IEICE TRANS. COMMUN., VOL.E102–B, NO.9 SEPTEMBER 2019

PAPER Special Section on Enhancing Information Centric Networking Technologies Towards Real-world Infrastructure

Location-Based Forwarding with Multi-Destinations in NDN Networks

Yoshiki KURIHARA[†a)], **Yuki KOIZUMI**[†], ***Members***, **Toru HASEGAWA**[†], ***Fellow***, ***and*** **Mayutan ARUMAITHURAI**[††], ***Nonmember***

SUMMARY Location-based forwarding is a key driver for location-based services. This paper designs forwarding information data structures for location-based forwarding in Internet Service Provider (ISP) scale networks based on Named Data Networking (NDN). Its important feature is a naming scheme which represents locations by leveraging space-filling curves.

key words: *information centric networking (ICN), location-based service, location-based forwarding*

1. Introduction

As Internet of Things (IoT) devices have been deployed everywhere, *location-based services* (*LBSs*) based on cloud computing [1] and crowd sourcing [2] have been studied. Since data obtained by IoT devices is location dependent, how to specify and retrieve such data is a research issue. Hereafter, we refer to such data as *location data*. *Location-based forwarding* is promising to support location data retrieval thanks to its nature that packets are forwarded to a location of a device rather than its address. This paper addresses location-based forwarding, focusing on crowd-sourcing-based LBSs, because they have a potential advantage of providing location privacy of users, which is a critical issue for LBSs. Please note that this paper focuses on forwarding, whereas the other papers of the authors have addressed location privacy [3], [4].

Location-based forwarding mechanisms have been studied for a long time for vehicular ad-hoc networks (VANETs) [5]; however, they are not appropriate solutions to the objective of this paper, i.e., on-demand location data retrieval from a crowd of IoT devices. First, most studies assume that locations of vehicles are tracked by a central server and their locations are resolved by the server [5]. This incurs extra round trip delay as well as communication overhead for the tracking. Second, their forwarding mechanisms deliver a packet to a vehicle of interest in a unicast manner, whereas our forwarding mechanism aims at delivering a packet to multiple IoT devices at a specified location in a multicast manner. Third, greedy forwarding adopted by most mechanisms is not so scalable as to be used in Internet Service Provider (ISP)-scale networks, whereas LBSs are generally provided in such networks.

This paper takes the following approaches to develop a location-based forwarding mechanism in ISP-scale networks. We assume Named Data Networking (NDN) [6] as underlying network architecture because NDN natively represents application names as network names. First, the mechanism supports multi-source retrieval so that location data of multiple IoT devices is retrieved at once as the pioneering work in NDN does so [7]. This reduces the number of request packets by leveraging the multicast nature. Second, this paper adopts a routing framework, whereas the existing mechanisms based on NDN use greedy forwarding [7], [8].

The routing framework posits a research issue about a naming scheme, which is not well addressed by the existing studies. Road addresses [8], [9] are not general enough for general services other than vehicular communications, whereas an x-y coordinate system [7] raises a scalability problem. Since a destination location is specified by two fields, i.e., an x-coordinate and a y-coordinate numbers, location-based forwarding is regarded as packet classification. A problem derived from this is a large data structure for such packet classification. The size is proportional to the multiplication of the numbers of values of the two fields [10].

To circumvent the problem, in our previous paper [11], we adopt a *Z-order* system [12], which is a space-filling curve, to represent locations. In contrast to x-y coordinate systems, a Z-order system enables to identify locations with one-dimensional quaternary numbers, thereby reducing location-based forwarding to prefix matching of names of locations. Since a Z-order number specifies a square of a specified size, this enables it for a collector of location data to resize an area as she or he wishes. In our previous work, we design a location data naming scheme for both locations and data types, a hierarchical PATRICIA-based data structure for the both names and prefix search rules to support multiple destinations in a specified area.

In this paper, we extend our previous work in [11] to precisely design the forwarding frame work. The extensions are summarized as follows: First, we rethink location-based forwarding in a bottom up manner and conclude that a space-filling curve like Z-order is an appropriate naming scheme. We compare sizes of forwarding data structures of the Z-order system with those of an x-y coordinate system

Manuscript received October 5, 2018.
Manuscript revised January 21, 2019.
Manuscript publicized March 22, 2019.

[†]The authors are with Graduate School of Information Science and Technology, Osaka University, Suita-shi, 565-0871 Japan.
[††]The author is with Institut für Informatik, Georg-August-Universität Göttingen, Germany.
a) E-mail: y-kurihara@ist.osaka-u.ac.jp

DOI: 10.1587/transcom.2018EIP0004

to evaluate how the space-filling curve reduces the sizes in a quantitative way. Second, we carefully re-design a hierarchical PATRICIA data structure and a procedure of updating it especially when location prefix aggregation is used, whereas how it is updated is not focused on in our previous work. Third, we precisely design a data retrieval mechanism from multiple destinations by considering how data pieces are replied, whereas our previous work only focuses on how a request is delivered to multiple destinations. Finally, we validate usefulness of resizing areas of interest, which is an advantage of Z-order, by evaluating performance of a car-based application.

The rest of the paper is organized as follows: First, Sect. 2 describes the design rationale and then Sect. 3 describes the architecture. We design the data structure based on PATRICIA in Sect. 4. Section 5 analyzes scalability of the proposed mechanism. Section 6 describes the related work and Sect. 7 concludes the paper.

2. Design Rationale of Naming Scheme

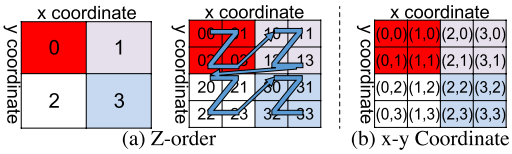
2.1 Requirements to Applications

Location-based forwarding is designed to support on-demand retrieval of location data obtained by multiple IoT devices in a specified location. A *location* is defined as an area of the specified size. Hereafter, we interchangeably use an area and a location. A motivation of on-demand retrieval rather than cloud-based upload [13] is explained by the following use case: An example use case is that a police department asks vehicles which moved around the place of a traffic accident one minute before to upload their dashcam videos. It is assumed that a witness of the accident let the police know the traffic accident via an emergency call. On the contrary, if the cloud-based upload is used, all the vehicles other than those of the police's interest should upload periodically all recorded dashcam videos or their metadata to the cloud [14]. We believe that this periodical update is more redundant than on-demand retrieval which this paper proposes.

