**Master's Thesis**

Title

# Mobility-Controlled Flying Routers for Information Centric Networking

Supervisor

Professor Masayuki Murata

Author

Taku Kitagawa

February 6th, 2017

Department of Information Networking

Graduate School of Information Science and Technology

Osaka University

Master's Thesis

# Mobility-Controlled Flying Routers for Information Centric Networking

Taku Kitagawa

## Abstract

Information Centric Networking (ICN) is expected as a novel network architecture in the future. Unlike existing location-oriented architectures represented by IP networks, ICN has a major feature that routing is controlled by a content name rather than a node address. Although ICN has various advantages from its design concept, we especially focus on the fact that ICN can realize various control mechanisms utilizing the high flexibility of name. In the existing works, data control and configuration at nodes are mainly considered, but the scope of this work extends to the physical control of equipment in the current situation where Internet of Things (IoT) gathers much attention from many researchers. In order to explore the feasibility of flexible control over ICN, we consider a physical movement control over ICN relay nodes installed on a drone. We name it "Flying Router (FR)". Then, we propose and design Router-Movable Information Centric Networking (RMICN) as a method to realize communication between disjoint networks, which are difficult to communicate with each other because of their distance, using FRs. In addition, we aim to show advantages of controlling router's movement with ICN. In this paper, we compare RMICN and the same approach based on Delay Tolerant Networking (DTN), which is commonly used as a communication method between disjoint networks, to evaluate the proposed RMICN. As a result, we could show design benefits of controlling router's movement with ICN and shorten content retrieval time in simulation.

## Keywords

ICN (Information Centric Networking)

NDN (Named Data Networking)

in-network processing

disjoint networks

movable router

path planning

# Contents

# List of Figures

## List of Tables

# 1 Introduction

Many years have passed since Internet became widespread, and IP is persistently used as a communication protocol even today. The original purpose using IP was to communicate between hosts such as Telnet and e-mail, so location-oriented communication protocols were considered natural, and therefore they became popular. Over time, however, Network usage patterns are changing from location-oriented style to content-oriented style to focus on contents. This is because attractive services using Internet appeared one after another due to the explosive spread of Internet. On services such as web and video distribution, the users are not interested in with which nodes they are communicating and they are only interested in contents delivered by services (web pages, movies, music, etc.). For this reason, it is considered problematic that the current network usage and the current network architecture are out of alignment in recent years.

Various problems have arisen due to the divergence of the network usage and architecture. One of the most widespread examples is the increase in processing load on servers and relay nodes. In recent years, the amount of traffic in the network is increasing explosively due to the popularity of smartphones and the capacity enlargement of contents, but ordinary IP protocols have limited mechanism for handling these problems. There is a Content Delivery Network (CDN) as a technology for constructing an optimal network for content delivery, but it is unable to deal with traffic of the entire network because it is implemented at the application layer and it takes high cost that a service provider individually deploys it as necessary.

Information Centric Networking (ICN) gathers attention as a novel network infrastructure that efficiently handles enormous contents. ICN is designed as a content-oriented network architecture and it has many advantages over IP because it eliminates the dissociation of the current network usage and the current network architectures. In particular, a control mechanism based on the name of an interesting unit has gained much attention in recent years. Thus, ICN is expected as a communication infrastructure of IoT which needs various device cooperation and configuration because it can handle in-network processing with name. Moreover, not only general data control and signaling in in-network processing but also physical control of devices are taken into consideration in recent years, and then research works related to control with ICN become more exciting.

Physical control of end devices has been considered in previous works, however, physical control of relay nodes is not much considered. In order to explore the feasibility of flexible control over ICN, we consider ”movement” as physical control of relay nodes and we think about Flying Router (FR), which is an ICN router installed on a drone, and its physical movement by ICN control in this research. Then, we propose and design RMICN (Router-Movable Information Centric Networking) as a method to realize communication between disjoint networks, which are difficult to communicate with each other because of their distance, via movement of FRs. In addition, we set our goal to show advantages of controlling router’s movement with ICN.

# 2 Related Work

## 2.1 Information Centric Networking / Named Data Networking

Unlike existing location-oriented architectures such as IP networks, there is a major feature that routing is controlled by content name rather than node address in ICN. Named Data Networking (NDN) and Content Centric Networking (CCN) [1, 2] are well-known as active research works as ICN projects, and we also focus on NDN as ICN (There is little difference in design concepts between NDN and CCN, because their original projects are the same). The characteristics of NDN are showed as follows:

- Request / response type communication is realized based on the name of contents
- Relay nodes in the network layer can cache contents.
- Content itself is encrypted and relay nodes authenticate content.

NDN adopts a hierarchical name structure similar to URI as content name. Then, communication is realized by exchanging a request message with name and a content with the corresponding name. The content request message is referred to as Interest, and content message is referred to as Data. NDN has a mechanism for handling contents efficiently, and the best example is caching contents at relay nodes. Content cannot be identified in the network layer in IP, but it is possible with NDN. Therefore it is possible to reduce load by caching contents in relay nodes and returning them according to other requests. In terms of security, unlike the connection oriented security mechanisms in IP networks such as SSL and IPSec, content itself rather than terminal-to-terminal connection is authenticated and encrypted in ICN. Therefore it can be said that ICN normally has the function to carry contents safely in any communication environment and protocol.

Figure 1 is protocol stacks of IP networking and NDN (cited from the paper [1]). In both cases, the network layer needs to match upper and lower layers, and so realizing a thin-waist stack, e.g., simplified network layer, is cited as the reason for the success of any communication protocols. Following this fact, NDN also constitutes a thin-waist stack like IP networking, and it makes requests through Face which is in the second layer lower than IP (Also interface of IP communication can be used as Face). Besides, a strategy layer is newly added to NDN as a point which is different from the protocol stack of IP networking. The strategy layer in NDN decides how to transfer Interest and Data. In IP networks, messages are transferred according to routing tables constructed by routing protocols, but in NDN, in addition to this routing processing, the strategy layer defines when and where to forward the message and what kind of processing intervenes during forwarding messages.

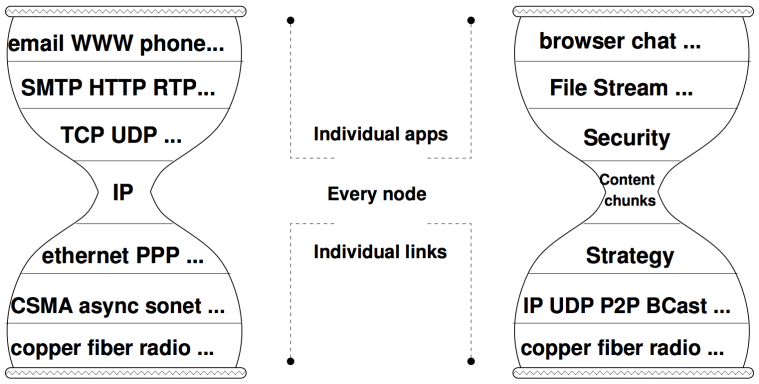


Figure 1: Protocol stacks of IP networking and NDN (cited from the paper [1])

The forwarder of NDN has the following main five tables in order to realize communication using Interest and Data.

**FIB (Forwarding Information Base)**

FIB is a table that determines Face which is the forwarding destination of Interest received from downstream. A node receiving an Interest transfers it to upstream with reference to FIB.

**PIT (Pending Interest Table)**

PIT is a table that stores information of Faces where Interest is received and sent. A node receiving a Data returns it to the requester by referring to PIT.

**CS (Content Store)**

CS is a table that stores caches of Data. A node receiving Interest returns the Data to the requester when the corresponding entry is in CS.

**RIB (Routing Information Base)**

RIB is a table that stores routes. RIB is updated manually or by routing protocols. Entries of FIB are generated from entries of RIB.

**Strategy Choice Table**

Strategy Choice Table is a table that stores strategies corresponding to each content. A node receiving an Interest or a Data applies a strategy to content by referring to Strategy Choice Table.

The names of contents are used for storing and referring to these tables. PIT and CS use complete content names, FIB and RIB, and Strategy Choice Table use the prefix of content names. On the other hand, at the end nodes, in order to realize exchange of Interest and Data, the prefix of the content name and the listener (application) are registered in advance to the face managed by the end nodes. Then, when a message matching the prefix of the registered name arrives, the callback function defined in the listener is called for each type (Interest or Data), so that the application can receive the forwarded message.

## 2.2 Realizing Controllability with ICN

In recent years, there has been research works to utilize names in ICN not only for forwarding but also for in-network processing. We consider the controlled object as two types, end devices and relay nodes. In ICN, forwarding and in-network processing can be performed seamlessly by sending control messages to the control object and the control content with flexible name, and it is possible to bring benefits that are not found in location-oriented communication architectures. For example, considering the case of specifying and controlling a certain device, in a location-oriented communication protocol such as IP, a conversion table which maps from the device name to the device address is required in the application layer. This is a disadvantage because the development of application becomes complicated in spite of the demand for just controlling devices. In the following, we introduce existing researches in ICN regarding controlling objects, and we also show the application scenarios of our proposal.

### 2.2.1 Control of End Devices

As existing works for controlling end devices, there are VoCCN (Voice over CCN) [3], Dash over CCN [4], etc. VoCCN delivers voice calls similar to VoIP over CCN, and its characteristic is that signaling is carried out when establishing a voice call using name. Normally, VoIP carries out communication of signaling data and voice data through separate paths, but VoCCN can make these distinctions by name, and it can limit the number of used paths to one. An example of the signaling name is shown in Table 1. This is an example in which an INVITE message in SIP protocol is encrypted and transmitted to Bob (the other person on the phone).

Table 1: An example of the signaling name in VoCCN

| Name |
|---|
| `/⟨domain⟩/sip/bob/invite/E_pkB(sk)/E_sk(SIP_INVITE_message)` |

On the other hand, Dash over CCN studies streaming delivery of video and audio using CCN. In the features of the approach, setting the quality of contents in the name field of Interest makes it possible to change the settings such as the resolution and the bit rate of the contents of video and audio on every chunk. An example of the name is shown in Table 2. It can be seen that the name is configured in units of segments and the setting of video quality such as resolution is also included. In IP communication, when changing the setting like this, it is necessary to separately transmit an IP packet with a message indicating the setting. It may occur configuration lag. In ICN, however, it is possible to designate content request and its quality at the same time in chunk units, so it can seamlessly retrieve a stream corresponding to the name.

Table 2: An example of the control name in Dash over CCN

| Name |
|---|
| `/DashOverCCN/hfp/www-itec.uni-klu.ac.at/ip/datasets/Mmsys12/BigBuckBunny/bunny_2s/bunny_2s_150kbit/bunny_2s1.m4s/` |

In addition, applicable control in IoT environments has also been studied because IoT gathers attention from many researchers in recent years [5, 6]. Because IoT has many devices connected to the network, it can be said that a content-oriented communication protocol like NDN is more suitable than a location-oriented communication protocol such as IP. The paper [5] considers the application of NDN to the IoT environment, and as a part of it, physical control of the end device in IoT is described. As an example of physical control of the end device, control of lighting in a smart home is cited. In this example, among the lighting fixtures in the living room, control of turning on / off the table lamp is performed. Its name is shown in Table 3.

Table 3: An example of the name for controlling lighting in a smart home

| Name |
|---|
| `/LivingRoom/Lighting/TableLamp/ON` |
| `/LivingRoom/Lighting/TableLamp/OFF` |

In IoT environments, besides, ICN gathers attention not only in data control and signaling but also in physical control such as temperature adjustment of air conditioners, change of angle of surveillance cameras, operation of cleaning robots, and so on.

### 2.2.2 Control of Relay Nodes

As existing works for controlling end devices, there is NFD Management protocol [7] in NFD (NDN Forwarding Daemon) [8] which is the forwarding daemon of NDN. The NFD can control configurable components such as Face, RIB, Strategy Choice Table through its controller, and the NFD Management protocol is used as a protocol supporting communication with various managers using Interest / Data for controlling them. Control Command [9] is defined as a control format, and NFD is controlled by configuring the name of the signed control Interest like Table 4. Control Command designates that it is Interest for NFD control in {`prefix`}, and designates a manager which is control target in {`management-module`}, and designates control name, control parameters and options after that. This makes it possible to control Face, FIB, Strategy Choice Table which are components of NDN by name.

Table 4: The namespace of Control Command in NFD Management Protocol

| Name |
|---|
| `/{prefix}/{management-module}/{command-verb}/{control-parameters}/{command-interest-components}` |

However, there is control of relay nodes as configuration of routers above, but physical control of relay nodes with ICN is not much considered. This is because relay nodes just forward messages and does not have the application layer..Therefore, in this research, we focus on physical control of relay nodes with ICN in order to verify the feasibility of flexible control over ICN, and we aim to control a movable router as concrete physical control. Since ICN is location-free unlike IP, it is thought that ICN has a high affinity to movable routers. Because, movement of routers reduces the effect of routing using a location identifier such as an IP address, while routing using a content name does not depend on location, and so it is not easily affected by movement of a router.

## 2.3 Delay Tolerant Networking (DTN)

By controlling routers' movement targeted in our research, it becomes possible to provide connectivity between disjoint networks that are not connected to each other. This is realized by a router, receiving a message from one network and buffering the message. The physical router moves to the position where the router can connect with the other network, and then transfers the message to the other network. Delay Tolerant Networking (DTN) is different from our networking with the mobility controllable router proposed in this research, but it is cited as one of existing technologies to provide communication between such disjoint networks. There are various types of DTN, but especially among them, Message Ferry [10] and Data Mule [11] are proposed as networking to move relay nodes similar to our research. Message Ferry is a networking system that sends and receives messages between nodes that can not communicate directly because their positions are far apart like the Figure 2. Message Ferry uses relay nodes (ferries) moving on a predetermined route for transporting messages. A relay node receives messages if it can communicate with a node in a cluster and the relay node sends the messages if it holds the message destined for the connected node. On the other hand, Data Mule is a networking system similar to Message Ferry, but the different points are that it is aimed at collecting data accumulated in a fixed sensor and that available routes may change dynamically.

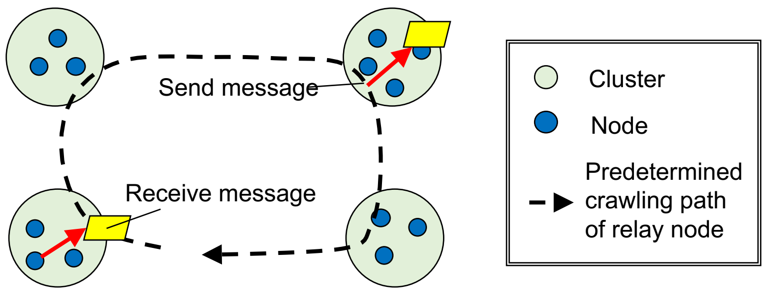


Figure 2: Message Ferry (One of DTN Methods)

As described above, there are several methods for providing connectivity between disjoint networks by moving relay nodes in DTN. As a related research topic, there is an active discussion regarding path planning of movable router: how the movement routes of relay nodes are determined [12–15]. In Message Ferry and Data Mule, however, relay nodes do not determine the traveling route depending on the characteristics of messages and contents, but they carry out location-oriented path planning considering only the position of each node. On the other hand, the networking with movable routers proposed in this paper is essentially different from DTN because routers carry out message-driven mobility control, and furthermore, it can bring content-oriented benefits to communication between decoupled networks by content-oriented mobility control with ICN.

# 3 Router-Movable Information Centric Networking (RMICN)

The purpose of our research is to propose a networking system (RMICN: Router-Movable Information Centric Networking) that carries out controlling movement of routers based on content, and clarifies the benefits of disjoint-networks communication by performing movement control of routers with ICN. This chapter shows use cases to what kind of scenario RMICN can be applied, the design goals of RMICN, and the target environment and the method of constructing a network by movable routers.

## 3.1 Use Cases

In this section, we describe use cases to what kind of scenario RMICN can be applied. RMICN can be basically applied in the case where there are disjoint networks which cannot be connected to each other because their distance is far away. In particular, we will introduce sensor networks and disaster networks as cases suitable for drones which are used for the body of movable router in this paper.

