卒業論文 2012 年度 (平成 24 年度)

T-Ring:センサデータの時間的特殊性に着目した多次元データ分散管理システム

指導教員

慶應義塾大学環境情報学部

徳田 英幸

村井 純

楠本 博之

中村 修

高汐 一紀

Rodney D. Van Meter III

植原 啓介

三次 仁

中澤 仁

武田 圭史

慶應義塾大学 環境情報学部

寺山淳基

romen@ht.sfc.keio.ac.jp

卒業論文要旨 2012 年度 (平成 24 年度)

T-Ring:センサデータの時間的特殊性に着目した多次元データ分散管理システム

近年，センサネットワークは環境モニタリングや，スマートグリッドなど，我々の生活を豊かにするインフラとして整備されつつある．また，センサノードの小型化やセンサを搭載したスマートフォンの普及により，センサ自体やセンサネットワークが，家庭，個人の規模で利用されつつあり，センサネットワークが我々を支える生活基盤と成り得る将来は遠くはない．

本研究は，センサネットワークがより一般的な生活基盤となり，他の組織や家庭などとデータを公開，共有し，センサの具体的な所在を気にすることなく，自由にデータを利用できる世の中を想定する．この環境下で，公開，共有されたセンサデータを公衆センサデータとする．

この公衆センサデータの管理を行うには，センサデータを緯度，経度，センサタイプを含んだ多次元のデータとして扱わなければならないが，従来の多次元データ管理の研究において，センサデータを対象とした研究は数が多くない．また，センサデータを対象にしていても，センサデータの時間的特殊性という性質に着目している研究は存在しない．

本研究は，このセンサデータの時間的特殊性に着目した多次元データ管理システムである T-Ring を提案する．T-Ring は時間を多次元データにおける 1 つの属性とせず，保存先変更の基準として扱うことで，データの保存，取得時の計算量を減らし，処理時間を減少させる．さらに，本研究ではこの T-Ring システムの性能を評価することで，有用性を証明する．

キーワード：

公衆センサデータ, 多次元データ管理，P2P，分散ネットワーク，構造化オーバーレイネットワーク

慶応義塾大学環境情報学部

寺山淳基

Abstract of Bachelor's Thesis Academic Year 2012

T-Ring:Sensor Data Oriented Decentralized Multidimensional Indexing Focusing on Particularity of Time

In recent years, sensor networks are being developed as an infrastructure to enrich our lives such as environmental monitoring and smart grids. In addition, by the spread of sensor equipped smartphones and downsizing of sensor nodes, sensor networks (or the sensor itself) are being used in the scale of family or personal. In the future, sensor networks will become the foundation of our life supporting.

This research assumes that sensor networks could become the foundation and that everyone could get data from anywhere without being concerned about which sensor network they belong to. In this environment, these published and shared sensor data are called "public sensor data" in this paper.

To manage this public sensor data, sensor data should be operated as Multidimensional Indexing which has latitude, longitude, sensor type and so on. However, in general works of Multidimensional Indexing, these researches were hardly intended on sensor data. No reserach focused on particularity of sensor time.

This research proposes T-Ring which is Multidimensional Indexing system for public sensor data. This is the first research focusing on particularity of sensor time. T-Ring doesn't use time attributtion in multidimensional data and use a criterion of deciding the place of storage. T-Ring reduces amount of calculations on storing and retrieving of data and processing speed. This paper evaluates this system and proves feasibility.

Keywords :

Public Sensor Data, Multidimensional Indexing, P2P, Decentralize Network, Structured Overlay Network

Junki Terayama

Faculty of Environment and Information Studies
Keio University

目次

図目次

表目次

第 1 章

序論

本章では，まずセンサデータの増加などの本研究の背景を述べ，次いで，センサデータの公衆一般化において発生する問題の解決という本研究の目的について言及し，最後に本論文の構成を示す.

1.1 本研究の背景

現在，センサネットワークは環境モニタリングなどの分野で大学や企業などの機関で広く利用されている．また，近年，センサノードの低価格化やスマートフォンやタブレット端末の普及に付随したセンサデータの量の増加により，今後，センサネットワークが先に述べた組織だけでなく，一般の企業や家庭に利用され，より一般的な生活基盤になることも十分に考え得る．

また，現在は，センサネットワークを超えたデータの授受を行うことが困難であるが，プライバシーポリシーなどの標準化が進み，あるセンサデータがどのセンサネットワークに所属しているかを気にすることなく利用可能になり，センサデータが公衆化すると考えられている．

以上に挙げたセンサネットワークの一般化，公衆化が実現した際に，その公衆化したセンサデータをどのように管理するべきかということが議論されている．cosm などで用いられているシステムはセンサデータを集中管理している．その一方で Synapse などでは分散管理を行なっている．

しかし，センサデータには時間的特殊性が存在する．一般的な文書や音楽などのコンテンツは，変更がなされない限り，時間によって中身が変わることがないので，そのコンテンツを取得する際に，同じオブジェクトを参照することになる．その一方でセンサデータは，データの値と時間のセットで 1 つの意味を成すものであるので，データの値が同じであっても，同じオブジェクトを参照してはならない．現行の分散管理では，この時間的特殊性が考慮されておらず，システムの動作環境次第では，パフォーマンスの低下が発生する．

1.2 本研究の目的

本研究では，最初に，一般公衆化したセンサネットワークのデータ管理において，データを多次元データとして扱うことの必要性を示す．次に，センサデータと音楽や文書のデータの時間的特殊性における違いについて言及する．そして，センサデータの時間的特殊性を考慮した多次元データ管理手法である T-Ring を提案する．最後に，評価を行い T-Ring の有用性を証明する．

1.3 本論文の構成

本論文では，2 章において，センサネットワークの一般公衆化とアプリケーションの分類から，そのデータ管理において必要な機能要件を示し，センサデータの時間的特殊性について述べる．3 章では，センサデータではない公衆コンテンツの管理と DHT を用いた広域公衆センサデータについての関連研究を紹介する．4 章，5 章では，センサデータの時間的特殊性に着目したデータ管理システムを示す．6 章で，提案手法に対する評価を行い，7 章で本研究のまとめを行う．

第 2 章

公衆センサデータ管理

本章では，最初にセンサネットワークの一般公衆化を説明し，公衆センサデータ管理における必須な機能要件である地理的探索について述べる．次に，公衆センサデータ管理のシステム設計の際の重要な概念であるデータ管理の時間的密度を解説し，最後にセンサデータの時間的特殊性とそれに起因する公衆センサデータ管理の問題点を述べる．

2.1 センサネットワークの一般公衆化

現在，センサネットワークは大学，企業，公的機関の研究や提供するサービスに広く使用されている．近年注目されているスマートシティに用いられるスマートグリッドでは，中枢機能としてセンサネットワークを利用した電力消費のモニタリングが行われている．これにより，電子機器に対する負荷の制御やユーザに対する注意喚起などを行なっている．また，農業分野においても育成環境のモニタリングに活用されている．広大な農地にセンサを設置し，それらが無線ネットワークを構築することにより，照度や土中の湿度などの監視を行っている．

近年では，センサネットワークはこのような従来型の利用だけではなく，各家庭や個人の規模でも利用されるようになった．このような利用状況の変遷に寄与した原因として，センサノードの小型化，低価格化やスマートフォンやタブレット端末（図 2.1）の普及（図 2.3，図 2.4）が考えられる．センサの低価格化により実現したセンサネットワークの例として，Andong らの研究 [1] がある．この研究では自転車利用時のカロリー計算を，心拍計や高度計などの，ユーザの体に装着し，動きを制御する可能性のある機器や，金銭的なコストの高い機器を一切使わず，スマートフォンに搭載された加速度，コンパス，GPS など数個のセンサノードのみでセンサネットワークを構築している．これらのセンサ情報と，公的機関などが提供している高度や気温などのデータを駆使し，Andong らのアルゴリズムで計算を行い，98% 以上の精度のカロリー計算を実現している．また，センサネットワークの個人利用化の例として，Kay らによる Lullaby[2] がある．Lullaby は，個人の寝室に数個のカメラ，マイクなどのセンサ群を設置し，睡眠環境をモニタリングすることで，ユーザに対してのフィードバックや，医師に対する有用な情報提供を可能にしている．

以上に挙げたように，センサネットワークは大学，企業，公的機関などの大規模利用だけでなく，各家庭や個人などの小規模利用もなされ，一般的な生活基盤になりつつある．

また，これらのセンサデータが所属するセンサネットワークを超えて公開され，共用利用されている．本研究では，これをセンサデータの公衆化と呼ぶ．cosm[3] 図 2.2 で行われているサービスでは，サービス利用者が各々で管理しているセンサネットワークのデータを公開，共有し，提供されている API から自由にそのデータを利用することが可能である．

センサネットワークの一般化が進み，センサデータの公衆化が広まることにより，多くの研究機関や個人に利益をもたらすと考えられている．

2.2 公衆センサデータ管理

本節では，まず，公衆センサデータの定義を行う．次いで，管理を行うに必要な機能要件についての議論を行う．最後に，公衆化したセンサデータを利用するアプリケーションの分類を行う．

