 1–5.
[6] Y. Teranishi, Y. Saito, S. Murono, and N. Nishinaga, "JOSE: An Open Testbed for Field Trials of Large-scale IoT Services," *Journal of the National Institute of Information and Communications Technology Vol*, vol. 62, no. 2, pp. 151–159, 2015.