Requirements to realize on-demand use cases are summarized as below: First, location-based forwarding must be scalable enough to support a huge number of IoT devices. Second, it must be able to specify arbitrary size of areas. Collectors are interested in data obtained by IoT devices in areas of various sizes. The size of an area depends on use cases. For example, detecting congestion in a road ahead of 2 km requires to collect location data in a small area like a 100 m square, whereas environmental monitoring requires to collect location data in a large area like a 1 km square. Third, multi-source retrieval, which is not natively provided by NDN, should be provided because multiple IoT devices exist in a specified area.

2.2 Design Rationale

A key issue about the naming scheme is which coordinate system is used between a space-filling curve and an x-y coordinate system, both of which are illustrated in Fig. 1, to represent a location. This section discusses them from the perspectives of the first and the second requirements. Assuming that an area is divided to the smallest squares called *cells*, most studies use an x-y coordinate system and identify each cell with two-dimensional numbers, i.e., x-y coordinates. In contrast, Z-order [12], which is a space-filling curve, identifies each cell with one-dimensional numbers, i.e., Z-order numbers. Z-order recursively divides the largest square into smaller ones and each digit represents one of divided squares. For example, *31* means one of sixteen squares divided twice from the largest square. We choose Z-order, as described below.



(a) Z-order (b) x-y Coordinate

Fig. 1 Coordinate systems.

We discuss how to resize an area of an x-y coordinate system. Military Grid Reference System (MGRS), a popular x-y coordinate system, prepares different x-y coordinate systems for different cell sizes. The number of digits is used for deciding cell sizes. For example, the numbers of digits of 1 m, 10 m and 100 m squares are five, four and three, respectively. This raises a couple of problems for compact forwarding data structures. First, forwarding information bases (FIBs) are provided individually for all the coordinate systems. Second, forwarding based on x-y coordinates is regarded as packet filtering with two fields. A size of a FIB is large such that it is proportional to multiplication of the numbers of values of two fields [10].

On the contrary, Z-order naturally supports resizing an area. The location of a square is identified by using a quaternary number and its size is specified by using its number of digits. For example, a square size with the certain number of digits is four times smaller than that with the one less number of digits. This feature solves the above two problems. First, different sizes of areas are represented by changing the number of digits of Z-order number. A single FIB is enough for supporting various sizes of areas. Second, forwarding based on Z-order is prefix matching rather than packet filtering. This contributes to reducing FIB sizes.

3. Architecture

3.1 System Model

The system model is illustrated in Fig. 2. A *collector* retrieves location data of IoT devices in a specified area. Location-based forwarding is used by the collector to send requests to such IoT devices. In Fig. 2, cameras and dashcam devices of cars are IoT devices and the car at the rightmost side is a collector. These IoT devices provide image data

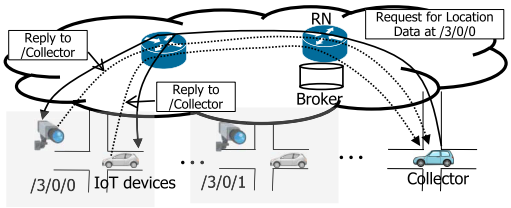


Fig. 2 System model.

pieces of high and low quality. We use an NDN network as an underlying network because name-based routing/forwarding avoids round-trip delays incurred by name resolution in IP networks. Location-based forwarding is responsible for delivering a packet to IoT devices to request location data, assuming that location data pieces are sent back to the collector by using COPSS as described later in the next subsection.

3.1.1 Multicast Communication

Two types of solutions are proposed to support multicasting in NDN networks: COPSS [15], which is a core-based multicast protocol based on publish/subscribe communication, and extension of NDN unicasting [3], [16]. We adopt COPSS because the extension of NDN unicasting incurs complicated procedures. A PIT entry is created when an Interest packet is received and then deleted when the corresponding Data packet is received. In the case of multicasting, it is difficult for a router to decide when the PIT entry is deleted because other Data packets may be received in the future. On the contrary, COPSS naturally provides multicast communication from a publisher to multiple subscribers. However, since the multicast communication is one way, how multiple IoT devices send replies to a collector is an important issue.

There are two design choices: extending COPSS and using it as it is. The first choice is extending COPSS so that it supports request-response communication; however, this extension is as difficult as the extension of NDN unicasting communication to multicasing one, which is described in the preceding paragraph. Thus we choose the second choice. The advantages of using COPSS are summarized below: First, one way communication of COPSS does not incur any extra packet transmission other than a publish packet with a location data piece. Second, COPSS supports collectors' mobility by leveraging a *broker* function [15] which records publish packets previously sent to a root router of a multicast tree called a *Rendezvous Node (RN)*. This naturally recovers publish packet losses caused by mobility of collectors. The broker enables it for the collector to receive such publish packets after when it moves to another place.

Multiple replies from IoT devices are sent by using COPSS as illustrated in Fig. 2. In order to receive replies from IoT devices, a collector subscribes to its name like `/Collector` before sending a publish packet of requesting location data pieces. When a collector sends a publish packet, it is delivered to all the IoT devices in a specified square. The publish packet is used to request location data to IoT devices, and the IoT devices receiving the publish packet send back publish packets of containing location data pieces to the collector's name. Here, the collector's name is conveyed as user data by the publish packet of requesting location data pieces.