### 3.1.1 Sensor Networks

There are various uses in sensor networks and they are considered as a major factor of IoT. Especially among them, smart agriculture applying sensor networks to agriculture is expected as a use case. Smart agriculture aims at reducing cost and increasing quality of agriculture utilizing robot technology and ICT, and sensor networks in Figure 3 are used for the purpose of collecting information on agricultural products (e.g., temperature, humidity and soil moisture content). However, it is easy to assume that agriculture is deployed in a broad area and a depopulated area with poor communication infrastructures, so it is not easy to make connectivity among sensor nodes. Therefore, introducing movement-controllable routers to provide connectivity among sensor nodes (between disjoint networks) is considered to be useful to this service scenarios which have attracted much attention recently.

### 3.1.2 Disaster Networks

When the function of a communication infrastructure is stopped or destroyed due to a disaster, networks that were originally connected may be disjoint. Therefore, as a use case of movement-

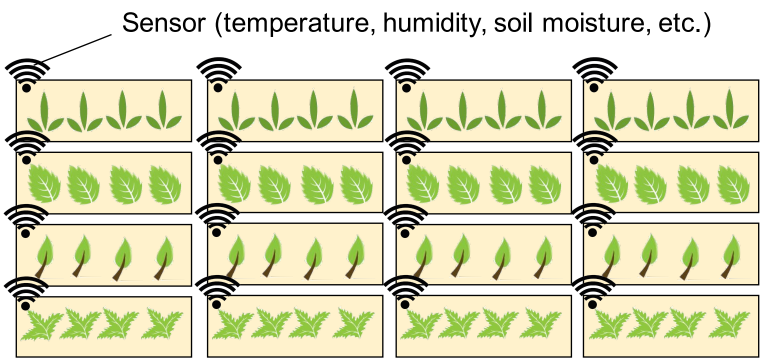


Figure 3: Sensor networks (smart agriculture)

controllable routers, communication between disjoint networks occurring at the time of a disaster is focused on. In the case of a disaster, people may evacuate to a big building, obtain information on disasters, and exchange safety information. movement-controllable routers can provide connectivity between divided points such as evacuation places, municipalities and victims in Figure 4.

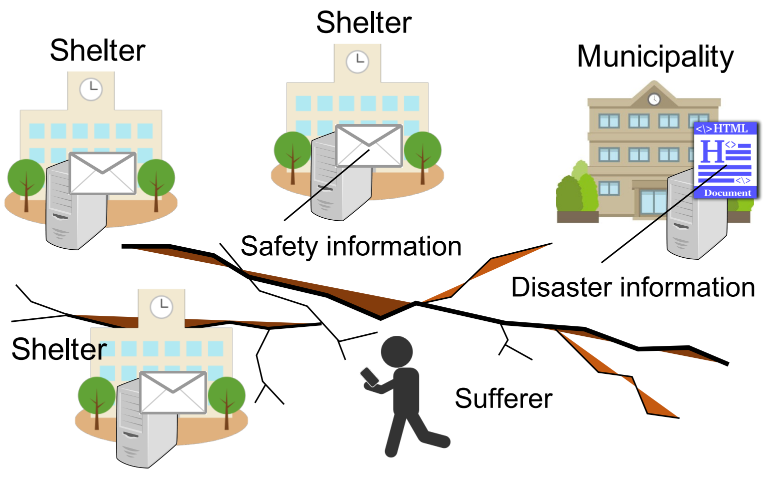


Figure 4: Disaster networks

## 3.2 Design Goals

In this paper, we aim to design RMICN which is a network with movable routers as a seamless expansion of ICN. As a result, RMICN can realize all features that ICN provides, and it can also handle disjoint-networks communication. In addition, since RMICN incorporates ICN for movement control of routers, it is possible to incorporate the content-based features for controlling movement path and communication. In this paper, we focus on the following main three features and leverage them as the design policy of RMICN.

**Control with Name**

RMICN can incorporate flexible control using ICN name into its networking. Movement control of routers is one of them. Besides, RMICN is possible to realize various controls other than movement control using naming schemes that ICN provides. For example, by including the meaning of the highly urgent content to its name, routers can recognize the content highly urgent, and the priority and the movement speed can be improved when delivering it. In addition, if someone wants to retrieve sensor information at a certain arbitrary location, it is possible to seamlessly expand the function to routers without losing its router functions by defining the name and adding software (the function to move to the location and create content of sensing result) and hardware (sensor) to the router. In this paper, we focus on the point that ICN can seamlessly incorporate these practical controls with name into its networking, and we design a foundation to realize these controls.

**Content-Based Parameter**

Since a primitive of ICN communication is content, path planning of movable routers using content-based parameters such as content arrival rate and retrieval time can be performed. It is impossible for location-oriented communication architectures to judge whether a message handled in the network layer represents a request of a content, a content itself or what kind of contents is represented. On the other hand, it is possible for ICN to do so because ICN uses names of contents as route identifiers and it can identify contents in the network layer. Since a communication unit that is important for users is content, it is considered significant to construct a network that, for example, informs users in disjoint networks of content retrieval time and delivers the content within that time. Therefore, we exploit this aspect that RMICN can perform path planning utilizing content-based parameters, set content retrieval time for each content, and design RMICN to deliver content within that time in this paper.

**Content Cache**

It is possible to cache contents in routers in ICN. We mentioned that movable routers provide connectivity between disjoint networks via their mobility, besides, it is possible to reduce needs to move to content providers to obtain contents. It is assumed that a mobile body responsible for router movement normally operates with a mobile battery, and so reducing the movement distance by caches is useful in terms of reducing power consumption. In addition, reducing the movement distance also shortens content retrieval time, and so contents cache is also useful in terms of user experience. In this paper, we focus on this aspect that ICN can cache contents in routers, and we design RMICN to shorten the average time of content retrieval.

These features are the advantageous because RMICN emphasizes on the essence of ICN. As a common point of the advantages of ICN, all the above three features are completed in the network layer. In order to realize these features in a location-based protocol stack such as an IP network, not only network layer but also the application layer need to be implemented, so that independence of each layer, which is the principle of a protocol stack, is unable to be kept. On the other hand, movable routers in RMICN can independently provide these features as functions of themselves, and it is possible to provide a highly applicable network which does not depend on any applications.

## 3.3 Target Environmental Model and Constructing Networks with Movable Routers

RMICN uses NDN as an architecture of ICN and a drone as a mobile body of a router for performing movement control of routers with ICN. Then, a NDN router installed on a drone is configured as a movable router, which is called a Flying Router (FR). RMICN defines a method to provide connectivity using multiple FRs in an environment including disjoint networks, and we show an example of the target scenario in Figure 5.

As shown in Figure 5, RMICN recognizes the target area as a hexagon, and supposes a situation in which networks (disjoint networks) that do not have mutual connectivity exist dispersedly in the area. Each disjoint network is composed of one or more wireless nodes, and one of them always becomes a gateway (GW) and takes charge of communication with FRs. Depot in the figure means a departure point when FRs move, and all FRs in Depot exchange information with each other by relaying a Depot Router (DR) which is a representative node of FRs. The position of Depot can be dynamically and statically set, and if it is set dynamically, the average position of all the disjoint networks is set as the position of Depot.

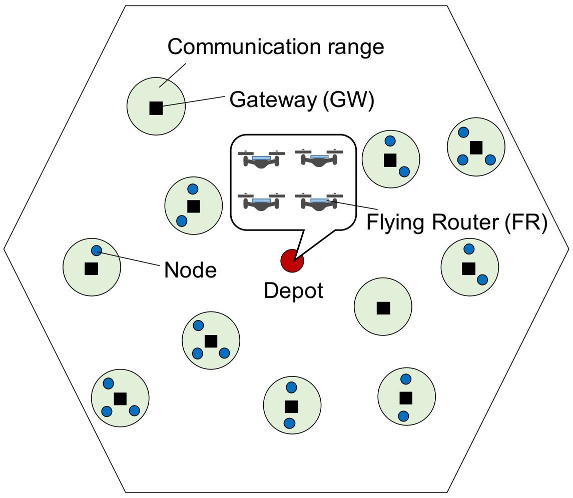


Figure 5: Target scenario

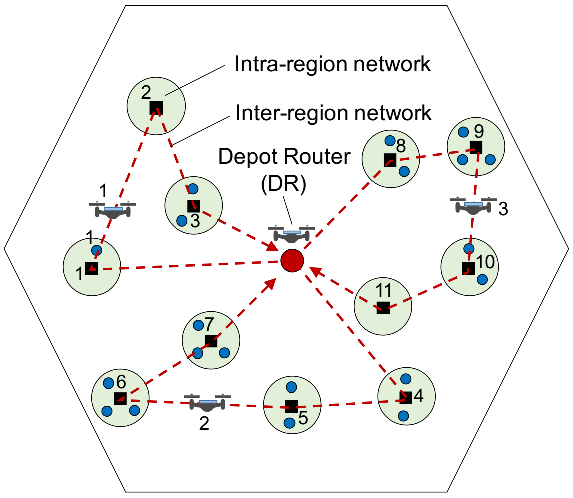


Figure 6: An Inter-region network

In this paper, a disjoint network as mentioned above is specially called an Intra-region network, and a network between intra-region networks constructed by movement of FRs is called an Inter-region network. Figure 6 shows a Inter-region network that is configured when the example of Figure 5 is targeted. RMICN constructs an Inter-region network in the following five steps basically.

(1) In order to discover Intra-region networks existing in the target area, a DR allocates and circulates the subareas for each FR. The phase of discovering Intra-region networks is called Discovery Phase.

(2) The DR generates as many paths whose departure point is Depot as the number of FRs except for the DR. A network formed by circulating each path by the FR is called an Inter-region subnetwork.

(3) A DR sets the Inter-region subnetworks that are in charge of crawling for each FR one by one, and circulates the path for each FR. The phase of communicating between Intra-region networks by FRs is called Crawling Phase.

(4) The FR sends and receives messages to the nodes (the DR or GWs) that has become connectable by its moving.

(5) The FR returns to Depot by each crawling of its own Inter-region subnetwork, connects to

the DR, and exchanges the collected route information and messages.

This is the method of realizing communication among Intra-region networks via an Inter-region network. As methods for finding the Inter-region subnetwork in which each FR is in charge of the crawling, there are two approaches depending on whether FRs decide their own crawling path autonomously or whether the representative node decides crawling paths of all the FRs centrally. As described above, the latter is adopted in terms of facilitating cooperation among FRs and the crawling paths of all FRs are determined by a DR in RMICN. A DR is a special Flying Router that constructs an Inter-region network by gathering information of the target area with FRs, and it instructs each FR to circulate each Inter-region subnetwork. Then, the FR which has received the crawling instruction from a DR carries out crawling as instructed and exchanges route information and messages when it becomes possible to connect with the GW constituting an Inter-region network. Finally, the FR after the crawling is connected to the DR at Depot and exchanges the route information and the messages of the whole area. The path planning in RMICN does not take into account the communication range of wireless nodes as considered in [12]. The aim of our research is to show the advantages that ICN brings to movement control of routers rather than to demonstrate the accuracy of the proposed path planning compared to location-based path planning approaches.

In addition, RMICN can be scaled according to the size of the targeted disjoint-networks. Figure7 represents scale of RMICN. RMICN can be easily scaled, and as the method, a Flying Router called Rendezvous Router is installed at every two apexes of hexagons of areas respectively, and communication between the hexagons is performed only by Rendezvous Routers. The behavior of Rendezvous Routers is equal to GWs, so that without moving, route information and messages are exchanged when connecting with FRs. Instead of simply enlarging the target area of RMICN, it is possible to prevent the calculation time of an Inter-region network from increasing exponentially by configuring hexagons by joining together. In addition, there is an advantage that contents retrieval time is relatively shortened, especially in the use of frequent regional communication.

## 3.4 Supporting Functions by Seamless Extension of ICN

As mentioned in Section 3.2 that RMICN is designed as a seamless extension of ICN, RMICN's extension to ICN is only mobility control function and connection driven communication function

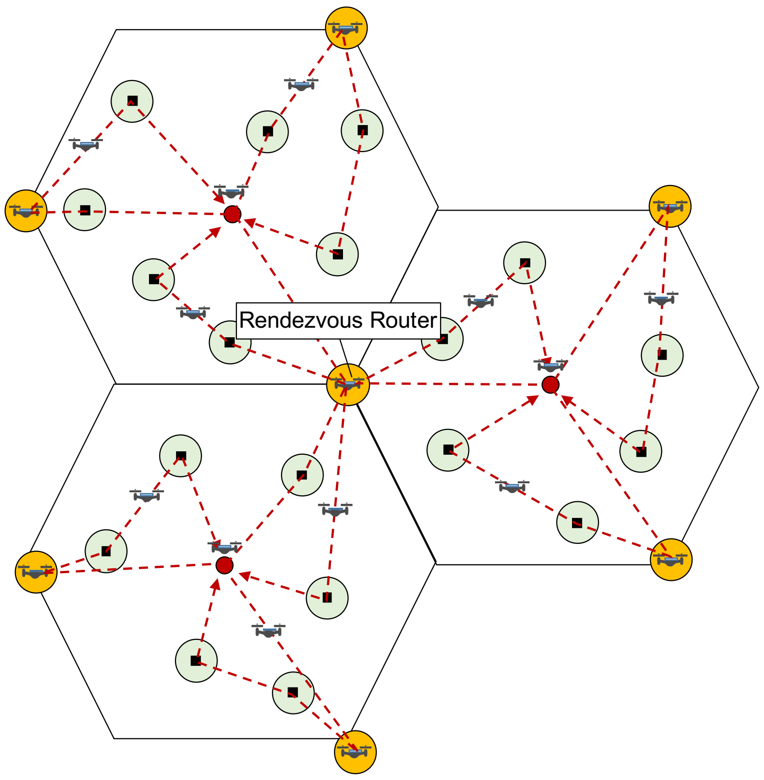


Figure 7: Scale of RMICN

of a router. Therefore, since RMICN does not affect the communication protocol of ICN at all, it can be said that functions realizable by ICN can also be realized in RMICN. In this paper, we explain in particular that RMICN supports high security and push-type communication.

### 3.4.1 Security

In Section 2.1, we mentioned that ICN can provide high security regardless of communication path and communication protocol unlike IP networking, because ICN makes contents themselves secure. In a network such as RMICN where connections are frequently disconnected, it can be said that it is not suitable to secure paths like the security extension on IP networking. This is because when securing paths, it is necessary to authenticate at the time of connection and establish a session at the start of communication, and so overheads involved in these processes frequently occur in disjoint-networks communications that repeat connection and disconnection. On the other hand, since RMICN signs and encrypts contents themselves, connections authentication and sessions establishment are unnecessary, and it is possible to send and receive messages immediately.

In NDN, each node refers to the KeyLocator included in a message to obtain the name of the key corresponding to the content and it retrieves Data with the actual key by sending the Interest with the name to verify the signed content [16]. In this way, NDN also retrieves keys for security use by exchanging Interest and Data, and so it can be said that security function of ICN operates normally even on RMICN.

### 3.4.2 Push-Type Communication

Since ICN is based on pull-type communication as a general rule, push-type communication can not be performed basically. However, depending on use cases, there are many cases where push-type communication is required. For example, in sensor networks, it is considered that sensors periodically transmit sensing information, besides in disaster networks, person's safety information may be transmitted before requesting. So, there are some papers examining push-type communication using ICN such as a method of Pub/Sub [17], a method handling special PIT entry [18], and a method of embedding content in name of Interest [19]. Any of them can be applied in RMICN. In this thesis, we introduce the paper [17] which proposes Pub/Sub type communication using NDN as a method to realize push-type communication for RMICN.

In COPSS (Content-Oriented Pub/Sub System) proposed in the paper [17], a central node called Rendezvous Node (RN) for Pub/Sub type communication is established and furthermore it adds a new data structure called Subscription Table (ST), which maps a content name to the face destined to a subscriber. A subscriber sends Subscribe Message including the name of a content in the direction of RN when the subscriber wants to subscribe the content, and the node receiving the Subscribe Message records the name and the received face in ST. When a push-type content is published in a publisher node, the node sends a Publish Message including the content in the direction of RN. Then if the name is registered in ST, the node receiving the Publish Message sends to the corresponding face. In COPSS, push-type communication based on Pub/Sub is realized by the above procedure, and it is possible to operate COPSS in RMICN using DR as RN and implementing ST as a new data structure.