図 2.1　スマートフォンやタブレット端末（参考：iphone5[4]，GALAXY Note2[5]）

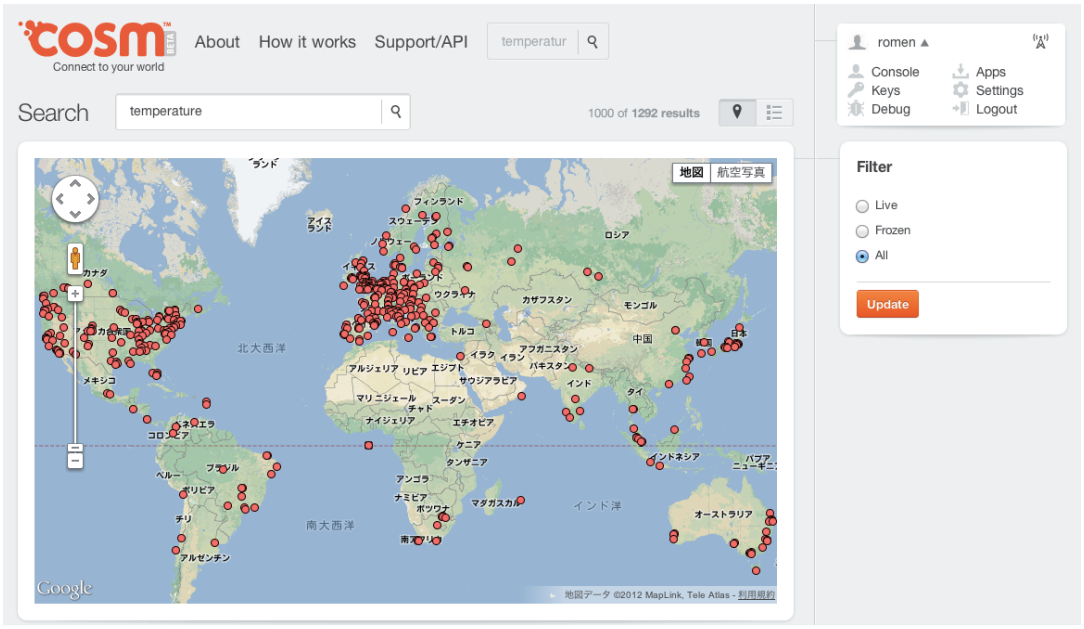


図 2.2　Cosm によるセンサデータの集中管理（参考：Cosm[3]）

2.2.1　公衆センサデータの定義

センサデータは [7][8][9] でも言及されているようにプライバシーの対象であり，公開すべきでないデータと公開してもよいデータに分けられる．2.1 でも述べたように，スマートフォンの普及がセンサデータの増大を引き起こした一因であると考えられている．しかし，スマートフォンは個人的なものである．Beresford ら [10] は，位置情報のプライバシー性について述べている．GPS の

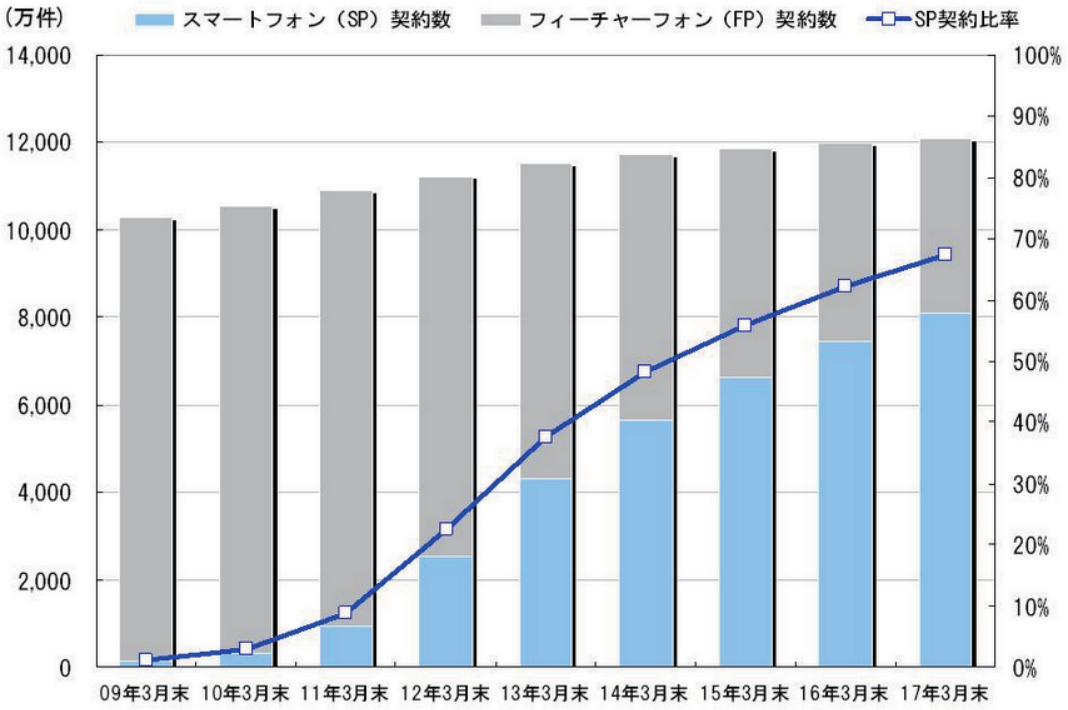


図 2.3　スマートフォン契約台数推移（参考：MM 総研 [6]）

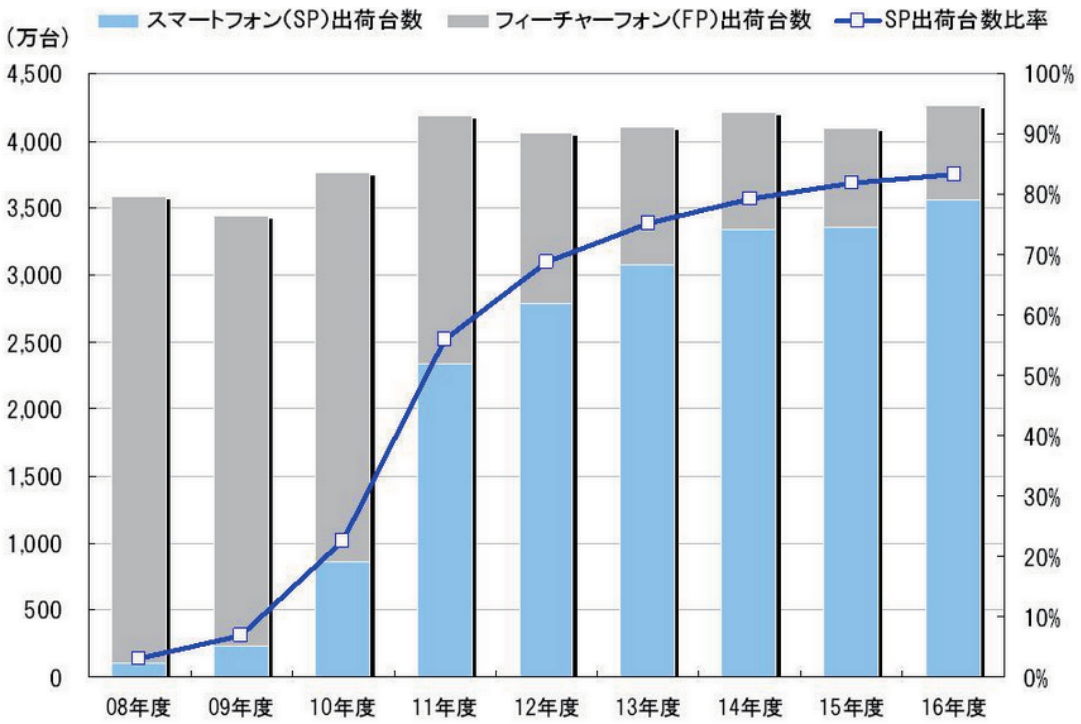


図 2.4　スマートフォン出荷台数推移（参考：MM 総研 [6]）

値も位置情報であり，無条件に公開するべきでない．この公開・非公開のポリシーについては本研究の対象外とする．本研究における公衆センサデータは，ユーザが公開を許可したデータであり，他人が無条件に利用することが可能なセンサデータとする．

2.2.2　公衆センサデータの地理的探索

センサデータの公開がなされ，公衆センサデータが偏在する環境において利用されるアプリケーションを考慮した際に，不可欠な機能要件として，センサデータの地理的探索がある．ある領域に存在する全てのセンサノードからデータを取得する場合，センサノードの緯度，経度，半径などの地理的な指定が可能でなければならない．地理的探索については図 2.5 が例である．

図 2.5　地理的探索

2.2.3　公衆センサデータ管理におけるデータ探索のシード

センサデータの地理的探索を行う場合，センサデータの取得に，対象のセンサノードの物理アドレスを指定する方法は適さない．物理アドレスの指定によるデータの取得を行う場合，自身の管理外のセンサネットワークからデータを取得する際に，事前に対象ノードの物理アドレスを取得する必要がある．地理的探索により，複数のセンサネットワークの複数のセンサノードからデータを取得を行うには，それらすべてのノードの物理アドレスを事前に保持しなければならず，現実的ではない．よって，公衆センサデータ管理においては，物理アドレスではなく，事前共有の必要のない情報からセンサノードの指定を行えなければならない．