3.2 Naming Scheme

A location data piece consists of its location, data type, and data piece. A name of location data called a *location data name* is a human readable hierarchical name and consists of a *location name* and a *data name*, which are delimited by the reserved word `#dat`, e.g., `/3/0/0/#dat/img/high`. A location name is specified according to Z-order [12]. A Z-order number is a quaternary number, $z_1 z_2 \ldots z_m$, where z_i is the i-th digit. Z-order numbers represent squares which are recursively divided from the largest square. As illustrated in Fig. 1(a), the most significant digit z_1 corresponds to one of the four squares, which are divided from the largest square and numbered by 0, 1, 2 and 3. Each of the four squares is divided into four squares and they are numbered from $z_1 0$ to $z_1 3$. Smaller squares are recursively numbered in the same way. The n-th digit corresponds to squares divided n times from the largest square. The quaternary number is naturally mapped onto an NDN-like hierarchical name $/z_1/z_2/\ldots/z_m$.

3.3 Location-Based Forwarding

3.3.1 Overview

We design a location-based forwarding mechanism by leveraging COPSS [15]. A collector and IoT devices rendezvous at the RN which works as a rendezvous point. An IoT device subscribes to a location data name by sending a *subscribe packet* to the RN, of which name is `/RN`. Each router records a received location data name in its subscription table (ST), which is similar to a multicast forwarding table of IP multicast. An entry of an ST is a pair of a location data name and a list of face IDs as illustrated in Fig. 3. A multicast tree for each location data name is constructed from the RN to all IoT devices. Hereafter, we call location data names, location names and data names in STs *location data prefixes*, *location prefixes* and *data prefixes*, respectively.

3.3.2 Design Principles

This section describes the principles of extending COPSS.

a) Prefix Search Rule: An important feature of location-based forwarding is that a publish packet is delivered to IoT devices in a square of which size is determined by the digit number of a location name. That is, a packet to a location name is delivered to IoT devices which subscribe to the location name as well as those which subscribe to longer location names. For example, a publish packet to `/3/0` is delivered to all IoT devices of which location names are `/3/0/0`, `/3/0/1`, `/3/0/2` and `/3/0/3` in Fig. 3. Please note

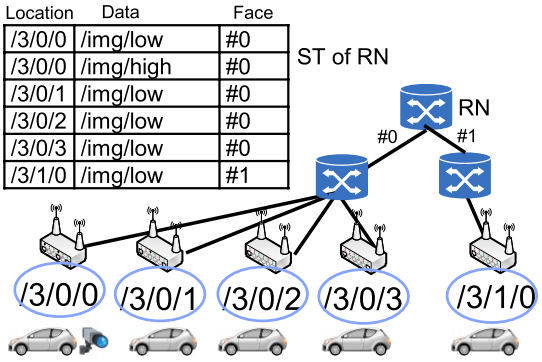


Fig. 3 Subscription table.

that location names of IoT devices are recorded as location prefixes in an ST and thus a location name of a publish packet is equal to or shorter than location prefixes matched in an ST. A location name is a *prefix* of location prefixes in an ST, although it felt that this naming scheme is strange. We design a rule suitable to the above prefix search for location names as well as data names.

b) PATRICIA-based Data Structure: A publish packet is only forwarded to a corresponding face when both a location name and a data name match a location prefix and a data prefix in an entry of an ST, respectively. Compactness of the data structure of STs is an important requirement. We leverage a hierarchical PATRICIA tree [10] to support frequent location and data prefix changes caused by joins and leaves of IoT devices rather than Bloom filters used by naive COPSS [15]. Hereafter, we refer to a PATRICIA tree just as a PATRICIA.

c) Location Prefix Aggregation: Although a PATRICIA naturally reduces the number of vertices in a tree by making location prefixes to share a common sequence of ancestral vertices, it does not fully exploit a hierarchical structure of Z-order numbers. We design a location prefix aggregation rule to reduce the number of location prefixes rather than vertices. Location prefixes are aggregated to a single common location prefix if the following two conditions are satisfied: First, the least significant digits of the location names/prefixes cover 0 to 3. Second, all the data names are the same for these four location names. For example, four location names `/3/0/0`, `/3/0/1`, `/3/0/2`, and `/3/0/3` have the same data name `/img/low` in Fig. 3, if location data name `/img/high` were not subscribed. In this case, the four location data names would be aggregated to `/3/0/#dat/img/low`. Practically, data name `/img/high` prevents the second condition from being satisfied and thus the four location data names cannot be aggregated. To circumvent this, we relax the second condition in the next section.

3.3.3 Relaxation of Location Prefix Aggregation

The second condition of location prefix aggregation is relaxed so that location prefix aggregation works for many four tuples of location data prefixes. The relaxed condition is that all the data name components of the predefined number from the first one are the same. This relaxation comes from the observation that such preceding data name components are enough to decide a face for upstream routers like an RN. At an upstream router RN, data name component `/img` decides face #0 irrespective of the second data name components, i.e., `high` and `low`. Obviously, a wrong choice of the predefined number causes erroneous packet forwarding; however, our previous work shows that such unnecessary packet forwarding is negligible [11].

4. Design of Subscription Table

To realize location-based forwarding, routers have to store pairs of a location data prefix and a list of faces associated with the prefix and store the information in their STs. On receiving a publish packet, routers look up their STs and obtain lists of faces associated with the name in the received publish packet. This section describes the design of an ST which fully supports location data prefixes and extends it so that location prefixes aggregation is supported. Precisely, an ST works as a routing information base (RIB). It is used to record all the location data names and is stored at a slow memory device like a DRAM device. On the contrary, a FIB is created from the ST/RIB by the location prefix aggregation. It is used to forward publish packets and is stored at a fast memory device. Hereafter, we refer to a RIB and a FIB as an ST and a FIB, respectively.

4.1 Definition of Subscription Table

This subsection defines the content of an ST and the prefix searching rule of an ST and a FIB.

4.1.1 Content of Subscription Table

Each entry i in an ST holds a pair of a location data prefix P_i and a list of requesting faces F_i. A location data prefix P_i consists of a location prefix L_i and a data prefix D_i by concatenating them with the reserved word, `#dat`, and it is denoted by $P_i = /L_i/\texttt{\#dat}/D_i$. L_i and D_i can be expressed as sequences of Z-order digits $L_i = \{z_1^i, \ldots, z_m^i\}$ and those of data name components $D_i = \{c_1^i, \ldots, c_n^i\}$, respectively.