## 3.5 Leveraging Content-Based Essence

In Section 3.2, incorporating the content-based features of ICN is mentioned as a design policy of RMICN. In addition, we described that we focus on control with name, content-based parameter, and content cache especially among them. In this section, we will describe how these content-based features are utilized in RMICN.

### 3.5.1 Control with Name

We descibed that a DR establishes the movement paths of FRs and it makes each FR crawl along the path for constructing an Inter-region network in RMICN. RMICN realizes the movement control of the FR, including the crawling command, to utilizing names and strategies in NDN. Regarding movement control of routers using name, we studied in our previous research report [20], but the movement control was realized in the application layer instead of strategies. In addition, the movement control was driven by receiving a normal message. In this thesis, unlike our previous paper which involves moving FRs by only the characteristics of normal messages (such as destined location), we propose moving FRs by path planning and movement commands to perform communication considering cooperation of multiple FRs and handling of multiple messages. Specifically, we propose that a DR makes an FR perform desired movement by sending an Interest whose details of the movement of the FR is represented by its name from the DR to the FR. Then, the FR receiving the Interest performs movement control according to the name by its strategy. NDN can control the equipment by its name, and NDN routers can seamlessly perform forwarding and control of routers by separating forwarding messages and control messages in the namespace.

In addition, as the other utilization of names and strategies, we also propose a control mechanism to characterize contents which refer to the names of Interest and Data received in the process of crawling of FRs. In this paper, we consider a process of retrieving contents autonomously interrupting the crawling instructed from a DR if the requested contents are geographically close. Message-based and content-based autonomous movement control utilizing names is a unique characteristic of ICN routers and it is expected to lead to the realization of a flexible infrastructure to enable communication between disjoint networks.. In addition, as an extensional control utilizing this feature, we also propose an infrastructure that can easily provide extended functions for a wide variety of applications in FRs. For example, considering the smart agriculture mentioned in Section 3.1, the router itself serves as a movable actuator to provide functions such as an aerial photographs of agricultural crops, spraying of agricultural chemicals, fertilizer, and herbicide as contents. Then, users can use these functions only by sending an Interest to an FR.

### 3.5.2 Content-Based Parameter

As described in Section 3.2, RMICN uses content retrieval time as a content-based parameter and performs path planning based on the standard of content retrieval time. In RMICN, it is possible to regard the flow from the request to the retrieval of a content as the following three phases:

- Collecting Interest
- Collecting Data
- Delivering Data

In the above description, we show FRs which crawls in an Inter-region subnetwork calculated by a DR constructs an Inter-region network by exchanging information and messages with the DR at each crawling. Considering that all the FRs meet the other FRs at Depot for each crawling, it can be seen that all contents are completely retrieved by all FRs performing crawling for the three times. Considering from the user's point of view, in the case where the content retrieval time is the latest, that is, the Interest is transmitted immediately after the FR passes the Intra-region network which the user belongs to, the crawling to retrieve the content may take four times longer in the worst case. Therefore, if the interval of meeting at Depot is $T_r$, the maximum retrieval time of content can be set to $4T_r$. So, RMICN can notify users of the estimated content retrieval time. In addition, RMICN performs autonomous movement control as described in Section 3.5.1 so as not to exceed the content retrieval time. This can be realized because the communication primitive of NDN is one-to-one exchange of Interest and Data.

### 3.5.3 Content Cache

Since RMICN is a seamless extension of ICN, it is possible to utilize content cache. According to the described method above of constructing an Inter-region network, RMICN can provide the following three cases in the cache effect (omission of phases) from the user's viewpoint.

- The case that one phase (Collecting Data) can be omitted
  - A user requests a content which was requested once or more from other than the Inter-region subnetwork which the user belongs to (that is, there is a cache in the DR).
- The case that two phases (Collecting Data and Delivering Data) can be omitted
  - A user requests a content which was requested once or more from the Inter-region subnetwork which the user belongs to (that is, there is a cache in the FR constructing the Inter-region subnetwork which the user belongs to).
- The case that all phases (Collecting Interest, Collecting Data, and Delivering Data) can be omitted
  - A user requests a content which was requested once or more from the Intra-region network which the user belongs to (that is, there is a cache in the GW constructing the Intra-region network which the user belongs to).

In the above cases, however, the cache size in FRs and GW is infinite, and the effect differs depending on the cache size and the cache replacement algorithm.

# 4 Architectural Design

In this thesis, we design RMICN extending NDN. In this section, we first describe the components to be added as an extension to NDN in Section 4.1, and then we describe the communication method between disjoint networks using those components in Section 4.2. Furthermore, Section 4.3 describes further practical controls of FR using names and strategies.

## 4.1 Enhancement Components

The class diagram of nodes constituting RMICN is shown in Figure 8. As shown in Figure 8, FRs and GWs which are constituent nodes of RMICN can be realized by adding unique data structures, managers, and strategies in addition to functions supported by NDN. The data structures mean structures for storing data necessary for communication in RMICN, and the managers mean objects for providing the function of actually realizing communication in RMICN using the data structures. Then, by calling the functions of the managers in unique strategies, we realize cooperation between NDN communication and processing such as movement control of routers in RMICN.

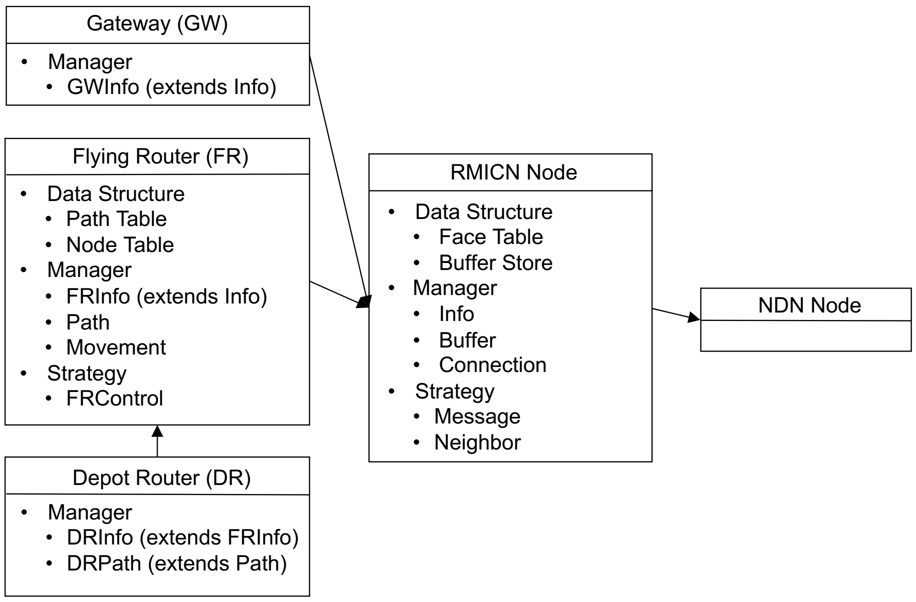


Figure 8: A class diagram of nodes constituting RMICN

There are four types of data structures, respectively called Face Table, Buffer Store, Path Table, and Node Table. A role of each data structure is shown in Table 5.

Table 5: Custom data structures

| Data Structure | Role | Implemented by |
|---|---|---|
| Face Table | Store the node name and connection state of the node for each Face | All nodes |
| Buffer Store | Store messages waiting for each Face | All nodes |
| Path Table | Store Inter-region subnetworks that compose the Inter-region network | FR・DR |
| Node Table | Store the name, type, and location of all the nodes that construct the Inter-region network | FR・DR |

There are five types of managers, respectively called Info Manager, Buffer Manager, Connection Manager, Path Manager, and Movement Manager. A role of each manager is shown in Table 6.

Table 6: Custom managers

| Manager | Role | Implemented by |
|---|---|---|
| Info Manager | Manage information necessary for RMICN | All nodes |
| Buffer Manager | Manage buffers for each Face | All nodes |
| Connection Manager | Manage the connection state of Faces | All nodes |
| Path Manager | Manage paths of the Inter-region network | FR・DR |
| Movement Manager | Manage movement of nodes | FR・DR |

There are three types of strategies, respectively called Message Strategy, Neighbor Strategy, and FRControl Strategy. A role of each strategy is shown in Table 7.

Nodes of RMICN operate by implementing these data structures, managers, and strategies, but these are different depending on the type of the node. All nodes need Face Table and Buffer Store as data structures, Info Manager, besides, Buffer Manager, and Connection Manager as managers, besides, Message Strategy and Neighbor Strategy as strategies. For GW, there is no need for additional, but for FR and DR, Path Table and Node Table as data structures, Path Manager and Movement Manager as managers, FRControl Strategy as strategies are necessary in addition.

Table 7: Custom strategies

| Strategy | Role | Implemented by |
|---|---|---|
| Message Strategy | Perform forwarding normal messages | All nodes |
| Neighbor Strategy | Perform exchanging information such as route information with connected nodes | All nodes |
| FRControl Strategy | Perform movement control of FRs | FR・DR |

Since GW, FR, and DR require different functions when managing data structures and exchanging information, GW has GWInfo Manager that inherits from Info Manager, and FR has FRInfo Manager that inherits from Info Manager, and DR has DRInfo Manager which inherits from FRInfo Manager. Likewise, since FR and DR require different functions when performing path control of FRs, DR has DRPath Manager that inherits from Path Manager.

### 4.1.1 Custom Data Structures

In order to construct a network involving movement of routers, RMICN adds the following data structures to the nodes which are not found in NDN. Their details are shown below:

**Face Table**

Face Table is a table for storing the Face ID, the name of the destination node, and the connection state by each node, and it is managed by Info Manager. In RMICN, each node has a unique name, and uniquely creates Face for each node (DR, FR, GW or Node) that may be connected. The nodes that may be connected mean, for FR, DR and GW in its Inter-region subnetwork, for GW, FR and the Nodes in its Intra-region network. It should be noted that the connection state is represented by a logical type and it is used when referring to whether the face is connected when the node sends a message. For example, in the Figure 6, it is possible that Face Tables of Node1, GW1, FR, and DR will look like Table 8 - Table 11.

**Buffer Store**

When forwarding a message, RMICN forwards the message if the destination Face is connected, but it buffers the message if it is not connected. Buffer Store is a buffer space of messages created for each face, and it is managed by Buffer Manager. Then, when a Face becomes a connected state

Table 8: An example of custom data structure: Face Table of Node1 in Figure 6

| Face ID | Node Name | Connected |
|---|---|---|
| face1 | GW1 | true |

Table 9: An example of custom data structure: Face Table of GW1 in Figure 6

| Face ID | Node Name | Connected |
|---|---|---|
| face1 | FR1 | false |
| face2 | Node1 | true |

Table 10: An example of custom data structure: Face Table of FR1 in Figure 6

| Face ID | Node Name | Connected |
|---|---|---|
| face1 | DR | false |
| face2 | GW1 | false |
| face3 | GW2 | false |
| face4 | GW3 | false |

Table 11: An example of custom data structure: Face Table of DR in Figure 6

| Face ID | Node Name | Connected |
|---|---|---|
| face1 | FR1 | false |
| face2 | FR2 | false |
| face3 | FR3 | false |

by connection event, the node sends all messages to the face in Buffer Store.

**Path Table**

Path Table is lists representing an Inter-region network when a DR is a gateway or representing an Inter-region subnetwork when an FR is a gateway. It is managed by FRInfo Manager. Path Table is implemented in all FRs and a DR, but since GWs and Nodes do not require the configuration information of an Inter-region network, they do not implement Path Table. The list representing Inter-region subnetwork has meaning in order which means each FR crawls along GWs in that order in the Inter-region subnetwork. As an example in Figure 6, Path Table of the DR is shown in Table 12, and Path Table of FR1 is shown in Table 13.

Table 12: An example of custom data structure: Path Table of DR in Figure 6

| FR Name | Path |
|---|---|
| FR1 | GW1→GW2→GW3 |
| FR2 | GW4→GW5→GW6→GW7 |
| FR3 | GW8→GW9→GW10→GW11 |

Table 13: An example of custom data structure: Path Table of FR1 in Figure 6

| FR Name | Path |
|---|---|
| FR1 | GW1→GW2→GW3 |

**Node Table**

Node Table is a table for storing name, type, and location of all the nodes constituting the Inter-region network, and it is managed by FRInfo Manager. For the same reason as Path Table, only FRs and a DR implement Node Table. In addition, the location entry stores the position of Depot for a DR, the place where it is located for GWs, and whether or not at Depot for FRs. For example, Node Table of all FRs and a DR in Figure 6 is shown as Table 14.

Table 14: An example of custom data structure: Node Table of FR and DR in Figure 6

| Node Name | Node Type | Location |
|---|---|---|
| FR1 | FR | Not Depot |
| FR2 | FR | Not Depot |
| FR3 | FR | Not Depot |
| DR | DR | Depot |
| GW1 | GW | Loc1 |
| GW2 | GW | Loc2 |
| ... | ... | ... |
| GW11 | GW | Loc11 |

### 4.1.2 Custom Managers

In this paper, we add a module called a manager to the node as a mechanism that provides a function to construct a network that actually involves router movement using the data structure defined in Section 4.1.1. Details of each manager are shown below:

**Info Manager**

Info Manager is a manager whose main purposes are the management of data structures and the exchange of information between nodes. It is implemented in a RMICN node. However, functions on the management of data structures and the exchange of information between nodes depend on the types of node (Node, GW, FR, or DR), so that GWInfo Manager inheriting from Info Manager is implemented for GW, FRInfo Manager inheriting from Info Manager is implemented for FR, and DRInfo Manager inheriting from FRInfo Manager is implemented for DR. Figure 9 shows the class diagram of Info Manager implemented in each node.

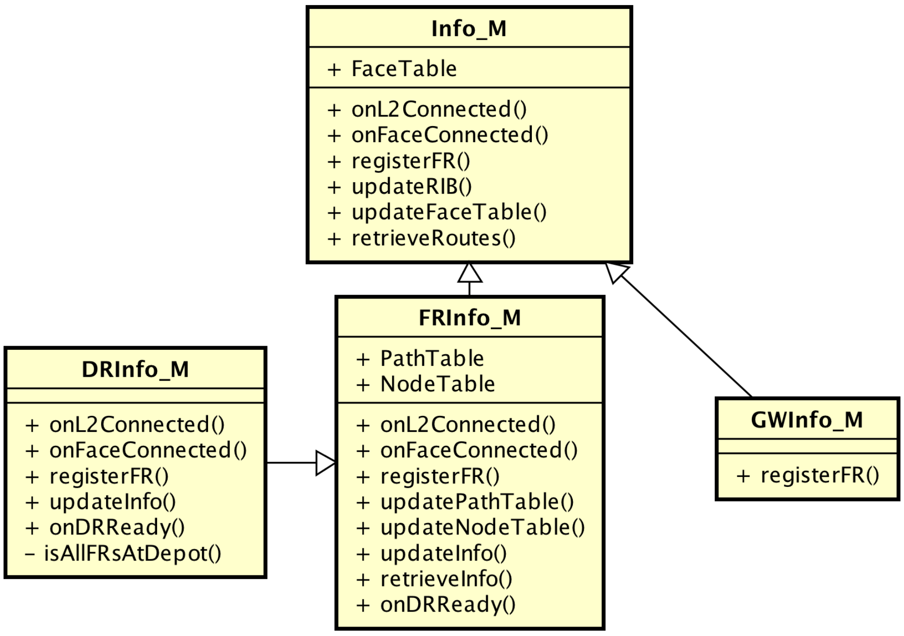


Figure 9: A class diagram of Info Manager

Figure 9 shows the main fields and methods implemented in each Info Manager. However, as mentioned above, since the functions of the managing the data structure and the exchanging information are different depending on the types of node, different methods are implemented, and methods are redefined to express the functions required by each Info Manager. Table 15 gives an overview of the role of each method. In the case that the implementation of the method is empty, it is represented by '-'.

