An Information Look up System using Geographic Location-based Distributed Routing Table

by

KUMIKO KOBAYASHI

Submitted to

The University of Electro-Communications

In partial fulfillment of the requirements

for the degree of

Doctor of Philosophy

Department of Information Management Science

Graduate School of Information Systems

The University of Electro-Communications

March 2014

An Information Look up System using Geographic Location-based Distributed Routing Table

Approved by Supervisory Committee:

地理位置情報に基づく分散ルーティングテーブルを用いた情報検索システム

小林　久美子

概　要

本論文では，都市部において移動車両が収集した情報を有効活用することを目的とし，収集されたデータの場所に対応するエリアノードから構成されるオーバレイネットワークのための地理的な位置に基づく分散型ルーティング (GDR) を提案している．GDR を用いたシステムでは，グリッド状エリア毎に移動車両が取得した情報を管理するノードを 1 つ配置する．各ノードは地理的な位置関係を反映したオーバーレイネットワークを構築し，2 次元空間において緯度と経度の座標に基づいた情報検索を提供する．その前提条件として，GDR は都市部の道路で構成されたその形状がグリッドとしてモデル化できるアンダーレイネットワーク上で動作することを想定している．GDR では，エリアノードの ID を，従来法で用いられて来た空間充填曲線 (SFC) を使用せずに，地理的な位置を反映するように生成し，デカルトフォーマット (x, y) で表す．N 個のノードで構成される GDR は，各ノードはサイズ $\log N$ のルーティングテーブルを持ち，$O(\log N)$ 回のホップ数でノード検索が可能である．

論文では，提案方式の性能を，ホップ数ならびにノード間距離の総和である中継距離を評価関数としてそれらの平均・分散によって評価する．さらに，代表的な分散型ルーティングシステムである，Chord，Kademlia, CAN に対しても，同じ評価関数の平均・分散を新たに理論的に導出する．それらのシステムと比較した結果，GDR の検索ホップ数の平均と分散は Chord と

Kademlia と同じであるが，検索ホップ時にたどるノード間距離の合計である中継距離の平均と分散は Chord と Kademlia より小さいことが分かった．また，GDR の中継距離の平均と分散は CAN と同じであるが，検索ホップ数の平均と分散は CAN よりも小さいことが分かった．

それらの結果から，GDR の平均中継距離は chord の約 1/2 倍，Kademlia の約 2/3 倍小さく，平均ホップ数は CAN の約 $(3/4)\log N/\sqrt{N}$ 倍小さくできることが明らかになった．

さらに，SFC を使用して生成した ID を用いたシステムにおいて Chord および Kademlia のシミュレーションを行った結果に対しても同様であった．

アンダーレイネットワークがグリッド状であれば，物理アドレスの指定は容易である．しかし，あるノードが故障した場合にそのアドレスの修正や変更は困難である．そこで，GDR のロバスト性を高めるためのエージェントリストを提案している．GDR の各ノードに対して，それが有するルーティングテーブルに登録されているノード（登録ノード）ごとに，その代理として，登録ノードに隣接する 2 つのノード（エージェントノード）を登録したエージェントリストを付け加える．エージェントリストのサイズは $2\log N$ である．GDR のノードは，エージェントリストを利用してオーバーレイネットワーク上で故障したノードを回避できる．計算機シミュレーションより，エージェントリストを加えた GDR はノードに障害が発生した場合に対応できることを示した．

最後に，GDR の応用として *Wall Pass* (*WP*) アルゴリズムを提案している．*WP* アルゴリズムは移動端末からの問い合わせ結果を移動端末が進行する方向の隣接エリアにメッセージを転送する機能である．GDR システムを構成するエリアノードは，サッカーの wall pass における wall player のように振る舞い，各ノードはメッセージの種類とその位置情報に基づいて受信した

メッセージを処理する．Manhattan Model に基づく移動端末を用いた計算機シミュレーション結果から，*WP* アルゴリズムによりシステムの通信負荷が削減されることを示した.

An Information Look up System using Geographic Location-based Distributed Routing Table

Kumiko Kobayashi

Abstract

In this thesis, we propose an information look up system using geographic location-based distributed routing (GDR) table that collects and manages information gathered by moving vehicles in urban areas. Throughout this thesis, we assume the underlay network of the GDR system can be modeled as a grid. This assumption makes a sense for an urban area where the roads are paved on a grid pattern. The system uses area nodes placed on several locations where each node manages location-oriented information on a designated non-overlapping area. The GDR system provides an information lookup based on the geographic latitude and longitude coordinates. A geographic coordinate is assigned for a node as its identifier (ID), and each node manages an overlay routing table. The routing table consists of pointers to other nodes in the network in order to forward messages to the geographically nearest overlay node toward its final destination. In a system with N nodes, each node has a routing table of size $\log N$ and a search is possible in $O(\log N)$.

We evaluate the mean and the variance of the *path length* and the *relay length* of GDR, CAN, Chord and Kademlia, under the assumptions that the ID is in

cartesian format (x, y), all nodes are active, and the source node and the destination node are chosen independently with equal probability.

We show that regardless of the ID format (i.e. even though the ID is in cartesian format or the ID is generated by using Space Filling Curve (SFC)), GDR, Chord and Kademlia have the same mean and the same variance of the *path length*, while the mean and the variance of the *relay length* of GDR are smaller than those of Chord and Kademlia. Furthermore, while GDR and CAN have the same mean and the same variance of the *relay length*, the mean and the variance of the *path length* of GDR are smaller than those of CAN. We show that the mean *relay length* of GDR is about half of that of Chord, and about $2/3$ of that of Kademlia, and the mean *path length* is about $(3/4)\log N/\sqrt{N}$ of that of CAN.

In addition, the GDR system has a routing redundancy to increase robustness. When a node fails, its neighbor node behaves as an agent for the failing node. To know the agent node of the failing node, each node has an agent list which is the records of the agent nodes of the nodes of its routing table. Since the number of the agent nodes is 2, the size of the agent list is $2\log N$. If an underlay network can be modeled as a grid, it is easy to assign a physical address for a node. However, if a node fails, it is difficult to modify or change its physical address. In the GDR system, the nodes can avoid a failed node by using its agent list on the overlay network.

We also present an application of the GDR system. In order to send a reply to a terminal after it moves to the neighboring area, we proposed $Wall\ Pass$ (WP) algorithm. We consider a node as a wall player of wall pass in football. We evaluated the performance of the GDR system when the mobile mobile terminals are moving. The results show that WP algorithm can decrease the communication overhead.

Contents

List of Notations and Terminology

D_{c_x}	the relay length of a Chord system for the horizontal direction
D_{c_y}	the relay length of a Chord system for the vertical direction
D_c	the relay length of a Chord system. $D_c = D_{c\_x} + D_{c\_y}$
H_{c_x}	the path length of a Chord system for the horizontal direction
H_{c_y}	the path length of a Chord system for the vertical direction
H_c	the path length of a Chord system. $H_c = H_{c\_x} + H_{c\_y}$
D_{k_x}	the relay length of a Kademlia system for the horizontal direction
D_{k_y}	the relay length of a Kademlia system for the vertical direction
D_k	the relay length of a Kademlia system. $D_k = D_{k\_x} + D_{k\_y}$
H_{k_x}	the path length of a Kademlia system for the horizontal direction
H_{k_y}	the path length of a Kademlia system for the vertical direction
H_k	the path length of a Kademlia system. $H_k = H_{k\_x} + H_{k\_y}$
D_{n_x}	the relay length of a CAN system for the horizontal direction

D_{n_y}	the relay length of a CAN system for the vertical direction
D_n	the relay length of a CAN system. $D_n = D_{n\_x} + D_{n\_y}$
H_{n_x}	the path length of a CAN system for the horizontal direction
H_{n_y}	the path length of a CAN system for the vertical direction
H_n	the path length of a CAN system. $H_n = H_{n\_x} + H_{n\_y}$
D_{g_x}	the relay length of a GDR system for the horizontal direction
D_{g_y}	the relay length of a GDR system for the vertical direction
D_g	the relay length of a GDR system. $D_g = D_{g\_x} + D_{g\_y}$
H_{g_x}	the path length of a GDR system for the horizontal direction
H_{g_y}	the path length of a GDR system for the vertical direction
H_g	the path length of a GDR system. $H_g = H_{g\_x} + H_{g\_y}$
$\mathbb{E}(X)$	the mean of a random variable X
$\mathbb{V}(X)$	the variance of a random variable X
$node(x, y)$	a node at longitude position x and latitude position y
T_h	a routing table which is used to forward messages to the horizontal direction
T_v	a routing table which is used to forward messages to the vertical direction

T_{h_a}	a agent list which is used to modify T_h, if node fails in T_h
T_{v_a}	a agent list which is used to modify T_v, if node fails in T_v
hop	the number of nodes receiving a query, where these nodes are selected from routing table of a query sending node, until the query arrives at the destination node on the overlay network
path length	the number of hops on the overlay network
relay length	the sum of geographical distances between consecutive two nodes in a *path length*

Chapter 1

Introduction

1.1 Background

The ubiquitous nature of mobile devices and their ability to gather various location-oriented information through embedded sensors has led to several new services over wireless network infrastructures. For example, service providers may gather location-oriented information from mobile devices and process this information for traffic, weather, and environment services and applications. Under the present status of technologies, most service providers control the collection and processing of the information intensively on central servers. However, if all information were stored on a single machine, its memory complexity would increase in proportion to the number of items. In fact, it is expected that the demand for the information gathered by mobile devices will significantly increase in the near future. Therefore, we have to investigate more scalable and reliable schemes.

Furthermore, as various devices or machines are able to communicate with each other autonomously over networks, a new service called Machine-to-Machine

(M2M) service, is becoming popular. In M2M, it would be possible for a machine to access remote location-oriented information by communicating with other networked machines. To realize M2M services, the system infrastructure is required to collect and store the information from each device. In addition, to exchange the location-oriented information efficiently in the system, it is important for the system to be provided a geographical distributed information lookup and routing.

1.2 Architecture of location-oriented information service

An example of the location-oriented information services is Probe information system [1]. The system regard the vehicles as the moving sensors. The location-oriented information through the moving sensors is gathered over the wireless network infrastructure. In this case, the location-oriented information is controlled the collection and processing of the information intensively on central server.

The location-oriented information is sent from a vehicle to the access points of the wireless network. If the access points could collect and store the location-oriented information, it is possible to build a distributed system. Moreover, the distributed system provide a geographical distributed information lookup and routing, it can exchange the location-oriented information efficiently.

1.3 Distributed Routing Table

Routing is a mechanism to select the best node or path to send a message to its destination in a network. In routing, normally, upon receiving a message, a network node searches for a relay node which has the closest distance to the destination node of the message in its routing table entries. The selected nodes along the path repeat the process until the message arrives at its final destination.

A distributed system, which is composed by the collection of the participating nodes for the same purpose, often called the overlay network or overlay system. The typical methods for lookup a data in the distributed systems are a flooding search and a distributed indexes. [2] – [4]

In a flooding search, lookup queries are sent to all nodes participating in the system until the corresponding node is found. If a node receives a query, it floods the query to other nodes up to a fixed number of hops. The advantage of flooding-based systems is that there is not necessary to maintain the network. It is called unstructured overlay network. In a system with N nodes, a search is possible in $O(N^2)$.

A flooding search does not need to manage the network. However, it becomes a problem in terms of the communication overhead. In a distributed indexes, Distributed Hash Table(DHT) is a suitable method for the problem. DHT has the systematic and proactive procedures. It is called a structured overlay network. DHT provides a distributed indexes of various data among many nodes, independent of their actual locations.

In a structured overlay network, a link between a content identifier(ID) and an IP address of a node is usually based on DHT. The nodes in a distributed system are organizing themselves and they manage the references for efficient routing to other nodes. In a system with N nodes, a query is forwarded to a destination node according to DHT routing with $O(\log N)$. Well-known examples of DHT are Chord[5] and Kademlia[6].

We focus on structured distributed routing (SDR) and information lookup systems, such as Chord and Kademlia. They can reduce the number of steps needed to locate information. For example, in a system consisting of N nodes, queries are routed via a small number of nodes to the destination node. Because the routing table in each node has $O(\log N)$ references, a datum can be located by routing across not more than $O(\log N)$ hops. However, the distances calculated from their routing table do not necessarily represent the actual geographical distances.

1.4 Objectives

In this thesis, we are concerned with an SDR system provides a geographical distributed routing and information lookup. To evaluate the distances on the identifier (ID) space, we use the *path length* which is defined to be the number of hops on the overlay network. Furthermore, to evaluate the distances on the actual geographical space, we use the *relay length* which is defined to be the sum of geographical distances between consecutive two nodes in a *path length*.

Our goal is two-folded:

- To propose an SDR system provides a geographical distributed routing and information lookup which the value of the *path length* and the *relay length* as small as possible ; and

- To provide the functions to increase robustness if some nodes fails in the system.

1.5 Approach

We propose an information lookup system using geographic location-based distributed routing (GDR) table that collects and manages information gathered by moving vehicles in urban areas.

In generally, the *relay length* doesn't necessarily represent the actual geographical distance between two nodes since the physical network for an overlay network usually has a complex structures. However, throughout this thesis, we assume the underlay network of the GDR system can be modeled as a grid. This assumption makes a sense for an urban area where the roads are paved on a grid pattern. Therefore, in this thesis, the *relay length* can be interpreted as the actual geographical distance between them.

In this thesis, it is very important to evaluate the *relay length*. Because greater *relay length* means greater propagation time. The value of *path length* and *relay length* should be as small as possible.

The GDR system provides an information lookup based on the geographic latitude and longitude coordinates. Each node is given the geographic coordinates as its ID, and manages an overlay routing table. The routing table consists of pointers to other nodes in the network in order to forward messages to the geographically nearest overlay node toward its final destination. In a system with N nodes, each node has a routing table of size $\log N$ and a search is possible in $O(\log N)$. In fact, a proposal on this kind of system has been reported, e.g., CAN[7]. Our GDR differs from CAN mainly in the routing algorithm and the distance metrics.

We compare our GDR system only to Chord, Kademlia, and CAN, since these three systems contain the original methods. Most of SDR systems adopt the ID generator and the distance metric that are used in Chord, Kademlia, and CAN. In fact, Pastry[8], Tapestry[9], and PHT[14] use, as their ID generators, the hash table used in Chord. GeoPeer[20] and GeoIBM[21] use the cartesian format ID used in CAN.

Moreover, LDHT[15] uses the distance metric based on Chord and Kademlia. Geophony[16] uses the XOR-based metric used in Kademlia. A node of Mill[19] maintains two routing tables, each of which consists of other node's IDs sorted clockwise or counterclockwise based on the logarithmic distance metric used in Chord.

It is noted that the node distances calculated in the systems described above do not necessarily represent the actual geographical distances.

To evaluate the distances on the ID space, we use two parameters: *path length* and *relay length*. The *path length* is defined as the number of hops that processes and forwards the incoming message on the overlay network. Hence, the *path length* is closely related to the total amount of processing time required while forwarding message from a source node to its destination. The *relay length*, on the other hand, is defined as the distance between source node and destination node to indicate the actual geographical distance between those nodes.

To the best of our knowledge, no theoretical analysis for evaluating the SDR systems have been done so far. Therefore, we evaluate the basic performance of the GDR system in the theoretical analysis.

We evaluate the *path length* and the *relay length* of GDR and other systems based on Chord, Kademlia and CAN, under the assumption that the ID is in cartesian format (x, y), all nodes are active, and the source node and the destination node are chosen independently with equal probability. We show that regardless of the ID format(i.e. even though the ID is in cartesian format or the ID is generated by using Space Filling Curve (SFC)[24]), GDR, Chord and Kademlia have the same mean and the same variance of the *path length*, while the mean and the variance of the *relay length* of GDR are smaller than those of Chord and Kademlia. Furthermore, while GDR and CAN have the same mean and the same variance of the *relay length*, the mean and the variance of the *path length* of GDR are smaller than those of CAN.

In addition, the GDR system has a routing redundancy to increase robustness. When a node fails, its neighbor node behaves as an agent for the failing node. To know the agent node of the failing node, each node has an agent list which is the records of the agent nodes of the nodes of its routing table. Since the number of the agent nodes is 2, the size of the agent list is $2\log N$.

If an underlay network can be modeled as a grid, it is easy to assign a physical address for a node. However, if a node fails, it is difficult to modify or change its physical address. In the GDR system, the nodes can avoid a failed node by using its agent list on the overlay network.

We also present an application of the GDR system. In order to send a reply to a terminal after it moves to the neighboring area, we propose *Wall Pass* (*WP*) algorithm. We consider a node as a wall player of wall pass in football. We evaluate the performance of the GDR system when the mobile terminals are moving. The results show that *WP* algorithm can decrease the communication overhead.

1.6 Organization of the Thesis

This thesis is organized as follows.

Chapter 2 discusses some proposals on structured distributed routing (SDR) and information lookup systems and their differences with GDR.

Chapter 3 describes the GDR system in details and evaluate the routing performance of the GDR system in the theoretical analysis.

Chapter 4 describes the details of node fails and evaluates the performance.

In Chapter 5, we evaluate an application of the GDR system.

Chapter 6 summarizes this thesis.

Chapter 2

Structured Distributed Rooting (SDR) and Information Look up system

2.1 Introduction

As we mentioned in Chapter 1, some SDR systems have been proposed to provide a distributed information lookup and routing. They are Chord[5], Kademlia[6], CAN[7], Pastry[8], Tapestry[9], Greedy forwarding[10], GPSR[11], GLS[12], GHT[13], PHT[14], LDHT[15], Geophony[16], SFC[24], Mill[19], GeoPeer[20], and GeoIGM[21].

They can be further classified into three types; namely basic structured distributed routing, geographic routing in wireless mobile ad-hoc network and sensor network, and location-based routing.

In this Chapter 2, we explain the properties of those systems based on their

ID format, and routing table size. We also explain the difference of our system (GDR) to those systems.

2.2 Basic Structured Distributed Routing

2.2.1 Chord

Chord[5] puts nodes on a ring geometry where the IDs of the nodes are computed by using a consistent hashing[26]. A Chord's node maintains a routing table that points to other nodes at logarithmic distance. In case of m-bit ID, the 1-D ID space size is $2^m (= N)$. It is ordered based on an ID circle of modulo 2^m, known as Chord ring. Each node maintains a routing table called finger table of size m. The i-th finger table of node j has information of node $(j + 2^{(i-1)}) \bmod 2^m$, where $1 \leq i \leq m$. In a system with N nodes, node search is possible in $O(\log N)$. Hash function is also used by other systems to create node IDs, such as Pastry[8] and Tapestry[9].

2.2.2 Kademlia

Kademlia[6] runs on a mesh network with an arbitrary ID. The ID space is treated as a binary tree. In case of m-bit ID, the ID space size is $2^m (= N)$. Each node maintains a routing table called k-buckets of size m. The i-th k-bucket of node j has information of nodes which have the first $m - i$ bits matched to j, where $1 \leq i \leq m$. The node information of the maximum k unit is stored in each k-bucket. In a system with N nodes, node search is possible in $O(\log N)$ in XOR-based metric to reduce the lookup ID space.

2.2.3 CAN

CAN[7] uses d-dimensional coordinate space. Each node in CAN only has the addresses of the neighboring nodes. In a system with N nodes, CAN is able to search for destination in $O(\sqrt{N})$.

2.3 Geographic routing in wireless mobile ad-hoc network and sensor network

2.3.1 Greedy forwarding

Greedy forwarding[10] is a popular method for geographic routing in wireless mobile ad-hoc network and sensor network. In greedy forwarding, each node knows its own geographic location and adjacent nodes that are located within its radio range. A source node sends a query to an adjacent node, which is the closest node to destination, and repeats this process until the query reaches the destination node. Greedy forwarding may fail if a forwarding node fails to find the closest adjacent node to destination.

2.3.2 GPSR

GPSR[11] recovers from greedy forwarding failure using *perimeter mode*. In *perimeter mode*, a query is navigated to a node which is closer to destination node around a *void* by the long-known *right-hand rule*.In both greedy forwarding and GPSR, the source node requires know the position of destination node using a location server.

2.3.3 GLS

GLS[12] is a distributed Geographic Location Service, which tracks mobile node locations on wireless mobile ad-hoc network. To enable routing to some nodes, GLS uses a hierarchy of square grids of increasing size and the node IDs are assigned by using hash function. Each node selects three nodes as its location servers in each level of the grid hierarchy. The ID space is considered to be circular. When node X would like to know the current location of node Y, node X send a query to a node which is the closest in ID space to node Y at successively higher levels in the grid hierarchy.

2.3.4 GHT

GHT[13] is a data-centric storage(DCS) in wireless sensor network with a Geographic Hash Table. It hashes the key k into geographic coordinates, and stores the relevant data by name at the nodes. The same key k hash k to the same location. GHT uses GPSR as its underlying routing system, where a node has to know its own geographic position.

2.4 Location-based routing

2.4.1 PHT

In PHT[14], a data structure is designed to support range queries. PHT uses the lookup interface to construct a binary trie-based structure. Each node of the trie is labeled with a prefix. A node with label l is thus assigned to the node to which l is mapped by Distributed Hash Table(DHT).

2.4.2 LDHT

LDHT[15] exploits network locality based on Chord and Kademlia. To embed network topology information into its ID, each node is assigned a locality-aware prefix-based ID.

2.4.3 Geophony

Geophony[16] is a hierarchical version of Symphony[22] based on Cyclone[23] algorithm. In Geophony, suffix-based location ID is assigned in order to map geographic areas and coordinates. Those systems are implemented by combining ID assignment and the existing systems.

2.4.4 LL-Net

LL-Net[17] is a location-based P2P network for context-aware services. The LL-Net defines an area as a square region divided by latitude and longitude. Each area is identified by (xID, yID). Areas are hierarchically grouped into higher-level areas in order to route a query efficiently. Each peer constructs inter-area links to nearby areas densely and to far areas thinly. In a system with N nodes, each node links $(3/2)\log N + k$ nodes, where $k = 0$, 2, and 5.