4.1.2 Prefix Search Rule

When a router receives a publish packet of which name consists of L_p and D_p, the router searches its ST for entries e whose location prefix L_e and data prefix D_e satisfy the following conditions, where p and e stand for publish packet and entry, respectively:

$$E = \left\{ e \mid z_1^e = z_1^p \wedge L_p \subseteq L_e \wedge c_1^e = c_1^p \wedge D_p \subseteq D_e \right\}. \quad (1)$$

Note that the subset sign in this context represents subsequences. The router then forwards the publish packet to

faces recorded in $e \in E$. In other words, location data prefixes are matched if both location and data names L_p and D_p in the publish packet are one of the prefixes of L_e and D_e in e. Logical expressions $L_p \subseteq L_e$ and $z_1^e = z_1^p$ mean that location name L_p is a prefix of location prefix L_e. The sequence of digits from the first digit of prefix L_e is the same as location name L_p, where the number of digits is that of location name L_p. For instance, location name `/3/0` of a publish packet and location prefix `/3/0/1` in an ST satisfy the above logical expression. When a publish packet of which name is `/3/0/#dat/im/high` arrives, an entry that has the location data prefix like `/3/0/1/#dat/img/high` is matched.

4.2 Design of Data Structure

4.2.1 PATRICIA

We use a *PATRICIA*, which is a tree to store a set of strings, as a fundamental data structure to store pairs of $\{P_i, F_i\}$. STs must satisfy the following two requirements: The first requirement is that they must be able to store a huge number of entries in a space-efficient manner. The second one is that they must be able to deal with frequent changes in entries, such as insertion and removal of entries caused by installation, removal and movement of IoT devices.

There are several alternatives to realize an ST: First, Chen et al. [15] use Bloom filters to realize an ST. Although Bloom filters can store many entries in a space-efficient manner, the entries cannot be removed. The second alternative is geometric algorithms for packet classification [10], which maps classification rules to a n-dimensional hyper-rectangle. Though this approach satisfies the above-mentioned two requirements, it requires both location and data prefixes to be interpreted onto intervals on number lines. Hash functions can map prefixes onto numbers, however they cannot be applied since they generally do not preserve hierarchical relations among components of prefixes. In contrast, a PATRICIA supports space-efficient accommodation of many entries as well as insertion and removal of entries. A PATRICIA is component-based and its vertices are labeled with components rather than characters, whereas a character-based PATRICIA is used in our previous work.

4.2.2 Hierarchical PATRICIA

To realize location-based forwarding, routers forward publish packets to faces that match both location and data names of the packet. Assuming that the number of data prefixes is much smaller than that of location prefixes, we develop a *hierarchical PATRICIA*, which holds location prefixes at the first-level PATRICIA and data prefixes at the second-level PATRICIA. Hereafter, we refer to PATRICIAs storing location prefixes and data prefixes as *location PATRICIA* and *data PATRICIA*, respectively.

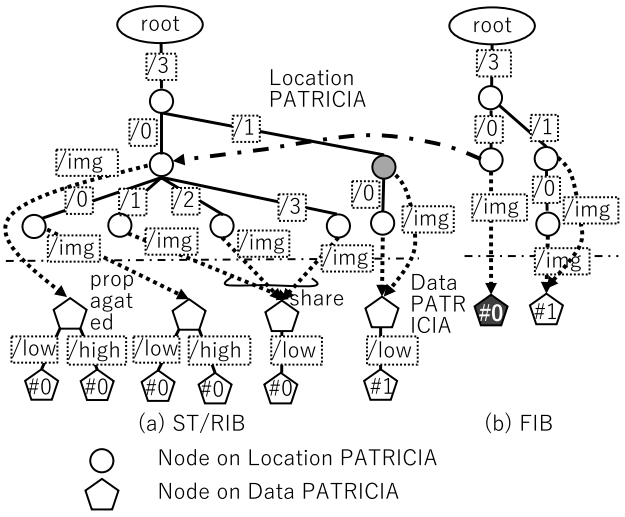


Fig. 4 Examples of RIB and FIB.

4.2.3 Meshed PATRICIA

Our prefix search rule in Eq. (1) requires traversing all descendant vertices on the hierarchical PATRICIA. To avoid such traverses and accelerate searches on the ST, data PATRICIAs and face IDs are propagated to ancestor vertices. In the case that the published location name is $\{z_1^s, \ldots, z_m^s\}$, all vertices associated with $\{z_1^s, \ldots, z_m^s, \ldots\}$ and data PATRICIAs under such vertices have to be traversed. Propagating data PATRICIAs and face IDs to ancestor vertices enables to stop the search when a vertex of $\{z_1^s, \ldots, z_m^s\}$ is found. Searches on the ST is, hence, accelerated. How to propagate data PATRICIAs is explained later in Sect. 4.3.

Figure 4(a) illustrates an example of an ST after an RN receives six subscribe packets illustrated in Fig. 3. Circles and pentagons represent vertices on the location and data PATRICIAs. Face IDs are denoted by numbers with a number sign, e.g., #1. Solid and dashed lines represent edges on PATRICIAs and pointers from the location to data PATRICIAs, respectively. Texts in a rectangle on edges are the labels of the edges.

4.3 Data Structure

An ST is denoted by $\mathcal{S} = \{\mathcal{L}, \{\mathcal{D}_1, \ldots, \mathcal{D}_n\}\}$, where $\mathcal{L}$ and $\mathcal{D}_n$ are the location and data PATRICIAs. Vertices corresponding to sequences of $\{z_1^i, \ldots, z_m^i\}$ and $\{c_1^i, \ldots, c_n^i\}$ are denoted by $v(z_m^i)$ and $v(c_n^i)$, respectively.