Table 15: Methods implemented in Info Manager

| Method | Manager | Role |
|---|---|---|
| onL2Connected() | Info | - |

| | | |
|---|---|---|
| onL2Connected() | FRInfo | This is an event litsener called to detect a new second layer connection not recognized as a face. First, when the current phase is Discovery Phase, calling stop() of Movement Manager stops movement of the FR. Then, in order to communicate with connected node, it creates a Face to the node. Besides, it sends an Interest with the name ”`/Neighbor/{FaceID}/RegisterFR/{Name}`” in order to register its own node (FR) as a communication node to the connected node, from the created Face. Here, ID of the Face from which the Interest is transmitted is inserted into FaceID, and the name of its node is inserted into name (The same applies hereafter). |
| onL2Connected() | DRInfo | - |
| onFaceConnected() | Info | This is an event listener called when a Face is connected. In order to obtain route information from the connected node, it sends an Interest named ”`/Neighbor/{FaceID}/RetrieveRoutes`” from the connected Face. |
| onFaceConnected() | FRInfo | Regardless of the current phase (Discover or Crawling), it does nothing if the type of the connected node is Node, FR, or DR. When the type is GW, in the case of Discover Phase or the case of Crawling Phase and its GW is included in its own Path Table (that is, the GW belongs to its Inter-region subnetwork), it performs the same processing as onFaceConnected() in Info Manager. |
| onFaceConnected() | DRInfo | If the type of the connected node is FR, it sends an Interest named ”`/Neighbor/{FaceID}/RetrieveInfo`” from the connected Face to obtain route information and entries of Node Table of the connected node. |
| registerFR() | Info | This is a method called when a request of FR registration is received. In this method, we refuse the registration (only GW and DR will accept FR registration). |

| | | |
|---|---|---|
| registerFR() | GWInfo | This is a method called when a request of FR registration is received. By calling updateInfo(), it registers the FR information in Face Table. |
| registerFR() | FRInfo | This is a method called when a request of FR registration is received or FR registration is completed. If it is called by a request of registration, the registration is refused and finished, but if it is called at the completion of registration, by calling updateFaceTable() and updateNodeTable(), information of the connected node can be stored in Face Table and Node Table. Then, when the current phase is Crawling Phase, it calls onFaceConnected() of Connection Manager and sends an Interest named ”`/Neighbor/{FaceID}/onFaceConnected`” to the connected node. |
| registerFR() | DRInfo | This is a method called when a request of FR registration is received. It performs the same processing as registerFR() of GWInfo Manager. |
| updateRIB() | Info | It updates RIB. |
| updateFaceTable() | Info | It updates Face Table. |
| retrieveRoutes() | Info | This is a method called when exchanging route information with the connected node. It converts routing information owned by itself into a character string to store it as a message. Specifically, it converts the list of content names registered in RIB and the list of content names provided by the applications registered in Face into a character string in JSON format. However, it excludes the route information whose next hop is the connected node. |
| updatePathTable() | FRInfo | It updates Path Table. |
| updateNodeTable() | FRInfo | It updates Node Table. |
| updateInfo() | FRInfo | It updates RIB and Node Table by calling updateRIB() and updateNodeTable(). |

| | | |
|---|---|---|
| updateInfo() | DRInfo | It updates RIB and Node Table by calling updateRIB() and updateNodeTable(). Then, it also calls isAllFRsAtDepot() and if all FRs are at Depot, it calls onDRReady() as the trigger. |
| retrieveInfo() | FRInfo | This is a method called when exchanging route information with the connected node. It converts route information and entries of Node Table into a character string. Route information is converted by calling retrieveRoutes() and entries of Node Table are converted into a character string in JSON format except for the entries whose node type is FR or DR and whose name is not one of the own nodes. |
| onDRReady() | FRInfo | This is a method called when a DR retrieves information from all FRs and becomes available for the next crawling. In order to retrieve information from the DR, it sends an Interest named ”url /Neighbor/FaceID/RetrieveInfo” to the DR. |
| onDRReady() | DRInfo | In order to advertise that the DR retrieves information from all FRs and it is ready to move onto the next crawling, it sends an Interest named ”`/Neighbor/{FaceID}/onDRReady`” to all FRs. Then, in order to move to the next crawling, it calls onDR-Ready() of Path Manager. |
| isAllFRsAtDepot() | DRInfo | This is a method to check whether all FRs are at Depot. As processing, it refers to Node Table, and if the locations of entries corresponding to all FRs are Depot, TRUE is returned. |

**Path Manager**

Path Manager is a manager having functions for configuring an Inter-region network. The functions of Path Manager depend on the role of FR. In the case of DR, it configures an Inter-region network from the location of an Intra-region network and the number of FRs. On the other hand, in the case of FR, it performs processing about autonomous route change. Therefore, note that DR has DRPath Manager that inherits from Path Manager as Path Manager. Besides, not only the configuration of the Inter-region network but also the discovery processing of Intra-region networks are performed by Path Manager. The class diagram of Path Manager implemented for FR and DR is shown in Figure 10. For the field variables and the methods in the figure, only major ones are indicated.

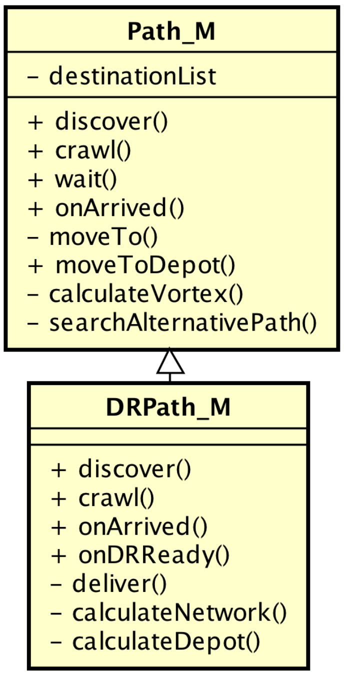


Figure 10: A class diagram of Path Manager

Table 16 gives an overview of the role of each method.

Table 16: Methods implemented in Path Manager

| Method | Manager | Role |
|---|---|---|
| discover() | Path | This is a method to discover Intra-region networks within a subarea given from the DR. In this method, first of all, the phase is set to Discovery Phase, and the location information of its own node (FR) stored in Node Table is set to ”Not Depot”. Then, after computing the route searching in a vortex shape in the subarea by calculateVortex(), it moves its node along the calculated path by calling moveTo() and onArrived(). When all the movements are completed, movement to Depot is started by calling moveToDepot() and the location information of its own node stored in Node Table is set to ”Depot”. |
| discover() | DRPath | This is a method to discover Intra-region networks from the target area. In this method, first of all, the phase is set to Discovery Phase, and the location information of all FRs in Node Table is set to ”Not Depot”. Then, it sends an Interest named ”`/FRControl/{name}/Discover/{Loc1} /{Loc2}/...`” to each FR in order to search each FR for Intra-region networks in each subarea into which the target area is divided by the number of FRs. Here, the name is one of the destined FRs, and the loc* is the vertex of the polygon constituting the subarea. |

| | | |
|---|---|---|
| crawl() | Path | This is a method to crawl along the Inter-region subnetwork (the list of the Intra-region networks) given from the DR. In this method, first of all, the phase is set to Crawling Phase and the location information of its own node (FR) stored in Node Table to ”Not Depot”. Then, the given list of the Intra-region networks is stored in Path Table, and it moves its node along the order of the Intra-region networks in the list by calling moveTo() and onArrived(). When all the movements are completed, movement to Depot is started, and location information of its own node stored in Node Table is set to ”Depot”. |
| crawl() | DRPath | This is a method to make each FR crawl along its Inter-region subnetwork stored in Path Table. In this method, first of all, the phase is set to Crawling Phase, and the location information of all FRs in Node Table is set to ”Not Depot”. Then, in order to make each FR crawl along the Intra-Region Subnetwork, It sends an Interest named ”`/FRControl/{name}/Crawl/{GW1}/{GW2}/...`” to each FR. Here, the name is one of the destined FRs, and the GW* is the name of the GW constituting the Intra-region network. |
| wait() | Path | This is a method to wait for $T_w$ seconds for route information and message exchange at an Intra-region network and Depot. |

| | | |
|---|---|---|
| onArrived() | Path | This is an event listener informing that its own node has arrived at the destination given by moveTo() or moveToDepot() called by Movement Manager. As processing, first of all, the arrived destination is deleted from the destinationList storing the destination list. At this time, if the destination is Depot, the method finishes. If not, it calls wait() for exchanging route information and message, and then it calls searchAlternativePath() to perform autonomous path change if it is Crawling Phase. Finally, if there is a next destination in the destinationList, it moves its own node to the next destination by calling moveTo(). If not, it sets its location information in Node Table to ”Depot” by calling updateNodeTable() of Info Manager, and moves its own node to Depot by calling moveToDepot(). |
| onArrived() | DRPath | OnArrived() in DRPath Manager is called when the node changes the location of Depot and completes moving to the new location of Depot. As processing, however, it does nothing. |
| moveTo() | Path | This is a method to move to a given destination. As processing, it calls moveTo() of Movement Manager. |
| moveToDepot() | Path | This is a method to move to Depot. As processing, it calls moveToDepot() of Movement Manager. |
| calculateVortex() | Path | This is a method to calculate the crawling path in a given subarea for discovering Intra-region networks. It calculates the path on which the FR moves in a vortex shape toward the center in the subarea spacing the range of its own communication range and moves to Depot from the center in the subarea. |
| searchAlternativePath() | Path | This is a method to select a better path by the algorithm described in Section 5.2. If a better path is found, the path is stored at the head of the destinationList. |

| | | |
|---|---|---|
| onDRReady() | DRPath | This is an event listener called when all FRs have finished crawling for discovering Intra-region networks or message exchanges and they are met at Depot. As processing, in the case of Discovery Phase, the case means that the discovery of Intra-region networks is over, so it updates the location of Depot by calling calculateDepot() and then calls deliver(). On the other hand, in the case of Crawling Phase, it calls crawl() and continues the crawl of the FRs. |
| deliver() | DRPath | This is a method to perform delivery of messages by making all FRs crawl along the paths by calling crawl() after the paths meaning Inter-region network is calculated by calculateNetwork(). It moves its own node to the newly set location of Depot by calling moveToDepot () at the end of the method. |
| calculateNetwork() | DRPath | This is a method to calculate the path of the Inter-region network by the algorithm described in Section 5.1 from the location of the Intra-region networks and the number of the FRs obtained by referring to Node Table. The calculated Inter-region network is stored in Path Table. |
| calculateDepot() | DRPath | This is a method to calculate the location of Depot from the location of the Intra-region networks obtained by referring to Node Table. After calculation, the location of Depot is stored in Node Table and an Interest named ”`/Neighbor/{FaceID}/onDepotUpdated`” is sent to all FRs to inform of the new location of Depot . |

**Buffer Manager**

In RMICN, messages being transferred to disconnected nodes need to be temporarily buffered. Buffer Manager is a manager that manages Buffer Store to buffer messages. The node determining the destination of the message transfers it (Interest only) if the destination is connected. In this case, Data is discarded without being transferred, because of the specification of NDN, that is, Data is transferred automatically when the destination is connected. On the other hand, when the destination is not yet connected, the message is stored in the buffer for each destination face prepared in Buffer Store. When a Face connected event occurs, all messages (Interest and Data) stored in the Face's buffer in Buffer Store are transferred. The class diagram of Buffer Manager is shown in Figure 11. For the field variables and methods in the figure, only major ones are indicated.

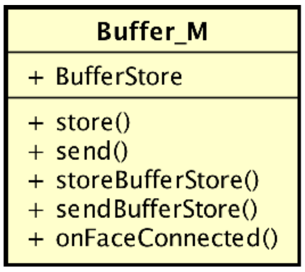


Figure 11: A class diagram of Buffer Manager

Table 17 describes an overview of the role of each method.

Table 17: Methods implemented in Buffer Manager

| Method | Role |
|---|---|
| store() | If the forwarding destination Face of the given message is connected, it sends the message by calling send() only if the type of the message is Interest. If the Face is not connected, it calls storeBufferStore() to store the message in Buffer Store. |
| send() | This is a method to send a message from the forwarding destination Face. When sending Interest, it records the Face in out-record of PIT, and when sending Data, it consumes the corresponding PIT entry . |
| storeBufferStore() | It stores the message in Buffer Store. |

|  |  |
|---|---|
| sendBufferStore() | It sends all messages with the given Face as the forwarding destination in Buffer Store by calling send(). |
| onFaceConnected() | This is an event listener called when a Face is connected. It calls sendBufferStore() to send all messages whose forwarding destination Face is the connected one in BufferStore. |

**Connection Manager**

Connection Manager monitors the second layer connection, and it notifies other managers performing processing according to a connection event by calling a callback function when the second layer is connected. The class diagram of Connection Manager is shown in Figure 12. For the field variables and methods in the figure, only major ones are indicated.

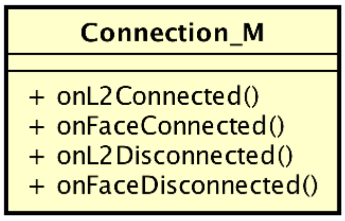


Figure 12: A class diagram of Connection Manager

Table 18 describes an overview of the role of each method.

Table 18: Methods implemented in Connection Manager

| Method | Role |
|---|---|
| onL2Connected() | This is a method which is called when the second layer connection is detected. onFaceConnected() is called if a Face corresponding to the identifier of connected second layer is in Face Table, otherwise onL2Connected() of the other manager registered as the listeners is called when a new second layer connection is detected. |

| | |
|---|---|
| onFaceConnected() | This is a method which is called when the connection of the second layer registered in Face Table is detected. In this method, first of all, it sets the connection state of the Face in Face Table to True and then it calls onFaceConnected() from the other managers registered as listeners. In addition, since onFaceConnected() is used as a trigger for information exchange in Info Manager, it can be seen that there is no information exchange trigger between nodes that never disconnect, e.g., GW and Node in an Intra-region network. Therefore, onFaceConnected() is called even when the connection state is True for a certain period of time so that information exchange can be performed between such nodes. |
| onL2Disconnected() | This is a method which is called when the disconnection of the second layer is detected. It calls onFaceDisconnected() if the Face corresponding to the identifier of the connected second layer is in Face Table. |
| onFaceDisconnected() | This is a method which is called when the disconnection of the second layer registered in Face Table is detected. It sets the connection state of the corresponding Face to False in Face Table. |

**Movement Manager**

Movement Manager is a manager performing movement control of the movement body (i.e., drone) of the FR. As for the movement control, it implements only simple functions that move linearly to the given destination and suspend and restart the current movement. Then, it calls the callback function of Path Manager instructing when the movement is completed. The class diagram of Movement Manager is shown in Figure 13. For the field variables and methods in the figure, only major ones are indicated.

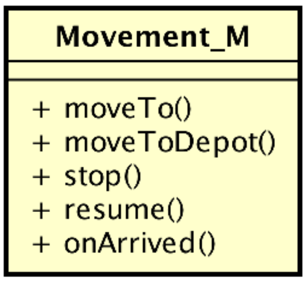


Figure 13: A class diagram of Movement Manager

Table 19 describes an overview of the role of each method.

Table 19: Methods implemented in Movement Manager

| Method | Role |
|---|---|
| moveTo() | This is a method to move to the given location. This method first registers the caller Path Manager as a listener, and if the destination is a node, it obtains the location of the destination by referring to Node Table and it holds the location information as the current movement destination. Then, it executes the control command of drone to move to the location. |
| moveToDepot() | This is a method to move to Depot. It gives the name of DR to the argument of moveTo(). |
| stop() | It suspends the current movement. |
| resume() | It resumes the current movement. |
| onArrived() | This is a method which is called when its own node arrives at the destination set by moveTo(). It calls onArrived() of Path Manager that instructs the current movement to inform that the node arrived at the destination. |

### 4.1.3 Custom Strategies

In RMICN, exchanging information between nodes, exchanging messages, and controlling movement of FRs are peformed by calling methods of managers described in Section 4.1.2 from strategies. Strategy in NDN can be set for each content as shown in Figure 1. Strategy in NDN can define strategies to decide when and where to transfer Interest and Data and what kind of processing to intervene at the NDN node. It is specifically able to perform various controls at the relay node by customizing the trigger method to be called when Interest or Data is received. In addition, NDN can freely decide strategies to apply by namespace using Strategy Choice Table, and it is possible to realize various control using the same route by changing only namespace. RMICN implements Message Strategy, Neighbor Strategy, and FRControl Strategy to realize strategic communication between disjoint networks. The details are shown below.