2.4.5 SFC

SFC[24] is a useful method for 2-D space searching. It divides a 2-D plane onto grid areas, assigns an ID to each area and maps the 2-D ID space into 1-D ID space.

2.4.6 Mill

In Mill[19], ID as a ring is generated by using z-ordering method[25], which is a kind of SFC. When ID-space size is $2^m (= N)$, each node has size of $m/2$ both clockwise and counterclockwise information of the nodes in a routing table. Therefore, the number of nodes in a network is M, node search is possible in $(\log N)/2$ while $M \leq 2\sqrt{N}$.

2.4.7 GeoPeer

GeoPeer[20] reduce the search in $O(\sqrt{N})$ hops in a 2-dimensional CAN. The ID of a node corresponds to its physical location. To reduce the search hops, the nodes select long range contacts(LRCs). If the network size is N, the average number of LRCs per node is $3 \log N$.

2.4.8 GeoIGM

In GeoIGM[21], the nodes are organized on a tesseral address space and into a hierarchal quad-tree overlay to maintain its spatial relationships. Although they insist that their architecture provide an efficient routing, the evaluation results have not been shown yet.

2.5 The differences of related work

GDR, our proposed system, each node utilizes the coordinates as an ID, and manages an overlay routing table. An ID is generated to reflect the geographical location without using SFC. The ID is in cartesian format (x, y), which represents the longitude x and latitude y. In case of area of size $N = n \times n$, each node has two routing tables of size $\log n$ for the horizontal direction and the vertical direction and a search is possible in $O(\log N)$.

In this thesis, we only consider routing systems for non-mobile networks. We summarized the differences of those systems in Table 2.1. The table shows that the main difference of GDR compared to other systems is the routing method to the destination. The distances shown on the routing table represent the actual geographical distances. The routing method and table creation method of GDR is not based on other systems, which provides an efficient routing method based on the actual geographical distances without increasing the size of routing table.

The routing method and also the table creation method in our system is not based on other systems. GDR, our proposed system, each node utilizes the coordinates as an ID, and manages an overlay routing table. An ID is generated to reflect the geographical location without using SFC. The ID is in cartesian format (x, y), which represents the longitude x and latitude y. In case of area of size $N = n \times n$, each node has two routing tables of size $\log n$ for the horizontal direction and the vertical direction and a search is possible in $O(\log N)$. A message forwarding direction can be either horizontal or vertical.
GDR uses cartesian format instead of hash table in Chord. GDR also uses an efficient routing on the actual geographical distances instead of an routing in Kademlia and CAN.

Table 2.1: The differences of related work, where ID space size is $N(= n \times n)$.

system	ID format	routing table size	routing direction to the destination
GDR	cartesian	$2 \log n$ $(= \log N)$	the closest node to the horizontal direction first, and the closest node to the vertical direcion.
Chord	Hash ID as a ring	$\log N$	clockwise on Chord ring
Kademlia	an arbitrary ID, binary tree	$\log N$	XOR-based metric to reduce the ID space
CAN	cartesian	4	the closest neighboring node to the destination
PHT	prefix-based ID, binary trie	depend on DHT	
LDHT	prefix-based ID	depend on DHT	
Geophony	suffix-based ID	depend on Symphony	
LL-Net	cartesian	$\frac{3}{2} \log N + k$ $(k = 0, 2, 5)$	the closest node in hierarchy area to the destination
SFC	SFC ID	depend on the system	
Mill	SFC ID as a ring	$\log N$	clockwise and counterclockwise on SFC ring
GeoPeer	cartesian	$3 \log N$	the closest node in LRCs to the destination
GeoIGM	hierarchal quad-tree	under the evaluation	

Chapter 3

GDR: A Geographic location-based Distributed Routing

3.1 Introduction

We propose a geographic location-based distributed routing (GDR) system. Figure 3.1 shows an architecture of the GDR system. In the GDR system, one network node is installed in each area and an area node manages location-oriented information of a non-overlapping area. The area nodes and the mobile terminals know their position by GPS sensor. Each area node is identified with its geographic coordinates in a cartesian format. For example, an area node at longitude position x and latitude position y is identified as *node* (x, y).

The area nodes communicate each other, where each area node maintains routing table to enable area nodes to search for location-oriented information of other places. The mobile terminals communicate the area node in wireless. The size of area is larger than the radio range. When the mobile terminal is in the radio range,

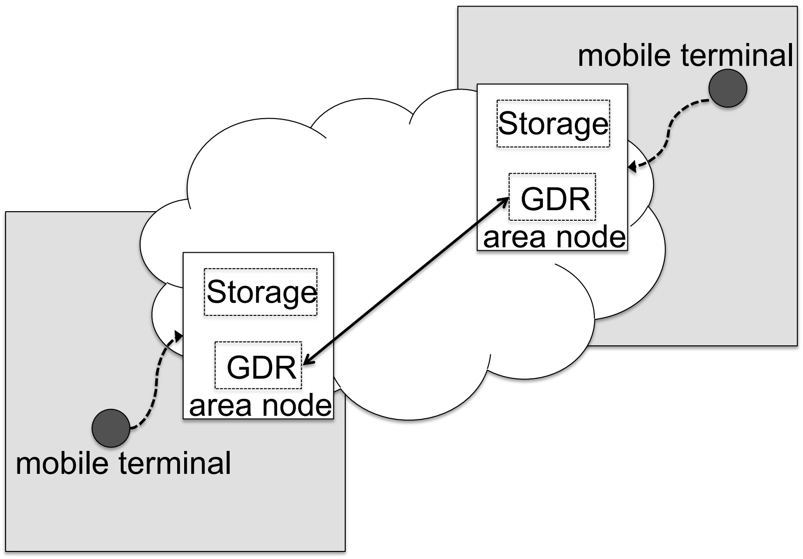


Figure 3.1: Architecture of the GDR system.

it can store the information and send a query to the node, and receive a reply from the node. The area nodes are immovable. They serve as super nodes that collect data from other mobile terminals. Hence, the routing tables of area nodes may be updated occasionally so that the overhead is relatively small. The errors of location information affect the location-oriented information without any effect on the GDR system.

Each area node receives a request from a mobile terminal and sends a reply. When the requested information is not available in the corresponding area node, that area node has to search for the information at other area nodes. To do so, every area node has to maintain a routing table and a search method.

We assume the system works on an urban area where the landscape of the area consists of roads that can be modeled as a grid. Therefore, we use grid network to analyze the performance of the proposed system.

Hereafter, we focus on the area nodes and call the area nodes as the nodes.

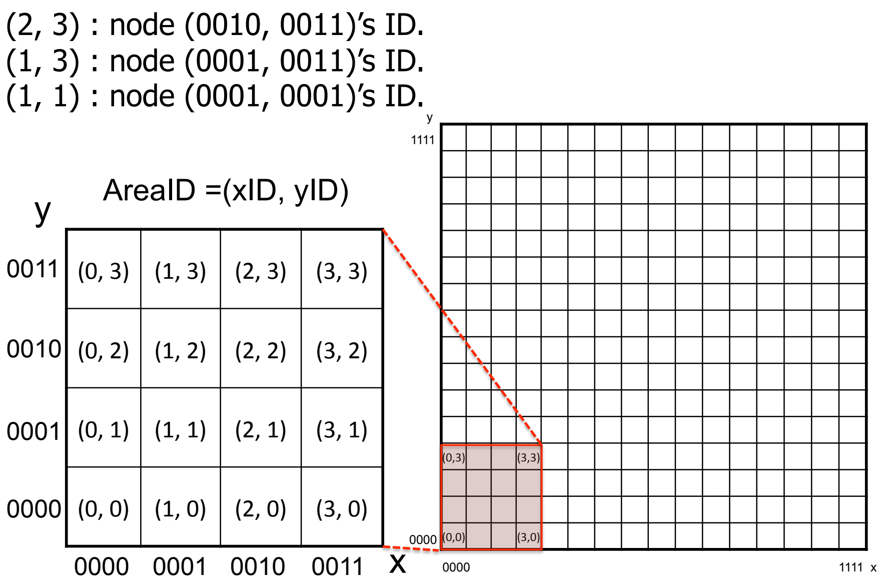


Figure 3.2: AreaID mapping, where r=4, n=16, and N=256.

The following subsections explain the routing table construction and the routing mechanism used in the GDR system.

3.2 Routing Table

We assume a grid network of size $N = n \times n$, where $n = 2^r$ and r is a positive integer (see Figure 3.2). Each node has node ID as (xID, yID). A message forwarding direction can be either horizontal or vertical. Hence, each node has to maintain two routing tables, where one table is used to forward messages to the horizontal direction(T_h) and the other one is used to the vertical direction(T_v). The size of each table is $\log n$. Hence, each node has a routing table of size $\log N$.

The routing table has one field, which consists of several records of IDs of nodes where a message will be forwarded. The records are determined with the following rules:

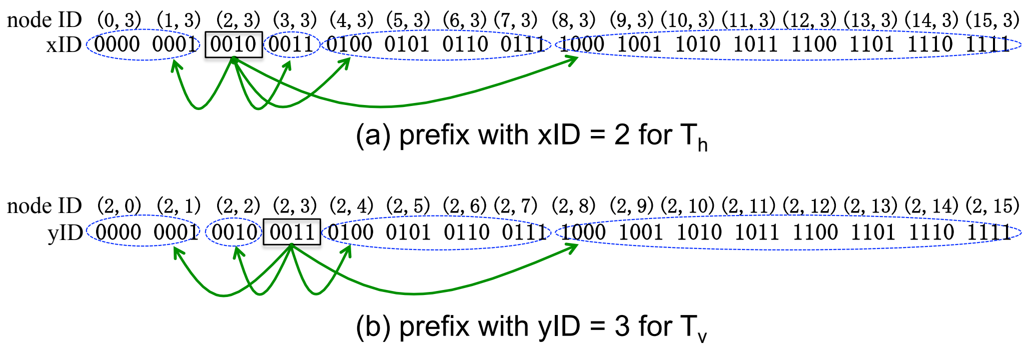

node ID (0, 3) (1, 3) (2, 3) (3, 3) (4, 3) (5, 3) (6, 3) (7, 3) (8, 3) (9, 3) (10, 3) (11, 3) (12, 3) (13, 3) (14, 3) (15, 3)
xID 0000 0001 0010 0011 0100 0101 0110 0111 1000 1001 1010 1011 1100 1101 1110 1111
(a) prefix with xID = 2 for T_h
node ID (2, 0) (2, 1) (2, 2) (2, 3) (2, 4) (2, 5) (2, 6) (2, 7) (2, 8) (2, 9) (2, 10) (2, 11) (2, 12) (2, 13) (2, 14) (2, 15)
yID 0000 0001 0010 0011 0100 0101 0110 0111 1000 1001 1010 1011 1100 1101 1110 1111
(b) prefix with yID = 3 for T_v

Figure 3.3: The partitioned xIDs and yIDs with prefix of $(2, 3)$'s xID and yID, where r=4 and n=16.

1. For (X, Y), partition the xIDs of the horizontal direction's node by common $r - i$ bit length prefix with X to G_{x_i}, and partition the yIDs of the vertical direction's node by common $r - i$ bit length prefix with Y to G_{y_i}, where $1 \leq i \leq r$.

2. From each partitioned group G_{x_i} and G_{y_i}, select nodes (x^*ID, Y) and (X, y^*ID) such that

$$|X - x^*ID| \leq |X - xID| \;\; for \;\; xID \in G_{x\_i},$$

$$|Y - y^*ID| \leq |Y - yID| \;\; for \;\; xID \in G_{y\_i}.$$

   That is, x^*ID is the nearest node in G_{x_i} to X and y^*ID is the nearest node in G_{y_i} to Y.

According to these rules, the maximum size of the table's records is r.

Example In case of $r = 4$, $1 \leq i \leq r$, Figure 3.3 (a) shows the node IDs, its xIDs in binary, and the partitioned xIDs with prefix of $(2, 3)$'s xID (=0010). Also, Figure 3.3 (b) shows the node IDs, its yIDs in binary, and the partitioned yIDs

Table 3.1: $(2,3)$'s routing table.

T_h	T_v
$(3,3)$	$(2,2)$
$(1,3)$	$(2,1)$
$(4,3)$	$(2,4)$
$(8,3)$	$(2,8)$

Table 3.2: $(1,3)$'s routing table.

T_h	T_v
$(0,3)$	$(1,2)$
$(2,3)$	$(1,1)$
$(4,3)$	$(1,4)$
$(8,3)$	$(1,8)$

with prefix of $(2,3)$'s *yID* (=0011). First, in Figure 3.3 (a), $(2,3)$ partitions the *xID*s of the horizontal direction's node by common 3, 2, 1, and 0 bit length prefix with $(2,3)$'s *xID*(= 0010) to G_{x_1}, G_{x_2}, G_{x_3}, and G_{x_4}, respectively. Then, select $(3,3)$, $(1,3)$, $(4,3)$, and $(8,3)$ for T_h from G_{x_1}, G_{x_2}, G_{x_3}, and G_{x_4}, respectively. Next, in Figure 3.3 (b), $(2,3)$ partitions the *yID*s of the vertical direction's node by common 3, 2, 1, and 0 bit length prefix with $(3,2)$'s *yID*(= 0011) to G_{y_1}, G_{y_2}, G_{y_3}, and G_{y_4}, respectively. Then, select $(2,2)$, $(2,1)$, $(2,4)$, and $(2,8)$ for T_v from G_{y_1}, G_{y_2}, G_{y_3}, and G_{y_4}, respectively. Therefore, Table 3.1 shows $(2,3)$'s routing table.

3.3 Routing Mechanism

Routing is a mechanism to select the best node or path to send a message to its destination in a network. In routing, normally, upon receiving a message, a

```
/* search the routing table for the next forwarding node to the destination id */
for(i=0; i<m ; i++){
  xor[i] = routing_table[i] xor destination id ;
}

if(xor[i] = 0)
   return routing_table[i];  //  this is the destination node id
else{
   if(the destination id > the node id){
     j = i of the biggest xor[i] ;
     return routing_table[j] ; // the next forwarding node id
   }
  else{
     j = i of the smallest xor[i] ;
     return routing_table[j] ; // the next forwarding node id
  }
}
```

Figure 3.4: Pseudocode to search the routing table.

network node searches for receiving node in its routing table entries the closest distance to the destination node of the message. The selected nodes along the path repeat the process until the message arrives at its final destination.

In GDR, a message is forwarded according to T_h first. After that, we search for destination in T_v. Figure 3.4 shows the pseudocode of the routing algorithm. We illustrate the routing mechanism by using an example based on Figure 3.2 as follows.

When $(2,3)$ wants to send a query to $(1,1)$, according to T_h of $(2,3)$ in Table 3.1, $(2,3)$ forwards the query to $(1,3)$, whose *xID* is equal to that of $(1,1)$. Then, based on T_v of $(1,3)$ in Table 3.2, $(1,3)$ forwards the query to $(1,1)$.

3.4 Theoretical Analysis

In this section, we compare our GDR system with other systems based on Chord, Kademlia, and CAN, because those three systems contain original methods that are also being used by other systems([8],[9], and [14]–[21]).

We evaluate the routing performance of GDR and compare it with Chord, Kademlia and CAN based on the means and the variances of their the *path length*s and the *relay length*s.

3.4.1 Definition of the Path Length and the Relay Length

To evaluate the distances on the identifier (ID) space, we use two parameters: *path length* and *relay length*. The *path length* is defined as the number of hops that processes and forwards the incoming message on the overlay network. Hence, the *path length* is closely related to the total amount of processing time required while forwarding message from a source node to its destination. The *relay length*, on the other hand, is defined as the distances between source node and destination node to indicate the actual geographical distance between those nodes. Greater *relay length* means greater propagation time. Therefore, the value of *path length* and *relay length* should be as small as possible.

The *relay length* between the source node (x_s, y_s) and the destination node (x_d, y_d) is measured by Manhattan distance[27]:

$$\mid x_d - x_s \mid + \mid y_d - y_s \mid . \tag{3.1}$$

We denote the *path length* of Chord, Kademlia, CAN, and GDR by H_c, H_k,

H_n, and H_g, respectively. Then, H_c, H_k, H_n, and H_g are defined to be follows:

$$\begin{aligned} & H_c((x_s, y_s), (x_d, y_d)) \\ =& H_{c\_x}((x_s, y_s), (x_d, y_s)) + H_{c\_y}((x_d, y_s), (x_d, y_d)), \quad (3.2) \\ & H_k((x_s, y_s), (x_d, y_d)) \\ =& H_{k\_x}((x_s, y_s), (x_d, y_s)) + H_{k\_y}((x_d, y_s), (x_d, y_d)), \quad (3.3) \\ & H_n((x_s, y_s), (x_d, y_d)) \\ =& H_{n\_x}((x_s, y_s), (x_d, y_s)) + H_{n\_y}((x_d, y_s), (x_d, y_d)), \quad (3.4) \\ & H_g((x_s, y_s), (x_d, y_d)) \\ =& H_{g\_x}((x_s, y_s), (x_d, y_s)) + H_{g\_y}((x_d, y_s), (x_d, y_d)). \quad (3.5) \end{aligned}$$

Hereafter, we focus on the horizontal direction of the *path length*s.

Suppose that the distance $\mid x_d - x_s \mid$ has a binary representation $(B_1, B_2, \cdots, B_r)$ such that $\mid x_d - x_s \mid = \sum_{i=1}^{r} B_i 2^{r-i}$, $B_i = 0$ or 1 for $1 \leq i \leq r$, and the mod distance $(x_d - x_s) \bmod 2^r$ has a binary representation $(C_1, C_2, \cdots, C_r)$. Moreover, if x_s and x_d have binary representations $(x_{s_1}, x_{s_2}, \cdots, x_{s_r})$ and $(x_{d_1}, x_{d_2}, \cdots, x_{d_r})$, respectively, let $x_d \oplus x_s$ be a bitwise representation $(x_{d_1} \oplus x_{s_1}, x_{d_2} \oplus x_{s_2}, \cdots, x_{d_r} \oplus x_{s_r})$ as $(D_1, D_2, \cdots, D_r)$.

Then, we can show the following lemma.

Lemma 1 For H_{n_x}, H_{g_x}, H_{c_x} and H_{k_x}, we have

$$H_{n\_x}((x_s, y_s), (x_d, y_s)) = \mid x_d - x_s \mid, \quad (3.6)$$

$$H_{g\_x}((x_s, y_s), (x_d, y_s)) = \sum_{k=1}^{r} B_k, \tag{3.7}$$

$$H_{c\_x}((x_s, y_s), (x_d, y_s)) = \sum_{k=1}^{r} C_k, \tag{3.8}$$

$$H_{k\_x}((x_s, y_s), (x_d, y_s)) = \sum_{k=1}^{r} D_k. \tag{3.9}$$

Proof It is obvious from the routing table construction of Chord, Kademlia, GDR, and CAN. □

Similarly, we denote the *relay length* of Chord, Kademlia, CAN, and GDR by D_c, D_k, D_n, and D_g, respectively. Then, D_c, D_k, D_n, and D_g are defined to be follows:

$$D_c((x_s, y_s), (x_d, y_d))$$
$$= D_{c\_x}((x_s, y_s), (x_d, y_s)) + D_{c\_y}((x_d, y_s), (x_d, y_d)), \tag{3.10}$$
$$D_k((x_s, y_s), (x_d, y_d))$$
$$= D_{k\_x}((x_s, y_s), (x_d, y_s)) + D_{k\_y}((x_d, y_s), (x_d, y_d)), \tag{3.11}$$
$$D_n((x_s, y_s), (x_d, y_d))$$
$$= D_{n\_x}((x_s, y_s), (x_d, y_s)) + D_{n\_y}((x_d, y_s), (x_d, y_d)), \tag{3.12}$$
$$D_g((x_s, y_s), (x_d, y_d))$$
$$= D_{g\_x}((x_s, y_s), (x_d, y_s)) + D_{g\_y}((x_d, y_s), (x_d, y_d)). \tag{3.13}$$

Hereafter, we focus on the horizontal direction of the *relay length*s.

Then, we can show the following lemma.

Lemma 2 *For D_{n_x}, D_{g_x}, D_{c_x} and D_{k_x}, we have*

$$D_{n\_x}((x_s, y_s), (x_d, y_s)) = D_{g\_x}((x_s, y_s), (x_d, y_s)) = \mid x_d - x_s \mid, \tag{3.14}$$

$$D_{c\_x}((x_s, y_s), (x_d, y_s)) = \begin{cases} (x_d - x_s) \bmod 2^r, & \text{if } x_s \le x_d, \quad (3.15a) \\ L_{x_s, x_d}, & \text{if } x_d < x_s, \quad (3.15b) \end{cases}$$

where $L_{x_s,x_d} = 2^{l_1} + \cdots + 2^{l_{k-1}} + (2^r - 2^{l_k}) + 2^{l_{k+1}} + \cdots + 2^{l_t}$, $(2^r - x_s + x_d \triangleq 2^{l_1} + 2^{l_2} + \cdots + 2^{l_t})$ *and* $k = min\{P \mid x_s + \sum_{j=1}^{P} 2^{l_j} > 2^r\}$.

$$D_{k\_x}((x_s, y_s), (x_d, y_s)) = x_d \oplus x_s. \tag{3.16}$$

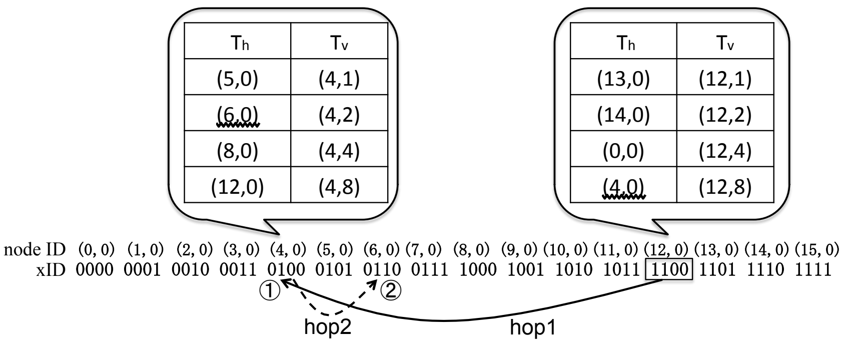
Proof It is obvious from ID with geographic coordinates in a cartesian format and distance metric based on Chord, Kademlia, CAN, and GDR. $\square$

We illustrate the *path length*s and the *relay length*s of GDR, Chord, Kademlia, and CAN as follows. If $r = 4$, then $n = 16$. The routing table size of horizontal direction that each node has is 2 for CAN, and $4(= \log n)$ for Chord, Kademlia, and GDR.