4.3.1 ST Update

Every time a router receives a subscribe packet s, of which location and data names are $L_s = \{z_1^s, \ldots, z_m^s\}$ and $D_s = \{c_1^s, \ldots, c_n^s\}$ and incoming face is f_s, the router updates its ST in the following three steps:

1) Creating Vertices on the Location PATRICIA: The router searches for a vertex $v(z_m^s)$ by traversing the location

PATRICIA $\mathcal{L}$ from its root vertex. If $\mathcal{L}$ does not have $v(z_m^s)$, i.e., the algorithm reaches a leaf $v(z_i^s)$ before it finds $v(z_m^s)$ $(i < m)$, missing vertices between $v(z_i^s)$ and $v(z_m^s)$, including $v(z_m^s)$, are created on $\mathcal{L}$. For example in Fig. 4(a), vertices corresponding to the location prefix `/3`, `/3/1` and `/3/1/0` are created when a subscribe packet of which name L_s is `/3/1/0` is received. Note that a vertex that is the only child, like a light gray colored vertex in Fig. 4(a), can be merged with its parent in a naive PATRICIA. Such vertices are not merged in our location PATRICIAs since a publish packet of which location name is `/3/1/0` must be also forwarded to faces associated with `/3/1` according to our prefix search rule.

2) Creating a Data PATRICIA: If a data PATRICIA $\mathcal{D}_s$, where D_s is stored, does not exist, it is created and connected to $v(z_m^s)$ on $\mathcal{L}$. Next, $v(c_n^s)$ and missing vertices are created on $\mathcal{D}_s$. Note that the vertices on $\mathcal{L}$, such as $v(z_m^s)$, also act as the root vertex of $\mathcal{D}_s$. Finally, face ID f_s is stored at $v(c_n^s)$. If multiple location prefixes have the same data PATRICIA with the same face IDs, the data PATRICIA is linked from the most descendent vertices and shared by them. For example, the three location prefixes `/3/0/1`, `/3/0/2` and `/3/0/3` share the same data PATRICIA.

3) Propagating Data PATRICIAs and Face IDs: A router searches the hierarchical PATRICIA for descendent vertices of the matched location prefix and data PATRICIAs linked from them to support the prefix search rule of location prefixes. For example, when a publish packet of location data name `/3/1/#dat/img/low` is received, the router searches the both vertices on the location PATRICIA like `/3/1` and its child `/3/1/0` for the data PATRICIA which records data name `/img/low`. To avoid such time-consuming search, data PATRICAs linked from any vertex on the location PATRICIA are copied to ancestor vertices.

Precisely, the router recursively iterates the following procedure from the most descendent vertex to ancestor vertices: If the parent vertex, i.e., vertex `/3/1`, has only the child vertex, i.e., `/3/1/0`, the edge to the root of the data PATRICIA is added to the parent vertex. We refer to such edge establishment as propagation of a data PATRICIA. On the contrary, if the parent vertex has multiple child vertices, the router creates a new data PATRICIA from data PATRICIAs which are linked from the child vertices. In Fig. 4(a), the four data PATRICIAs, three of which are shared, are merged and created as a single data PATRICIA. In Fig. 4, the leftmost data PATRICIA is created one according to this procedure. Then the data PATRICIA is linked from the parent vertex on the location PATRICIA, e.g., `/3/0`.

This propagation enables to skip traversing all descendant vertices on the location PATRICIA.

4) Removing Vertices on hierarchical PATRICIA: Since a soft-state approach is used, a router removes a location data name after it is expired according to the ordinary remove procedure of a PATRICIA. In addition to the procedure, the router removes vertices and pointers which are created for propagating data PATRICIAs. First, the router identifies the data PATRICIA which is linked from the most descendent vertex on the location PATRICIA. Second, the router removes propagated data PATRICIAs from the ancestor vertices of the identified vertex on the location PATRICIA. It removes the data name of the expired location data name from the data PATICIAs linked from them. If a data PATRICIA is shared by the identified vertex and ancestor vertices, only the edges from the ancestor vertices are removed. For example, when location data name `/3/0/0/#dat/img/high` is expired, the router removes data name `/img/high` from the data PATRICA linked from the vertex of `/3/0`. On the contrary, when location data name `/3/1/0/#dat/img/low` is expired, only the edge from the vertex `/3/1` is removed.

4.3.2 FIB Update

A FIB is a hierarchical PARTICIA created by applying the location prefix aggregation to a hierarchical PATRICIA of an ST. This subsection describes how a FIB is created, assuming that the predefined number of data name components is one. The FIB in Fig. 4(b) is created from the ST in Fig. 4(a) in the following way: Since the vertices and the edges of the FIB are the same as those of the ST except for those removed by the location prefix aggregation, a router does the same procedure when a subscribe packet is received. After inserting its location data name, the router searches the location PATRICIA in the FIB for candidates of four vertices from the most descendent vertex of the inserted location data name. Here, location data name `/3/0/0/#dat/img/low` is assumed to be inserted. The four child vertices of the vertex `/3/0` are the candidates and thus the router compares the first data name components of the data PATRICIAs in the FIB linked from the four vertices and face IDs of them. If the both comparisons are true, these four vertices can be aggregated. In this example, the router identifies the data prefix `/img` as the common data prefix and creates it. The router creates the data PATRICIA for the four prefixes, i.e., the dark gray colored pentagon with the face ID `#0`, adds the edge from the aggregated vertex, i.e., `/3/0`, to the pentagon and removes the descendent vertices from the vertex.

The router adds the backup edge from the aggregated vertex in the FIB to the vertex of the same location prefix, i.e., `/3/0`, in the ST for a future location data name expiration. If some location data names of aggregated location data prefixes are expired, the location prefix aggregation becomes invalid. In this case, the vertices and the edges which were removed from the FIB are copied back from the ST to the FIB using the backup edge between the above vertices.

5. Evaluation

We conduct two kinds of evaluation: One is to evaluate the efficiency of the proposed naming scheme and the reduction in the size of FIBs which are stored in a fast memory device. The other is to evaluate how the functionality of resizing areas of interest reduces the number of publish packets which request location data using an example of dashcam video retrieval aplication. In the rest of section, we refer

to a hierarchical PACTRICIA in a FIB just as a hierarchical PATRICIA.