**Message Strategy**

Message Strategy is a strategy for transferring normal messages. The strategy targets messages whose namespace is "`/`" (However, since Strategy Choice Table decides the strategy to select with the longest match, if they match the namespace targeted by Neighbor Strategy and FRControl Strategy to be described later, those strategies are applied instead of Message Strategy). In Message Strategy, regardless of the types of Interest and Data, all received messages are processed by Buffer Manager. Specifically, Message Strategy acquires the forwarding destination Face of a message by referring to FIB, and it calls store() of Buffer Manager with the message and the acquired Face as an argument.

**Neighbor Strategy**

Neighbor Strategy is a strategy for exchanging information and advertising information between connected neighboring nodes. The strategy targets messages whose namespace is "`/Neighbor/`" and provides processes shown in Table 20 (indicating '-' if there is no processing corresponding to the name). Managers of each node cooperate with managers of other nodes by sending and receiving Interest and Data with the name corresponding to each process via the Neighbor Strategy.

Table 20: Processes provided by Neighbor Strategy

| Name | Type | Process |
|---|---|---|
| `/Neighbor/{FaceID}/RegisterFR/{Name}` | Interest | Processing of the request of FR registration is performed. It calls registerFR() of Info Manager and returns Data storing the result. Note that FaceID is the ID of the Face from which the message is sent, and Name is the name of the requesting FR (the same applies below). |
| `/Neighbor/{FaceID}/RegisterFR/{Name}` | Data | Processing of the response of FR registration is performed. When the registration request is permitted, it calls registerFR() of Info Manager with the type, name, and location information of the registration destination node stored as Data. If the registration request is rejected, it does nothing. Regardless of the result of the registration request, in the case of the Discovery Phase, it resumes the movement of the FR that was interrupted by onL2Connected() of FRInfo Manager by calling resume() of Movement Manager. |
| `/Neighbor/{FaceID}/onFaceConnected` | Interest | Processing of notification request of Face connection event is performed. It calls onFaceConnected() of Connection Manager, and then returns empty Data. |
| `/Neighbor/{FaceID}/onFaceConnected` | Data | - |
| `/Neighbor/{FaceID}/RetrieveRoutes` | Interest | Processing of the request for retrieving route information is performed. It calls retrieveRoutes() of Info Manager, and then returns Data that stores the result (route information). |

| | | |
|---|---|---|
| `/Neighbor/{FaceID}/RetrieveRoutes` | Data | Processing of the response to the request for retrieving route information is performed. It calls updateRIB() of Info Manager. |
| `/Neighbor/{FaceID}/RetrieveInfo` | Interest | Processing of the request for retrieving Route information and entries of Node Table. After casting Info Manager to FRInfo Manager, it calls retrieveInfo(), and then returns Data that stores the result (route information and entries of Node Table). |
| `/Neighbor/{FaceID}/RetrieveInfo` | Data | Processing of the response to the request for retrieving Route information and entries of Node Table. After casting Info Manager to FRInfo Manager, It calls updateInfo() with those information stored in Data as an argument. |
| `/Neighbor/{FaceID}/onDRReady` | Interest | Processing of the request to advertise that the DR becomes available for the next crawling. After casting Info Manager to FRInfo Manager, it calls onDRReady(), and then returns empty Data. |
| `/Neighbor/{FaceID}/onDRReady` | Data | - |
| `/Neighbor/{FaceID}/onDepotUpdated/{Loc}` | Interest | Processing of the request to advertise that the location of Depot has been updated. After casting Info Manager to FRInfo Manager, it calls updateNodeTable() to update the location of DR stored in Node Table. After that, it returns empty Data. Note that the Loc refers to the updated location of Depot. |
| `/Neighbor/{FaceID}/onDepotUpdated/{Loc}` | Data | - |

**FRControl Strategy**

FRControl Strategy is a strategy that carries out control with FR movement. The strategy targets messages whose namespace is "`/FRControl/`" and provides processes shown in Table 21 (indicating '-' if there is no processing corresponding to the name). It provides processing for discovering Intra-region networks in the target area and for delivering messages by making FRs crawling in the Inter-region network.

Table 21: Processes provided by FRControl Strategy

| Name | Type | Process |
|---|---|---|
| `/FRControl/{Name}/Discover/{Loc1}/{Loc2}/...` | Interest | In order to discover Intra-region networks from the target area, it makes the FR specified by Name crawling in the subarea composed of a list of location information given by parameters. After returning empty Data, it calls discover() of Path Manager. If the name of the FR is different from Name, it does nothing. |
| `/FRControl/{Name}/Discover/{Loc1}/{Loc2}/...` | Data | - |
| `/FRControl/{Name}/Crawl/{GW1}/{GW2}/...` | Interest | In order to communicate via the Inter-region network, it makes the FR specified by Name crawling along a list of GWs given by parameters. After returning empty Data, it calls crawl() of Path Manager. If the name of the FR is different from Name, it does nothing. |
| `/FRControl/{Name}/Crawl/{GW1}/{GW2}/...` | Data | - |

## 4.2 Processing Sequence

In this section, we show the procedure for communicating between disjoint networks by FRs using our own components introduced in Section 4.1.

### 4.2.1 Discovering Intra-region networks (Discovery Phase)

In RMICN, when the location of Intra-region networks is not statically set, discovery of Intra-region networks and setting of the location of Depot are performed. A target area is given as a hexagon, and its center point is assumed to be the temporary Depot. Then DR calls discover() of DRPath Manager to start discovery process of Intra-region networks. Figure 14 shows a sequence diagram starting from discover() of DRPath Manager.

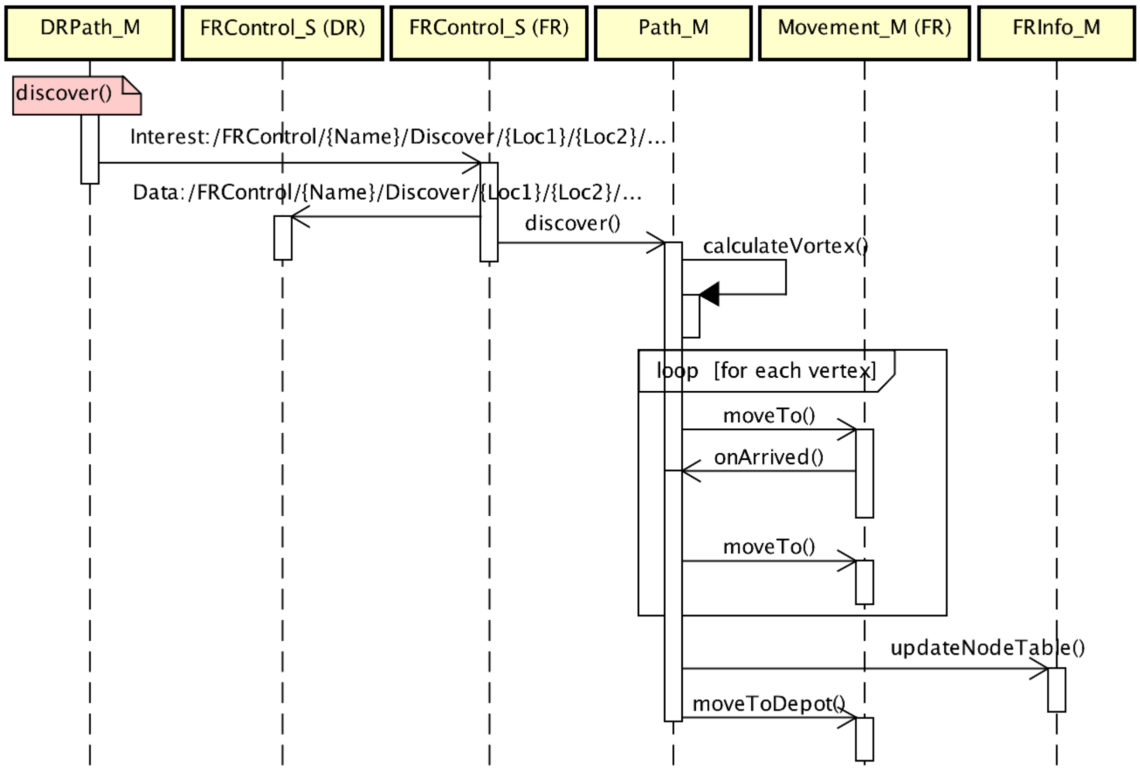


Figure 14: A sequence diagram starting from discover() of DRPath Manager

discover () of DRPath Manager divides the given hexagon by the number of FRs and allocate each divided subarea (a polygon) to each FR by sending Interest to FRControl Strategy. Then, the FR assigned to the polygon area moves following the path which is a vortex path toward the center in the polygon area spacing the range of its own communication range shown in Figure 15 calculated by cacalculateVortex(). Through the movement, the FR tries to discover Intra-region networks.

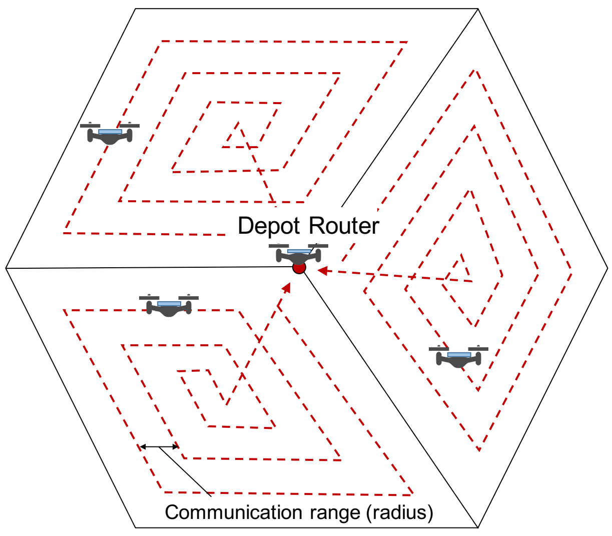


Figure 15: Discover Intra-region networks

In order to discover Intra-region networks, Connection Manager monitors the second layer connections. When detecting a second layer connection, Connection Manager calls its own onL2Connected(). Figure 16 shows a sequence diagram starting from onL2Connected() of Connection Manager.

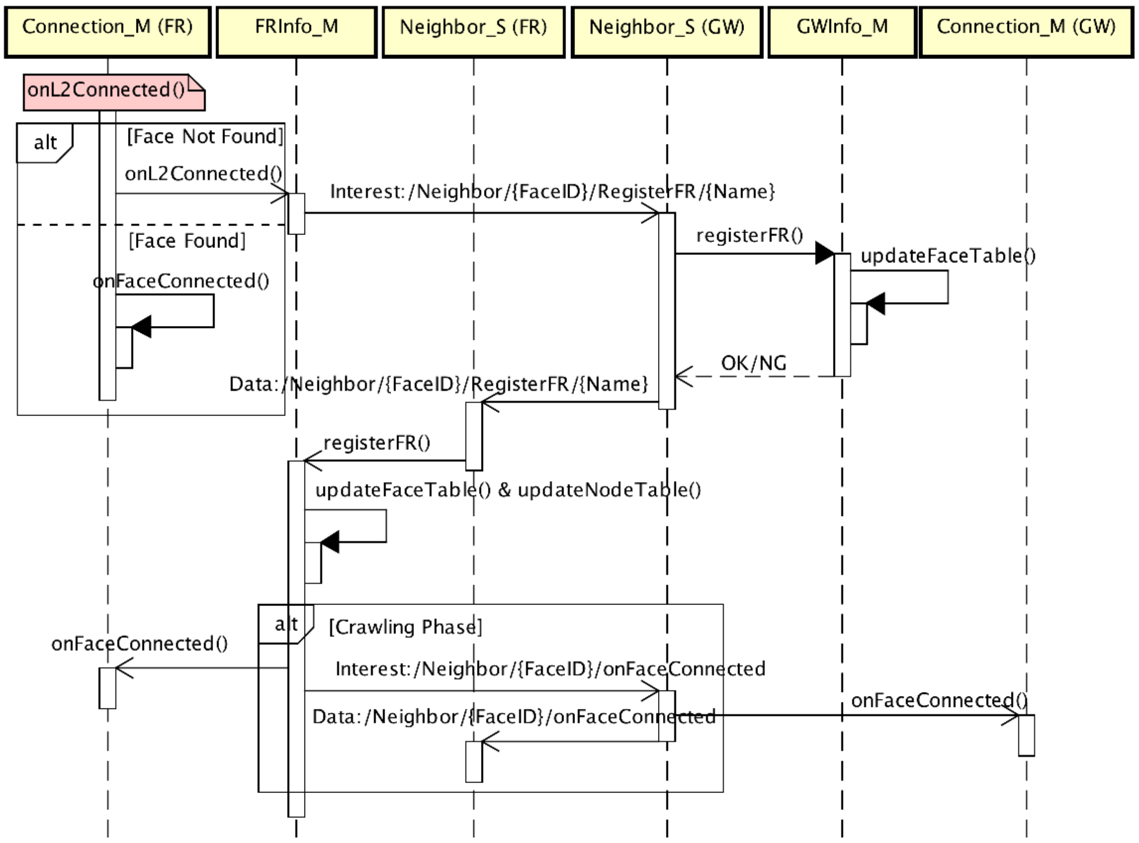


Figure 16: A sequence diagram starting from onL2Connected() of Connection Manager

In Figure 16, description of onL2Connected() of Connection Manager in GW is omitted, but in the actual design, Connection Manager in GW and Connection Manager in FR are the same implementation. As shown in Table 15, however, since onL2Connected() in GWInfo Manager is implemented same as FRInfo Manager, and so when FR and GW are actually connected at the second layer, they trace almost same sequence shown in Figure 16.

onL2Connected() of Connection Manager checks whether Face is already created with the identifier of the second layer, and if it is created, it calls its own onFaceConnected(), if not, it calls onL2Connected() of FRInfo Manager registered as a listener. In Discovery Phase, where FRs discover Intra-region network, Face should not be registered. So, onL2Connected() of FRInfo Manager is called here. When onL2Connected() of FRInfo Manager is called, FR calls registerFR() implemented in the connected GWInfo Manager in order to register itself as a communication node. Since nodes other than GW and DR refuse to register FRs by registerFR(), FRs are not registered in nodes in an Intra-region network, and FRs do not register each other. When the FR registration to the connected node is completed, the information of the connected node is registered in the table of itself (FR). registerFR() stores the name, the type, and the location information of the connected node in Node Table, and stores the Face ID and the node name in Face Table. Although it is not described in Figure 16, in Discovery Phase, the movement of the FR is stopped by calling stop() of Movement Manager when onL2Connected() of FRInfo Manager is called, and the movement of the FR is resumed by calling resume() of Movement Manager when receiving Data named ”`/Neighbor/{FaceID}/RegisterFR/{Name}`”. This is why exchange of the message named ”`/Neighbor/{FaceID}/RegisterFR/{Name}`” is performed normally at the time of discovery of the Intra-region network.

discover() shown in Figure 14 moves the FR straight one by one side of the vortex path in order to discover Intra-region networks. After finishing crawling along all sides, the FR sets its own location information in Node Table to ”Depot” and returns to Depot by calling moveToDepot() in Movement Manager.

After crawling for the discovery of Intra-region networks, the FR that returned to Depot connects with the DR. On this occasion, onL2Connected() of Connection Manager is called as shown in Figure 16. However, since the DR ordered the FR to discover Intra-region networks, the DR and the FR have been connected before. Therefore, as the event handler, onFaceConnected() is called. Figure 17 shows a sequence diagram starting from onFaceConnectet() of Connection Manager when the DR and the FR are connected.