Figure 3.5(a) through Figure 3.5(d) show the routing of a query from $(12, 0)$ to $(6, 0)$ based on Chord, Kademlia, CAN, and GDR, respectively.

In Figure 3.5(a), $(12, 0)$ forwards the query to $(4, 0)$, which is the nearest node towards $(6, 0)$ according to T_h of $(12, 0)$(①). Then, $(4, 0)$ forwards the query to $(6, 0)$ in T_h of $(4, 0)$(②). In this case, $H_{c\_x}((12, 0), (4, 0)) = 2$, and $D_{c\_x}((12, 0), (4, 0)) =\mid 4 - 12 \mid + \mid 6 - 4 \mid= 10$.

In Figure 3.5(b), according to T_h of $(12, 0)$, $(12, 0)$ forwards the query to $(4, 0)$, which is the nearest node towards $(6, 0)$ based on XOR distance (③). Then, $(4, 0)$ forwards the query to $(6, 0)$ in T_h of $(4, 0)$(④). In this case, $H_{k\_x}((12, 0), (4, 0)) = 2$, and $D_{k\_x}((12, 0), (4, 0)) =\mid 4 - 12 \mid + \mid 6 - 4 \mid= 10$.

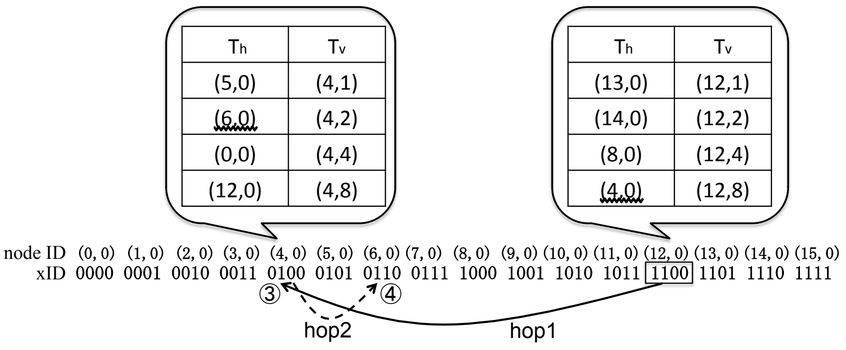


Th	Tv
(5,0)	(4,1)
(6,0)	(4,2)
(8,0)	(4,4)
(12,0)	(4,8)

Th	Tv
(13,0)	(12,1)
(14,0)	(12,2)
(0,0)	(12,4)
(4,0)	(12,8)

node ID (0, 0) (1, 0) (2, 0) (3, 0) (4, 0) (5, 0) (6, 0) (7, 0) (8, 0) (9, 0) (10, 0) (11, 0) (12, 0) (13, 0) (14, 0) (15, 0)
xID 0000 0001 0010 0011 0100 0101 0110 0111 1000 1001 1010 1011 1100 1101 1110 1111
①
②
hop2
hop1

(a) Chord routing

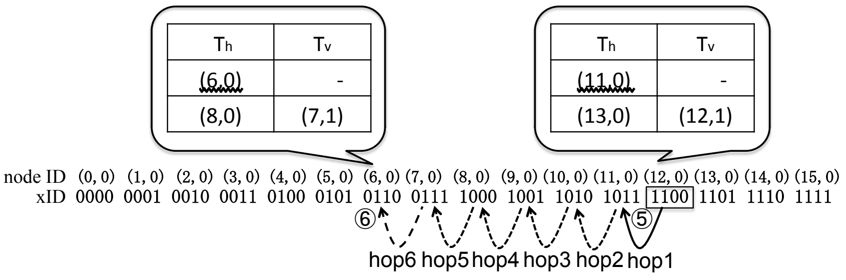


Th	Tv
(5,0)	(4,1)
(6,0)	(4,2)
(0,0)	(4,4)
(12,0)	(4,8)

Th	Tv
(13,0)	(12,1)
(14,0)	(12,2)
(8,0)	(12,4)
(4,0)	(12,8)

node ID (0, 0) (1, 0) (2, 0) (3, 0) (4, 0) (5, 0) (6, 0) (7, 0) (8, 0) (9, 0) (10, 0) (11, 0) (12, 0) (13, 0) (14, 0) (15, 0)
xID 0000 0001 0010 0011 0100 0101 0110 0111 1000 1001 1010 1011 1100 1101 1110 1111
③
④
hop2
hop1

(b) Kademlia routing

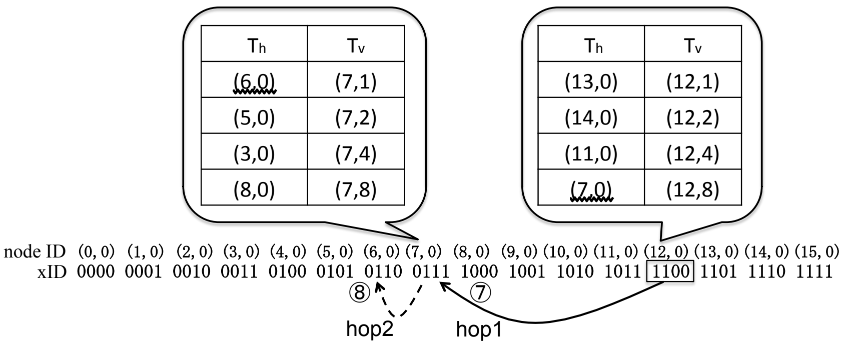


Th	Tv
(6,0)	-
(8,0)	(7,1)

Th	Tv
(11,0)	-
(13,0)	(12,1)

node ID (0, 0) (1, 0) (2, 0) (3, 0) (4, 0) (5, 0) (6, 0) (7, 0) (8, 0) (9, 0) (10, 0) (11, 0) (12, 0) (13, 0) (14, 0) (15, 0)
xID 0000 0001 0010 0011 0100 0101 0110 0111 1000 1001 1010 1011 1100 1101 1110 1111
⑥
⑤
hop6 hop5 hop4 hop3 hop2 hop1

(c) CAN routing



Th	Tv
(6,0)	(7,1)
(5,0)	(7,2)
(3,0)	(7,4)
(8,0)	(7,8)

Th	Tv
(13,0)	(12,1)
(14,0)	(12,2)
(11,0)	(12,4)
(7,0)	(12,8)

node ID (0, 0) (1, 0) (2, 0) (3, 0) (4, 0) (5, 0) (6, 0) (7, 0) (8, 0) (9, 0) (10, 0) (11, 0) (12, 0) (13, 0) (14, 0) (15, 0)
xID 0000 0001 0010 0011 0100 0101 0110 0111 1000 1001 1010 1011 1100 1101 1110 1111
⑧
⑦
hop2
hop1

(d) GDR routing

Figure 3.5: The routing of a query from $(12, 0)$ to $(6, 0)$.

Table 3.3: The *path length*s and the *relay length*s in Figure 3.5.

system	*path length*	*relay length*
Chord(Figure 3.5(a))	2	10 (0b1010)
Kademlia(Figure 3.5(b))	2	10 (0b1010)
CAN(Figure 3.5(c))	6	6
GDR(Figure 3.5(d))	2	6 (0b0110)

In Figure 3.5(c), according to T_h of $(12, 0)$, $(12, 0)$ forwards the query to $(11, 0)$, which is the nearest node towards $(6, 0)$ (⑤). Each node routes in a similar way in order to approach $(6, 0)$. Therefore, $(7, 0)$ forwards a query to $(6, 0)$ in T_h of $(7, 0)$(⑥). In this case, $H_{c\_x}((12, 0), (4, 0)) = 6$, and $D_{n\_x}((12, 0), (4, 0)) =\mid 11 - 12 \mid + \mid 10 - 11 \mid + \mid 10 - 9 \mid + \mid 8 - 9 \mid + \mid 7 - 8 \mid + \mid 6 - 7 \mid= 6$.

In Figure 3.5(d), according to T_h of $(12, 0)$, $(12, 0)$ forwards the query to $(7, 0)$, which is the nearest node towards $(6, 0)$ (⑦). Then, $(7, 0)$ forwards the message to $(6, 0)$ in T_h of $(7, 0)$(⑧). In this case, $H_{k\_x}((12, 0), (4, 0)) = 2$, and $D_{g\_x}((12, 0), (4, 0)) =\mid 7 - 12 \mid + \mid 6 - 7 \mid= 6$.

The *path length*s and the *relay length*s in Figure 3.5 are summarized in Table 3.3. Table 3.3 satisfies Lemma1 and Lemma2.

3.4.2 The Path Length

We analyze the statistic behavior of the *path length*s of Chord, Kademlia, CAN, and GDR. Especially, we explore the mean and the variance of the *path length*s for these system. When the number of the nodes in the horizontal direction is $2^r (= n)$, we assume that all nodes are active, and the source node and the destination node are chosen independently with equal probability. Then, we have the following theorem.

Theorem 1 For the means and the variances of *path length*s of those systems, we have

$$\mathbb{E}(H_{c\_x}) = \mathbb{E}(H_{k\_x}) = \mathbb{E}(H_{g\_x}) = \frac{1}{2}\log n, \tag{3.17}$$

$$\mathbb{V}(H_{c\_x}) = \mathbb{V}(H_{k\_x}) = \mathbb{V}(H_{g\_x}) = \frac{1}{4}\log n, \tag{3.18}$$

and,

$$\mathbb{E}(H_{n\_x}) = \frac{1}{3}(n - \frac{1}{n}), \tag{3.19}$$

$$\mathbb{V}(H_{n\_x}) = \frac{1}{18}(n^2 - \frac{2}{n^2} + 1). \tag{3.20}$$

Proof In case of Chord, Kademlia, and GDR, each node has a routing table of size $r(= \log n)$. From Eqs.(3.7) – (3.9) in Lemma1, the *path length* $h(X)$ is defined to be $h(X) = X_1 + X_2 + \cdots + X_r$. The random variable X_r takes value either 0 or 1 with equal probability, thus X_r can take the value 1 with probability $1/2$. Then,

$$\mathbb{E}[h(X)] = \frac{r}{2}. \tag{3.21}$$

Therefore,

$$\mathbb{E}(H_{c\_x}) = \mathbb{E}(H_{k\_x}) = \mathbb{E}(H_{g\_x}) = \frac{1}{2}\log n. \tag{3.22}$$

In case of CAN,

$$\begin{aligned}\mathbb{E}(H_{n\_x}) &= \sum_{i,j=0}^{n-1} P(I=i \wedge J=j) H_{n\_x}((x_i, y_s), (x_j, y_s)) \\ &= \frac{1}{n^2} \sum_{i,j=0}^{n-1} H_{n\_x}((x_i, y_s), (x_j, y_s)). \qquad (3.23)\end{aligned}$$

From Eq. (3.6) in Lemma1,

$$\begin{aligned}\sum_{i,j=0}^{n-1} H_{n\_x}((x_i, y_s), (x_j, y_s)) &= 2 \times \sum_{k=0}^{n} k(n-k) \\ &= \frac{1}{3}(n-1)n(n+1). \qquad (3.24)\end{aligned}$$

Hence, (3.19) follows from (3.23) and (3.24).

It is known that the variance is defined to be follows:

$$\begin{aligned}\mathbb{V}(H_{A\_x}) &= \sum_{i,j=0}^{n-1} P(I=i \wedge J=j)\{H_{A\_x}((x_i, 0), (x_j, 0)) - \mathbb{E}(H_{A\_x})\}^2 \\ &= \frac{1}{n^2} \sum_{i,j=0}^{n-1} H_{A\_x}((x_i, 0), (x_j, 0))^2 - \mathbb{E}(H_{A\_x})^2. \qquad (3.25)\end{aligned}$$

In case of Chord, Kademlia, and GDR, the first term of (3.25) is calculated as

$$\frac{1}{n^2} \times \frac{n^2}{4}(1 + \log n)\log n = \frac{1}{4}(1 + \log n)\log n. \qquad (3.26)$$

Hence, (3.18) follows from (3.17), (3.25), and (3.26).

In case of CAN, the first term of (3.25) is given by

$$\frac{1}{n^2} \times 2\{n \sum_{k=0}^{n} k^2 - \sum_{k=0}^{n} k^3\} = \frac{1}{6}(n^2 - 1). \tag{3.27}$$

Therefore, (3.20) follows from (3.19), (3.25), and (3.27).

We have completed the proof. □

When the number of areas is N, the number of nodes in the horizontal and the vertical directions is $\sqrt{N}$. From Theorem 1, the mean and the variance of the *path length* are given as follows:

$$\mathbb{E}(H_c) = \mathbb{E}(H_k) = \mathbb{E}(H_g) = \frac{1}{2}\log N, \tag{3.28}$$

$$\mathbb{V}(H_c) = \mathbb{V}(H_k) = \mathbb{V}(H_g) = \frac{1}{4}\log N, \tag{3.29}$$

and,

$$\mathbb{E}(H_n) = \frac{2}{3}(\sqrt{N} - \frac{1}{\sqrt{N}}), \tag{3.30}$$

$$\mathbb{V}(H_n) = \frac{1}{9}(N - \frac{2}{N} + 1). \tag{3.31}$$

Equations (3.28) and (3.29) show that Chord, Kademlia and GDR have the same mean and variance of the *path length*. Meanwhile, Eqs. (3.30) and (3.31) show that GDR can improve the mean and the variance of the *path length* in CAN about $(3/4)\log N/\sqrt{N}$ times and $(9/4)N\log N/(N^2 + N - 2)$ times, respectively.

3.4.3 The Relay Length

We analyze the statistic behavior of the *relay length*s of Chord, Kademlia, CAN, and GDR. Especially, we explore the mean and the variance of the *relay length*s for these system. When the number of the nodes in the horizontal direction is $2^r (= n)$, suppose that the source node and the destination node are chosen independently with equal probability. Then, we can show following theorem.

Theorem 2 For the means and the variances of *relay length*s of those systems, we have

$$\mathbb{E}(D_{n\_x}) = \mathbb{E}(D_{g\_x}) = \frac{1}{3}(n - \frac{1}{n}), \tag{3.32}$$
$$\mathbb{E}(D_{k\_x}) = \frac{1}{2}(n - 1), \tag{3.33}$$
$$\mathbb{E}(D_{c\_x}) = \frac{1}{3}(2n + \frac{1}{n}) - 1, \tag{3.34}$$

and,

$$\mathbb{V}(D_{n\_x}) = \mathbb{V}(D_{g\_x}) = \frac{1}{18}(n^2 - \frac{2}{n^2} + 1), \tag{3.35}$$
$$\mathbb{V}(D_{k\_x}) = \frac{1}{12}(n^2 - 1). \tag{3.36}$$

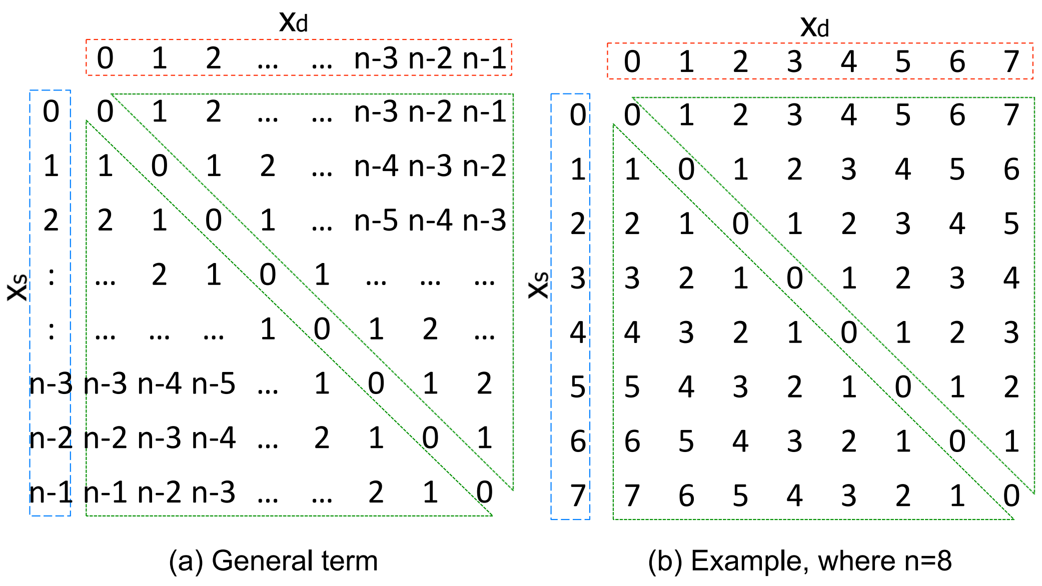
Proof We denote D_{c_x}, D_{k_x}, D_{n_x} and D_{g_x} by D_{A_x}, the mean of the *relay length* of D_{A_x} is defined as follows:

$$\begin{aligned}\mathbb{E}(D_{A\_x}) &= \sum_{i,j=0}^{n-1} P(I=i \wedge J=j) D_{A\_x}((x_i, y_s), (x_j, y_s)) \\ &= \frac{1}{n^2} \sum_{i,j=0}^{n-1} D_{A\_x}((x_i, y_s), (x_j, y_s)). \end{aligned} \tag{3.37}$$

In case of CAN and GDR, from Eq. (3.14) in Lemma2,

$$\begin{aligned}&\sum_{i,j=0}^{n-1} D_{c\_x}((x_i, y_s), (x_j, y_s)) \\ =&\sum_{i,j=0}^{n-1} D_{g\_x}((x_i, y_s), (x_j, y_s)) \\ =&2 \times \{(n-1)1 + (n-2)2 + \cdots + 1(n-1)\} \\ =&2 \times \sum_{k=1}^{n-1} k(n-k) \\ =&2 \times \sum_{k=0}^{n} k(n-k) \\ =&\frac{1}{3}(n-1)n(n+1). \end{aligned} \tag{3.38}$$

Hence, (3.32) follows from (3.37) and (3.38). Figure 3.6(a) shows general term by Eq. (3.14) in Lemma2. Also, Figure 3.6(b) shows an example of $D_{g\_x}((x_s, y_s), (x_d, y_s))$ and $D_{n\_x}((x_s, y_s), (x_d, y_s))$, where $n = 8$ and $0 \le x_s, x_d \le 7$.



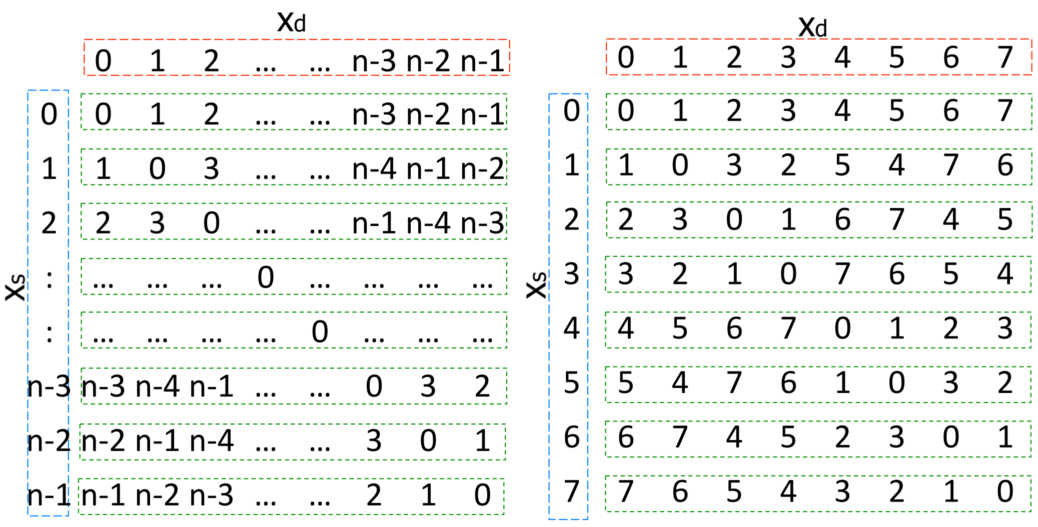
$$\begin{array}{c|cccccccc}
X_s \backslash X_d & 0 & 1 & 2 & \dots & \dots & n-3 & n-2 & n-1 \\
\hline
0 & 0 & 1 & 2 & \dots & \dots & n-3 & n-2 & n-1 \\
1 & 1 & 0 & 1 & 2 & \dots & n-4 & n-3 & n-2 \\
2 & 2 & 1 & 0 & 1 & \dots & n-5 & n-4 & n-3 \\
: & \dots & 2 & 1 & 0 & 1 & \dots & \dots & \dots \\
: & \dots & \dots & \dots & 1 & 0 & 1 & 2 & \dots \\
n-3 & n-3 & n-4 & n-5 & \dots & 1 & 0 & 1 & 2 \\
n-2 & n-2 & n-3 & n-4 & \dots & 2 & 1 & 0 & 1 \\
n-1 & n-1 & n-2 & n-3 & \dots & \dots & 2 & 1 & 0
\end{array}
\qquad
\begin{array}{c|cccccccc}
X_s \backslash X_d & 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 \\
\hline
0 & 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 \\
1 & 1 & 0 & 1 & 2 & 3 & 4 & 5 & 6 \\
2 & 2 & 1 & 0 & 1 & 2 & 3 & 4 & 5 \\
3 & 3 & 2 & 1 & 0 & 1 & 2 & 3 & 4 \\
4 & 4 & 3 & 2 & 1 & 0 & 1 & 2 & 3 \\
5 & 5 & 4 & 3 & 2 & 1 & 0 & 1 & 2 \\
6 & 6 & 5 & 4 & 3 & 2 & 1 & 0 & 1 \\
7 & 7 & 6 & 5 & 4 & 3 & 2 & 1 & 0
\end{array}$$

Figure 3.6: $D_{g\_x}((x_s, y_s), (x_d, y_s))$ and $D_{n\_x}((x_s, y_s), (x_d, y_s))$.