2.3 システム設計におけるデータ管理の時間的密度の必要性

本節では，データ管理の時間的密度の定義を行った後に，公衆センサデータ管理との関連性に言及する．最後に，システム設計におけるデータ管理の時間的密度の必要性について述べる．

2.3.1 データ管理の時間的密度

データ管理の時間的密度とは，単位時間あたりの，保存ストレージを持つ任意のマシンに対するデータの保存や取得のクエリの集中度を表す．この値が高ければ密度が高く，マシンのパフォーマンスの低下に繋がる．また，本研究では以後，保存ストレージを持つ任意のマシンを保存ピアと表現することとする．

2.3.2 公衆センサデータ管理とデータ管理の時間的密度

公衆センサデータ管理では，データ管理の時間的密度はシステム設計においての指標である．公衆センサデータ管理においては，複数台の保存ピアが協調して保存場所を決定するため，特定の保存ピアに対してクエリが集中し，著しいパフォーマンスの低下が発生すると，ボトルネックとなり，システム全体のパフォーマンスの低下を招く．

その一方で，データをどの保存ピアに保存するのかの決定にも計算コストがかかる．特定のピアに対してクエリが集中しないように保存ピア決定の計算が行われれば，その分計算コストが嵩む．

2.3.3 公衆センサデータを利用したアプリケーションと取得されるデータの時間の関係性

公衆センサデータなどの，時間とデータのセットで意味を成すビッグデータを用いたアプリケーションと取得されるデータの時間には特徴がある．公衆センサデータを用いて，天候情報を提供するアプリケーションを考えた場合，ユーザは，過去のデータよりも，現在に近いデータの方が利用する頻度が高いということは自明である．このように，アプリケーションごとに取得されやすいデータの時間が決まっている．仮に，現在に近いデータが全て同じ保存ピアに保存される場合，その保存ピアに対して取得のクエリが集中するので，データ管理の時間的密度が高くなる．よって，公衆センサデータ管理のシステム設計を行う際には，データ量も勘案し，特定の保存ピアに，どの程度クエリが集中するのか，または，させるのか，さらには，設計するシステムが，アプリケーション内におけるどのような時間のデータを管理するシステムであるのかなど，データ管理の時間的密度を考慮し，システムを設計しなければならない．

2.4　公衆センサデータ管理における問題

本節では，公衆センサデータとしてセンサデータを管理する場合のデータの捉え方について述べ，次いで，センサデータの他のデータとの違いについて言及し，最後に，その違いから起因する問題点について述べる．

2.4.1　多次元データとしてのセンサデータ

2000 年前後から，多次元のインデックスの指定がされたクエリをいかに処理するかが議論されている．そして，2000 年代中盤からは，P2P 技術を利用した多次元データ管理が，この分野の話題の中心となっている．この多次元データ管理の研究動向ついては，3.1 において詳細に述べる．

多次元データとは，任意のデータが複数の属性を持っているということである．例として，音楽データを考える．従来的なデータの概念としてこの音楽データを考えると，この音楽データには一意の名前が付けられており，ユーザはこの名前を参照することで，データにアクセスすることが可能になる．しかし，データ量が多大になると，名前による検索のみでなく，他の属性による絞込み（探索）が必要になる．例としては，データを保存するファイルの拡張子，歌手の名前，言語など様々である．これが，データの属性となり，多次元データとして管理されることとなる．多次元データにおける検索の例を図 2.6 に示す．

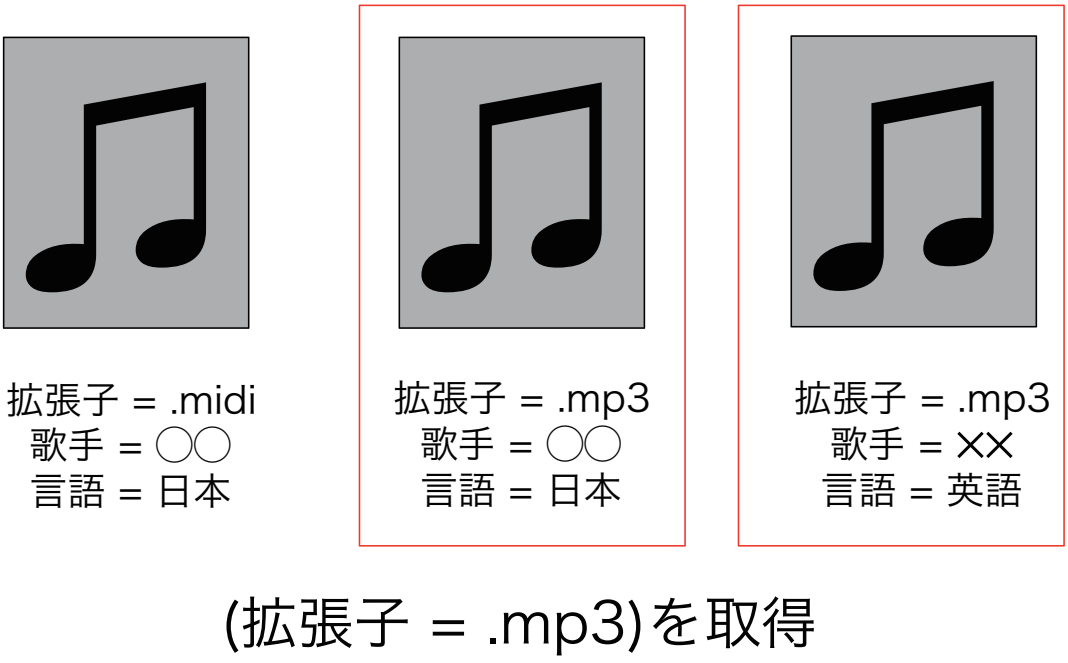


図 2.6　音楽を例とした多次元データ検索

公衆センサデータを管理するにあたり，センサデータは多次元として扱う必要がある．2.2.2 で述べた地理的探索などのように，センサデータはデータの値のみでなく，緯度，経度，センサタイプ，データの時間などのセットとして管理するものである．

2.4.2 センサデータの時間的特殊性

公衆センサデータを管理するには，さらに，センサデータの時間的特徴について考慮しなければならない．

従来の CDN（コンテンツデリバリーネットワーク）などで管理されているデータは，文書や音楽，動画などである．これらを本研究では，一般データと呼ぶ．一般データは変更が加えられない限り，内容が変化することはないので，ある任意の一般データを取得する際は，いつ誰が取得しようとも同一のオブジェクトを参照すればよい．

その一方で，センサデータは，時間とデータの値のセットである 1 つの意味を成すものである．異なる時間でセンサデータの値が同一であっても，それらをデータとして同一のものとして扱うことができない．よって，データの値が同じであっても，時間によって全て異なるオブジェクトとして扱わなければならない．この一般データとセンサデータの違いについては，図 2.7 において示す．

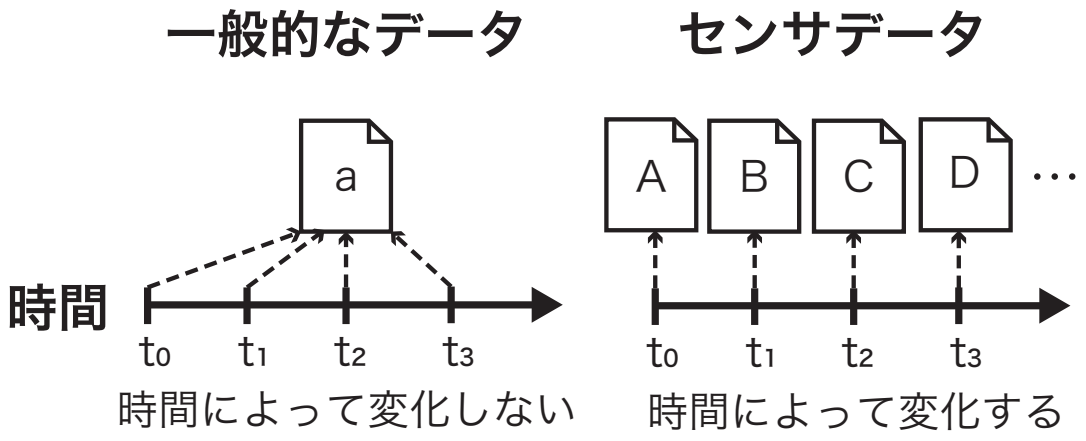


図 2.7　センサデータの時間的特殊性

さらに，センサデータはある程度，時間的に継続して監視を行うことで 1 つのコンテクストを成すことが多い．例えば，環境モニタリングアプリーケーションが存在し，何かイベントが発生した時にユーザに通知するシステムであったとする．このアプリーケーションで用いているセンサノードは，数十から数百 ms で環境における何かしらの情報をセンシングしている．このような環境において，異常を発見するのに，あるノードにおける，ある一時のセンシングされた値を参照しても，それが正しい値であるのかどうかを判断することはできない．時間的に継続して監視をし続け，その中である異常な値を発した時間にイベントの発生と判断することが可能になる．