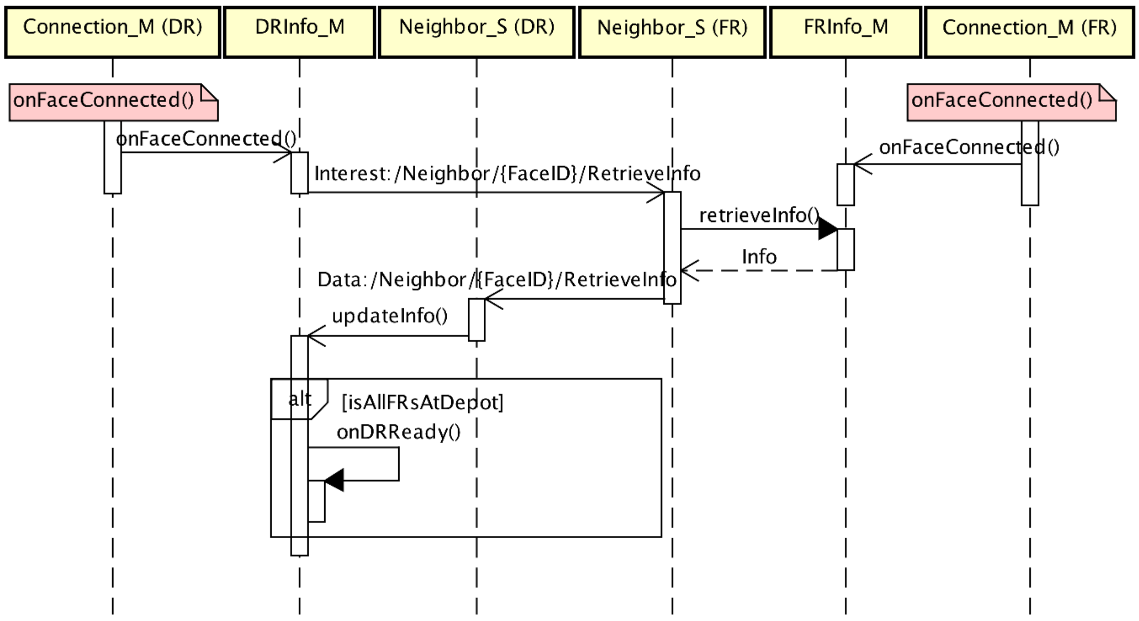


Figure 17: A sequence diagram starting from onFaceConnectet() of Connection Manager when DR and FR are connected

onFaceConnected() of Connection Manager calls onFaceConnected() of Info Manager which is one of the event listeners. If the type of the connected node is DR, onFaceConnected() in FRInfo Manager does nothing because it refrains from retrieving information until the other information gathers at the DR. Then, onFaceConnected() of DRInfo Manager calls retrieveInfo() of FRInfo Manager by sending an Interest in order to retrieve route information and entries of Node Table. Note that, in Discovery Phase, RIB has no entries, but Node Table has the entry of its own FR and information on Intra-region networks found in the area. When the DR obtains the information from all the FRs, since the location information of all the FRs is ”Depot” in Node Table, onDRReady() is called in DRInfo Manager. Figure 18 shows a sequence diagram starting from onDRReady() of DRInfo Manager.

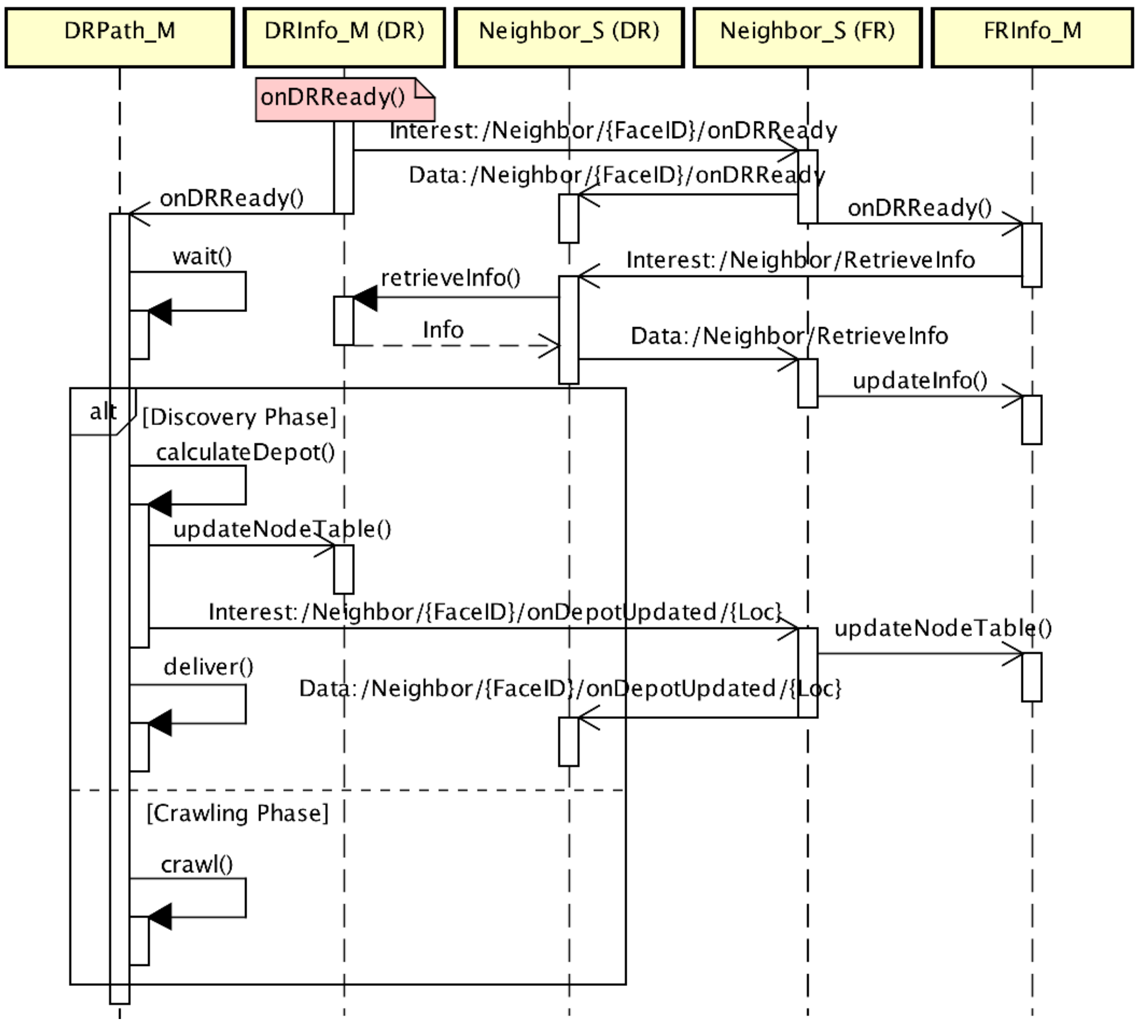


Figure 18: A sequence diagram starting from onDRReady() of DRInfo Manager

When onDRReady() of DRInfo Manager is called, onDRReady() of DRPath Manager of its own node and FRInfo Manager of all FRs is called. At this time, onDRReady() of FRInfo Manager only retrieves information from the DR, but onDRReady() of DRPath Manager performs some processing to move into Crawling Phase, where FRs deliver message. First of all, onDR-Ready() of DRPath Manager waits for a certain period of time by calling wait() in order for FRs to reliably obtain information from the DR, then it calculates the new location of Depot by calling calculateDepot(). The location of Depot $(x_D, y_D)$ is an expression (1) when the number of entries of GW in Node Table is N and the location information of GW$_i$ is $(x_i, y_i)$.

$$(x_D, y_D) = \left( \frac{1}{N} \sum_{1 \le i \le N} x_i, \frac{1}{N} \sum_{1 \le i \le N} y_i \right) \tag{1}$$

When the calculation of the location of Depot is completed, the obtained location information of Depot is registered in Node Table, and FRs are also updated with the location information of Depot by calling updateNodeTable() of FRInfo Manager by receiving the Interest. The above is the discovery procedure of Intra-region networks in Discovery Phase, and the next section will discuss the message exchange method starting from deliver() in DRPath Manager.

### 4.2.2 Delivering Messages（Crawling Phase）

After Intra-region networks are statically set up or dynamically discovered, RMICN shifts to Crawling Phase, which carries out message delivery between disjoint networks. A DR given a set of Intra-region networks (Node Table) calls deliver() in DRPath Manager. Figure 19 shows a sequence diagram starting from deliver() in DRPath Manager.

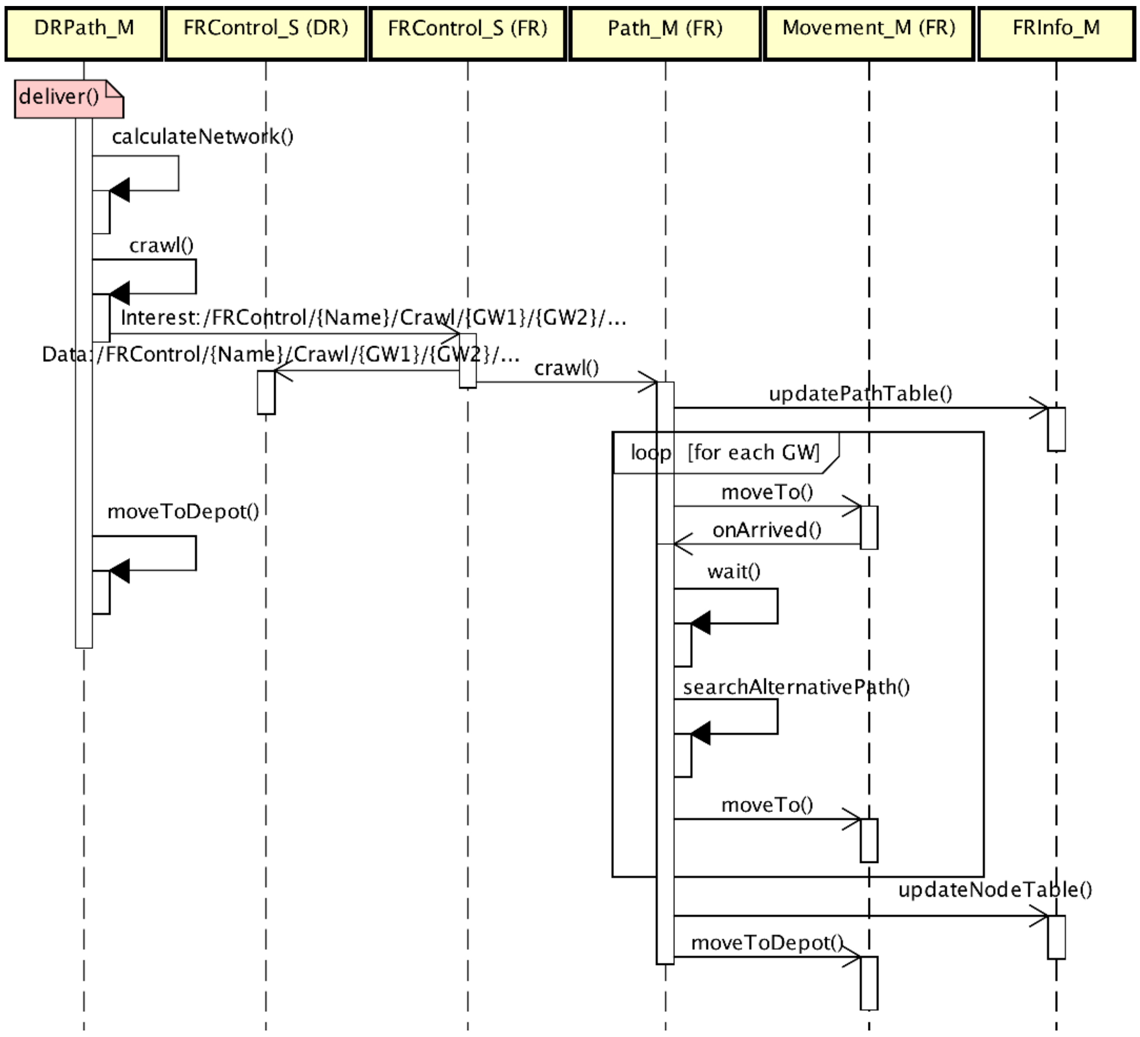


Figure 19: A sequence diagram starting from deliver() in DRPath Manager

deliver() of DRPath Manager searches the path of the Inter-region network by calling calculateNetwork() firstly, and registers the list of GWs for each Inter-region subnetwork in Path Table (For concrete calculation method of an Inter-region network is described in Section 5.1). After that, it calls crawl() of DRPath Manager to call crawl() of Path Manager of each FR by sending an Interest that makes each FR crawl along one Inter-region subnetwork. In crawl() of Path Manager, after registering the Inter-region subnetworks specified in the Interest in Path Table, it moves the FR through the GWs that constitute the Intra-Region Subnetwork in the order of the list.

When connecting with a GW while the FR is crawling, if there is no Face to communicate with the GW (that is, the FR connects with the GW at first), onFaceConnected of Connection Manager is called through the sequence of Figure 16. Figure 20 shows a sequence diagram starting from onFaceConnected() of Connection Manager when the FR and the GW are connected.

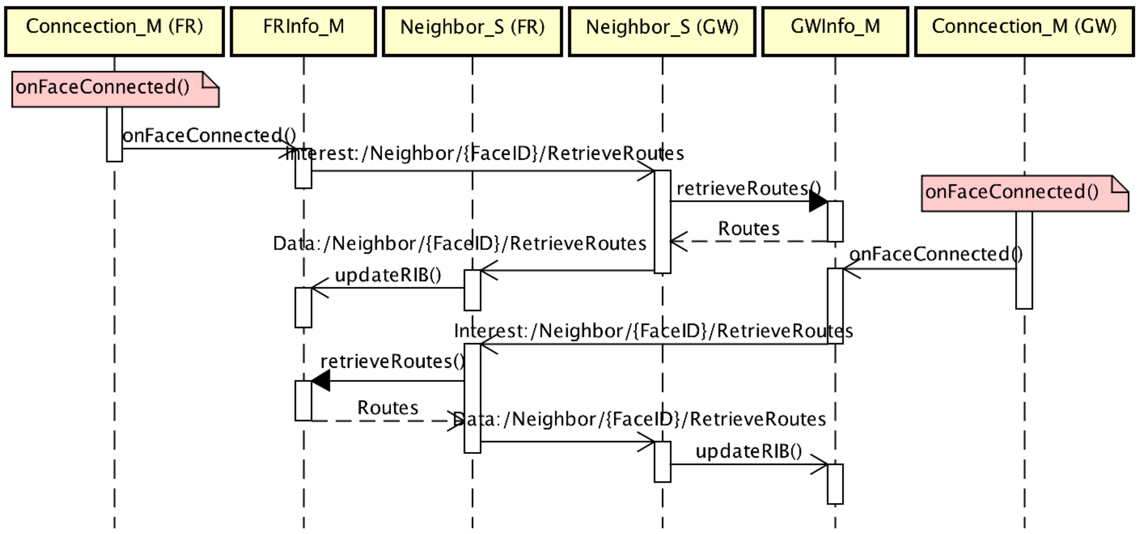


Figure 20: A sequence diagram starting from onFaceConnected() of Connection Manager when FR and GW are connected

onFaceConnected() of Connection Manager calls onFaceConnected() of Info Manager. Then, onFaceConnected() of Info Manager retrieves route information of the connected node by calling retrieveRoutes(). Unlike exchange of information between DR and FR, exchange of information between FR and GW is bilateral symmetry because it is only necessary to retrieve mutual information asynchronously. As a result, the FR can retrieve the route information in the Inter-region network composed by GWs, and the GW can also retrieve the route information of the Inter-region network from the FR. Note that retrieveRoutes() implemented in Info Manager does not retrieve all the route information registered in RIB, but retrieves only those entries whose next hop is not the connected node. This is the same not only for exchanging route information between FR and GW but also between DR and FR, and between GW and Node.

On the other hand, message exchange is very simple in RMICN. Normal messages are picked up by Message Strategy, and only passed to Buffer Manager. A sequence diagram relating to message exchange is shown in Figure 21.