Also, in case of Kademlia, from Eq. (3.16) in Lemma2,

$$\begin{aligned}
\sum_{i,j=0}^{n-1} D_{k\_x}((x_i, y_s), (x_j, y_s)) &= n \times \{0 + 1 + 2 + \cdots + (n-1)\} \\
&= n \times \sum_{k=0}^{n-1} k \\
&= n \times \frac{1}{2} n(n-1) \\
&= \frac{1}{2} n^2 (n-1). \qquad (3.39)
\end{aligned}$$

Hence, (3.33) follows from (3.37) and (3.39). Figure 3.7(a) shows general term by Eq. (3.16) in Lemma2. Also, Figure 3.7(b) shows an example of $D_{k\_x}((x_s, y_s), (x_d, y_s))$, where $n = 8$ and $0 \leq x_s, x_d \leq 7$.



xs \ xd	0	1	2	...	...	n-3	n-2	n-1
0	0	1	2	...	...	n-3	n-2	n-1
1	1	0	3	...	...	n-4	n-1	n-2
2	2	3	0	...	...	n-1	n-4	n-3
:	...	...	...	0	...	...	...	...
:	...	...	...	...	0	...	...	...
n-3	n-3	n-4	n-1	...	...	0	3	2
n-2	n-2	n-1	n-4	...	...	3	0	1
n-1	n-1	n-2	n-3	...	...	2	1	0

(a) General term

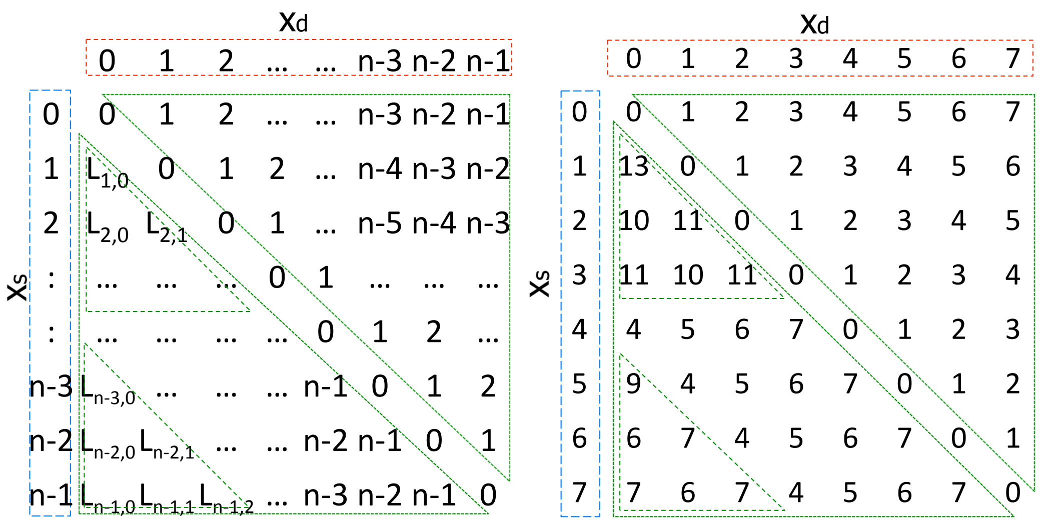
xs \ xd	0	1	2	3	4	5	6	7
0	0	1	2	3	4	5	6	7
1	1	0	3	2	5	4	7	6
2	2	3	0	1	6	7	4	5
3	3	2	1	0	7	6	5	4
4	4	5	6	7	0	1	2	3
5	5	4	7	6	1	0	3	2
6	6	7	4	5	2	3	0	1
7	7	6	5	4	3	2	1	0

(b) Example, where n=8

Figure 3.7: $D_{k\_x}((x_s, y_s), (x_d, y_s))$.

Also, in case of Chord, From Eq. (3.15a) and (3.15b) in Lemma2,

$$
\begin{aligned}
&\sum_{i,j=0}^{n-1} D_{c\_x}((x_i, y_s), (x_j, y_s)) \\
&= \sum_{i=0}^{n-1} \sum_{j=i}^{n-1} D_{c\_x}((x_i, y_s), (x_j, y_s)) \\
&\quad + \sum_{i=0}^{n-1} \sum_{j=0}^{i} D_{c\_x}((x_i, y_s), (x_j, y_s)) \\
&=(n-1)1 + (n-2)2 + \cdots + 1(n-1) \\
&\quad + (n-1)(n-1) + (n-2)(n-2) + \cdots 1 \cdot 1 + \Delta \\
&= \sum_{k=1}^{n-1} k(n-k) + \sum_{k=1}^{n-1} k^2 + \Delta \\
&= \sum_{k=0}^{n} k(n-k) + \sum_{k=0}^{n} k^2 + \frac{1}{6}n^3 - \frac{1}{2}n^2 + \frac{1}{3}n \\
&=\frac{2}{3}n^3 - n^2 + \frac{1}{3}n, \qquad (3.40)
\end{aligned}
$$



(a) General term (b) Example, where n=8

Figure 3.8: $D_{c\_x}((x_s, y_s), (x_d, y_s))$.

where Δ means correction term.

Hence, (3.34) follows from (3.37) and (3.40). Figure 3.8(a) shows general term by Eq. (3.15a) and (3.15b) in Lemma2. Also, Figure 3.8(b) shows an example of $D_{k\_c}((x_s, y_s), (x_d, y_s))$, where $n = 8$ and $0 \leq x_s, x_d \leq 7$.

It is known that the variance is defined to be follows:

$$\begin{aligned}\mathbb{V}(D_{A\_x}) &= \sum_{i,j=0}^{n-1} P(I = i \wedge J = j)\{D_{A\_x}((x_i, y_s), (x_j, y_s)) - \mathbb{E}(D_{A\_x})\}^2 \\ &= \frac{1}{n^2}\sum_{i,j=0}^{n-1} D_{A\_x}((x_i, y_s), (x_j, y_s))^2 - \mathbb{E}(D_{A\_x})^2. \qquad (3.41)\end{aligned}$$

In case of CAN and GDR, the first term of (3.41) is given by

$$\frac{1}{n^2} \times 2\{n\sum_{k=0}^{n} k^2 - \sum_{k=0}^{n} k^3\} = \frac{1}{6}(n^2 - 1). \qquad (3.42)$$

Hence, (3.35) follows from (3.32), (3.41), and (3.42).

In case of Kademlia, the first term of (3.41) is given by

$$\frac{1}{n^2} \times n \sum_{k=0}^{n-1} k^2 = \frac{1}{6}(2n^2 - 3n + 1). \tag{3.43}$$

Hence, (3.36) follows from (3.33), (3.41), and (3.43).

We have completed the proof. □

When the number of areas is N, the number of nodes in the horizontal and the vertical directions is $\sqrt{N}$. From Theorem 2, the mean and the variance of the *relay length* are given as follows:

$$\mathbb{E}(D_c) = \frac{2}{3}(2\sqrt{N} + \frac{1}{\sqrt{N}}) - 2, \tag{3.44}$$

$$\mathbb{E}(D_k) = \sqrt{N} - 1, \tag{3.45}$$

$$\mathbb{E}(D_n) = \mathbb{E}(D_g) = \frac{2}{3}(\sqrt{N} - \frac{1}{\sqrt{N}}). \tag{3.46}$$

And,

$$\mathbb{V}(D_k) = \frac{1}{6}(N - 1), \tag{3.47}$$

$$\mathbb{V}(D_n) = \mathbb{V}(D_g) = \frac{1}{9}(N - \frac{2}{N} + 1). \tag{3.48}$$

We can conclude that the mean *relay length* of GDR is about half of that of Chord, and about 2/3 of that of Kademlia. Meanwhile, CAN and GDR have the same mean and the variance of the *relay length*.

3.5 Conclusion

We proposed a geographic location-based distributed routing (GDR) system. The GDR system provides an information lookup based on latitude and longitude coordinates. We evaluate the mean and the variance of the *path length* and the *relay length* of GDR, CAN, Chord and Kademlia, under the assumptions that the ID is in cartesian format (x, y), all nodes are active, and the source node and the destination node are chosen independently with equal probability.

We show that GDR, Chord and Kademlia have the same mean and the same variance of the *path length*, while the mean and the variance of the *relay length* of GDR are smaller than those of Chord and Kademlia. Furthermore, while GDR and CAN have the same mean and the same variance of the *relay length*, the mean and the variance of the *path length* of GDR are smaller than those of CAN. We show that the mean *relay length* of GDR is about half of that of Chord, and about 2/3 of that of Kademlia, and the mean *path length* is about $(3/4)\log N/\sqrt{N}$ of that of CAN.

Chapter 4

Simulation

4.1 Introduction

In Chapter 3, we showed the basic performance of the GDR system in the theoretical analysis if all nodes are active, the source node and the destination node are chosen independently with equal probability, and in the ID is in cartesian format. However, the problem is opened if some nodes fails. When a node fails, if the node is the destination or the forwarding node, a link error occur while the routing table of the other nodes are updated.

The rest of this chapter is organized as follows. In section 4.2, we simulate if the source node is fixed, and in the ID is generated by using SFC. Section 4.3 describes the details of node fails and evaluates the performance.

4.2 Search Simulation

4.2.1 fix search

To confirm that the routing performance of GDR does not depend on the position of the source node and the destination node, we compare the mean and the variance of the *relay length* when the source node is fixed. When the ID space size of the 1-D in the horizontal and vertical directions is 4, we fix the source of the search in $(I,\ J)$ $(0 \leq I \leq 3,\ 0 \leq J \leq 3)$ and search from $(0,0)$ to $(3,3)$ node by node. Table 4.1 shows the mean and the variance of the *relay length*.

In Table 4.1, the mean and the variance of the *relay length* are constant even in the case of Kademlia that uses XOR distance for routing. On the other hand, GDR and Kademlia have the same maximum value of the mean and the variance of the *relay length*, whereas Chord and Kademlia have the same minimum value of the mean and the variance of the *relay length*. Because Chord is a one-way clockwise routing, the *relay length* becomes large by the position of the source node and the destination node. However, since DGR is a two-way approachable routing, the *relay length* is fixed by the position of the source node and the destination node. Furthermore, the mean and the variance of the *relay length* are small, which shows that the routing performance of GDR does not depend on the position of the source node and the destination node.

4.2.2 2-D search

To show the advantages that geometrically neighboring node IDs are certainly consecutive, we compare the performance of GDR with Chord and Kademlia in

Table 4.1: The mean and the variance of the *relay length*.

source	D_c		D_k		D_g	
node	Avg	Var	Avg	Var	Avg	Var
(0,0)	3.00	2.50	3.00	2.50	3.00	2.50
(1,0)	3.50	4.75	3.00	2.50	2.50	1.75
(2,0)	3.00	2.50	3.00	2.50	2.50	1.75
(3,0)	3.50	2.75	3.00	2.50	3.00	2.50
(0,1)	3.25	4.95	3.00	2.50	2.50	1.75
(1,1)	4.00	7.00	3.00	2.50	2.00	1.00
(2,1)	3.75	4.44	3.00	2.50	2.00	1.00
(3,1)	4.00	5.00	3.00	2.50	2.50	1.75
(0,2)	3.00	2.50	3.00	2.50	2.50	1.75
(1,2)	3.50	4.75	3.00	2.50	2.00	1.00
(2,2)	3.00	2.5	3.00	2.50	2.00	1.00
(3,2)	3.50	2.75	3.00	2.50	2.50	1.75
(0,3)	3.50	2.75	3.00	2.50	3.00	2.50
(1,3)	4.00	5.00	3.00	2.50	2.50	1.75
(2,3)	3.50	2.75	3.00	2.50	2.50	1.75
(3,3)	4.00	3.00	3.00	2.50	3.00	2.50

which the ID is generated by using SFC. We executed the behavior of Chord and Kademlia of size $N(= n \times n = 2^{2r})$ when the ID is generated by using z-ordering method, where $2 \leq r \leq 6$. We executed $10{\times}N$ random lookups to Chord and Kademlia. During the simulation, we measured the *relay length* for each lookup. Figure 4.1 plots the mean of the *relay length*. In Figure 4.1, Eqs. (3.44), (3.45), and (3.46) are presented as Chord-for-xy, Kademlia-for-xy, and GDR-for-xy, respectively. Also, the simulation results of Chord and Kademlia when the ID is generated by using SFC as Chord-sim-z and Kademlia-sim-z, respectively.

Figure 4.1 shows that the mean *relay length* of GDR is about $1/2$ times smaller than that of Chord, and about $2/3$ times smaller than that of Kademlia.

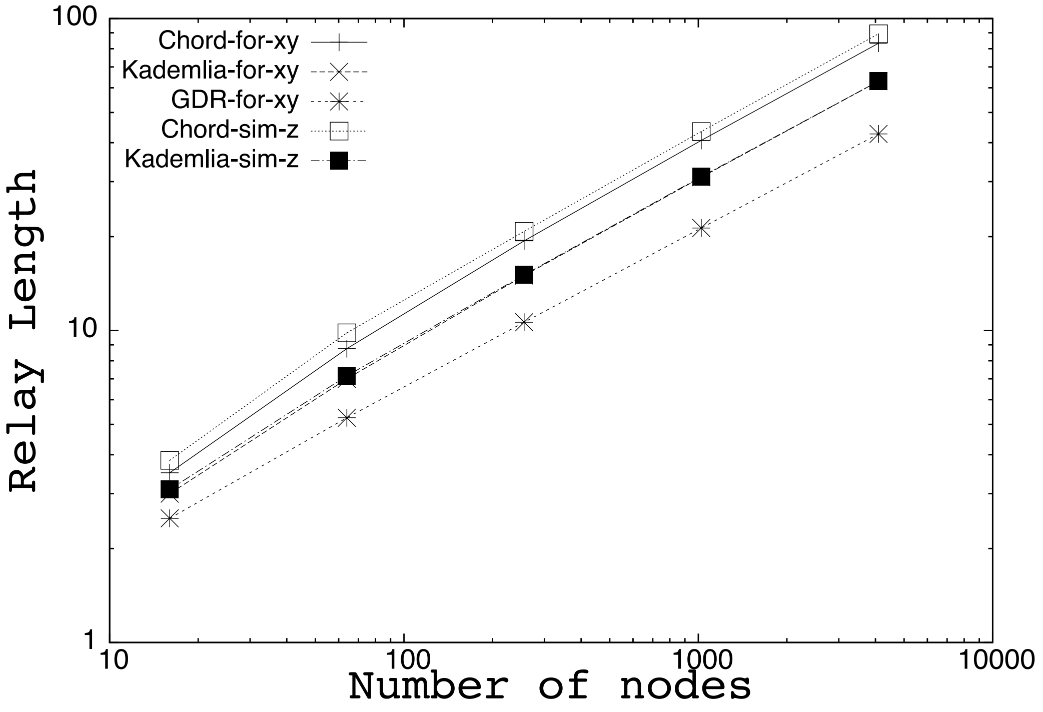


Figure 4.1: The Relay Length.

4.3 Node Fails

When a node fails, if the node is the destination or the forwarding node, a link error occurs until the routing table of the other nodes are updated. To increase robustness, the system is required to have a routing redundancy.

For instance, each node of the Chord system has the records of nodes as a successor list in addition to the records of nodes of its routing table. For each node, a node connected in clockwise to it is called successor. The first node in a successor list is the first node in a corresponding routing table, and the second node in the list is the successor of the first node in the routing table. If the first successor of a node doesn't respond, the node substitutes the second entry in tis successor list for the first successor. In an implementation of the Chord system, the size of successor list is chosen as $2 \log N$ for the foreseeable maximum number of nodes N.

In this section, we describe how the GDR system handles when some nodes fails.

4.3.1 State of Node

A node has the following states.

active : A node is working.

down : A node is not working without sending any notification to its neighbor nodes.

leave : A node is not working after sending notification to its neighbor nodes. On receiving a notification from a *leaving* node, the neighbor node updates its routing table and behaves as an agent for the *leaving* node.

join : A node is working after sending notification to its neighbor nodes. On receiving a notification from a *joining* node, the neighbor node updates its routing table and stops as an agent for the *joining* node.

update : A node updates a routing table and behaves as an agent for the *down* or *leaving* node.

4.3.2 Agent node

When a node fails, its neighbor node behaves as an agent for the failing node. To know the agent node of the failing node, each node has an agent list which is the records of the agent nodes of the nodes of its routing table. In the GDR system, each node has two routing tables, T_h and T_v. T_{h_a} and T_{v_a} are the agent list of the nodes of T_h and T_v, respectively.

The agent nodes are determined with the following rules:

1. For (X, Y), partition the xIDs of the horizontal direction's node by common $r - i$ bit length prefix with X to G_{x_i}, where $1 \leq i \leq r$.

2. From each partitioned group G_{x_i}, select nodes (x^*ID, Y) such that

$$|X - x^*ID| \leq |X - xID| \ \ for \ \ xID \in G_{x\_i},$$

where $i = 1, 2$. That is, x^*ID is the nearest node in G_{x_i} to X.

According to these rules, the number of the agent nodes is 2.

Example In case of $r = 4$, Table 3.1 shows T_h and T_v of $(2, 3)$. Figure 4.2 (a) - (d) show the node IDs, its xIDs in binary, and the partitioned xIDs with prefix of $(3, 3)$'s xID, $(1, 3)$'s xID, $(4, 3)$'s xID, and $(8, 3)$'s xID in Table 3.1, respectively. Also, figure 4.3 show the node IDs, its xIDs in binary, and the partitioned xIDs with prefix of $(2, 2)$'s xID, $(2, 1)$'s xID, $(2, 4)$'s xID, and $(2, 8)$'s xID in Table 3.1, respectively. First, In figure 4.2 (a), $(2, 3)$ partitions the xIDs of the horizontal direction's node by common 3 and 2 bit length prefix with $(3, 3)$'s $xID(= 0011)$ to G_{x_1} and G_{x_2}, respectively. Then, select $(2, 3)$ for T_{h_a1} and $(1, 3)$ for T_{h_a2} from G_{x_1} and G_{x_2}, respectively. Next, In Figure 4.3, $(2, 2)$ partitions the xIDs of the horizontal direction's node by common 3 and 2 bit length prefix with $(2, 2)$'s $xID(= 0010)$ to G_{x_1} and G_{x_2}, respectively. Then, select $(3, 2)$ for T_{h_a1} and $(1, 2)$ for T_{h_a2} from G_{x_1} and G_{x_2}, respectively. Similarly, the agent nodes of the other

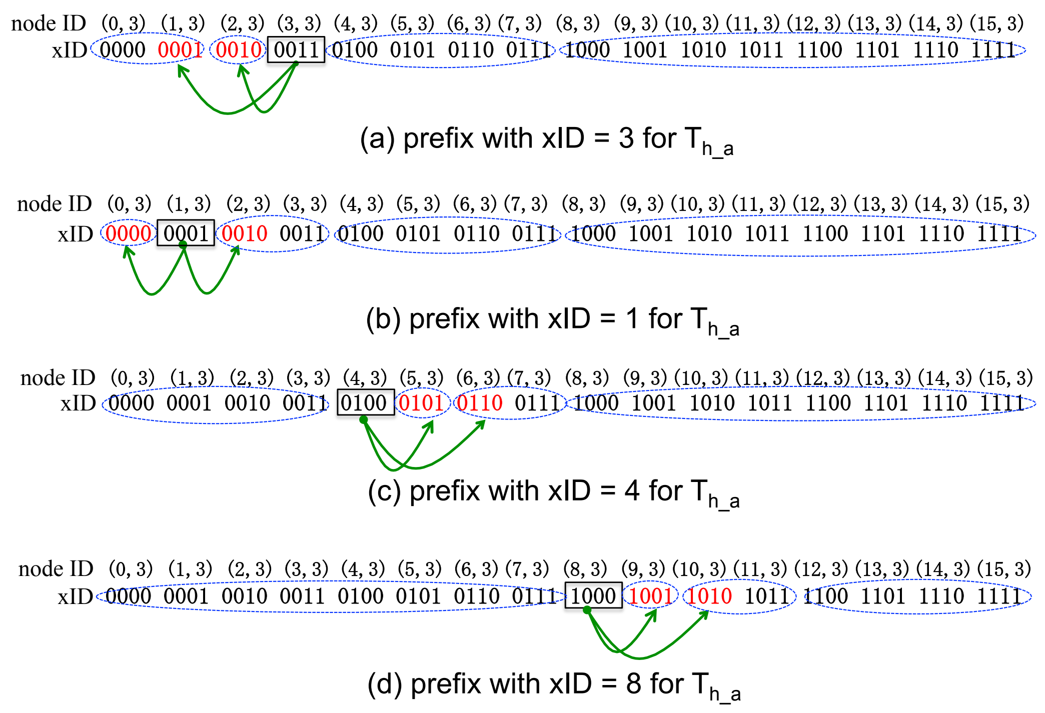

node ID (0, 3) (1, 3) (2, 3) (3, 3) (4, 3) (5, 3) (6, 3) (7, 3) (8, 3) (9, 3) (10, 3) (11, 3) (12, 3) (13, 3) (14, 3) (15, 3)
xID 0000 0001 0010 0011 0100 0101 0110 0111 1000 1001 1010 1011 1100 1101 1110 1111
(a) prefix with xID = 3 for T_h_a
(b) prefix with xID = 1 for T_h_a
(c) prefix with xID = 4 for T_h_a
(d) prefix with xID = 8 for T_h_a

Figure 4.2: The partitioned *xID*s with prefix of *xID* of $(3,3)$, $(1,3)$, $(4,3)$, and $(8,3)$, where r=4 and n=16.

nodes in Table 3.1 are determined. Therefore, Table 4.2 shows the routing table and the agent list of $(2,3)$.