よって，センサデータを管理する機構を考える場合には，一般のデータと違い，センサデータは時間によってデータ量が膨大になるという特徴と，センサデータを用いるアプリケーションは時間的に継続したデータを参照することが多いという特徴の 2 つを斟酌しなくてはならない．

2.4.3 センサデータの時間的特殊性に起因した広域センサデータ管理の現状

公衆広域センサデータは多次元として扱う必要があるが，多次元データ管理の手法を提案している研究の中にセンサデータを対象としたアルゴリズムはほとんど存在しない．また，センサデータを対象としたアルゴリズムであっても，センサデータの時間的特殊性のようなパラメータに特殊性のある多次元データを扱うことを前提としたアルゴリズムは存在しない．本研究では，このセンサデータの時間的特殊性に起因した広域センサデータ管理の問題を解決する広域センサデータ管理システムである T-Ring を提案する．

2.5 まとめ

本章では，まず，センサネットワークの一般公衆化について述べた．次いで，公衆センサデータがを扱ったシステム設計をする際に，データ管理の時間的密度という要素を考慮すべきであることを示した．そして，センサデータが時間的特殊性を持った多次元データであることを述べ，また，それに起因するセンサデータ管理における問題を提起した．

第 3 章

関連研究

本節では，最初に多次元データ管理 Multidimensional Indexing（MI）の分野における関連研究を挙げる．次に，Contents Delivery Network（CDN）におけるコンテンツの特徴を利用したレプリケーションを行った研究を挙げる．最後に，センサデータを多次元データとして扱った研究を挙げる．

3.1 多次元データ管理

1990 年代から，データベースの分野において，多次元のデータをどのように管理するべきか Multidimensional Indexing（MI）[11] が議論されている．そして，地理学，ロボット工学，環境保護学などの様々な分野でこの MI は応用されている．1990 年代の MI の中心はデータが集中管理されている環境を想定したアルゴリズムであり，VA_File[12]，Hilbert R-tree[13]，GHT[14] などがその代表である．2000 年代からは，P2P を用いた分散型の MI が主流となる．2000 年代前半に提案された Chord[15]，CAN[16]，SkipGraph[17] が代表である．2000 年代後半から現在にいたるまでにこれ以外の数多くのアルゴリズムが提案されているが，大半がこのいずれかのアルゴリズムの変形であって，大きなアルゴリズムの変化はない．

センサデータが公衆化し，広域に管理される場合，2.2 節で述べられているように，データの値以外のインデックスが付与される必要がある．よって，公衆広域センサデータを多次元データとして管理しなければならない．

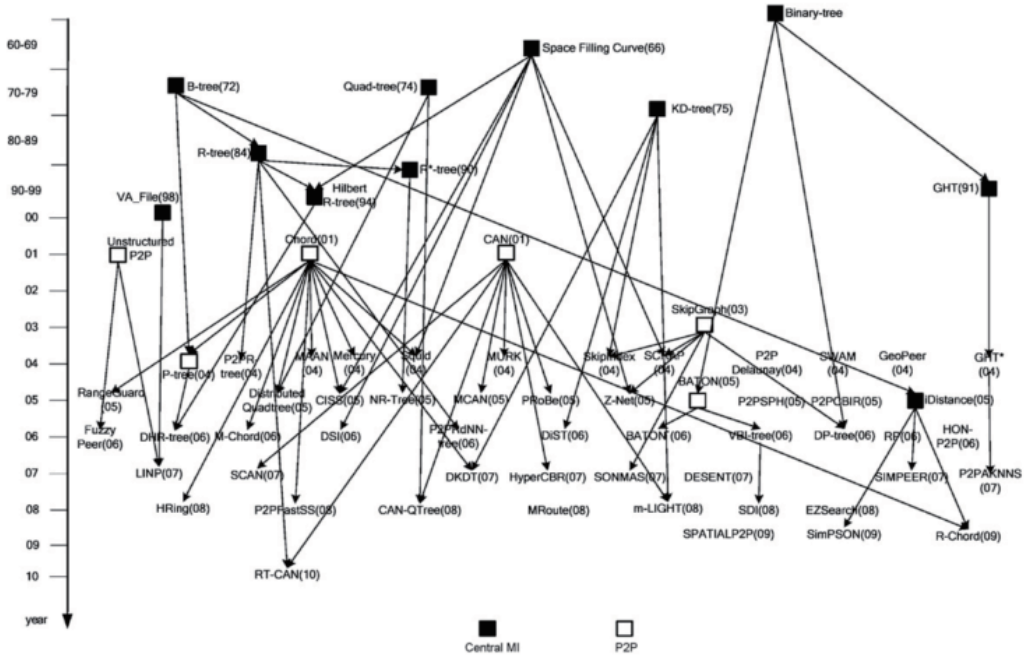


図 3.1　Multidimensional Indexing の歴史（参考：P2P-based multidimensional indexing methods. Journal of Systems and Software 2011[18]）

3.2 文書や音楽などのコンテンツの広域管理

この多次元データ管理は，Akamai[19] などに代表される，Contents Delivery Network（CDN）などで利用されている．CDN は世界中の数十，数百のデータセンターで P2P ネットワークを構

築し，ユーザーに音楽や動画などを提供している．コンテンツには，サイズ，言語などの多次元のインデックスが付与されている．CDN は，コンテンツの特徴を抽出し，レプリケーション先を決定し，効率的なコンテンツの配送を実現している．Cha ら [20] は，Flickr[21] のコンテンツがどのように伝播するのかの調査をしている．Scellato ら [22] は twitter[23] や facebook[24] などの SNS などに載せられたコンテンツとその発言などのいち情報から，コンテンツの地域性を割り出し，レプリケーションを効率的に行う手法を提案している．Akamai では，トラフィックの状況などを公開しており，時間帯や地域による傾向を捉えることができる．関連研究ではコンテンツの言語から地域性を抽出し，対象地域のデータセンターに予めコンテンツをレプリケーションすることにより，効率化を実現している．この他にも，コンテンツや利用状況の特徴を抽出するために [25][26][27][28] などのインターネット計測の分野の研究結果も利用されている．

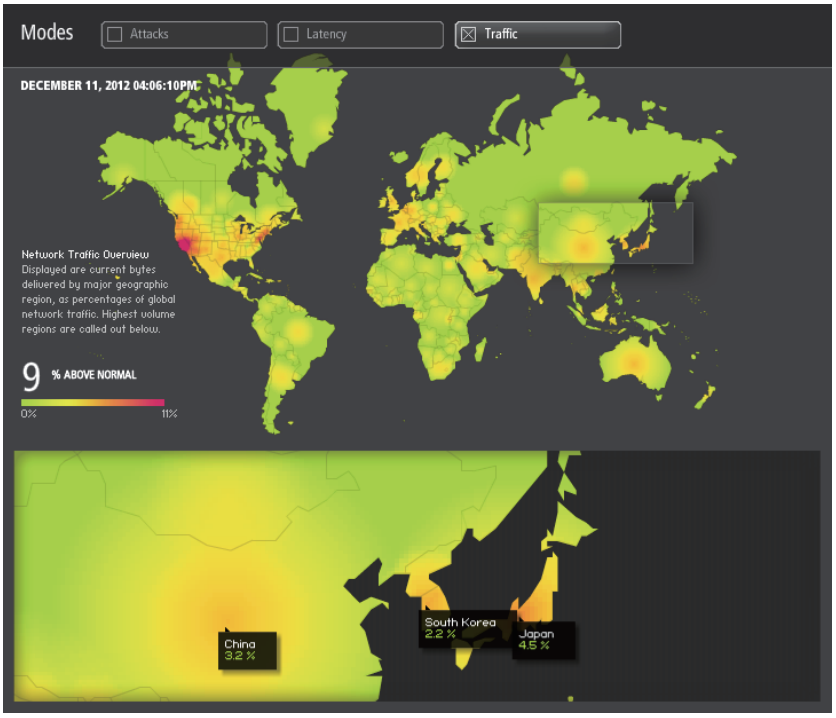


図 3.2　Akamai によるトラフィック情報の公開（参考：Akamai[19]）

センサデータは 2.3 節で取り上げた，音楽や動画とは異なる時間的特殊性に加え，言語などから由来する地域性も存在しないため，コンテンツの特徴からレプリケーションの場所を決定するという手法を取ることができない．

3.3　DHT を用いた公衆センサデータ管理

広域センサデータを多次元データとして，MI のアルゴリズムを用いた研究に Synapse[29] がある．この研究では，従来の，発せられたセンサデータを地理的に近い保存ストレージに保存すると

いう手法では，イベントの発生による突発的な人口過密が発生した際に，特定のセンサデータ保存ストレージに対する保存，取得のクエリが集中すると主張している．そこで，センサデータの緯度，経度，センサタイプ，データの時間を空間充填曲線 [30] 図 3.3 を用いて 1 次元化する．空間充填曲線は Lawder らが提案した，DHT アルゴリズムにおいて，多次元データを 1 次元化する手法で，これにより多次元データをハッシュ関数にかけることが可能になる．この手法は MI における基本的な手法であり，2009 年に Kantere らが提案した SPACIALP2P[31] という MI アルゴリズムでも用いられている．このハッシュ化した値を元に DHT ネットワークを構築することで，特定の保存ストレージに対するクエリの集中を防いでいる．