5.1 Application Scenario

This evaluation focuses on *dashcam video retrieval* as a typical application of the proposed architecture. Dashcam videos are useful not only as irrefutable evidences of traffic accidents but also as witnesses to accidents of others, and hence collecting dashcam videos has attracted attention [14]. One way to collect necessary videos is to upload all videos or metadata about all videos to a cloud service regardless of whether they will be accessed and to search for the necessary videos among the uploaded videos [14] but it obviously incurs heavy overhead in terms of the number of messages for uploading videos. In contrast, location-based forwarding allows to collect necessary videos in an on-demand manner, and hence it is more efficient in terms of the number of messages than approaches relying on cloud services. The dashcam video retrieval application based on location-based forwarding is referred to as the *location-based dashcam retrieval* application, hereafter.

A typical scenario of the location-based dashcam video retrieval application is summarized as follows: A witness of a traffic accident reports it to the police. Then, the police ask vehicles of which dashcams were potentially recording the accident by sending publish packets to the candidate area centered at the accident point, in which such vehicles are likely to be. The naming scheme of Z-order enables it for the police to send the small number of publish packets by leveraging the functionality of resizing areas of interest. In the rest of the section, we assume that the police send publish packets in the following way: First, assuming that the average speed of the cars in the city is known, the police determines the candidate area as the circle described below: Its center is the location of the accident and the radius is the multiplication of the elapsed time from its occurrence and the average speed. Second, the police recursively choose squares each of which is specified by a location name. In the first step, it finds the largest square which is included in the circle. In the following steps, it finds the next largest square which is included in the circle from which the already chosen squares are removed. This continues until the chosen squares cover the circle. Finally, the police send publish packets to the chosen squares and their number is the number of publish packets.

5.2 Simulation Condition

We conduct our simulation under a realistic scenario, which is based on a measurement result of vehicular mobility in the Luxembourg city [17] and cellular base station deployment information provided by the Grand Duchy of Luxembourg [18]. For notational simplicity, we refer to the Luxembourg city as Luxembourg, hereafter. The following subsections explain the simulation condition.

a) Simulation Area: The target area is Luxembourg, which is in a square area of 12 km width. The area is recursively subdivided into four squares eight times. That is, the width of the smallest squares is about 40 m.

b) Router-level Topology: We construct a router-level topology, which is a tree of depth 3, on the basis of the locations of large cellular base stations of which transmission power is higher than 50 W [18], assuming that routers are also deployed with such large cellular base stations. A router closest to the Luxembourg central station, which is the approximate center of Luxembourg, is selected as the first-level router. Four second-level routers and thirteen third-level routers are placed so that Luxembourg is uniformly covered with the routers and they are connected with the closest upstream routers. Finally, fourth-level routers are placed at all the cellular base stations and they are connected with the closest third-level routers. 73 base stations are deployed to cover the whole city and the radius of base stations' coverage areas is 150 meters.

c) IoT Devices and Data Names: In the first evaluation, 1 million IoT devices, each of which accommodates several data types, are deployed uniformly at random in the simulation area. For example, a car has many IoT devices such as a dashcam, an accelerometer, a thermometer and so on. We assume that the first data name componet specifies the data type like `/img` in the data name `/img/low` and routers choose one location name component as the predefined threshold of the location prefix aggregation. How the location prefix aggregation reduces size of hierarchical PATRICIAs depends on distribution of data types specified by the first data name components. Location prefixes are more likely to be aggregated if individual IoT devices support the same set of data types. Thus, we evaluate the reduction in the sizes of hierarchical PATRICIAs by changing the average number of data types which each IoT device has. More precisely, assuming that n data types are supported by IoT devices in the simulation area, we define a set of the probabilities, $\{p_1, \ldots, p_n\}$, each of which is the probability that each IoT device has data type i. We consider two cases of deployment of data types: a uniformly distributed case and a Zipf-distributed case. In the uniformly distributed case, equally popular data types are deployed, and hence all the probabilities p_i are set to an identical value. In the Zipf-distributed case, we set p_i according to a Zipf distribution, assuming that some of the data types are popular in the same way as content popularity in the Internet [19]. We assume that the most popular data types are supported by all the IoT devices, and thus p_i is normalized so that that of the most popular service is 1. Finally, 5 data types are supported in both of the cases.

In the second evaluation, 10 thousand cars are deployed uniformly at random on the roads in Luxembourg, which are obtained from the measurement result of Codecá et al. [17]. In order to simulate the mobility of the cars, we use a traffic simulator, SUMO (Simulation of Urban MObility) [20]. Each car selects a point on the roads uniformly at random as its destination and chooses a route from its deployed point to the destination so that it avoids traffic jams.

Table 1 The size of a FIB.

Router	Z-order system							x-y coordinate system
	Without location prefix aggregation	Uniformly distributed case (p_i)			Zipf-distributed case (α)			
		0.1	0.5	1.0	0.8	1.2	1.6	
First-level router	87,380	86,774	39,572	39,119	39,635	44,132	56,477	87,904
Second-level router	22,130	21,981	10,415	10,266	10,431	11,559	14,596	22,355
Third-level router	6,962	6,921	3,554	3,489	3,553	3,892	4,780	7,112
Fourth-level router	1,291	1,291	1,291	1,291	1,291	1,291	1,291	1,374

d) Comparison: For comparison, we use location-based forwarding with a general x-y coordinate system. The naming scheme and the ST for the x-y coordinate system are designed so that collectors can specify arbitrary size of areas and their locations. Since the ST for the x-y coordinate system works as a FIB, we refer to this ST as the FIB, hereafter. A location name for the x-y coordinate system consists of 3 components, i.e., a granularity number of area size, x and y coordinate numbers. In the same way as the Z-order system, the target area is recursively divided into small squares. A granularity number of area size indicates the number of subdivisions of the target area and x and y coordinate numbers specify a square among the subdivided squares. For instance, a location name `/2/0/1` for the x-y coordinate system indicates one of sixteen squares divided twice from the largest square and it is identical to `/0/2` in the case of the Z-order system.

To build a FIB for this naming scheme, we use a PATRICIA that holds granularity numbers at the first level edges, x coordinate numbers at the second level edges and y coordinate numbers at the third-level edges. Let us note that data names for the x-y coordinate system are identical to those for the Z-order system, and hence their data structures for data prefixes are also identical.