4.3.3 Leave Message

When node A is *leaving*, node A sends a *leave* message to node B which is the first node of its T_h. When node B receives a *leave* message, node B modifies its routing table to behave as an agent for node A. After that, node B sends an *OK* message to node A. Once node A receives an *OK* message from node B, it leaves the system.

Example When $(2,3)$ is *leaving*, $(2,3)$ sends a *leave* message to $(3,3)$ which is

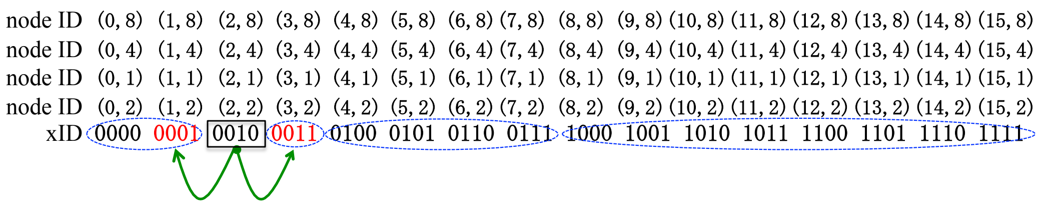

node ID (0, 8) (1, 8) (2, 8) (3, 8) (4, 8) (5, 8) (6, 8) (7, 8) (8, 8) (9, 8) (10, 8) (11, 8) (12, 8) (13, 8) (14, 8) (15, 8)
node ID (0, 4) (1, 4) (2, 4) (3, 4) (4, 4) (5, 4) (6, 4) (7, 4) (8, 4) (9, 4) (10, 4) (11, 4) (12, 4) (13, 4) (14, 4) (15, 4)
node ID (0, 1) (1, 1) (2, 1) (3, 1) (4, 1) (5, 1) (6, 1) (7, 1) (8, 1) (9, 1) (10, 1) (11, 1) (12, 1) (13, 1) (14, 1) (15, 1)
node ID (0, 2) (1, 2) (2, 2) (3, 2) (4, 2) (5, 2) (6, 2) (7, 2) (8, 2) (9, 2) (10, 2) (11, 2) (12, 2) (13, 2) (14, 2) (15, 2)
xID 0000 0001 0010 0011 0100 0101 0110 0111 1000 1001 1010 1011 1100 1101 1110 1111

Figure 4.3: The partitioned *xID*s with prefix of *xID* of $(2,2)$, $(2,1)$, $(2,4)$, and $(2,8)$, where r=4 and n=16.

Table 4.2: The routing table and the agent list of $(2,3)$.

T_h	T_{h_a}		T_v	T_{v_a}	
$(3,3)$	$(2,3)$	$(1,3)$	$(2,2)$	$(3,2)$	$(1,2)$
$(1,3)$	$(0,3)$	$(2,3)$	$(2,1)$	$(3,1)$	$(1,1)$
$(4,3)$	$(5,3)$	$(6,3)$	$(2,4)$	$(3,4)$	$(1,4)$
$(8,3)$	$(9,3)$	$(10,3)$	$(2,8)$	$(3,8)$	$(1,8)$

the first node of its T_h in Table 3.1. When node $(3,3)$ receives a *leave* message, $(3,3)$ modifies its routing table(Table 4.3) to behave as an agent for $(2,3)$. After that, $(3,3)$ sends an *OK* message to $(2,3)$. Once $(2,3)$ receives an *OK* message from $(3,3)$, it leaves the system.

4.3.4 Join Message

When node A is *joining*, node A sends a *join* message to node B which is the first node of its T_h. When node B receives a *join* message, node B modifies its routing table to stop behaving as an agent for node A. After that, node B sends an *OK* message to node A. Once node A receives an *OK* message from node B, it joins the system.

Example When $(2,3)$ is *joining*, $(2,3)$ sends a *join* message to $(3,3)$ which is the first node of its T_h in Table 3.1. When node $(3,3)$ receives a *join* message, $(3,3)$

Table 4.3: The modified routing table of $(3,3)$ when $(2,3)$ is *leaving*.

T_h	T_v	T_v of $(2,3)$
$(2,3) \rightarrow (3,3)$	$(3,2)$	$(2,2)$
$(1,3)$	$(3,1)$	$(2,1)$
$(4,3)$	$(3,4)$	$(2,4)$
$(8,3)$	$(3,8)$	$(2,8)$

Table 4.4: The modified routing table of $(3,3)$ when $(2,3)$ is *joining*.

T_h	T_v
$(3,3) \rightarrow (2,3)$	$(3,2)$
$(1,3)$	$(3,1)$
$(4,3)$	$(3,4)$
$(8,3)$	$(3,8)$

modifies its routing table(Table 4.4) to stop behaving as an agent for $(2,3)$. After that, $(3,3)$ sends an *OK* message to $(2,3)$. Once $(2,3)$ receives an *OK* message from $(3,3)$, it joins the system.

4.3.5 Update of Routing Table

The routing table is updated according to the following rules:

1. Node A sends a *confirm* message to the nodes of its T_h and T_v.

2. If any nodes in its T_h and T_v do not answer, node A sends a *confirm* message to node B which is the agent node of no answering node in its agent list.

3. When node B receives a *comfirm* message, node B modifies its routing table to behave as an agent for no answering node. After that, node B sends an *OK* message to node A.

Table 4.5: The modified routing table of $(9,3)$.

T_h	T_v	T_v of $(8,3)$
$(8,3) \rightarrow (9,3)$	$(9,2)$	$(8,2)$
$(1,3)$	$(9,1)$	$(8,1)$
$(4,3)$	$(9,4)$	$(8,4)$
$(8,3)$	$(9,8)$	$(8,8)$

Table 4.6: The modified routing table of $(2,3)$.

T_h	T_v
$(2,3)$	$(3,2)$
$(1,3)$	$(3,1)$
$(4,3)$	$(3,4)$
$(8,3) \rightarrow (9,3)$	$(3,8)$

4. Once node A receives an OK message from the node, it modifies the routing table.

Example $(2,3)$ sends a $confirm$ message to the nodes of its T_h and T_v in Table 3.1. If $(8,3)$ doesn't answer, $(2,3)$ sends a $confirm$ message to $(9,3)$ which is the agent node of $(8,3)$ in Table 4.2. When $(9,3)$ receives a $confirm$ message, $(9,3)$ modifies its routing table(Table 4.5) to behave as an agent for $(8,3)$. After that, $(9,3)$ sends an OK message to $(2,3)$. Once $(2,3)$ receives an an OK message from $(9,3)$, it modifies the routing table(Table 4.6).

4.3.6 Simulation Results

To confirm that the routing performance of GDR when some nodes fails, we consider the following two scenarios:

1 State of Nodes

We evaluate the performance in states of *active*, *down*, *leave*, and *update*. We simulated the behavior of the GDR of size $N(= n \times n = 2^{2r})$, where $2 \leq r \leq 5$. We executed $10\times N$ random lookups to the GDR. During the simulation, we measured the *path length* and the *relay length* for each lookup.

2 Agent node

We evaluate the performance in states of *active* and *down*. We simulated the behavior of the GDR of size $N = 1024$, each node maintains agent list of size $r = 2\log N$. We executed $10\times N$ random lookups to the GDR. During the simulation, we measured the *path length* and the *relay length* for each lookup.

4.3.6.1 State of Nodes

1 *down*

In case that when all nodes are *active*, we execute random lookups.

Next some nodes are *down* with probability $1/16$, we execute random lookups again.

Finally *active* nodes are *update*, we execute random lookups once again.

2 *leave*

In case that when all nodes are *active*, we execute random lookups.

Next some nodes are *leave* with probability $1/16$, we execute random lookups again.

Finally *active* nodes are *update*, we execute random lookups once again.

Table 4.7, Table 4.8, Table 4.9, and Table 4.10 show the mean and the variance of the *path length* and the *relay length*, and the lookup success rate in state of nodes are *down* with probability $1/16$, *leave* with probability $1/16$, *active*, and *update*, respectively. Figure 4.4, Figure 4.5, Figure 4.6, Figure 4.7, and Figure 4.8 show the mean of the *path length*, the variance of the *path length*, the mean of the *relay length*, the variance of the *relay length*, and the lookup success rate, respectively. The lookup success rate is obtained by eq. (4.1).

$$\begin{aligned} &lookup\ \ success\ \ rate\ \ [\%] \\ &= \frac{number\ \ of success\ \ in\ \ random\ \ lookups}{number\ \ of\ \ random\ \ lookups} \times 100. \end{aligned} \tag{4.1}$$

In Table 4.7, when state of node X is *down*, if X is the destination or the forwarding node, a link error occurs. On the other hand, in Table 4.8, if state of node X is *leave*, then the neighbor node Y behaves as an agent for X, a link error doesn't occur when Y is the sending or the forwarding node to X. That is why the mean and the variance of the *path length* and the *relay length* are almost the same, while each lookup success rate in Table 4.8 is higher than that in Table 4.7.

In Table 4.10, a link error doesn't occur after the routing table of all *active* nodes are updated. Therefore, the message arrives at its final destination. That is why the mean and the variance of the *path length* and the *relay length* in Table 4.10 are larger than those in Table 4.7 and Table 4.8. In addition, each lookup success rate becomes 100%.

Figure 4.4, Figure 4.5, Figure 4.6, and Figure 4.7 show Eq. (3.28), Eq. (3.29), Eq. (3.46) and Eq. (3.48), respectively.

Table 4.7: H_g, D_g, and lookup success rate when state of some nodes are *down* with probability $1/16$.

N	H_g		D_g		success
	Avg	Var	Avg	Var	rate [%]
16	1.83	0.83	2.26	1.50	86.2
64	3.01	1.39	5.32	7.11	85.9
256	3.92	2.00	10.24	26.81	81.0
1024	4.89	4.30	20.59	111.60	79.7

Table 4.8: H_g, D_g, and lookup success rate when state of some nodes are *leave* with probability $1/16$.

N	H_g		D_g		success
	Avg	Var	Avg	Var	rate [%]
16	1.80	0.86	2.24	1.54	88.3
64	2.99	1.39	5.29	7.13	86.7
256	3.89	1.99	10.20	26.63	82.9
1024	4.87	4.33	20.60	112.64	81.4

Table 4.9: H_g, D_g, and lookup success rate when state of all nodes are *active*.

N	H_g		D_g		success
	Avg	Var	Avg	Var	rate [%]
16	1.99	0.88	2.48	1.76	100.0
64	3.05	1.36	5.31	6.67	100.0
256	4.01	1.99	10.58	27.95	100.0
1024	5.08	5.61	21.23	112.76	100.0

Table 4.10: H_g, D_g, and lookup success rate when state of *active* nodes are *update*.

N	H_g		D_g		success
	Avg	Var	Avg	Var	rate [%]
16	1.85	0.83	2.45	1.86	100.0
64	3.01	1.32	5.49	7.23	100.0
256	3.98	2.01	10.69	28.20	100.0
1024	5.02	5.27	21.29	113.42	100.0

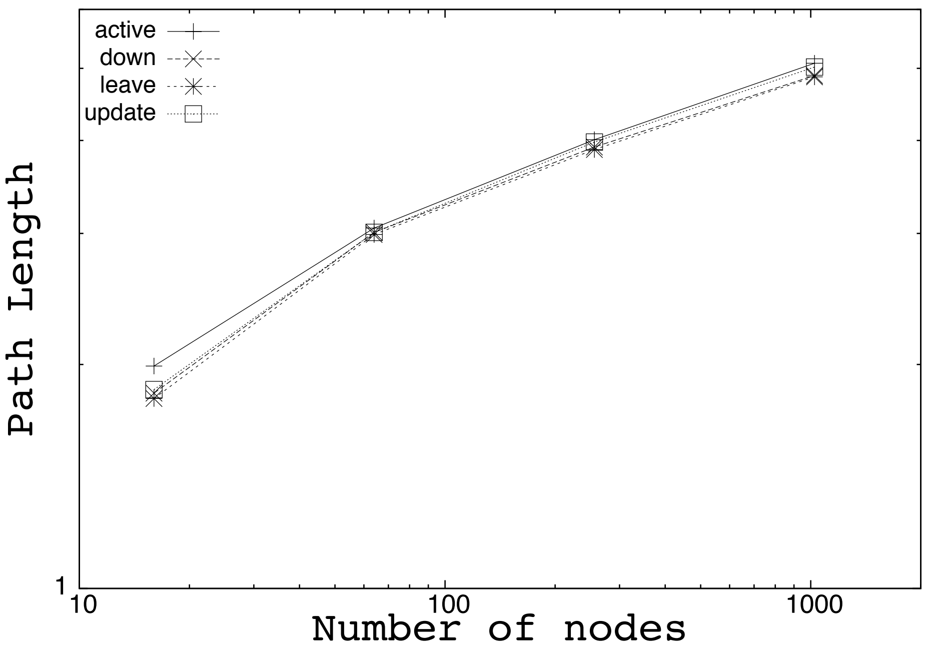


Figure 4.4: The mean of the Path Length.

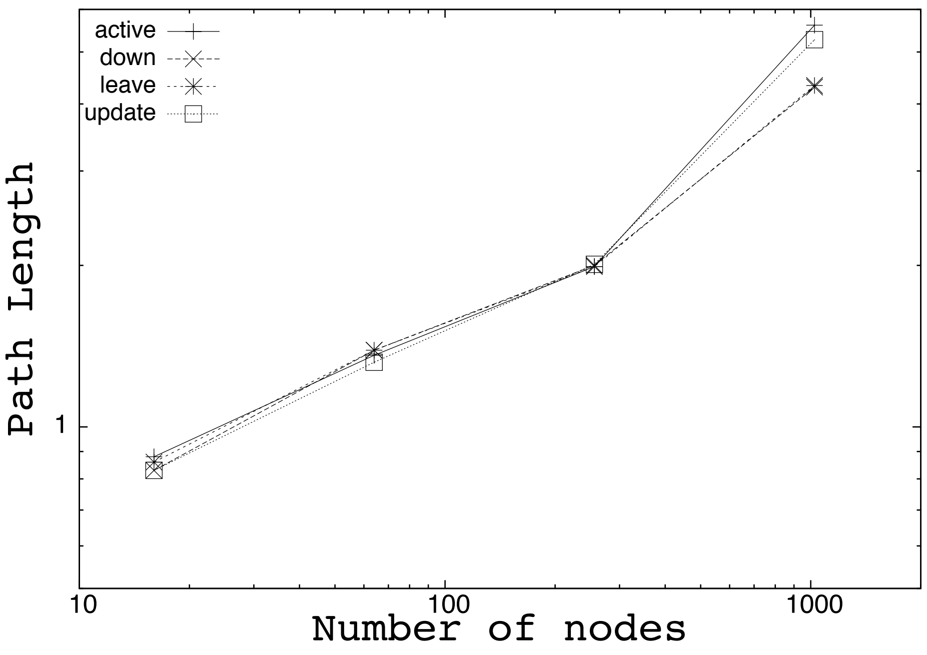


Figure 4.5: The variance of the Path Length.

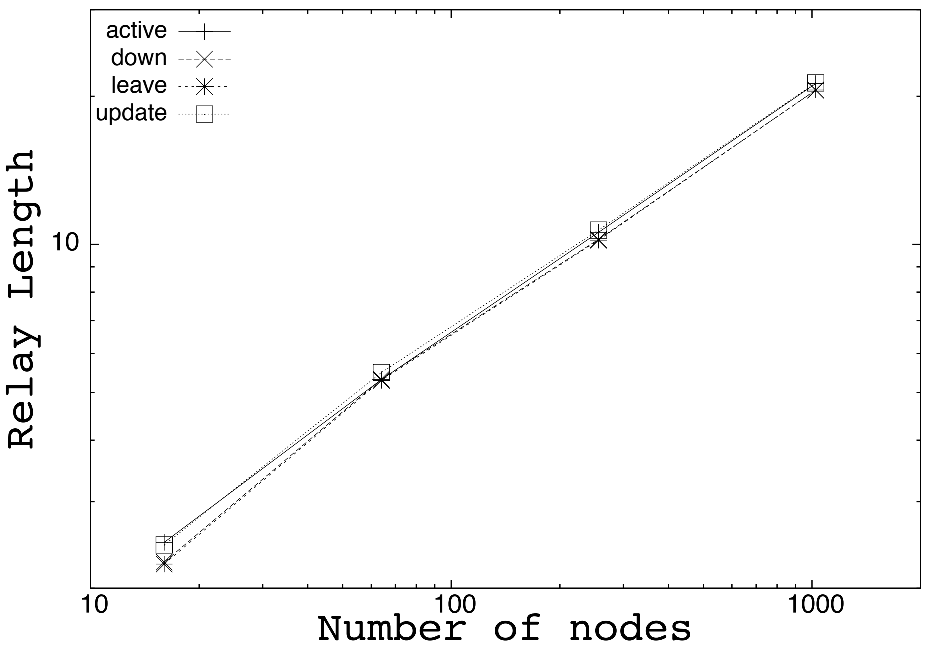


Figure 4.6: The mean of the Relay Length.

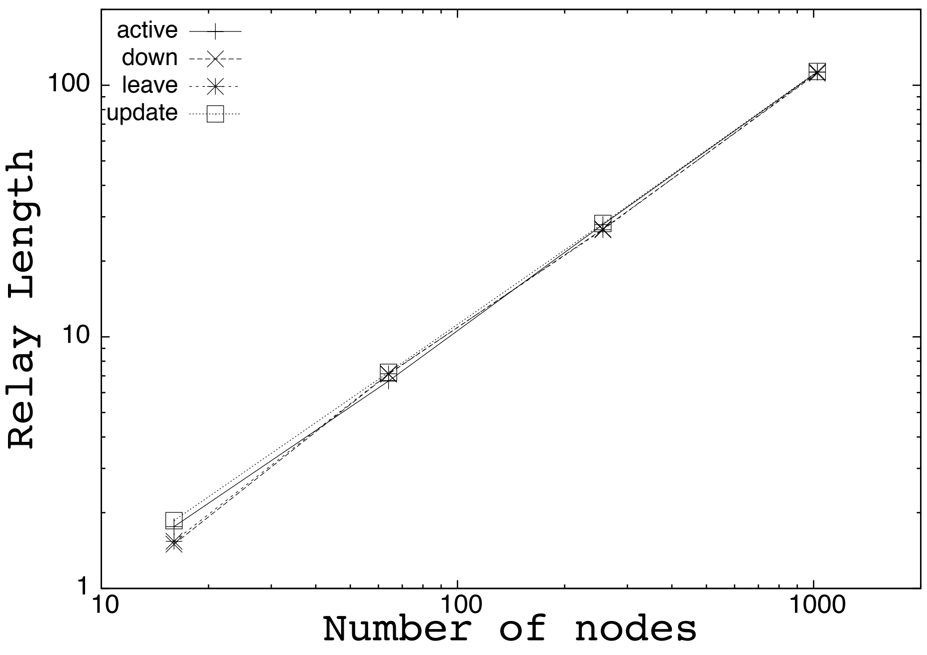


Figure 4.7: The variance of the Relay Length.

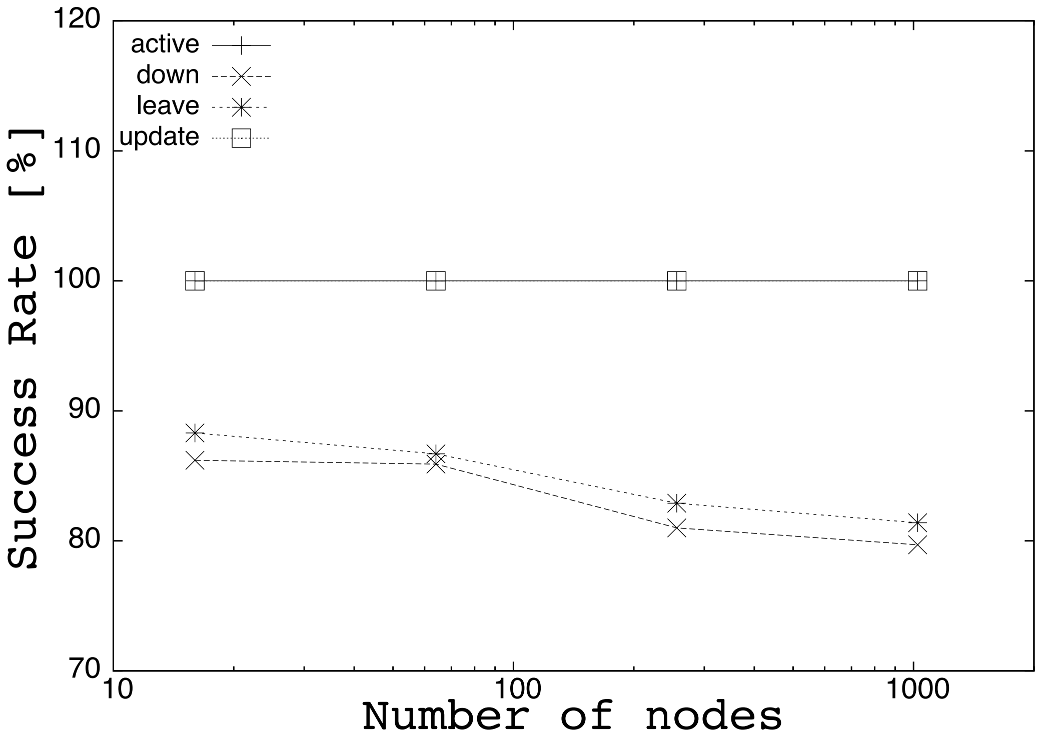


Figure 4.8: The Success Rate.