しかし，Synapse ではセンサデータの時間的特殊性が考慮されていないため，利用状況によっては，パフォーマンスが低下してしまう可能性がある．

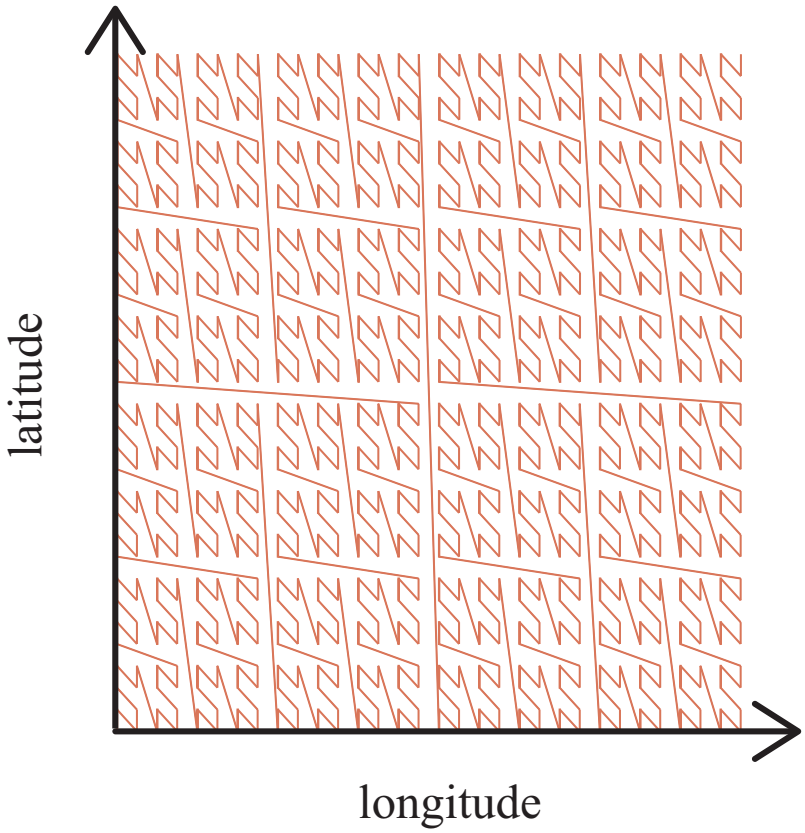


図 3.3　空間充填曲線:Z-order

3.4　まとめ

本章では，まず，公衆広域センサデータが多次元データであるということから，本研究の根幹を成す，Multidimensional Indexing（MI）という研究分野を紹介した．次に，Contents Delivery Network（CDN）という分散したデータセンターで多次元データを管理するネットワークという

観点から捉え，センサデータには CDN で扱われるような特殊性が存在しないことを述べた．最後に，公衆広域センサデータの分散管理手法を紹介し，センサデータの時間的特殊性が考慮されていないことに言及した．

第 4 章

T-Ring:時間を基準とした保存先変更を行う分散多次元センサデータ管理システム

本章では，最初に，本研究が提案する T-Ring という公衆センサデータ管理システムにおける，時間概念の扱いと，それに伴った保存アルゴリズムの概念説明を行う．次に，集中管理と Synapse とのデータ管理の時間的密度と計算コストの比較を行う．詳細な設計については次章で解説する．

4.1 T-Ring が対象とするデータの時間的密度

センサデータには，時間的特殊性が存在するので，Synapse のようなデータの時間的密度で保存ををするシステムの場合，任意の時間的範囲を持った取得のクエリを送った場合，保存ピア発見の計算量が多くなり，パフォーマンスが低下することがある．しかし，各データの分散は非常に強いので，保存ピアのデータの時間的密度は非常に低くなる．本研究では，保存ピア発見の計算量を少なくする手法を提案する．よって，T-Ring では Synapse より高いデータの時間的密度を持つシステムを構築する．

4.2 センサデータの時間的特殊性に着目した保存アルゴリズム

本節では，センサデータの時間的特殊性を考慮した時間に基づく保存アルゴリズムを提案する．

Synapse は広域センサデータ分散管理において，データ管理の時間的密度が非常に低いシステムである．しかし，センサデータの時間的特殊性を考えた場合，計算コストの高い保存アルゴリズムであると言える．T-Ring の提案する保存アルゴリズムは，取得のクエリの局所的集中のないアプリケーションにおいて有効なアルゴリズムである．

Synapse では，DHT における Z-order を用いた保存ピアの決定に，緯度，経度，センサタイプ，センサデータの時間の 4 次元を主な属性として用いて行なっている．つまり，データの時間を Z-order における 1 次元として扱っている．DHT による多次元データ管理では，1 次元化した値をハッシュ化し，その値に従って保存場所を決定する．よって，Synapse のように，センサデータの時間を Z-order における一つの属性として扱うと，センサデータ毎の情報空間上の位置が無関係になる．2.4.2 で述べた，センサデータは時間が経過するに従ってデータが増加し続けるという特徴を考慮すると，センサデータの保存を考えた場合，センサデータが断続的に送られている環境で，保存場所を決定する際に，センサの緯度や経度などに変化がないのにもかかわらず，保存するセンサデータ毎に保存する先を計算しなければならない．これは，センサデータが単位時間に送られる量が増えると計算によるオーバーヘッドが増加し，計算を行なっているマシンのパフォーマンスの低下を招く可能性がある．さらに， 2.4.2 で述べた，センサデータを利用したアプリケーションが時間的に連続したデータを扱う可能性が高いということを考慮すると，センサデータの検索を行う際に，実時間が近い複数のデータであるにもかかわらず，センサデータ毎に保存先の探索の計算を行わなければならなくなる．これについては図 4.1 の左図が Synapse の図解である．この図では，時間的に密接した A，B，C という 3 つのデータが存在している．この 3 つのデータを保存する際に，A は t0 という時間を Z-order の属性に加えて保存場所の計算を行う，B，C，においても同様に，それぞれの時間を Z-order の属性に加える．この図により，データの保存，取得どちらにおいても計算によるオーバヘッドが発生することが確認できる．

T-Ring では，任意のセンサノードから発せられるデータに対して，最初の保存先の保存ピアを，Synapse で必要な属性からセンサデータの時間を抜いた，緯度，経度，センサタイプで決定する．

これにより，センサデータが時間の経過によりデータが増加し続けたとしても，緯度，経度，センサタイプは変化せず常に一定であるので，Z-order の 1 次元化の計算は最初の 1 回のみで済み，それ以降はその値を使用し続ければよい．時間に関しては，あるセンサの一定の時間分のデータをある保存ピアに保存した後，一意に決められた別のピアに保存先の変更するという手法をとる．この一定時間という概念はセンサ毎にそのセンサを設置した管理者が決定することとする．T-Ring では，一定時間経過後，保存先はハッシュ値の情報空間上で隣の保存ピアに変更されることとする．例えば，あるユーザが環境にセンサを設置し，5 分ごとに保存場所を変更すると設定したとする．センサデータを漸次的に，ある保存ピアに保存し，5 分が経過する直前のセンサデータを保存する際に，現在保存を行なっている保存ピアに対し，隣の保存ピアのアドレスを同時に返すようにリクエストを送る．リクエストを受けた保存ピアはネットワークを構築する段階で，既に自分の隣の保存ピアのアドレスを知っているので，そのアドレスをリクエストの送信元の保存ピアにそのまま返す．これにより，5 分を超えた次のデータはそのアドレスに対して保存を行うことが可能になる．よって，取得をする際においても，Synapse では，15 分のデータを取得する際には，先頭のデータの保存ピアの探索から，15 分/センシングレート (回) の保存場所の探索を行わなければならなかったのに対し，T-Ring では探索を行わずに取得することが可能になる．図 4.1 の右図が T-Ring の図解である．A，B，C のそれぞれの時間の間隔を 5 分としたものが上の解説である．

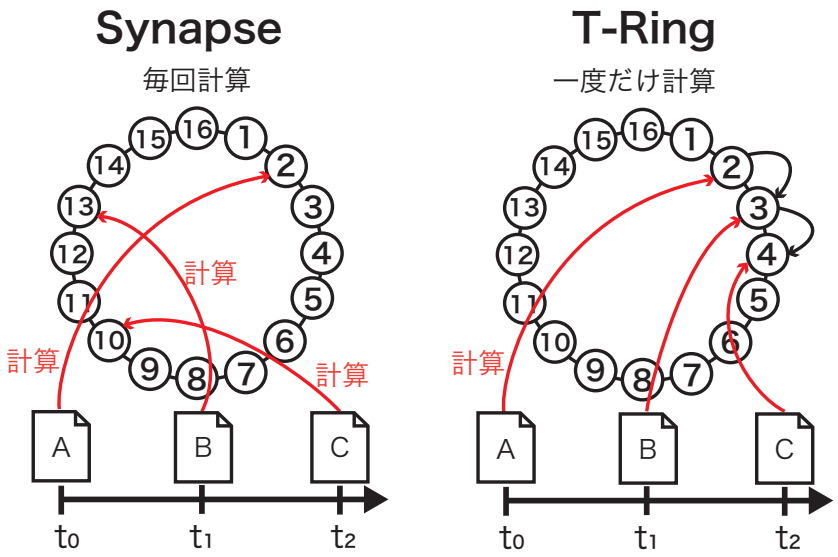


図 4.1　Synapse と T-Ring の保存アルゴリズムの比較

4.3　時間的密度の比較

4.2 で示されているように，分散管理を行わずに一極集中管理をしていた場合，全てのセンサデータが同一のマシンに保存されるため，時間的密度は非常に高い．また，Synapse では，保存ピ