5.3 Size of FIB

In order to investigate the efficiency of the proposed naming scheme and the location prefix aggregation scheme, we compare the Z-order with that the x-y coordinate systems by using the size of their FIBs as a metric of scalability. The size of a FIB is measured with the number of vertices in the Location PATRICIA in a FIB. Since the predefined number is one data name component and the number of data types is just 5, data PATRICIAs are stored in the most descendent vertices on location PARICIAs. Hereafter, we call the number of vertices on a location PATRICIA in a FIB as the size of a FIB.

Figure 5 shows the average sizes of FIBs of the Z-order system in the uniformly distributed case for the 4 level routers. The probability of the horizontal axis is defined as described below: The probability that an IoT device supports a data type is defined for all the IoT devices. The probability of 1, for instance, means that all IoT devices support each data type with probability 1. Thus all the five data types are supported by all the devices. How the location prefix aggregation scheme reduces sizes of FIBs depends on the probability. The location prefix aggregation scheme has a larger effect of reduction in the FIB size if the probability is high. Higher the probability becomes, the smaller the average size of FIBs becomes. This is interpreted that if the number of devices which support all the data types become large, location data prefixes received from the same face tend to have all the data types. It increases the probability that these location data prefixes are aggregated. On the contrary, the reduction effect decays in the case that data types are sparsely deployed due to reduction of the probability. Nevertheless, the location prefix aggregation is able to reduce the FIB size except for the case where data names are extremely sparsely deployed, for instance the case where the probability is less than 0.3.

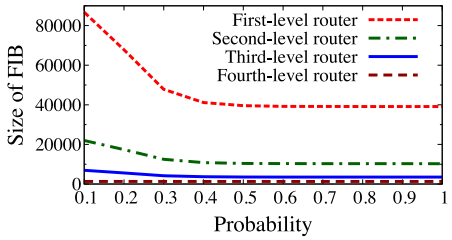


Fig. 5 The size of a FIB and the probability that IoT devices have each data type in the uniformly distributed case.

Table 1 summarizes more detailed results of the FIB size. We obtain the following two observations. First, the location prefix aggregation scheme reduces the FIB size unless some of data types are sparsely deployed. As we have discussed with the results in Fig. 5, the location prefix aggregation scheme reduces the FIB size if data names are uniformly and not sparsely deployed. In the case that the Zipf-distributed case and $\alpha = 1.6$, the probability p_i of the least popular data name is 0.17, and such sparsely deployed data names disturb the location prefix aggregation. However, the location prefix aggregation scheme also reduces the FIB size in the Zipf-distributed case unless some of data names are sparsely deployed.

Second, we observe that the naming scheme and the location prefix aggregation scheme contribute to realizing compact FIBs by comping the results of the Z-order system and those of the x-y coordinate system. Since location prefixes cannot be aggregated in the case of the x-y coordinate system, the FIB size of the x-y coordinate system is much larger than that of the Z-order system.

5.4 Applicability of Location-Based Forwarding

This subsection evaluates how the functionality of resiz-

Table 2 The elapsed time from an accident, the average moving distance of cars, the number of publish packets and the number of unicast packets.

Elapsed time from an accident (s)	Average moving distance (km)	Number of publish packets	Number of unicast packets
30	0.28	22.72	76.27
60	0.53	51.16	273.42
90	0.81	83.55	638
120	1.00	104.02	971.01

ing areas of interest reduces the number of request packets which are sent by a collector, focusing on the location-based dashcam retrieval application discussed in Sect. 5.1. In this simulation, we randomly select a point on the road of Luxembourg as a traffic accident point and generate a traffic accident. Cars on 100 m ahead and behind the accident points at the accident time are set to witness vehicles. We evaluate the average moving distance of the witness vehicles and the number of publish packets to collect data from the witness vehicles by changing the elapsed time from the accident, assuming that it takes time to report the accident to the police. Table 2 summarizes the average values of the results of 1,000 randomly chosen traffic accident. The table shows the number of packets in the following two cases: First, publish packets are sent by leveraging the functionality of resizing areas of interest, which is described in Sect. 5.1. Second, packets are sent to all the smallest square in a unicast manner. The comparison shows that the functionality of resizing reduces the number of packets requesting location data. For example, the number of publish packets is 8 times smaller than that of unicast packets when the elapsed time is 120 seconds.

The multiple data retrieval based on location-based forwarding enables to collect necessary data with a smaller number of publish packets than those of cloud-based services [14]. In the proposed retrieval, the average number of witness vehicles for one traffic accident is 9.1. In the ideal case, where all the movement of the witness cars can be tracked and the necessary data can be retrieved from each of them, the optimal number of messages to collect data is 9.1. On the contrary, cloud-based services [14] obviously incur a large number of messages to collect the necessary data because all vehicles upload data to a cloud regardless of whether they are witnesses or not. In the case of ViewMap [14], each vehicle uploads a piece of metadata once a minute, and hence it generates 10,000 messages in a minute. In this sense, the proposed architecture is useful for such location-based services.

6. Related Work

The paper is compared with four series of related studies. First, location-based forwarding is studied for MANETs [21] and VANETs [5]. Our location-based forwarding is based on a routing scheme, where routing information is disseminated, whereas that in MANETs and VANETs is based on greedy forwarding. Second, an inherent multicasting nature of location-based forwarding is leveraged to provide location anonymity, i.e., to prevent interests of locations from being leaked [2]–[4]. Multicasting the same packet to multiple IoT devices provides k-anonymity. The paper does not focus on location anonymity because our previous work does so [3], [4]. Third, packet classification is categorized into two types: trie-based algorithms and geometric algorithms [10]. We choose the hierarchical PATRICIA, a type of trie-based trees, because geometric algorithms assume that prefix search rules are defined by a range on a number line but it is difficult to arrange data names of the paper in such an order. Fourth, ViewMap [14] is a location-based dashcam video retrieval application similar to the dashcam application in this paper. Viewmap takes a cloud approach where references to dashcam videos are uploaded periodically from all vehicles, whereas our application takes a peer-to-peer approach where those are retrieved from the vehicles of interest in an on-demand manner.