4.3.6.2 Agent node

1 Lookup without agent list

In case that $N = 1024$, all nodes are *active*, we execute random lookups. Next some nodes are *down* with probability $p = 0.1, 0.2, 0.3, 04$, and 0.5. Before the routing table of all *active* nodes are updated, we execute random lookups again.

2 Lookup with agent list

In case that $N = 1024$, all nodes are *active*, we execute random lookups with the routing table as well as the agent list.

Table 4.11: Lookup without agent list: H_g, D_g, when state of some nodes are *down* with probability p, N=1024.

p	H_g		D_g		success
	Avg	Var	Avg	Var	rate [%]
0	5.08	5.61	21.23	112.76	100.0
0.1	4.78	4.19	20.00	109.97	72.35
0.2	4.56	3.78	19.35	105.92	63.34
0.3	4.29	2.86	18.61	114.19	58.04
0.4	4.17	3.40	17.83	106.70	56.61
0.5	3.88	2.28	16.56	101.43	55.40

Table 4.12: Lookup with agent list: H_g, D_g, when state of some nodes are *down* with probability p, N=1024.

p	$H_g(agent)$		$D_g(agent)$		success
	Avg	Var	Avg	Var	rate [%]
0	4.33	2.41	21.23	112.79	100.0
0.1	4.08	1.92	20.00	107.68	74.52
0.2	3.87	1.88	19.02	103.58	64.55
0.3	3.72	1.97	18.50	113.75	59.23
0.4	3.55	1.79	17.72	103.19	57.59
0.5	3.37	1.78	16.88	102.47	56.24

Next some nodes are *down* with probability $p = 0.1, 0.2, 0.3, 04$, and 0.5. Before the routing table of all *active* nodes are updated, we execute random lookups with the routing table as well as the agent list again.

Table 4.11 shows the mean and the variance of the *path length* and the *relay length*, and the lookup success rate when random lookups with the routing table in state of some nodes are *down* with probability p, and N=1024. Table 4.12 shows the mean and the variance of the *path length* and the *relay length*, and the lookup success rate when random lookups with the routing table as well as the agent list in state of some nodes are *down* with probability p, and N=1024.

Table 4.11 and Table 4.12 show that the success rate decreases as the number of failed node increases. Furthermore, the mean and the variance of the *path length* and the *relay length* in Table 4.11 are larger than those in Table 4.12. Moreover, each success rate in Table 4.12 is higher than that in Table 4.11 at the same p ($0 < p$).

4.4 Conclusion

In this Chapter, we evaluated the performance of the GDR system by simulation.

In fix search, the mean and the variance of the *relay length* are constant even in the case of Kademlia that uses XOR distance for routing. On the other hand, GDR and Kademlia have the same maximum value of the mean and the variance of the *relay length*, whereas Chord and Kademlia have the same minimum value of the mean and the variance of the *relay length*.

We also show that the ID is generated by using SFC, the mean *relay length* of GDR is about half of that of Chord, and about $2/3$ of that of Kademlia.

In addition, the GDR system has a routing redundancy to increase robustness. When a node fails, its neighbor node behaves as an agent for the failing node. To know the agent node of the failing node, each node has an agent list which is the records of the agent nodes of the nodes of its routing table. Since the number of the agent nodes is 2, the size of the agent list is $2 \log N$. If an underlay network can be modeled as a grid, it is easy to assign a physical address for a node. However, if a node fails, it is difficult to modify or change its physical address. In the GDR system, the nodes can avoid a failed node by using its agent list on the overlay network.

Chapter 5

Application

5.1 Introduction

In the GDR system, one network node is installed in each area and the nodes are immovable. They serve as super nodes that collect data from other mobile terminals. The mobile terminals communicate with the nodes in wireless manner. The size of the area is larger than the radio range. When the mobile terminal is in the radio range, it can store the information and send a query to the node, and receive a reply from the node.

However, as shown in Figure 5.1, if the mobile terminal moves to a neighboring area before it stores the information to the node, the node cannot collect the information gathered by the mobile terminal. Furthermore, if the mobile terminal moves to a neighboring area before it receives reply from the node, the node cannot send a reply to the mobile terminal.

In the GDR system, the nodes maintain the routing tables to enable the nodes to search for location-oriented information of other places. Therefore, as shown in

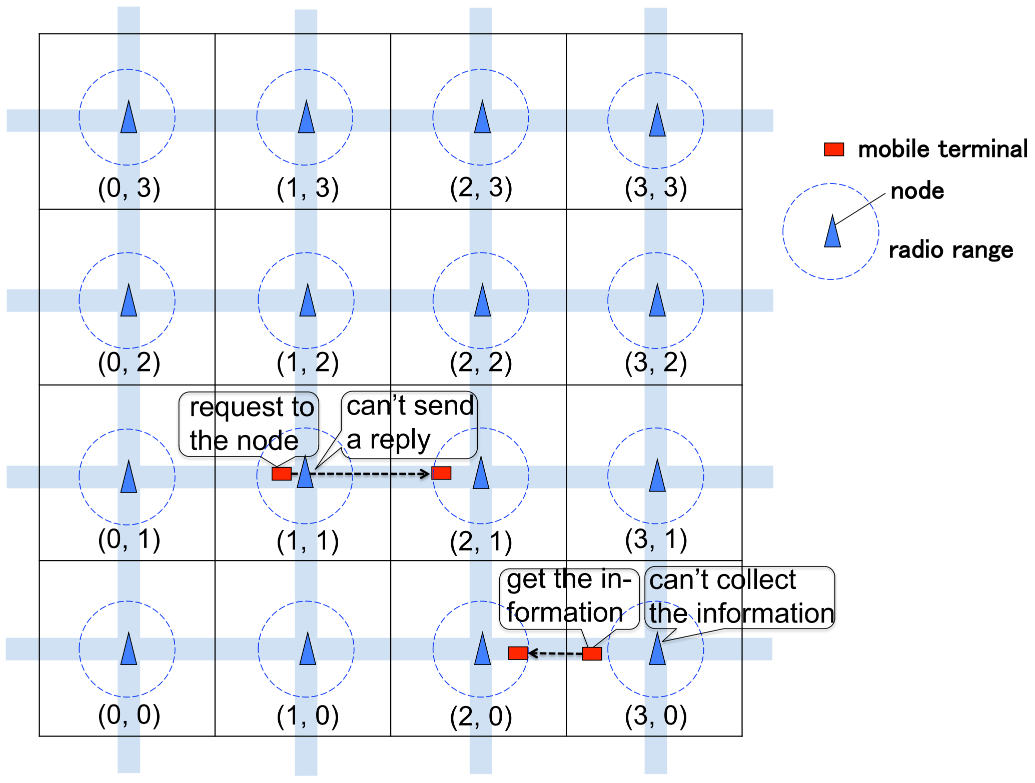


Figure 5.1: Communication between a node and a mobile terminal without a function to forward the messages.

Figure 5.2, if the nodes have a function to forward the messages, the nodes collect the information gathered by a mobile terminal after it moves to a neighboring area via the neighboring node. Furthermore, they can forward a reply to the neighboring node and enable to send the reply to a mobile terminal after it moves to the neighboring area.

The advantage of GDR system is that the mobile terminals do not need to manage the network. Thus, it is possible to decrease the communication overhead of the mobile terminals.

For instance, Mill[19] has a routing table of size m which consists of information of $m/2$ mobile terminals in each side, clockwise or counterclockwise. Then,

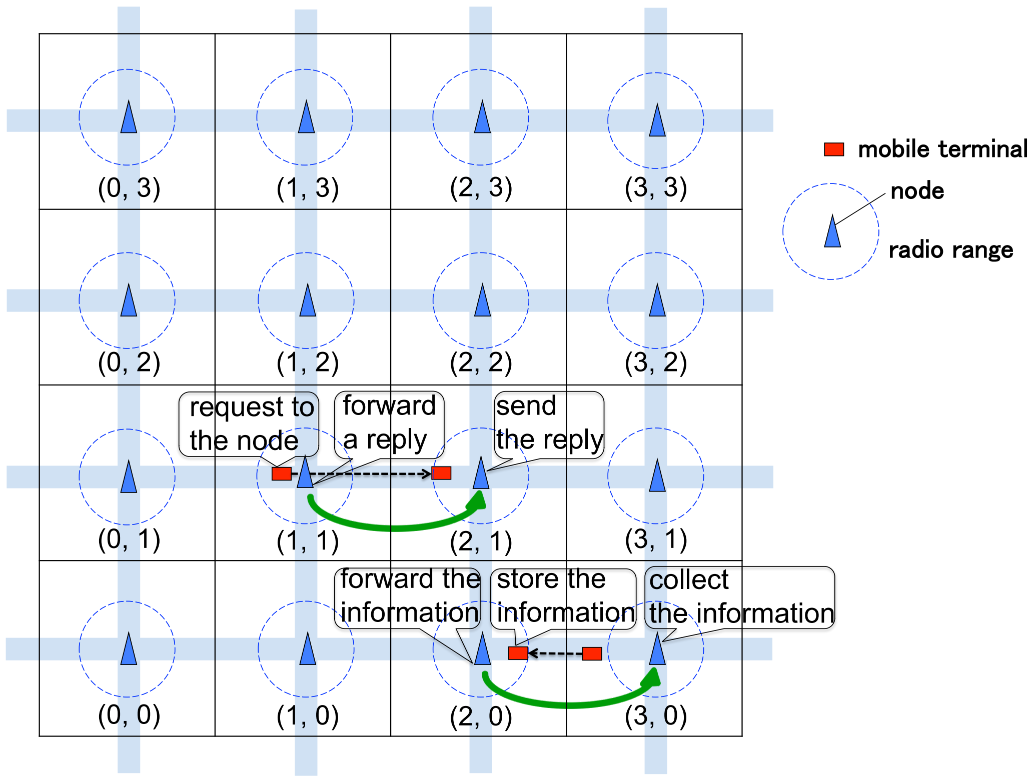


Figure 5.2: Communication between a node and a mobile terminal with a function to forward the messages.

a node search can be done in $(\log N)/2$ steps on average if the number of mobile terminals in the network is no more than $2\sqrt{N}$.

Once a mobile terminal obtains the information, the mobile terminal records the ID of the location where the information is generated. Then, this mobile terminal sends the ID and its IP-address to a clockwise side mobile terminal on the ID space. The clockwise side mobile terminal sends this message to the next clockwise side mobile terminal. Sending the message to the clockwise, a particular mobile terminal which handles the ID-space including the ID receives this message. The mobile terminal manages the ID with the IP-address. Therefore, the communication overhead of the mobile terminals increases.

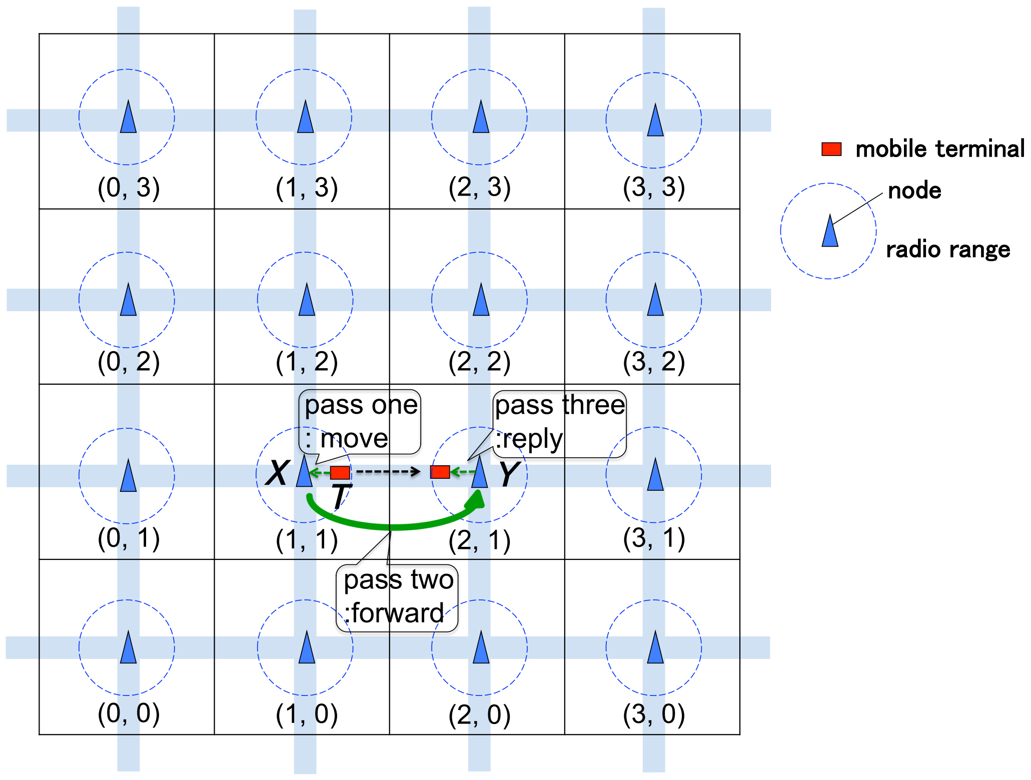


Figure 5.3: Wall pass in the GDR system.

The communication overhead of the mobile terminals doesn't decrease even if they have a routing table based on Chord or Kademlia.

In Chapter 3 and 4, we evaluated random lookups to analyze the basic performance of the nodes. However, practical lookups depend on the requirement from the mobile terminals. In this chapter, we evaluate the GDR system when the mobile terminals are moving to a neighboring area.

The rest of this chapter is organized as follows. In section 5.2, we describe the type of a message. Next in section 5.3, the detailed description of the proposed algorithm is given. Section 5.4 describes the mobility model of the mobile terminals. Section 5.5 evaluates the performance.

5.2 Type of Message

On receiving or sending a message m, a node X processes it according to its type. There are nine types of messages as follows:

$data$: m issued by a mobile terminal T is delivered to X and stored in its data storage.

$store$: m issued by another node Y is delivered to X and stored in its data storage.

$query$: m is a query issued by T.

$reply$: m is a reply which X gives to a query issued by T.

$request$: m is a request that X makes to Y.

$response$: m is a response of X to another node Y which is the first node that makes a request to X.

$move$: m is a notification issued by T that has given a query to X but makes a move out of its access area with no reply.

$retrieve$: m is a notification issued by T that has issued a move notification to another node and then reaches the access area of X.

$forward$: m is to be forwarded to a neighboring node of X

5.3 Wall Pass Algorithm

In order to send a reply to a mobile terminal after it moves to a neighboring area, we refer to the movement of wall pass in football. Wall pass is a triangular movement where a player A gets past an opponent by making a short pass to a wall player B and running toward the return pass. Wall pass is also known as a one-two pass. A passes a ball to B (pass one). After that A moves into space and receives a very quick pass from B (pass two). [28, 29]

In the GDR system, we consider a node as a wall player. In Figure 5.3, a mobile terminal T sends a *move* message to a node X (pass one). While T moves to a neighboring area, X sends a $forward$ message to another node Y (pass two). After that T receives a *reply* message from Y(pass three).

X receives the messages from T and Y. To process the messages coming from T and Y, X uses two $Wall\ Pass$ (WP) algorithm: $WP1$ and $WP2$, which will be described below. When X receives a message from T, X uses algorithm $WP1$. On the other hand, when X receives a message from Y, X uses algorithm $WP2$.

5.3.1 Algorithm $WP1$

First, we explain how a node processes the messages coming from a mobile terminal according to the flow chart in Figure 5.4.

Step 1:

When X receives a message from T, if its type is *data*, *query*, *move*, and *retrieve*, then go to Step 2, Step 3, Step4, and Step 5, respectively.

Step 2:

It knows which node is responsible for the message from its location information. If X is so, then X stores a data. Otherwise, X picks up a node Y that is the closest to the responsible node among the nodes in its routing table. After that X modifies its type from *data* to *store*, and then X sends it to Y.

Step 3:

It knows which node is responsible for the message from its location information. // It is strange here? what is it? If *X* is so, then *X* sends a message as a *reply* to T. // what is If X is so? Otherwise, *X* picks up a node *Y* that is the closest to the responsible node among the nodes in its routing table. After *X* modifies its type from *query* to *request*, then *X* sends it to Y.

Step 4:

X records that *T* moves to the neighboring node.

Step 5:

X records that *T* arrives.

5.3.2 Algorithm $WP2$

Next, we explain how a node processes a message coming from another node based on the flow chart in Figure 5.5.

Step 1:

When *X* receives a message from *Y*, if its type is *store*, *request*, *response*, and $forward$, then go to Step 2, Step3, Step 4, and Step 5, respectively.

Step 2:

It knows which node is responsible for the message from its location information. If *X* is so, then *X* stores a data. Otherwise, *X* picks up a node *Y* that is the closest to the responsible node among the nodes in its routing table, then *X* sends it as a *store* to Y.

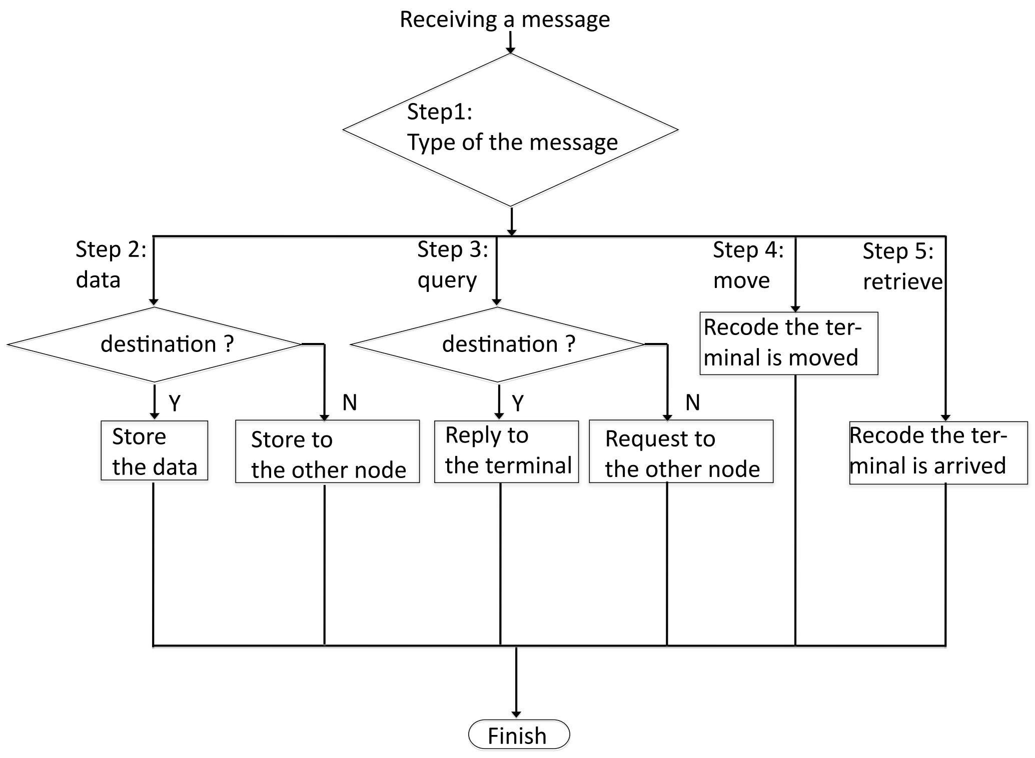


Figure 5.4: Flow chart for algorithm $WP1$.

Step 3:

It knows which node is responsible for the message from its location information. If X is so, then X sends a message as a $response$ to the first $request$ sending node. Otherwise, X picks up a node Y that is the closest to the responsible node among the nodes in its routing table, then X sends it as a $request$ to Y.

Step 4:

X checks the mobile terminal records. If T is in this area, then X sends a message as a $reply$ to T. If T has been moved to the neighboring area, then X sends a message as a $forward$ to the neighboring node.

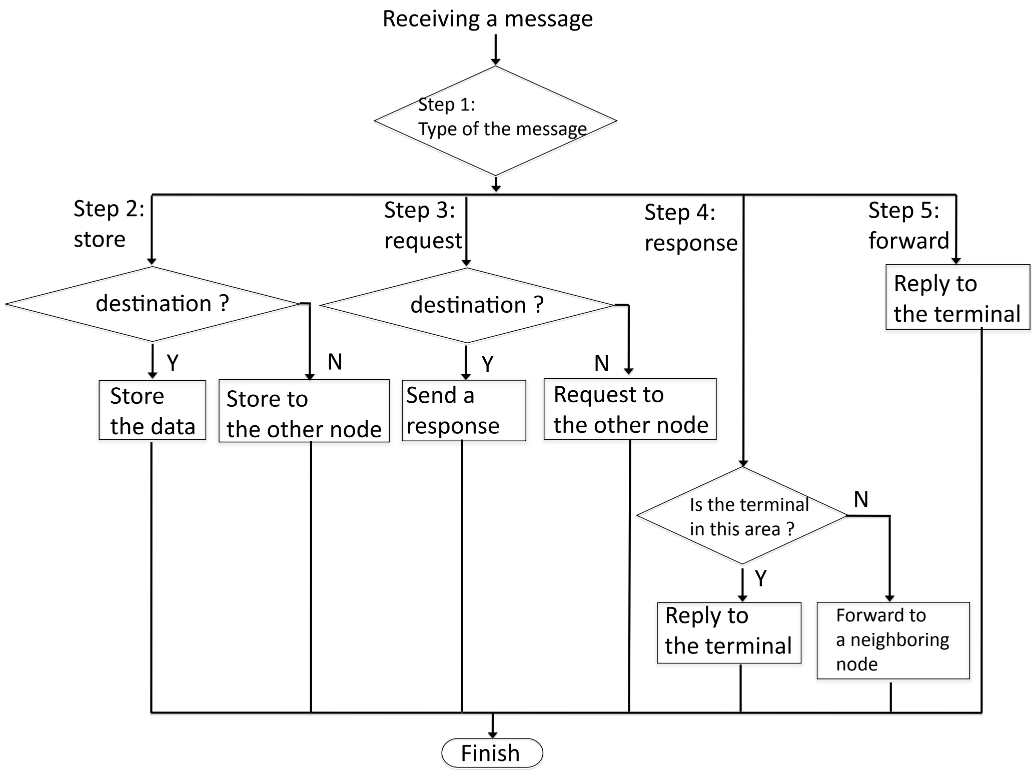


Figure 5.5: Flow chart for algorithm *WP*2.