アの決定がセンサデータごとに行われる．保存先はセンサデータごとに異なるため，時間的密度は低い．その一方で，T-Ring では，最初の保存先を決定した後に，保存先の変更が一定時間ごとに行われるため，任意の一定時間分のセンサデータは同一の保存ピアに保存される．よって時間的密度は，一極集中管理よりも低く，Synapse よりも高い．

4.4　保存先の決定における計算コストの比較

4.2 で示されているように，一極集中管理では，保存先は常にかわらないので，保存先の決定における計算コストは皆無である．Synapse では，保存先の決定がセンサデータごとになされる．よって，計算コストは非常に高いと言える．T-Ring は，最初の保存先の決定においてのみ，保存先の計算が行われる．一定時間が経過した後，次の保存先は一意に決定しているので，保存先変更の計算コストは低い．よって，T-Ring の計算コストは一極集中管理よりも高いが，Synapse よりも低い．

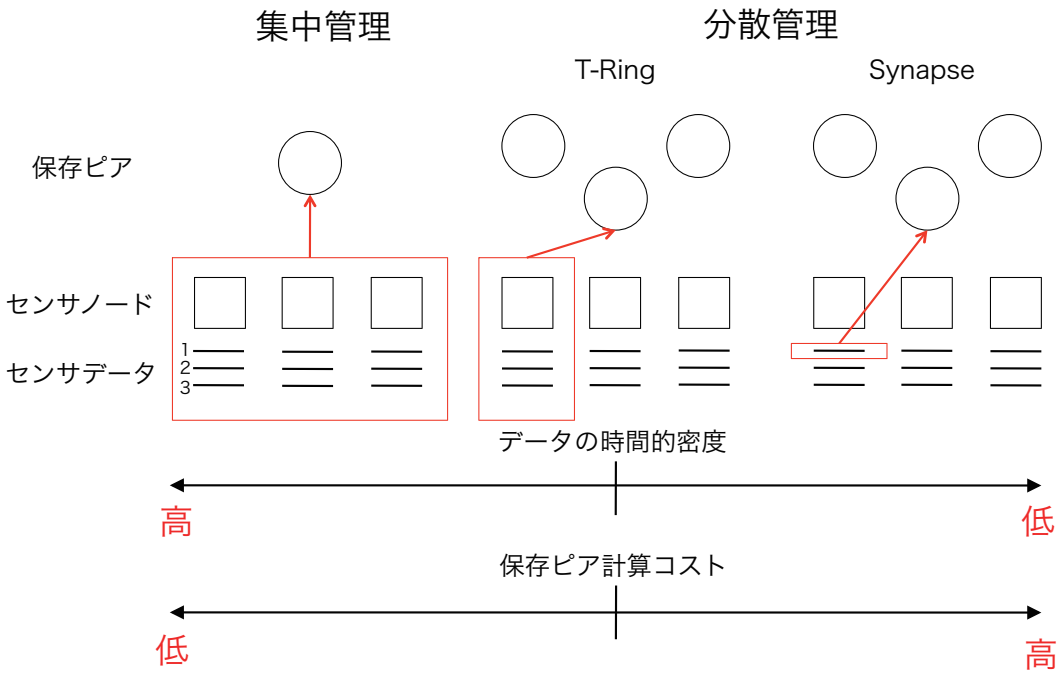


図 4.2　Synapse と T-Ring のデータの時間的密度と計算コストの比較

4.5　T-Ring の基本設計概念

本節では，設計の基本概念について述べる．

4.5.1 1D トーラス型構造化オーバレイネットワーク

T-Ring に参加する保存ピアは従来のセンサネットワークにおけるデータを集約格納するシンクノードのようなマシンを想定する．T-Ring を利用する団体や家庭，個人などは自身で管理するシンクノードを T-Ring の保存ピアとして提供する．これらの保存ピアが情報空間上で 1D トーラス型の繋がりを持ったオーバレイネットワークを構築する．

1D トーラス型のオーバレイネットワークの構築は DHT アルゴリズムである Chord に準拠している．保存ピアが新規に追加された際に，情報空間上での場所を決定するための Join 機能を有する．Join を行う保存ピアは，既にネットワークに参加している 1 台の保存ピアに対して Join リクエストを行う．Join リクエストを受けた保存ピアはリクエストを行った保存ピアの物理アドレスをハッシュ化し，その値から情報空間上の位置を決定する．情報空間は時計周りにハッシュ化後の値が大きくなるように構成される．情報空間上の位置が決定すると，Successor と Predecessor の張替えを行う．また，保存ピアが何らかの理由により通信不能となった場合にも，同様に Successor と Predecessor の張替えを行う．Successor List や Finger Table の作成，修正も行われる．

T-Ring に対して公開センサデータの保存，取得を行う際は，ユーザは自身の提供した保存ピアに対してリクエストを送る

4.5.2 センサ情報の 1 次元化とハッシュ化

T-Ring では保存や取得に関するリクエストを加工する必要性がある．本節では，センサ情報の 1 次元化とハッシュ化について説明する．

- センサ情報の 1 次元化
  T-Ring では，保存先の解決を行うにあたり，次に説明するハッシュ化を行うため，所与の多次元の属性を 1 次元化する処理が必要となる．この 1 次元化の処理に本研究では，空間充填曲線の Z-order を用いる．
- 1 次元値のハッシュ化
  一次元化した値のハッシュ化を行う．このハッシュ化された値の正の向きで最も近い保存ピアが担当の保存ピアとなる．T-Ring に参加する保存ピアも物理アドレスのハッシュ値に従って順序が決定される．T-Ring では全てこのハッシュ値に従って保存場所が決定される．

4.5.3 センサデータの時間的特殊性を考慮した時間に関する設計

前節にあるように，T-Ring では一定時間が経過すると，保存先が変更される．この保存先変更の時間はセンサのセンシングレートなどにより柔軟に設定されるべきである．T-Ring システムでは，センサを設置したユーザがセンサごとに保存先変更時間の設定を行う．この時間の概念を chunk と呼ぶ．chunk が 5 分であった場合，保存先の変更が 5 分ごとに行われる．前章において，

保存先の変更が一意に行われると述べたが，T-Ring においては，この「一意に決められた次の保存先」は，情報空間上において，物理アドレスをハッシュ化した値が正の向きに最も近い保存ピアとする．つまり，Chord アルゴリズムにおける Successor とする．

また，センサデータの取得を考えた際，時間の概念が chunk のみであると，データの保存開始時間から長期の時間が経過した任意の時間範囲のデータを取得する場合，最初の保存ピアから successor を辿らなければならないため，対象データの先頭データを保持する保存ピア探索のホップ数が大量になる．そこで，T-Ring では SP（Start Point）という時間の概念を用いている．これは，センサデータの保存開始点を変更するために存在する．SP が 60 分に設定された場合，最初の 60 分間は任意の保存ピアから保存される．60 分間が経過すると，保存開始ピアを再計算する．これにより，どのような時間範囲指定であっても，対象データの先頭データを保存する保存ピアを SP/chunk ホップ以内で発見することが可能になる．また，本研究では保存開始ピアが計算され，その計算結果により，最初に保存されるピアをマスターピアと呼び，n 回目の保存開始ピアの再計算により決められたマスターピアをマスターピア n とする．先の SP が 60 分の例の場合，あるセンサの 0 分 60 分のデータはマスターピア 0 が，60 分〜120 分のデータはマスターピア 2 が保存開始ピアとなる．

4.6 地理的探索クエリ

地理的探索を行うための取得クエリは，以下の項目を有する必要がある．

- Lat = 緯度
  領域の中心の緯度
- Long = 経度
  領域の中心の経度
- R = 半径
  領域の半径
- T = センサタイプ
  取得するデータのタイプ
- B = 開始時間
  時間範囲の開始日時
- E = 終了時間
  時間範囲の終了日時

クエリを言語化した場合，「(Lat, Long）を中心とする半径 R の領域から，B から E までの T のデータを取得する．」となる．図 4.3 はユーザの T-Ring を使ったセンサデータ取得のクエリ送信から実際に取得を行うまでの概念を表した図である．