7. Conclusion

This paper designs architecture for location-based forwarding for on-demand location data retrieval. The key contributions of the paper are summarized as follows: The paper develops a naming scheme to specify location data with its location and its data type and a data structure for forwarding with hierarchical PATRICIA. The paper designs a forwarding mechanism based on the naming scheme and evaluates its performance for an on-demand video retrieval application.

References

[1] G. Gartner and H. Huang, Progress in Location-Based Services 2016, Springer, Sept. 2016.
[2] B. Niu, Q. Li, X. Zhu, G. Cao, and H. Li, "Achieving k-anonymity in privacy-aware location-based services," Proc. IEEE INFOCOM, pp.754–762, April 2014.
[3] K. Ryu, Y. Koizumi, and T. Hasegawa, "Name-based geographical routing/forwarding support for location-based IoT services," Proc. IEEE HotPNS 2016, Nov. 2016.
[4] K. Kita, Y. Kurihara, Y. Koizumi, and T. Hasegawa, "Location privacy protection with a semi-honest anonymizer in information centric networking," Proc. ACM ICN 2018, Sept. 2018.
[5] S.M. Brial, C.J. Brtnardos, and C. Guerrero, "Position-based routing in vehiclular networks: A survey," J. Netw. Comput. Appl., vol.36, no.2, pp.685–697, May 2013.
[6] V. Jacobson, D.K. Smetters, J.D. Thornton, M.F. Plass, N.H. Briggs, and R.L. Braynard, "Networking named content," Proc. ACM CoNEXT, pp.1–12, Dec. 2009.
[7] G. Grassi, D. Pesavento, G. Pau, L. Zhang, and S. Fdida, "Navigo: Interest forwarding by geolocations in vehicular named data networking," Proc. IEEE WoWMoM, pp.1–12, Dec. 2015.
[8] L. Wang, R. Wakukawa, R. Kuntz, R. Vuyyuru, and L. Zhang, "Data naming in vehicle-to-vehicle communications," Proc. IEEE NOMEN, pp.328–333, March 2012.
[9] J. Chen, M. Jahanian, and K.K. Ramakrishnan, "Black ice! using information centric networks for timely vehicular safety information dissemination," Proc. IEEE LANMAN, June 2017.
[10] H.J. Chao and B. Liu, High Performance Switches and Routers, Wiley-IEEE Press, March 2007.
[11] Y. Kurihara, Y. Koizumi, and T. Hasegawa, "Compact data structures for location-based forwarding in NDN networks," Proc. IEEE Workshop on Information Centric Networking Solutions for Real World

Applications (ICN-SRA), May 2018.
[12] G.M. Morton, "A computer oriented geodetic data base; and a new technique in file sequencing," Technical Report, IBM, March 1966.
[13] A. Botta, W. de Donato, V. Persico, and A. Pescap, "Integration of cloud computing and internet of things: A survey," Future Generation Computer Systems, vol.56, pp.684–700, March 2016.
[14] M. Kim, J. Lim, H. Yu, K. Kim, Y. Kim, and S.B. Lee, "ViewMap: Sharing private in-vehicle dashcam videos," Proc. USENIX NSDI 2017, pp.163–176, March 2017.
[15] J. Chen, M. Arumaithurai, L. Jiao, X. Fu, and K. Ramakrishnan, "COPSS: An efficient content oriented publish/subscribe system," Proc. IEEE/ACM ANCS, pp.99–110, Oct. 2011.
[16] M. Amadeo, C. Campolo, and A. Molinaro, "Multi-source data retrieval in IoT via named data networking," Proc. ACM ICN, pp.67–76, 2014.
[17] L. Codecá, R. Frank, S. Faye, and T. Engel, "Luxembourg SUMO traffic (LuST) scenario: Traffic demand evaluation," IEEE Intell. Transp. Syst. Mag., vol.9, no.2, pp.52–63, April 2017.
[18] The National Geopotal of the Grand-duchy of Luxembourg, "geoportal.lu," https://www.geoportail.lu/
[19] C. Fricker, P. Robert, J. Roberts, and N. Sbihi, "Impact of traffic mix on caching performance in a content-centric network," Proc. IEEE NOMEN, pp.310–315, March 2012.
[20] D. Krajzewicz, J. Erdmann, M. Behrisch, and L. Bieker, "Recent development and applications of sumo-simulation of urban mobility," International Journal on Advances in Systems and Measurements, vol.5, no.3&4, pp.128–138, 2012.
[21] F. Cadger, K. Curran, J. Santos, and S. Moffett, "A survey of geographical routing in wireless ad-hoc networks," IEEE Commun. Surveys Tuts, vol.15, no.2, pp.621–653, Jan. 2013.

Yoshiki Kurihara received his Bachelor of Information Science degree from Osaka University, Japan, in 2016. He is currently pursuing the master degree at the Graduate School of Information Science and Technology, Osaka University. His research interests include information centric networking and location-based forwarding. He is a member of IEICE.

Yuki Koizumi is an associate professor of Graduate School of Information Science and Technology, Osaka University, Japan. He received his Master of Information Science and Ph.D. of Information Science degrees from Osaka University, Japan, in 2006 and 2009, respectively. His research interests include information-centric networking and mobile networking. He is a member of IEEE, ACM, and IEICE.

Toru Hasegawa is a professor of Graduate school of Information and Science, Osaka University. He received the B.E., the M.E. and Dr. Informatics degrees in information engineering from Kyoto University, Japan, in 1982, 1984 and 2000, respectively. After receiving the master degree, he worked as a research engineer at KDDI R&D labs (former KDD R&D labs.). His current interests are Information Centric Networking, privacy and so on. He received the Meritorious Award on Radio of ARIB in 2003, the best tutorial paper award in 2014 from IEICE and the best paper award in 2015 from IEICE. He is a fellow of IPSJ and IEICE.

Mayutan Arumaithurai is a senior researcher in the Computer Networks Group at Institute of Computer Science, University of Göttingen. Prior to that, he worked as a research scientist at Network Laboratories of NEC Europe Ltd. in Heidelberg, Germany. He received his doctoral degree from the University of Göttingen in 2010.