Step 5:

X sends a message as a *reply* to T.

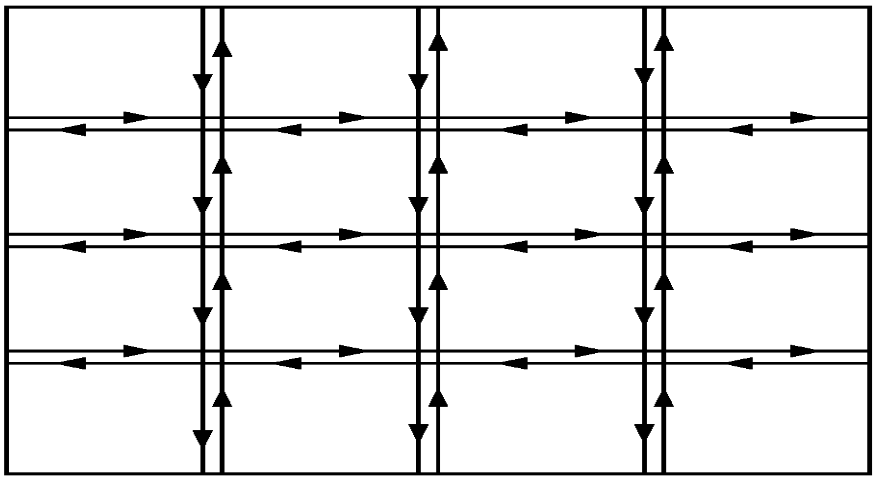


Figure 5.6: The map used in Manhattan Model[27].

5.4 Mobility Model

We assume the GDR system works on an urban area. Therefore, we use the Manhattan model[27] which is a mobility model for the urban traffic. It emulates the movement of the mobile terminals on the streets defined by map. The map used in Manhattan model is shown in Figure 5.6.

The map is composed of horizontal and vertical streets. Each street has two lanes for each direction (North and South direction for vertical streets, East and West for horizontal streets). The mobile terminals are allowed to move along the grid of the horizontal and the vertical streets on the map.

The mobile terminals are placed at each intersections initially. At an intersection of the map, they can turn left, turn right, stop, or go straight, with probability 0.2, 0.2, 0.2, or 0.4, respectively. If a mobile terminal moves beyond the boundary of the map, it returns reflectively. According to these rules, we generate a mobility scenario of the mobile terminals.

5.5 Simulation Results

To evaluate the performance when the mobile terminals are moving, we consider the following three scenarios:

1 Storing Data

We evaluate the performance of the system when the mobile terminals are moving. We simulate the behavior of Chord, Kademlia, CAN, and GDR for size N=16, 64, 256, and1024. We execute the number of mobile terminal is $M = N$, when the mobile terminals are in the radio range, they can store to a node ten times. During the simulation, we measure the *path length*, the *relay length*, and the number of *store* forwarding times for each store.

2 Sending Query

We evaluate the performance of the system when the mobile terminals are moving. We simulate the behavior of the GDR for size N=16, 64, 256, and1024. We execute the number of mobile terminal is $M = N$, when the mobile terminals are in the radio range, they can send a *query* to a node ten times randomly and receive each reply from a node successfully. During the simulation, we measure the *path length*, the *relay length*, and the number of *request* forwarding times for each lookup.

3 Response

We evaluate the performance when the mobile terminals are moving with or without WP algorithm. We simulate the behavior of the GDR for size N=16, 64, 256, and1024. We execute the number of mobile terminal is $M = N$, when the mobile terminals are in the radio range, they send a *query* to a node ten times randomly and receive each reply from a node. During the simulation, we measure the *path length* and the *relay length* for each lookup and the number of *query* sending times.

5.5.1 Storing Data

Table 5.1 and Table 5.2 show the mean and the variance of the *path length* and the *relay length* of Chord, Kademlia, CAN, and GDR, respectively.

Figure 5.7 and Figure 5.8 show cumulative distribution function of the *path length* and the *relay length* when storing data, respectively, where $N = M = 1024$.

Table 5.3 shows the mean and the variance of the *store* forwarding times when storing data.

Figure 5.9, Figure 5.10, and Figure 5.11 show the number of *store* forwarding times of each node, respectively, where $N = M = 64$.

In Table 5.2, the mean and the variance of the *relay length* of Chord are larger than those of Kademlia and GDR. Because Chord is a one-way clockwise routing, the *relay length* becomes large by the position of the source node and the destination node. However, since DGR is a two-way approachable routing, the *relay length* is fixed by the position of the source node and the destination node. Greater *relay length* means greater propagation time.

The results show that GDR and CAN have the same mean and variance of the *path length* and the *relay length*, while the mean and the variance of the *relay length* and the *relay length* of GDR are smaller than those of Chord and Kademlia.

Figure 5.7, Figure 5.8, Table 5.3, and Figure 5.9 – Figure 5.11show similar results. The reason for this is that CAN and GDR know the geometrically neigh-

Table 5.1: The mean and the variance of the *path length* when storing data.

N & M	H_c		H_k		H_n		H_g	
	Avg	Var	Avg	Var	Avg	Var	Avg	Var
16	0.98	0.64	0.86	0.51	0.67	0.22	0.67	0.22
64	1.41	1.45	1.18	0.90	0.74	0.19	0.72	0.19
256	1.85	2.78	1.31	1.18	0.76	0.18	0.76	0.18
1024	2.33	4.65	1.44	1.50	0.78	0.17	0.78	0.17

Table 5.2: The mean and the variance of the *relay length* when storing data.

N & M	D_c		D_k		D_n		D_g	
	Avg	Var	Avg	Var	Avg	Var	Avg	Var
16	1.51	2.65	1.06	1.10	0.67	0.22	0.67	0.22
64	3.48	18.31	1.85	4.47	0.74	0.19	0.72	0.19
256	7.73	99.63	2.45	11.79	0.76	0.18	0.76	0.18
1024	16.37	437.65	3.28	32.40	0.78	0.17	0.78	0.17

boring node IDs.

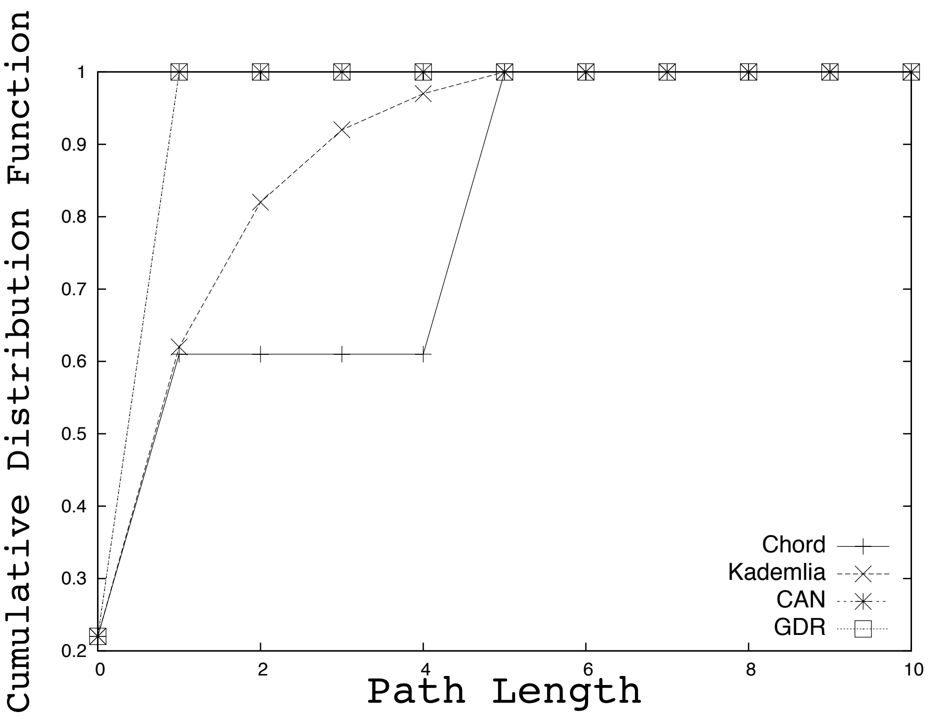


Figure 5.7: Cumulative Distribution Function of the *path length* when storing data ($N = M = 1024$).

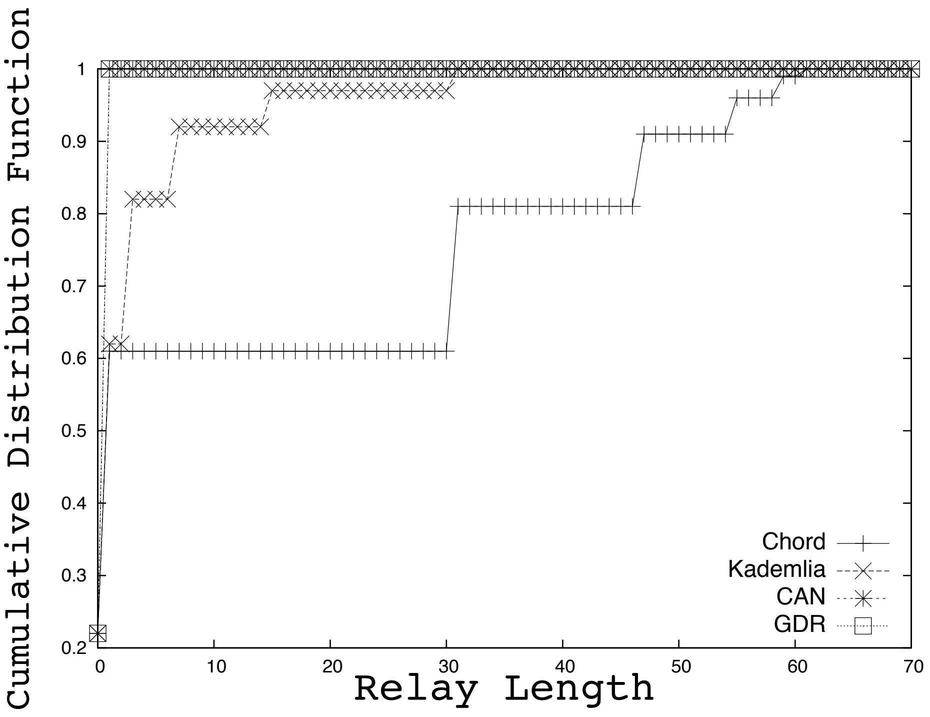
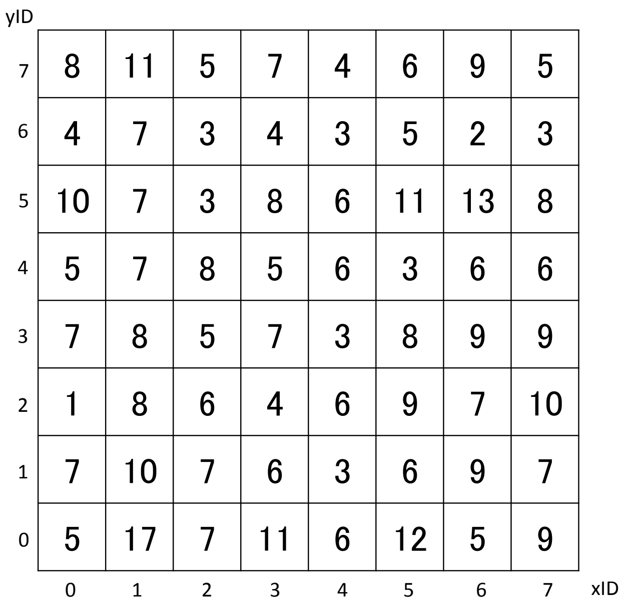


Figure 5.8: Cumulative Distribution Function of the *relay length* when storing data ($N = M = 1024$).

Table 5.3: The mean and the variance of the *store* forwarding times when storing data.

N & M	S_c		S_k		S_n		S_g	
	Avg	Var	Avg	Var	Avg	Var	Avg	Var
16	3.06	2.81	1.94	1.56	0.00	0.00	0.00	0.00
64	6.75	8.03	4.42	7.87	0.00	0.00	0.00	0.00
256	10.86	14.49	5.50	12.45	0.00	0.00	0.00	0.00
1024	15.53	14.50	6.62	15.31	0.00	0.00	0.00	0.00



yID \ xID	0	1	2	3	4	5	6	7
7	8	11	5	7	4	6	9	5
6	4	7	3	4	3	5	2	3
5	10	7	3	8	6	11	13	8
4	5	7	8	5	6	3	6	6
3	7	8	5	7	3	8	9	9
2	1	8	6	4	6	9	7	10
1	7	10	7	6	3	6	9	7
0	5	17	7	11	6	12	5	9

Figure 5.9: Chord : the number of *store* forwarding times of each node (N = M = 64).

yID								
7	11	3	11	6	6	8	6	7
6	2	0	1	1	2	2	0	4
5	9	3	5	9	5	5	2	9
4	5	2	3	5	6	3	2	9
3	8	2	3	4	3	4	0	7
2	1	5	2	5	5	4	4	5
1	4	0	2	1	1	3	0	5
0	5	9	5	8	6	8	5	7
	0	1	2	3	4	5	6	7 xID

Figure 5.10: Kademlia : the number of *store* forwarding times of each node ($N = M = 64$).

yID								
7	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
	0	1	2	3	4	5	6	7 xID

Figure 5.11: CAN and GDR : the number of *store* forwarding times of each node ($N = M = 64$).

Table 5.4: The mean and the variance of the *path length* when sending query.

N & M	H_c		H_k		H_n		H_g	
	Avg	Var	Avg	Var	Avg	Var	Avg	Var
16	2.03	1.07	2.06	1.00	2.61	2.04	2.13	1.06
64	3.02	1.46	3.02	1.42	5.19	7.69	2.99	1.53
256	4.04	2.03	4.01	2.00	10.55	28.58	3.99	2.02
1024	5.00	2.53	4.99	2.50	21.27	115.10	5.00	2.54

Table 5.5: The mean and the variance of the *relay length* when sending query.

N & M	D_c		D_k		D_n		D_g	
	Avg	Var	Avg	Var	Avg	Var	Avg	Var
16	3.56	3.08	2.49	2.31	2.61	2.04	2.61	2.04
64	8.71	19.70	6.98	10.35	5.19	7.69	5.19	7.69
256	19.69	94.19	15.03	43.22	10.55	28.58	10.55	28.58
1024	40.67	394.30	31.02	171.97	21.27	115.10	21.27	115.10

5.5.2 Sending Query

Table 5.4 and Table 5.5 show the mean and the variance of the *path length* and the *relay length* of Chord, Kademlia, CAN, and GDR, respectively. It shows the communication overhead of the GDR system. The mobile terminals in the GDR system do not need to manage the network. Therefore, it is possible to decrease the communication overhead of the mobile terminals. However, if the mobile terminals manage the network, the communication overhead of the mobile terminals increases as Table 5.4.

Figure 5.12 and Figure 5.13 show cumulative distribution function of the *path length* and the *relay length* when sending query, respectively, where $N = M =$

1024.

Table 5.6 shows the mean and the variance of the *request* forwarding times when sending query.

Figure 5.14, Figure 5.15, Figure 5.16, and Figure 5.17 show the number of *request* forwarding times of each node, respectively, where $N = M = 64$.

Table 5.4, Table 5.5, Figure 5.12, and Figure 5.13 show that GDR, Chord and Kademlia have the same mean and the same variance of the *path length*, while the mean and the variance of the *relay length* of GDR are smaller than those of Chord and Kademlia. Furthermore, while GDR and CAN have the same mean and the same variance of the *relay length*, the mean and the variance of the *path length* of GDR are smaller than those of CAN.

On the other hand, Table 5.6 shows that the variance of the *request* forwarding times of GDR are larger than these of Chord and Kademlia. Moreover, Figure 5.14 – Figure 5.17 show that the *request* forwarding times of the central nodes are larger than those of the boundary nodes in GDR. In the GDR system, the central nodes's IDs are often selected by other nodes for its routing table. As a result, the central nodes forward the *request* when random lookups.

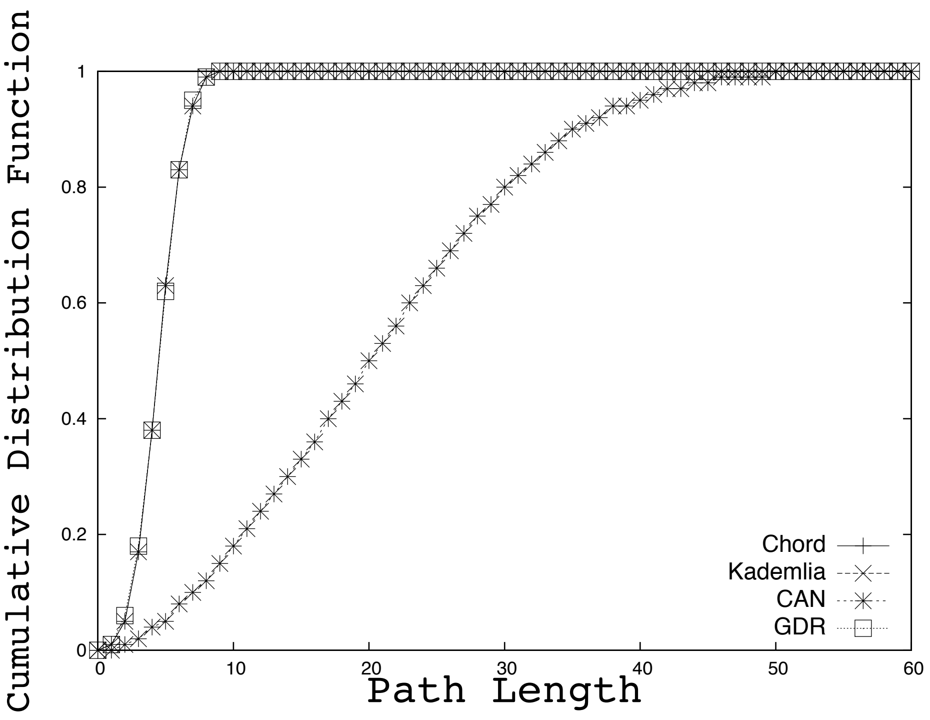


Figure 5.12: Cumulative Distribution Function of the *path length* when sending query ($N = M = 1024$).

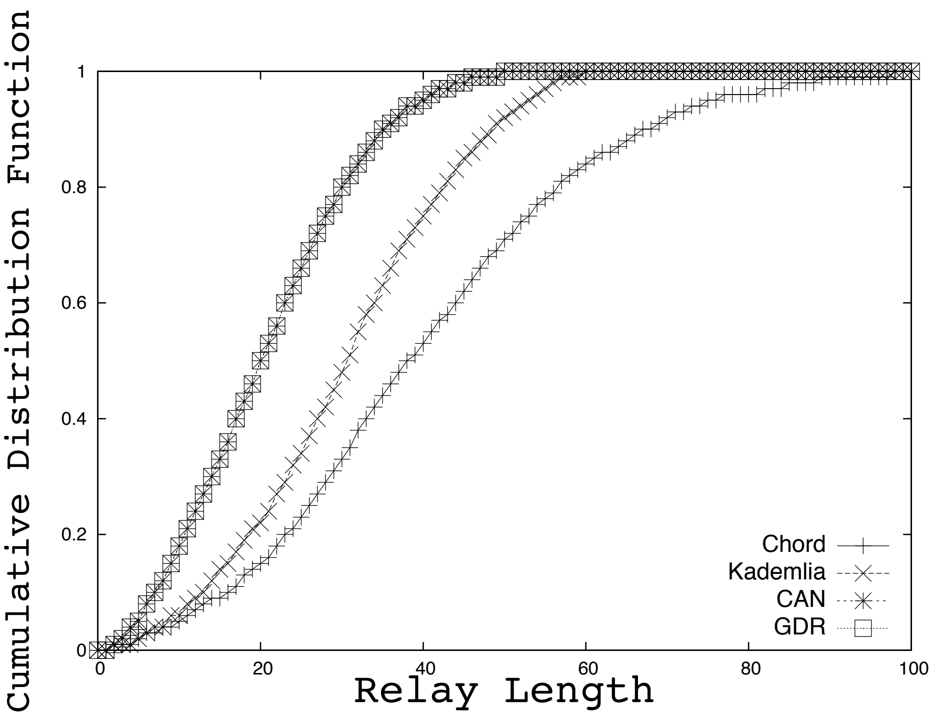
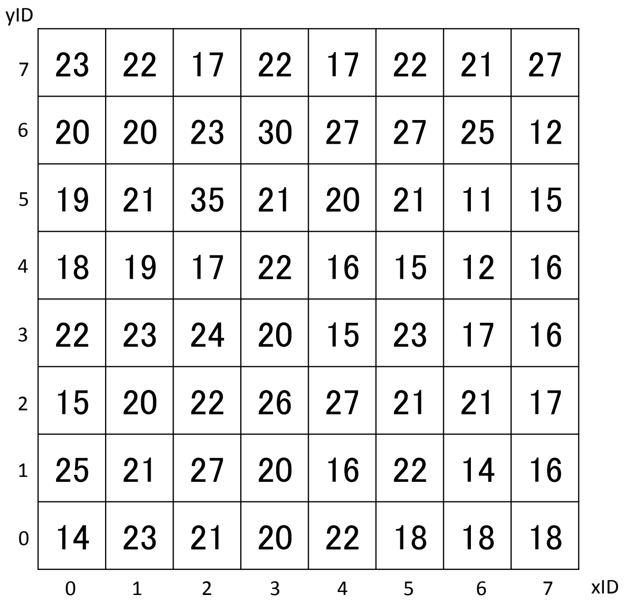


Figure 5.13: Cumulative Distribution Function of the *relay length* when sending query ($N = M = 1024$).