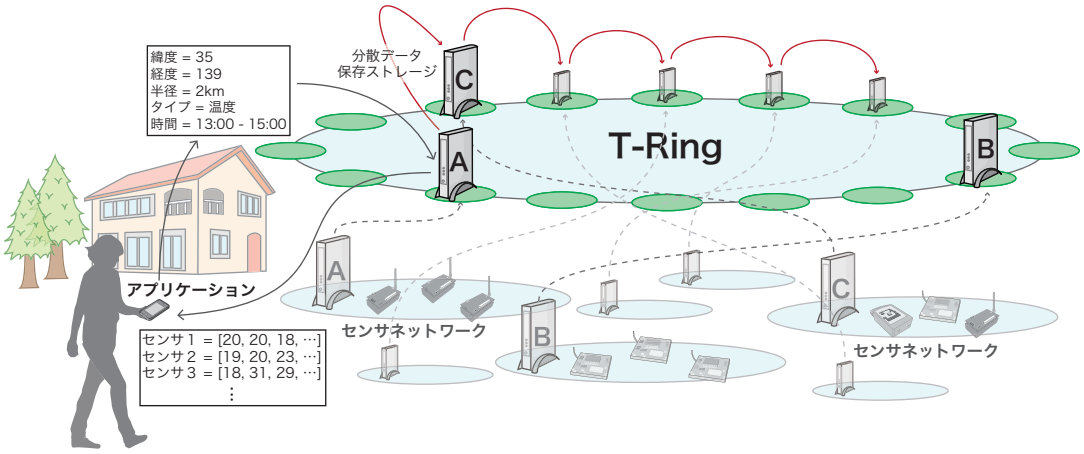


図 4.3　T-Ring 概念図

4.7　多次元の 1 次元化:Z-order

本節では，T-Ring において基本的な，多次元データを 1 次元値に変更する処理を行う Z-order について述べる．この処理を行うクラスは Zorder.java である．このクラスのインスタンスを生成するには，コンストラクタの引数として多次元データの各値（double の値）の List を必要とする．そして，getZorder() という関数を呼び出すと，まず，List で渡された値の要素数から，何次元のデータであるかを解析する．そして，各次元の値をバイナリの値（byte の配列）に変更する．次に，各配列の要素数を調べる．要素数の最も多い配列に要素数を合わせるために，他の配列に最も多い配列の要素数と同値になるまで 0 を埋める．最後に，各次元における配列の先頭を順に並べ，すべての次元の値が並べ終わったら，配列の次の要素について同様の作業を行う．これを要素数分繰り返す．これにより，最終的に完成した byte の配列を関数の結果として返す．これについての図解が図 4.4 である．Z-order のプログラムとその使用例は以下に掲載する．

Listing 4.1　Z-order

```
public class Zorder implements ZorderInterface {

  private ArrayList<Long> positions = new ArrayList<Long>();
  private ArrayList<ArrayList<Byte>> binaryPositions = new ArrayList<ArrayList<Byte>>();

  public Zorder(ArrayList<Long> positions) {
    this.positions = positions;
  }

  public byte[] getZorder() {
    convertPositionsToBinary();
    int length = adjustPositions();
    ArrayList<Byte> returnArray = new ArrayList<Byte>();
    for (int i = 0; i < length; i++) {
      for (int j = 0; j < this.positions.size(); j++) {
        returnArray.add((byte) (this.binaryPositions.get(j).get(i)));
      }
    }
```

```
    byte[] tmpByteArray = new byte[adjustPositions()
        * this.positions.size()];
    for (int i = 0; i < tmpByteArray.length; i++) {
      tmpByteArray[i] = returnArray.get(i);
    }
    return tmpByteArray;
  }

  private void convertPositionsToBinary() {
    for (int i = 0; i < this.positions.size(); i++) {
      this.binaryPositions.add(convertToBinary(this.positions.get(i)));
    }
  }

  private int adjustPositions() {
    int max = this.binaryPositions.get(0).size();
    for (int i = 0; i < this.binaryPositions.size(); i++) {
      if (max < this.binaryPositions.get(i).size()) {
        max = this.binaryPositions.get(i).size();
      }
    }
    int gap;
    for (int i = 0; i < this.binaryPositions.size(); i++) {
      gap = max - this.binaryPositions.get(i).size();
      for (int j = 0; j < gap; j++) {
        this.binaryPositions.get(i).add((byte) 0);
      }
    }
    return max;
  }

  private ArrayList<Byte> convertToBinary(long d) {
    ArrayList<Byte> byteArray = new ArrayList<Byte>();
    while (d >= 2) {
      byteArray.add((byte) (d % 2));
      d /= 2;
    }
    byteArray.add((byte) d);
    return byteArray;
  }
  public void debug() {
    getZorder();
    System.out.println("number of dimensions : " + this.positions.size());
    for (int i = 0; i < this.positions.size(); i++) {
      System.out.println("position of dimension " + i + " :"
          + this.positions.get(i));
    }
    for (int i = 0; i < this.positions.size(); i++) {
      System.out.println("converted position of dimension" + i + " : "
          + this.binaryPositions.get(i));
    }
    System.out.println("max binary length : " + adjustPositions());
    System.out.print("zorder : [");
    for (int i = 0; i < getZorder().length; i++) {
      System.out.print(getZorder()[i]+",");
    }
    System.out.println("]");
  }
}
```

Listing 4.2　Z-order 使用例

```
public class ZorderTest {
  public static void main(String[] argv) {
    ArrayList<Long> positions = new ArrayList<Long>();
    positions.add((long) 10);
    positions.add((long) 5);
    positions.add((long) 20);
    ZorderInterface zorder = new Zorder(positions);
    zorder.getZorder();
    zorder.debug();
```

```
  }
}
```

Listing 4.3　実行結果

```
position of dimension 0 :10
position of dimension 1 :5
position of dimension 2 :20
converted position of dimension0 : [0, 1, 0, 1, 0]
converted position of dimension1 : [1, 0, 1, 0, 0]
converted position of dimension2 : [0, 0, 1, 0, 1]
max binary length : 5
zorder : [0,1,0,1,0,0,0,1,1,1,0,0,0,0,1]
```

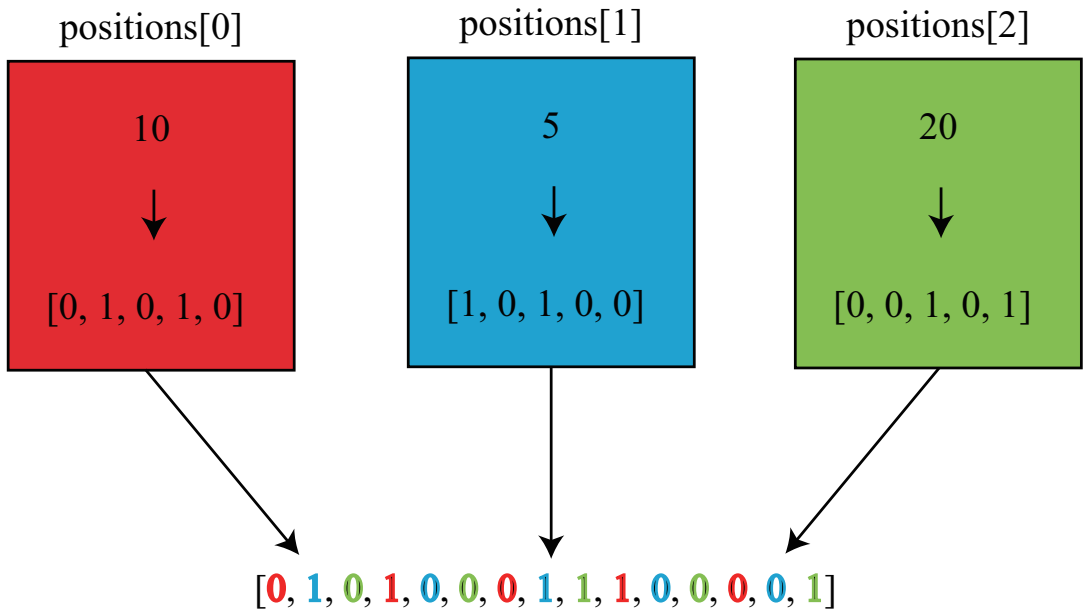


図 4.4　Z-order 図解

4.8　まとめ

本章では，まず，保存ピアの計算量を削減し，Synapse よりもデータ管理の時間的密度が高いアルゴリズムを提案することを述べた．次に，T-Ring において時間という属性を DHT における保存ピアの決定における一つ要素としてではなく，保存ピアの変更を行うための要素として扱うことを述べた．そして，時間的密度，計算コストの観点から集中管理，Synapse，T-Ring の 3 つの手法の比較を行った．また，センサデータの時間的特殊性に着目した多次元センサデータ分散管理システムである T-Ring の詳細な設計概念，設計実装について述べた．T-Ring は P2P のトポロジとして，代表的なアルゴリズムである Chord に準拠した 1D トーラスを用いている．保存ピアの参加や離脱なども同様に Chord に準拠している．また，多次元データを扱うにあたり，Z-order により 1 次元化に言及した．時間的特殊性については，chunk と SP という 2 つの時間概念を導入し，

保存ピア変更の基準とした．また，ユーザが T-Ring システムを用いて，データ取得のクエリを送る際の，形式を定義した．

第 5 章

T-Ring の設計実装

本章では，T-Ring における詳細な設計，実装について解説する．

5.1　システム構成

本節では，T-Ring システムの設計について述べる．T-Ring を構成する保存ピアはネットワークモジュール，センサ情報管理モジュール，データ保存モジュール，データ取得モジュール，保存ピア発見モジュール，センサ管理モジュール，アプリケーションモジュールの 7 つのモジュールによって構成される．図 5.1 がシステム構成図であり，各モジュール間でのデータの受け渡しが記述されている．