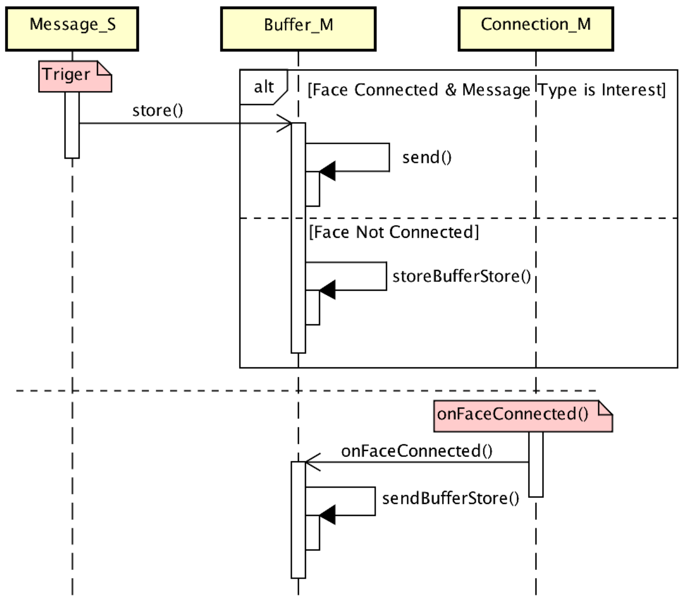


Figure 21: A sequence diagram relating to message exchange

store() of Buffer Manager called by Message Strategy sends the message if the Face is connected and furthermore the type of the message is Interest. If it is not connected, Buffer Manager stores the message in Buffer Store. When onFaceConnected() of Buffer Manager is called by Connection Manager, all messages that are destined to the Face in Buffer Store are sent.

The above is the method of information and message exchanging method in RMICN. As you can see from Figure 19, Path Manager executes wait() and searchAlternativePath() every time the FR visits an Intra-region network. wait() is called to wait for the time to exchange route information and messages by connecting a Face, and it has a function to wait for $T_w$ secs (Figure 18 in DRPath Manager is the same function and it is used for the same reason). searchAlternativePath() is a method that refers to Buffer Store and performs autonomously changing the crawling path when a better path is found that reduces the average retrieval time of contents. The details are explained in Section 5.2. Then, after finishing all the crawls, FRs return to Depot as shown in Figure 19, and exchange messages by the sequence of Figure 21, besides, advertises the information gathered from Inter-region subnetworks by the sequence of Figure 17. Then, when all FRs advertise information to the DR, RMICN shifts to the next crawling by the sequence of Figure 18. At this time, since the Inter-region network has already been constructed, crawls are performed by calling only crawl() instead of deliver().

## 4.3 Applicative Extended Control with Names and Strategies

In RMICN, FRs are used as movable routers that support disjoint-networks communication, but FR can also provide various application control based on strategies in the naming scheme although it is a relay node. In this section, we introduce content control and drone control as applicative extended control.

### 4.3.1 Content Control

Content control indirectly controls a FR by adding meta information to content. It can perform various control according to the desires of the content unit, for example, it increases the moving speed of the FR when carrying the content with high urgency. When there is a content to which a user wishes to give a characteristic, the user adds a name representing the characteristic to the original content name as a prefix, and the user sends it to the FR. Then, the strategy based on the received message performs the following processing as a base.

(1) When receiving a message, the strategy obtains the destination by referring to FIB using the name, excluding the prefix, representing the characteristic (However, there are cases where the destination does not exist in FIB depending on the characteristics, such as the characteristic to be distributed to all nodes).

(2) It performs control according to the characteristic (e.g., Increasing moving speed for content with high urgency).

(3) When forwarding a message, it removes the prefix representing the characteristic from the name. In addition, when the message is Interest, register the event listener on the Face in the same way as NDN applications so that other strategy does not take the returned data. Then, it sends the Interest directly from the Face and retrieves the Data returned by the registered event listener.

(4) When the Data is retrieved by an event listener, the destination is obtained by referring to PIT using the name, including the prefix, representing the characteristic. After that, the control according to the characteristic is performed, and the Data is returned to the destination.

### 4.3.2 Drone Control

Drone control performs control utilizing the drone which is the movement body of the FR. As control contents, for example, it is expected to retrieve aerial photographs and sensor information (temperature, humidity, radiation, etc.) at arbitrary places and to spray herbicides and fertilizers in smart agriculture. In this way, a drone is required to be on the arbitrary place after its movement. When strategy of drone control receives an Interest that needs to control movement to an arbitrary place, it performs the following processing.

(1) The Interest is discarded if the requested location expressed by the name of the Interest is outside the area of RMICN, but if it is within the area, the GW closest to the location is obtained by referring to Node Table.

(2) Next, it checks whether the GW is within its own Inter-region subnetwork referring to Path Table. If it is included, it moves the FR to the requested position either before or after visiting the GW at crawl and executes the drone control. If it is not included, the Interest is sent to the DR when it arrives at Depot.

(3) The DR obtains the GW which is the requested location nearest in the same procedure as the FR and transfers the Interest to the FR which is handling the GW as a part of its Inter-region subnetwork.

(4) The FR receiving the Interest moves to the location either before or after visiting the GW closest to the requested location and executes the drone control.

(5) As a result of the drone control, when Data such as aerial photographs and sensor information is generated, the Data is returned to the requesting Intra-region network by forwarding processing of RMICN.

# 5 Algorithms of Path Planning

In this section, we describe algorithms for calculating the crawling path of the Inter-region network and algorithms for autonomously changing the paths of the FRs crawling along the Inter-region subnetwork.

## 5.1 Calculating Inter-region network

In this section, we describe an algorithm for calculating the crawling path of the Inter-region subnetwork. Inter-region network should be divided into Inter-region subnetwork considering not only the number of FRs within each subnetwork but also total crawling distance of all FRs necessary for configuring the Inter-Region Networks. Therefore, in this research, we focus on the fact that this problem is very similar to Vehicle Routing Problem (VRP) [21], and we propose to calculate the Inter-region network by solving the VRP.

### 5.1.1 Vehicle Routing Problem (VRP)

VRP supposes a set of customers geographically separated from each other and multiple vehicles at base or Depot which deliver something to all customers. Then, VRP is a problem of obtaining the path until they return to Depot again. This problem is the most commonly used when considering packing pickup algorithms at shipping companies. For example, considering a VRP such as Figure 22, it is understood that this is the solution of VRP where four vehicles pick up packages crawling from Depot $c_0$ along thirteen customers. VRP is very similar to the problem of seeking an Inter-region network as you can see Figure 22. So, it is possible to model the problem in our research by taking a customer as an Intra-region network and a vehicle as an FR. Therefore, it can be used as a problem to calculate the Inter-region network as it is by ignoring the baggage amount owned by the customer or the cargo restriction of the vehicle in the VRP. However, VRP is NP-hard and it is known that it is difficult to obtain an optimal solution. In this paper, IACO (Improved Ant Colony Optimization) [22] is used as a meta heuristic method for calculating the Inter-region network.

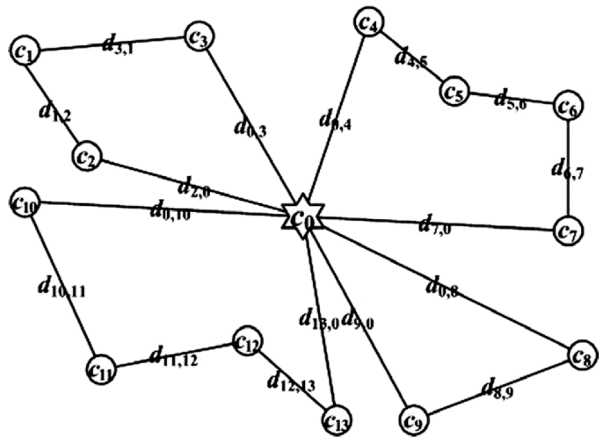


Figure 22: An example of VRP (cited from the paper [22])

### 5.1.2 Improved Ant Colony Optimization (IACO)

ACO is an algorithm to calculate the route by simulating the behavior of ants to search for bait, and it is simple to RMICN by considering an ant as an FR, a location of bait as an Intra-region network and an ant nest as Depot. IACO [22] introduces the concept of mutation in Generic Algorithm to ACO, and compares their proposal to other meta heuristic methods to solve VRP. The time to convergence is not fast but high accuracy. Therefore, in consideration of the importance of building an Inter-region network with a better route, IACO is adopted as an algorithm to calculate an Inter-region network in our research. In addition, although this is one of future works, if it is possible to reflect not only the location of bait (the location of Intra-region network) but also the amount and quality of bait (content) on the ant pheromone, it is thought that it becomes possible to construct the topology of Inter-region network reflecting the information.

## 5.2 Calculating Alternative Path

As shown in Figure 19, an FR crawling in the Crawling Phase moves between GWs of its Inter-region subnetwork, and autonomously changes the path every time when it arrives at the location of a GW by calling searchAlternativePath(). searchAlternativePath() autonomously changes the path by referring to Buffer Store within the guideline of the content retrieval time obtained from the crawling path of the Inter-region network. As described in Section 3.5.2, the estimate of the content retrieval time $T$ from the view point of users in RMICN is $T = 4T_r$, where $T_r$ is the time taken for all the FRs to meet together at Depot. Here, the number of the Inter-region subnetworks (that is, the number of FRs) is $N$, the number of GWs in the Inter-region subnetwork $i$ is $M_i$, the time taken for crawling along the Inter-region subnetwork $i$ is $CT_i$ and the time to wait in wait() of Path Manager is $T_w$. Then, $T_r$ is obtained by Eq.(2).

$$\begin{aligned} z &= \arg\max_{1 \leq i \leq N}(CT_i) \\ T_r &= CT_z + M_z \cdot t_w \end{aligned} \tag{2}$$

Therefore, the real time $DT_i$ that can be spent by autonomous path change in a single crawl along the Inter-region subnetwork $i$ is obtained by Eq.(3) .

$$DT_i = T_r - (CT_i + M_i \cdot t_w) \tag{3}$$

searchAlternativePath() retrieves a list of GWs, constituting its Inter-region subnetwork, that have already been visited in the current crawl and where the number of buffered messages destined is not zero. Then, within $DT_i$, the GWs which can reciprocate from the currently connected GW are searched from the list. At this time, Eq.(4) tells whether the selected GW can reciprocate from the current location..

$$DT_i > 2(MT + T_w) \tag{4}$$

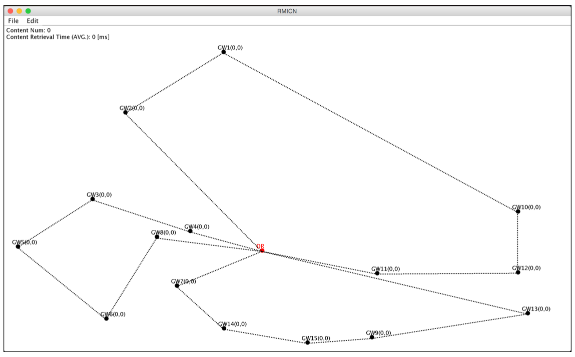
Here, $MT$ represents the time taken to move from the currently connected GW to the selected GW. Furthermore, if there are multiple Intra-region networks satisfying Eq.(4), the GW where the number of the buffered messages destined is the largest. Note that $DT_i$ decreases by $2(MT + T_w)$ when the FR changes the path, and the value will not be recovered until the current crawl is over.

# 6 Evaluation of RMICN

This chapter describes the evaluation of RMICN. In Section 3.2, we describe the design policy of RMICN, and as a design advantage, we describe that RMICN can construct a network that utilizes flexible control with name, content-based parameter, and content cache. In this chapter, therefore, we aim to evaluate 1) the degree of autonomous path change at FRs and 2) how much content retrieval time can be reduced due to the operation of in-network cache in RMICN.

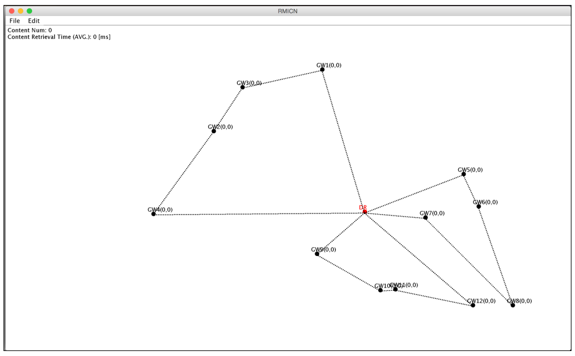
## 6.1 Evaluation Methods

In this paper, we evaluate the content retrieval time in RMICN in simulation environment. As a comparison target, we use DTN (Message Ferry) which is one of the methods to realize disjoint-network communication, and for fairness we give the Inter-region network in RMICN as a crawling path in DTN. Then, we use the five patterns of layout information (whose scale is 500 [m] square) assuming the real environment as the location of GWs constituting Intra-region networks, and the number of crawling FRs is set to three. Figure 23 - Figure 27 shows the location of wireless nodes used as the evaluation data and the calculated Inter-region network.



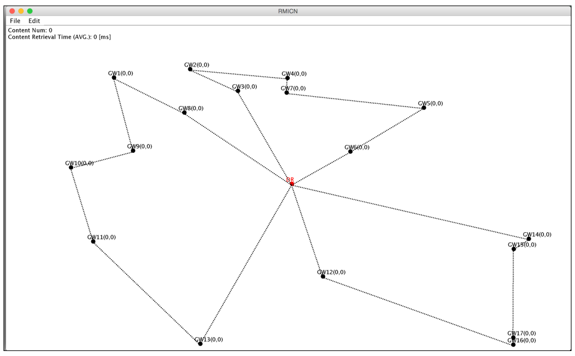
Content Num: 0
Content Retrieval Time (AVG.): 0 [ms]

Figure 23: Location of wireless nodes used in the evaluation and the calculated Inter-region network: CASE1



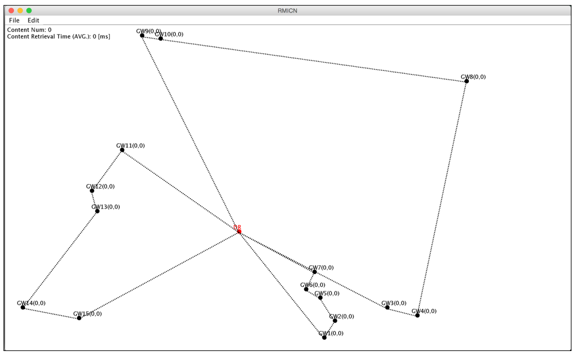
Content Num: 0
Content Retrieval Time (AVG.): 0 [ms]

Figure 24: Location of wireless nodes used in the evaluation and the calculated Inter-region network: CASE2



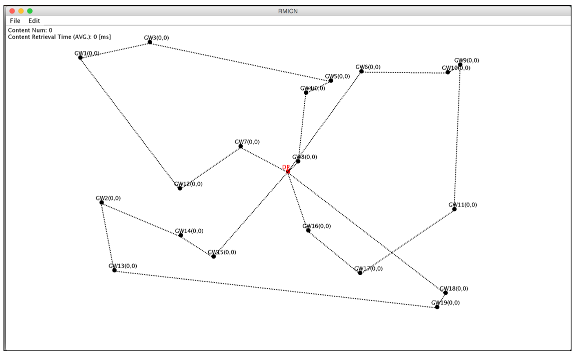
Content Num: 0
Content Retrieval Time (AVG.): 0 [ms]

Figure 25: Location of wireless nodes used in the evaluation and the calculated Inter-region network: CASE3



Content Num: 0
Content Retrieval Time (AVG.): 0 [ms]

Figure 26: Location of wireless nodes used in the evaluation and the calculated Inter-region network: CASE4



Content Num: 0
Content Retrieval Time (AVG.): 0 [ms]

Figure 27: Location of wireless nodes used in the evaluation and the calculated Inter-region network: CASE5

In the evaluation, the average retrieval time of contents used as an evaluation parameter is defined as the average of the time taken for the node in the Intra-region network to receive Data after transmitting Interest. As a traffic pattern, a pattern assuming the following two scenarios is used to evaluate autonomous path change of FRs and content cache. In both cases, which content is requested by which node is randomly determined.