Table 5.6: The mean and the variance of the *request* forwarding times when sending query.

N &	S_c		S_k		S_n		S_g	
M	Avg	Var	Avg	Var	Avg	Var	Avg	Var
16	10.94	14.43	11.25	16.44	16.75	73.44	12.00	32.88
64	20.27	19.95	20.31	20.09	42.03	260.75	20.03	89.72
256	30.45	35.10	30.14	34.32	95.51	1198.56	29.93	288.07
1024	39.97	38.85	39.93	39.65	202.68	4729.42	39.99	857.54



yID								
7	23	22	17	22	17	22	21	27
6	20	20	23	30	27	27	25	12
5	19	21	35	21	20	21	11	15
4	18	19	17	22	16	15	12	16
3	22	23	24	20	15	23	17	16
2	15	20	22	26	27	21	21	17
1	25	21	27	20	16	22	14	16
0	14	23	21	20	22	18	18	18
	0	1	2	3	4	5	6	7 xID

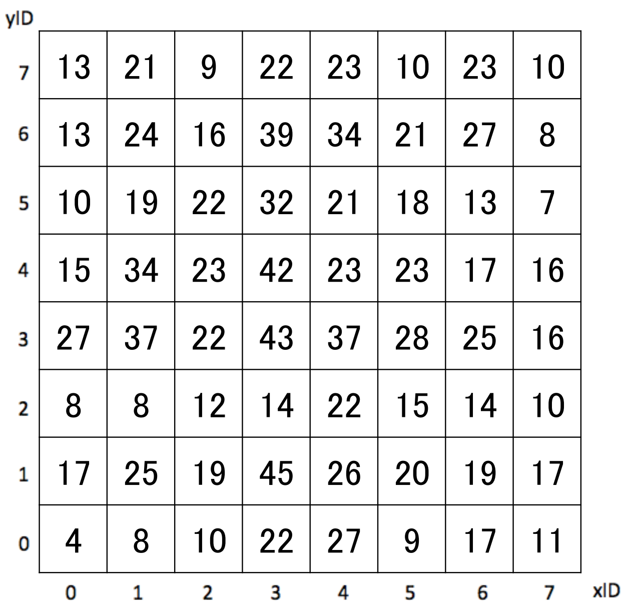
Figure 5.14: Chord : the number of *request* forwarding times of each node (N = M = 64).

yID								
7	24	27	23	15	15	18	23	20
6	19	19	17	29	27	26	25	13
5	23	22	33	26	24	21	16	12
4	18	20	16	20	16	15	11	14
3	30	25	21	17	21	22	16	17
2	16	16	23	17	23	23	20	15
1	23	22	27	24	20	20	18	18
0	14	20	21	21	22	20	24	17
	0	1	2	3	4	5	6	7 xID

Figure 5.15: Kademlia : the number of *request* forwarding times of each node ($N = M = 64$).

yID								
7	13	30	37	39	39	35	27	10
6	26	45	51	64	59	57	53	16
5	34	56	60	69	59	57	42	24
4	38	56	55	62	52	53	39	27
3	36	52	59	74	68	59	41	27
2	29	34	48	56	54	49	37	32
1	22	38	55	68	57	47	32	27
0	4	18	32	45	40	33	22	11
	0	1	2	3	4	5	6	7 xID

Figure 5.16: CAN : the number of *request* forwarding times of each node ($N = M = 64$).



yID								
7	13	21	9	22	23	10	23	10
6	13	24	16	39	34	21	27	8
5	10	19	22	32	21	18	13	7
4	15	34	23	42	23	23	17	16
3	27	37	22	43	37	28	25	16
2	8	8	12	14	22	15	14	10
1	17	25	19	45	26	20	19	17
0	4	8	10	22	27	9	17	11
	0	1	2	3	4	5	6	7 xID

Figure 5.17: GDR : the number of *request* forwarding times of each node ($N = M = 64$).

Table 5.7: The mean and the variance of the *path length* and the *relay length* with *WP* algorithm.

N	H_g		D_g	
	Avg	Var	Avg	Var
16	2.75	1.51	3.23	2.46
64	3.72	1.76	5.92	7.77
256	4.75	2.19	11.30	28.61
1024	5.78	2.71	22.05	115.07

Table 5.8: The mean and the variance of the *path length* and the *relay length* without *WP* algorithm.

N	H_g		D_g		increasing rate
	Avg	Var	Avg	Var	
16	2.16	1.10	2.65	2.04	1.81
64	2.98	1.55	5.18	7.69	1.82
256	3.99	2.01	10.56	28.55	1.80
1024	5.00	2.54	21.24	114.71	1.80

5.5.3 Response

Table 5.7 and Table 5.9 show the mean and the variance of the *path length* and the *relay length* with *WP* algorithm. Table 5.8 and Table 5.10 show the mean and the variance of the *path length* and the *relay length*, and the increasing rate without *WP* algorithm. In case of Table 5.9 and Table 5.10, the mobile terminals move to a neighboring area before it receive reply from a node with probability 0.1 in scenario 3.

The increasing rate is obtained by eq. (5.1).

Table 5.9: The mean and the variance of the *path length* and the *relay length* with *WP* algorithm.

N	H_g		D_g	
	Avg	Var	Avg	Var
16	2.68	1.53	3.14	2.27
64	3.70	1.85	5.84	7.37
256	4.86	2.24	11.56	28.33
1024	5.87	2.74	22.12	110.84

Table 5.10: The mean and the variance of the *path length* and the *relay length* without *WP* algorithm.

N	H_g		D_g		increasing rate
	Avg	Var	Avg	Var	
16	2.04	0.91	2.49	1.60	1.90
64	2.91	1.45	5.04	6.83	1.88
256	4.01	1.93	10.68	28.08	1.87
1024	5.01	2.45	21.24	110.68	1.88

$$\begin{aligned} & increasing\ \ rate \\ & = \frac{number\ \ of\ \ query\ \ sending\ \ times\ \ without\ \ WP\ \ algorthm}{number\ \ of\ \ query\ \ sending\ \ times\ \ with\ \ WP\ \ algorthm}. \end{aligned} \tag{5.1}$$

The mean and the variance of the *path length* and the *relay length* in Table 5.8 and Table 5.10 are smaller than those in Table 5.7 and Table 5.9. However, the increasing rates are 1.8 times and 1.9 times.

In scenario 3, the mobile terminals stop at an intersection with probability 0.2, then they can receive reply from a node. Otherwise, they move to a neighboring area before they receive reply from a node with probability 0.8. Therefore, they send a *query* again after they move to a neighboring area because the system doesn't have a function to forward the messages. That is why the increasing rates are almost the same in Table 5.8 and Table 5.10. It shows that *WP* algorithm can decrease the communication overhead.

5.6 Conclusion

In this Chapter, in order to send a reply to a terminal after it moves to a neighboring area, we proposed *Wall Pass* (*WP*) algorithm. We consider a node as a wall player of wall pass in football. We evaluated the performance of the GDR system when the mobile terminals are moving.

The results show that GDR and CAN have the same mean and variance of the *path length* and the *relay length* when storing data, while the mean and the variance of the *relay length* and the *relay length* of GDR are smaller than those of Chord and Kademlia.

On the other hand, when sending query, GDR, Chord and Kademlia have the same mean and variance of the *path length*, while the mean and the variance of the *relay length* of GDR are smaller than those of CAN. Furthermore, GDR and CAN have the same mean and variance of the *relay length*, while the mean and the variance of the *path length* of GDR are smaller than those of Chord and Kademlia. However, the variance of the *request* forwarding times of GDR is larger than that of Chord and Kademlia.

In addtion, *WP* algorithm can decrease the communication overhead of the system.

Chapter 6

Conclusion

We have focused on structured distributed routing (SDR) and information lookup systems.

In Chapter 2, we explained the properties of some SDR systems based on their ID format, and routing table size. We also explained the difference of our proposed system to those systems.

In Chapter 3, we proposed a geographic location-based distributed routing (GDR) system. The GDR system provides an information lookup based on the geographic latitude and longitude coordinates. Each node is given the geographic coordinates as its ID, and manages an overlay routing table. The routing table consists of pointers to other nodes in the network in order to forward messages to the geographically nearest overlay node toward its final destination. In a system with N nodes, each node has a routing table of size $\log N$ and a search is possible in $O(\log N)$.

We evaluated the mean and the variance of the *path length* and the *relay length* of GDR, CAN, Chord and Kademlia, under the assumptions that the ID is in cartesian format (x, y), all nodes are active, and the source node and the destination node are chosen independently with equal probability.

We showed that GDR, Chord and Kademlia have the same mean and the same variance of the *path length*, while the mean and the variance of the *relay length* of GDR are smaller than those of Chord and Kademlia. Furthermore, while GDR and CAN have the same mean and the same variance of the *relay length*, the mean and the variance of the *path length* of GDR are smaller than those of CAN. We showed that the mean *relay length* of GDR is about half of that of Chord, and about $2/3$ of that of Kademlia, and the mean *path length* is about $(3/4) \log N / \sqrt{N}$ of that of CAN.

In Chapter 4, we evaluated the performance of the GDR system by simulation.

In fix search, the mean and the variance of the *relay length* are constant even in the case of Kademlia that uses XOR distance for routing. On the other hand, GDR and Kademlia have the same maximum value of the mean and the variance of the *relay length*, whereas Chord and Kademlia have the same minimum value of the mean and the variance of the *relay length*.

We also showed that the ID is generated by using SFC, the mean *relay length* of GDR is about half of that of Chord, and about $2/3$ of that of Kademlia.

In addition, we showed that the GDR system has a routing redundancy to increase robustness. When a node fails, its neighbor node behaves as an agent for the failing node. To know the agent node of the failing node, each node has an agent list which is the records of the agent nodes of the nodes of its routing table. Since the number of the agent nodes is 2, the size of the agent list is $2 \log N$. If an underlay network can be modeled as a grid, it is easy to assign a physical address for a node. However, if a node fails, it is difficult to modify or change its physical address. In the GDR system, the nodes can avoid a failed node by using its agent list on the overlay network.

In Chapter 5, we presented an application of the GDR system. In order to send a reply to a terminal after it moves to the neighboring area, we proposed *Wall Pass* (*WP*) algorithm. We consider a node as a wall player of wall pass in football. We evaluated the performance of the GDR system when the mobile terminals are moving.

The results showed that GDR and CAN have the same mean and variance of the *path length* and the *relay length* when storing data, while the mean and the variance of the *relay length* and the *relay length* of GDR are smaller than those of Chord and Kademlia.

On the other hand, when sending query, GDR, Chord and Kademlia have the same mean and variance of the *path length*, while the mean and the variance of the *relay length* of GDR are smaller than those of CAN. Furthermore, GDR and CAN have the same mean and variance of the *relay length*, while the mean and the variance of the *path length* of GDR are smaller than those of Chord and Kademlia. However, the variance of the *request* forwarding times of GDR is larger than that of Chord and Kademlia.

In addtion, *WP* algorithm can decrease the communication overhead of the system.

References

[1] http://www.mlit.go.jp/road/ITS/j-html/news/event/20120515/documents/10.pdf.

[2] R. Steinmetz and K. Wehrle (Eds.), "Peer-to-Peer Systems and Applications," Springer-Verlag Berlin Heidelberg, 2005.

[3] J. F. Buford, H. Yu, and E. K. Lua, "P2P Networking and Applications," Morgan Kaufmann Publishers in an imprint of Elsevir, 2009.

[4] H. Esaki, "PEER TO PEER TEXT BOOK," An Impress Group Company, 2008 (in Japanese).

[5] I. Stoica, R. Morris, D. Liben-Nowell, D. R. Karger, M. F. Kaashoek, F. Dabek, and H. Balakrishnan, "Chord: A Scalable Peer-to-peer Lookup Service for Internet Applications," IEEE/ACM Transactions on Networking Vol. 11, No. 1, pp. 17-32, February 2003.

[6] P. Maymounkov and D. mazieres, "Kademlia: A peer-to-peer information system based on the XOR metric," The 1st International Workshop on Peer-to-Peer Systems(IPTPS02), pp.53-65, March 2002.

[7] S. Ratnasamy, P. Francis, M. Handley, R. M. Karp, and S. Shenker, “A scalable content-addressable network,” Proc. SIGCOMM, pp.161-172, August 2001.

[8] A. Rowstron and P. Druschel, “Pastry: Scalable, decentralized object location and routing for large-scale peer-to-peer systems,” IFIP/ACM International Conference on Distributed Systems Platforms (Middleware), pp.329-350, 2001.

[9] B. Zhao, J. Kubiatowicz, and A. Joseph, “Tapestry: An Infrastructure for Fault-tolerant Wide-area Location and Routing,” Technical Report: CSD-01-1141, U. C. Berkeley, 2001.

[10] G. G. Finn, “Routing and addressing problems in Large Metropolitan-scale internetworks,” ISI/RR-87-180, Information Sciences Institute, March 1987.

[11] B. Karp and H. T. Kung, “GPSR: Greedy Perimeter Stateless Routing for Wireless Networks,” Proc. ACM/IEEE MobiCom 2000, pp.243-245, August 2000.

[12] J. Li, J. Jannotti, D. S. J. De Couto, D. R. Karger, R. Morris, “A Scalable Location Service for Geographic Ad Hoc Routing,” Proc. ACM/IEEE MobiCom 2000, pp.120-130, Augast 2000.

[13] S. Ratnasamy, B. Karp, S. Shenker, D. Estrin, R. Govindan, L. Yun and F. Yu, “Data-Centric Storage in Sensornets with GHT, A Geographic Hash Table,” ACM Mobile Networks and Applications(MONET), Vol.8, No.4, pp.427-442 , August 2003.

[14] S. Ramabhadran, S. Ratnasamy, J .M .Hellerstein, and S. Shenker, “Prefix hash Tree: An indexing data structure over distributed hash tables,” Proc of the 23rd ACM Symposium on Principles of Distributed Computing (PODC 2004), July 2004.

[15] W. Wu, Y. Chen, X. Zhang, X. Shi, L. Cong, B. Deng, and X. Li, “LDHT: Locality-aware Distributed Hash Tables,” Information Networking, 2008. ICOIN 2008, pp.1 - 5, January 2008.

[16] J. P. Ahullo, P. G. Lopez, M. S. Artigas, and A. F. G. Skarmeta, “Supporting geographical queries onto DHTs,” The 33rd IEEE Conference on Local Computer Networks (LCN 2008), pp.435-442, October 2008.

[17] Y. Kaneko, K. Harumoto, S. Fukumura, S. Shimojo, and S. Nishio, “A Location-Based Peer-to-Peer Network for Context-Aware Services in a Ubiquitous Environment,” Proc. of Int’l Symposium on Applications and the Internet (SAINT 2005) Workshops, pp. 208-211, Feb. 2005.

[18] PIAX, http://www.piax.org/

[19] S. Matsuura, K. Fujikawa, and H. Sunahara, “Mill: A Geographical Location Oriented Overlay Network Managing Data of Ubiquitous Sensors,” IEICE TRANSACTIONS on Communications,Volume E90-B No.10. , pp.2720-2728, October 2007.

[20] F. Araujo and L. Rodrigues, “GeoPeer: a location-aware peer-to-peer system,” Network Computing and Applications, 2004 (NCA 2004), pp.39-46, August 2004.

[21] N. Cowzer and A. Quigley , “GeoIGM: A Location-Aware IGM Platform,” WETICE ’09, 18th IEEE International Workshops on Enabling Technologies: Infrastructures for Collaborative Enterprises, pp.105-110, June 2009.

[22] G. S. Manku, M. Bawa, and P. Raghavan, “Symphony: distributed hashing in a small world,” Proc. of the 4th conference on USENIX Symposium on Internet Technologies and Systems (USITS’03), pp.127-140, March 2003.

[23] M. S. Artigas, P. G. Lopez, J. P. Ahullo, and A. F. G. Skarmeta, “Cyclone: a novel design schema for hierarchical DHTs,” Fifth IEEE International Conference on Peer-to-Peer Computing (P2P 2005), pp.49-56, August 2005.

[24] H. Sagan, “Space-Filling Curves,” Springer-Verlag, New York, 1994.

[25] J. A. Orenstein, “Spatial query processing in an object-oriented database system,” SIGMOD ’86 Proceedings of the 1986 ACM SIGMOD international conference on Management of data, pp.326-336, 1986.

[26] D. Karger, E. Lehman, T. Leighton, M. Levine, D. Lewin, and R. Panigrahy, “Consistent Hashing and Random Trees: Distributed Caching Protocols for Relieving Hot Spots on the World Wide Web,” Proceedings of the Twenty-ninth Annual ACM Symposium on Theory ofComputing, pp. 654-663, 1997.

[27] Paul E. Black, “Manhattan distance” in “Dictionary of Algorithms and Data Structures,” http://xlinux.nist.gov/dads//.

[28] http://www.soccer-universe.com/wall-pass.html.

[29] http://www.sportsdefinitions.com/soccer/Wall-pass.html.

[30] F. Bai, N. Sadagopan, and A. Helmy, "IMPORTANT: A framework to systematically analyze the Impact of Mobility on Performance of Routing protocols for Adhoc Networks," IEEE INFOCOM'03, 2003.

[31] "Software: Mobility Simulation and Analysis Tools," http://nile.cise.ufl.edu/important/software.htm.

Acknowledgements

First of all, I would like to express my thankfulness to Professor Hiroyoshi Morita, who made it possible for me to become a Ph.D. candidate in computer science. I would like to thank him very much for the straight way of dealing with various issues as well as for all the fruitful discussions and advices that significantly influenced this thesis.

I would also like to express my thankfulness to Dr. I Gusti Bagus Baskara Nugraha. He gave me a lot of help and valuable comments on the manuscripts.

I wish to thank to Professor Hideki Koike, Professor Takashi Suehiro, Associate Professor Hisashi Koga and Associate Professor Satoshi Ohzahata for taking their time effort to read, evaluate, and make useful comments on the thesis.

I acknowledge Japan Radio Corporation for giving me the opportunity to make academic research, and all the people there for their support.

I would like to thank all the members in the math-sys and the applied network laboratory for their kindness and help. Especially I would like to thank Dr. Akiko Manada and Dr. Todorka Alexandrova for giving me many valuable comments on the presentation in English. My special thanks also go to Associate Professor Hiroyuki Kasai and Associate Professor Takahiro Ota for the many helpful advices.

Lastly, I would like to give my special thanks to my friends, my parents, and my family for their moral support and considerable encouragement.

Author Biography

Kumiko Kobayashi was born in Ibaraki, Japan. She received the B.Eng. degree from the University of Electro-Communications, Tokyo, Japan, in 1992. She withdrew from the Doctoral Program with the Completion of Course Requirements, Graduate School of Information Systems, the University of Electro-Communications, in 2012. In 1992, she joined Japan Radio Co., Ltd., Tokyo, Japan. She had been engaged in the research and the development for the systems of GPS, MPEG-2, MPEG-4, sensor network, distributed network, and applied radio. She had been a sub-working group member of Multimedia Platform Committee, The Telecommunication Technology Committee (TTC) from 2001 to 2004. Since 2009, she has been a commission member of Information and Communications Technology sub-Council, Information and Communications Council, Ministry of Internal Affairs and Communications (MIC). She is a member of the Institute of Electronics, Information and Communication Engineers (IEICE). Her current research interests are in radio systems and distributed network systems.

List of Publications Related to the Thesis

Journal Paper (refereed)

[1] **K. Kobayashi**, I. G. B. Baskara Nugraha, and H. Morita, "A Geographic Location-Based Distributed Routing System," *IEICE Trans. on Communications*, vol. E96-B, No. 1, pp. 88–98, Jan. 2013 .
(The contents of Chapter 3 and Chapter 4)

International Conference Paper (refereed)

[1] **K. Kobayashi**, I. G. B. Baskara Nugraha, and H. Morita, "Geographic location-based distributed routing system," *Proc. of Computer Aided Modeling and Design of Communication Links and Networks (CAMAD), 2011 IEEE 16th International Workshop*, pp.51–55, June 2011.
(The contents of Chapter 3 and Chapter 4)

[2] **K. Kobayashi**, I. G. B. Baskara Nugraha, and H. Morita, "Wall Pass Algorithm in a Geographic Location-Based Distributed Routing System," *Proc.*

of Networking and Distributed Computing (ICNDC), 2013 Fourth International Conference, to appear, Mar. 2014.
(The contents of Chapter 5)

[3] **K. Kobayashi**, I. G. B. Baskara Nugraha, and H. Morita, “A Routing redundancy in a Geographic Location-Based Distributed Routing System,” *5th International Conference on Smart Communications in Network Technologies (SaCoNet) 2014*, submitted, Jun. 2014.
(The contents of Chapter 4)

Technical Reports

[1] **K. Kobayashi**, I. G. B. Baskara Nugraha, and H. Morita, “Proposal of Distributed Management System for Location-related Information,” *IEICE Technical Report* , vol. 109, no. 79, IN2009-20, pp. 43–48, June 2009 (in Japanese).
(The contents of Chapter 3 and Chapter 4)

[2] **K. Kobayashi**, I. G. B. Baskara Nugraha, and H. Morita, “Proposal of a P2P Routing Based on Neighboring Areas,” *IEICE Technical Report* , vol. 110, no. 224, IN2010-72, pp. 39–44, Oct. 2010 (in Japanese).
(The contents of Chapter 3 and Chapter 4)

List of Figures

List of Tables