**Sensor Networks**

This is a pattern assuming sensor networks as a scenario. In this scenario, it is a major feature that the same content is not requested more than once. This is because content such as sensor information in sensor networks is strong in real time property. Therefore, it is expected to reduce content retrieval time only by autonomous path change of FRs.

**Disaster Networks**

This is a pattern assuming disaster networks as a scenario. In this scenario, it is a major feature that the same content is requested multiple times. This is because there are contents requested by many users such as disaster information. Therefore, it is expected to reduce content retrieval time not only by autonomous path change of FRs but also by contents cache.

Finally, Table 22 shows the parameters used in the evaluation.

Table 22: Parameters used for the evaluation

| Parameter | Value |
|---|---|
| Movement speed of all FRs | 50 [km/h] |
| Number of packets requesting contents | 1000 |
| Probability for requesting the same content | 0.1 |
| FR's standby time at Intra-region network: $T_w$ | 1 [s] |

## 6.2 Evaluation Results

In this section, the results of simulation carried out in Section 6.1 are indicated. The average of the content retrieval time in the scenario of sensor networks is shown in Table 23 and Figure 28, and the average of the content retrieval time in the scenario of disaster networks is shown in Table 24 and Figure 29.

Table 23: The average of content retrieval time in the scenario of sensor networks

| Method | CASE1 | CASE2 | CASE3 | CASE4 | CASE5 | AVG. |
|---|---|---|---|---|---|---|
| DTN [s] | 179.6 | 120.6 | 164.5 | 199.6 | 164.3 | - |
| RMICN [s] | 165.9 | 111.0 | 157.4 | 164.3 | 165.1 | - |
| Rate [%] | 92.3 | 92.1 | 95.7 | 82.3 | 100.5 | 92.6 |

Table 24: The average of content retrieval time in the scenario of disaster networks

| Method | CASE1 | CASE2 | CASE3 | CASE4 | CASE5 | AVG. |
|---|---|---|---|---|---|---|
| DTN [s] | 184.5 | 119.6 | 165.7 | 190.7 | 161.2 | - |
| RMICN [s] | 157.3 | 101.6 | 150.1 | 151.2 | 147.9 | - |
| Rate [%] | 85.2 | 84.9 | 90.7 | 79.3 | 91.7 | 86.4 |

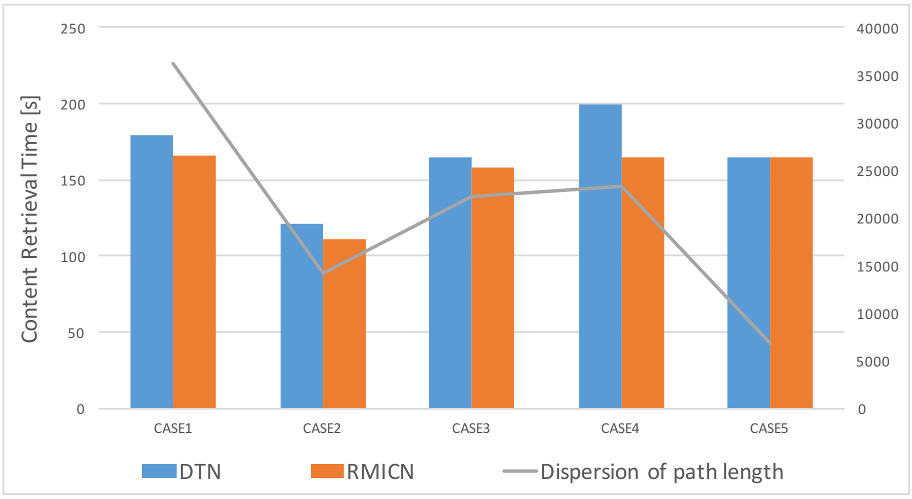


Figure 28: The average of content retrieval time in the scenario of sensor networks

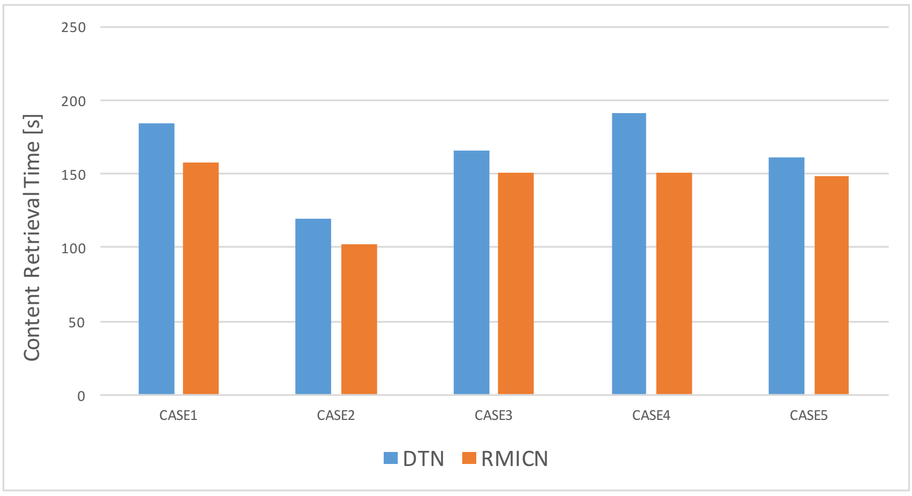


Figure 29: The average of content retrieval time in the scenario of disaster networks

As shown by these results, in RMICN, it is understood that content retrieval time is reduced more than DTN in many cases in both scenarios: sensor networks and disaster networks.

In the scenario of sensor networks, RMICN demonstrates smaller retrieval time in all five cases compared to DTN. However, the difference is marginal in CASE 5, because the variance of the path length of the Inter-region subnetwork is significantly smaller than one in other cases. Since autonomous path change of FRs in RMICN is performed when there is a margin for meeting time at Depot while crawling, if the meeting time is short, that is, the dispersion of the path length of the Inter-region subnetworks, the effect becomes trivial. However, when the dispersion of the path length of the Inter-region subnetwork is large, the effect becomes dominant, and so it is considered to be useful, for example, when the number of Intra-region networks increases and the construction accuracy of the Inter-region network is low.

In the scenario of disaster networks, the reduction of the content retrieval time was confirmed in all cases. Figure 30 shows comparison of content retrieval time of RMICN in sensor networks and disaster networks. This shows the effect of content cache only in RMICN except the utility of autonomous path change, the effect reduces content retrieval time by an average of 7.4 [%]. In this simulation, the probability for requesting the same content is set to 0.1, but in the case where this value becomes large, the content retrieval time can be further shortened.

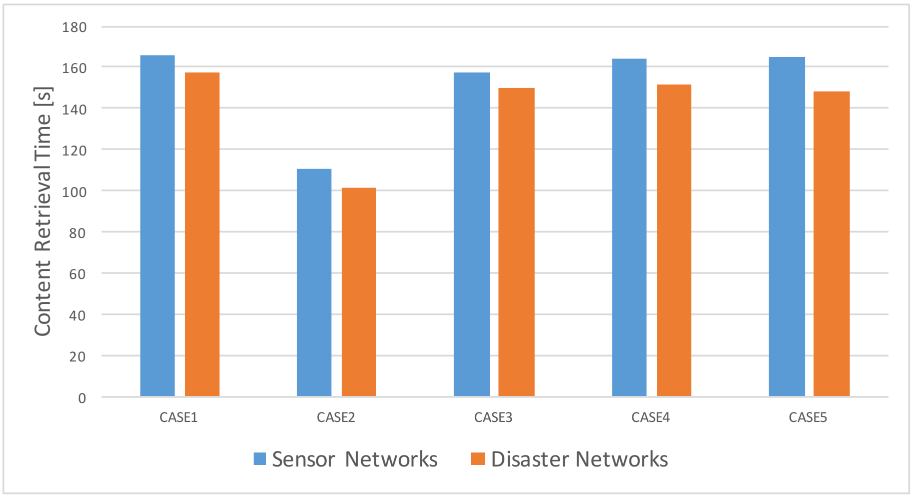


Figure 30: The effect of content cache in RMICN

Finally, we describe the calculation cost of path planning of FRs. The CPU of the computer used in the simulation is Intel Core i7 (3 GHz). The average computation time taken to calculate the Inter-region network is 1.73 [s]. On the other hand, the time taken to search autonomously changed paths by FRs is on the order of 0.1 [ms] on average, which can be negligible compared with the content retrieval time.

# 7 Conclusions and Future Work

In this research, we focused on flexible control utilizing the name scheme of ICN, and we proposed and designed RMICN, which is ICN with movement-controllable routers, and demonstrated the advantages of realizing movement control of routers using ICN. Since RMICN is realized by seamless extension of NDN, it can inherit all feasible functions from NDN. In addition, we also show that there are many advantages for disjoint-network communication by utilizing control with name, content-based parameter, and content cache that are unique features of ICN. With unique strategies defined in the naming scheme in NDN, it is possible to control the movement of the router by Interest, and possible to manipulate routers flexibly according to the characteristics of the contents. Furthermore, it is also possible to provide a flexible control foundation which is applicable to various applications. In addition, utilizing the content-based parameter, we propose path planning which delivers all contents within the guideline of the estimated content retrieval time because the communication primitive of NDN is a content and the communication is made by one-to-one exchange of Interest and Data. Besides, utilizing content cache, we actually simulated RMICN on a computer and confirmed that the retrieval time of the content is shortened due to the effect of the cache in the scenario where the same content is requested multiple times such as disaster networks.

As future tasks, we plan to further utilize content-based parameter for the method of constructing the Inter-region network and of autonomous path planning of routers. Specifically, we also aim to realize more flexible content delivery by creating a distribution of contents and content requests with FRs and changing the movement path of FRs according to the distribution.

# Acknowledgements

This thesis would not be accomplished without the united efforts and supports of a great number of people. First, I would like to express my the deepest appreciation to my supervisor, Professor Masayuki Murata of Osaka University, for his valuable time, advices, and continuous supports. Furthermore, I would like to show my tremendous gratitude to Professor Shingo Ata of Osaka City University, for his significant guidance, precise advices and various cares. Moreover, I would like to offer my significant thanks to Associate Professor EUM Suyong of Osaka University, for his instructive advice and the very kind English guidance. I also would like to express my gratitude to Associate Professor Shin'ichi Arakawa, Assistant Professor Yuchi Ohsita, Daichi Kominami and Specially Appointed Assistant Professor Naomi Kuze for their accurate suggestions and advices. Furthermore, I would like to offer my special thanks to Mr. Ryo Nakano of Hitachi, Ltd., Yokohama Research Laboratory. He offers us the data used for evaluation of our proposal. I could complete this thesis thanks to him. In addition, I am deeply grateful to Mr. Atsushi Ooka. He always leads me friendly to the right direction and gives me very useful points. Mr. Tatsuya Tanaka and Mr. GAO YUHAO also helps me and they are very good research fellows for me and I appreciate for them. Finally, I owe my gratitude to all my friends, colleagues and secretaries of the Advanced Network Architecture Research Laboratory of Osaka University. This research and development work was supported by the MIC/SCOPE #165007007.

# References

[1] V. Jacobson, D. K. Smetters, J. D. Thornton, M. F. Plass, N. H. Briggs, and R. L. Braynard, "Networking named content," in *Proceedings of the 5th International Conference on Emerging Networking Experiments and Technologies (CoNEXT)*, pp. 1–12, Dec. 2009.

[2] L. Zhang, D. Estrin, J. Burke, V. Jacobson, J. Thornton, D. K, *et al.*, "Named data networking (NDN) project," Tech. Rep. NDN-0001, PARC, Oct. 2010.

[3] V. Jacobson, D. K. Smetters, N. H. Briggs, M. Plass, P. Stewart, *et al.*, "VoCCN: Voice-over Content-Centric Networks," in *Proceedings of the 2009 Workshop on Re-architecting the Internet*, pp. 1–6, Dec. 2009.

[4] Y. Liu, J. Geurts, J.-C. Point, JCP-Consult, F. S. Lederer, B. Rainer, *et al.*, "Dynamic adaptive streaming over CCN: a caching and overhead analysis," in *Proceedings of 2013 IEEE International Conference on Communications*, pp. 3629–3633, Jun. 2013.

[5] W. Shang, A. Bannis, T. Liang, Z. Wang, Y. Yu, *et al.*, "Named data networking of things," in *Proceedings of 2016 IEEE First International Conference on Internet-of-Things Design and Implementation (IoTDI)*, pp. 117–128, Apr. 2016.

[6] M. Amadeo, C. Campolo, A. Iera, and A. Molinaro, "Named data networking for iot: An architectural perspective," in *Proceedings of 2014 European Conference on Networks and Communications (EuCNC)*, pp. 1–5, June 2014.

[7] NFD Team, "NFD Management protocol." `https://redmine.named-data.net/projects/nfd/wiki/Management`. Accessed: 2016-11-01.

[8] NFD Team, "NFD Developer's Guide." `https://named-data.net/wp-content/uploads/2016/10/ndn-0021-7-nfd-developer-guide.pdf`. Accessed: 2016-11-01.

[9] NFD Team, "Control Command." `https://redmine.named-data.net/projects/nfd/wiki/ControlCommand`. Accessed: 2016-11-01.

[10] W. Zhao and M. H. Ammar, "Message ferrying: proactive routing in highly-partitioned wireless ad hoc networks," in *Proceedings of The Ninth IEEE Workshop on Future Trends of Distributed Computing Systems*, pp. 308–314, May 2003.

[11] R. C. Shah, S. Roy, S. Jain, *et al.*, "Data mules: Modeling a three-tier architecture for sparse sensor networks," *Ad Hoc Networks*, vol. 1, pp. 215–233, Sep. 2003.

[12] R. Sugihara and R. K. Gupta, "Path planning of data mules in sensor networks," *ACM Transactions on Sensor Networks*, vol. 8, pp. 1–27, Aug. 2011.

[13] J. S. Liu, S. Y. Wu, and K. M. Chiu, "Path planning of a data mule in wireless sensor network using an improved implementation of clustering-based genetic algorithm," in *Proceedings of 2013 IEEE Symposium on Computational Intelligence in Control and Automation (CICA)*, pp. 30–37, Apr. 2013.

[14] O. Tekdas, V. Isler, J. H. Lim, and V. Isler, "Using mobile robots to harvest data from sensor fields," *IEEE Wireless Communications*, vol. 16, pp. 22–28, Feb. 2009.

[15] O. Tekdas, W. Yang, and V. Isler, "Robotic routers," in *Proceedings of 2008 IEEE International Conference on Robotics and Automation*, pp. 1513–1518, May 2008.

[16] J. Alex Halderman, "NDN: A Security Perspective." `https://named-data.net/wp-content/uploads/2015/06/fiapi-2015-security-perspective.pdf`. Accessed: 2015-11-16.

[17] J. Chen, M. Arumaithurai, L. Jiao, X. Fu, and K. Ramakrishnan, "Copss: An efficient content oriented publish/subscribe system," in *Proceedings of 2011 Architectures for Networking and Communications Systems (ANCS)*, pp. 99–110, Oct. 2011.

[18] K. Kim, S. Choi, S. Kim, *et al.*, "A push-enabling scheme for live streaming system in content-centric networking," in *Proceedings of the 2013 workshop on Student workshop*, pp. 49–52, Dec. 2013.

[19] J. François, T. Cholez, and et al., "CCN Traffic Optimization for IoT," in *Proceedings of 2013 Fourth International Conference on the Network of the Future (NoF)*, pp. 1–5, Oct. 2013.

[20] T. Kitagawa, S. Ata, and M. Murata, "Retrieving information with autonomously-flying routers in information-centric network," in *Proceedings of 2016 IEEE International Conference on Communications (ICC)*, pp. 1–6, May 2016.

[21] G. Laporte, "The vehicle routing problem: An overview of exact and approximate algorithms," *European Journal of Operational Research*, vol. 59, pp. 345–358, Jun. 1992.

[22] B. Yu, Z.-Z. Yang, and B. Yao, "An improved ant colony optimization for vehicle routing problem," *European Journal of Operational Research*, vol. 196, pp. 171–176, Jul. 2009.