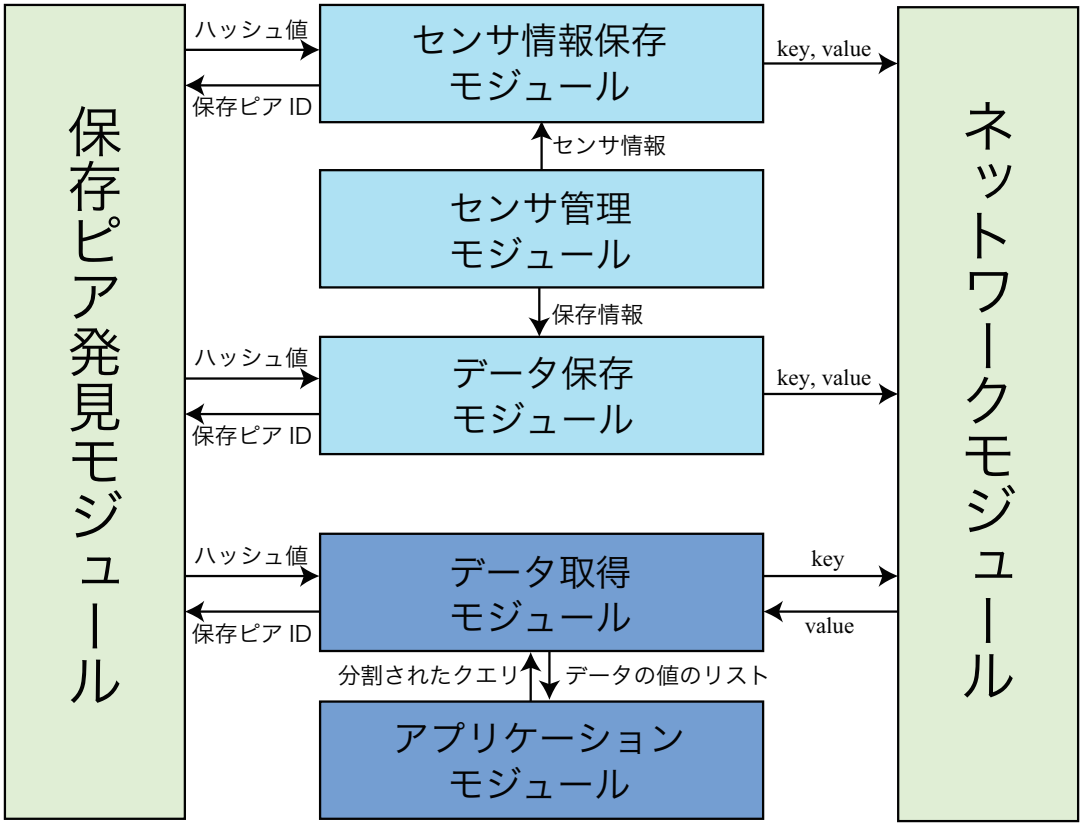


図 5.1　システム構成図

5.1.1　ネットワークモジュール

ネットワークモジュールは保存ピア同士で協調し，1D トーラス型のオーバーレイネットワークを構築及びデータに関する通信を行うモジュールである．本モジュールでは，他の保存ピアのデータ取得モジュールから，データ取得のクエリが送られた場合に，対象のデータをクエリ送信元の保存ピアに送信する．

本モジュールの実装はオープンソースである OpenChord の一部を参考にしている．保存ピアの

Join，Retrieve などの保存ピアによるネットワークの構築の部分において利用している．

5.1.2 センサ情報管理モジュール

本モジュールは，センサの緯度や経度など，センサデータの値以外の多次元データとしてのセンサ情報を適切な場所に保存することで，データの取得を可能にするモジュールである．

センサデータの取得を考えた際に，一般的な手法は，センサノードのアドレスを指定して取得するが，T-Ring が想定している環境では，センサデータをアドレス以外の他の要素から取得する．よって，取得を行う時には，取得されるセンサデータの情報が必要である．このために，T-Ring では，対象のセンサノードの緯度，経度，センサタイプ，マスターピア=0 で一次元化，ハッシュ化した値を担当する保存ピアに対して，このセンサ情報を登録する．登録はセンサデータの値の保存と同様に行う．センサデータの保存については，次節で詳細に述べる．保存におけるキーとして，緯度，経度，センサタイプ，マスターピア=0 の 4 次元値を 1 次元化，ハッシュ化した値を，バリューとして，そのセンサの chunk と SP の時間のセットを保存する．データの取得については，5.4 で詳細に述べるが，データの取得の際には，緯度，経度，センサタイプ，chunk，SP の時間を必要とする．緯度，経度，センサタイプ，マスターピア=0 の担当ピアに取得に必要なセンサ情報を登録することによって，chunk と SP の時間を知ることが可能になる．

本モジュールでは，最初に，センサに関する情報が登録されている SensorInfo.csv から，センサ ID をキーとして，センサデータの保存に必要である，緯度，経度，センサタイプ，chunk，SP の時間を取得する．取得された情報から，緯度，経度，センサタイプ，マスターピア=0 の 4 次元値を Z-order 関数によって処理し，返されたバイナリ値を保存ピア発見モジュールに渡す．保存ピア発見モジュールは，担当の保存ピアの保存ピア ID を返してくるので，そのピアに対して必要な情報を登録する．

5.1.3 データ保存モジュール

本モジュールはセンサデータを保存する保存するモジュールである．

T-Ring では，センサデータの時間に従って保存先を決めるため，センサ管理モジュールから，センサ ID，対象データの緯度，経度，センサタイプ，chunk，SP の時間，データの時間，センサの設定時間を取得する．本モジュールでは，フィールドとして，センサ ID をキーに，次に保存すべき保存ピアの ID をバリューとする Map を所持している．送られてきたセンサ ID と Map 内のセンサ ID を比較し，登録されていない場合は，データの時間，センサの設定時間，chunk から，まず，マスターピアの番号を決定する．次いで，そのマスターピアから Successor を辿る回数を計算する．そして，緯度，経度，センサタイプ，マスターピアの番号の 4 次元からハッシュ化をし，その値と Successor を辿る回数を保存ピア発見モジュールに送る．その結果として保存ピア ID を受け取る．データが保存されるセンサの ID と Map 内のセンサ ID が一致していた場合は，その Map のセンサ ID に対応する保存ピアの ID を取得する．次に，その ID の保存ピアに対して，キーを緯

度，経度，センサタイプの 1 次元化とハッシュ値，バリューをデータの値として保存する．

また，保存が行われると，次に保存するべき保存ピアの計算が行われる．ここからはセンサの ID とデータの時間だけを監視する．この計算を行うために，センサ ID をキーとして，最新のセンサデータが保存された保存ピアの ID，マスターピアの番号それぞれをバリューとして保存する 2 つのフィールドが存在する．次に送られてきたデータが前回のデータと比較して chunk が異なっていた場合，保存ピア発見モジュールに対して，最新の保存データが保存された保存ピア ID と Successor を辿る回数が送られる．これに対して，対象の保存ピアの ID が返される．この保存ピアの ID を次に保存すべき保存ピア ID として更新する．次に，chunk が同一であった場合，最新の保存データが保存された保存ピア ID を次に保存すべき保存ピア ID をとして更新する．マスターピアが異なっていた場合，センサ ID が登録されていない時と同様の処理を行い，保存ピア発見モジュールから返された保存ピア ID を次に保存すべき保存ピア ID として更新する．

これらの処理が繰り返えされることにより，データの保存が行われる．以上に述べた処理をフローチャートで示す．

5.1.4 データ取得モジュール

アプリケーションモジュールから送られるユーザからの取得のクエリは，範囲を持ったクエリであるので，本モジュールにより，クエリの緯度，経度，半径から分割を行う．これについては図 5.2 が例と図解である．そして，分割を行ったそれぞれの位置について，緯度，経度，センサタイプとクエリで指定された取得時間から，データ保存と同じ手順により，緯度，経度，センサタイプ，マスターピアの番号の 4 次元によるハッシュ値，Successor を辿る回数を保存ピア発見モジュールに送り，その結果として受け取った保存ピア ID に対して，緯度，経度，センサタイプで生成されるキーからバリューを受け取る．この受け取ったバリューのセットを，指定された方式に変換しアプリケーションモジュールに送り返す．指定された方式に関しては，アプリケーションモジュールの解説の際に説明する．

5.1.5 保存ピア発見モジュール

本モジュールは，センサ情報管理モジュール，データ保存モジュール，データ取得モジュールから受け取ったハッシュ値や Successor を辿る回数から，対象の保存ピアを発見し，そのアドレスを各モジュールに送り返す．

T-Ring では 2 種類の保存ピアの発見の方法が存在する，1 つ目は，Finger Table を用いた発見である．Finger Table の計算回数は $O(\log 2N)$ である．この発見はデータ保存モジュール，データ取得モジュールから送られた緯度，経度，センサタイプ，マスターピアの番号の 4 次元によるハッシュ値が前データと異なる場合に，マスターピアを探索するために行われる．また，Synapse における保存，取得では，センサデータ毎にこの計算がなされる．

2 つ目の方法は，データ保存モジュール，データ取得モジュールから送られた緯度，経度，セン

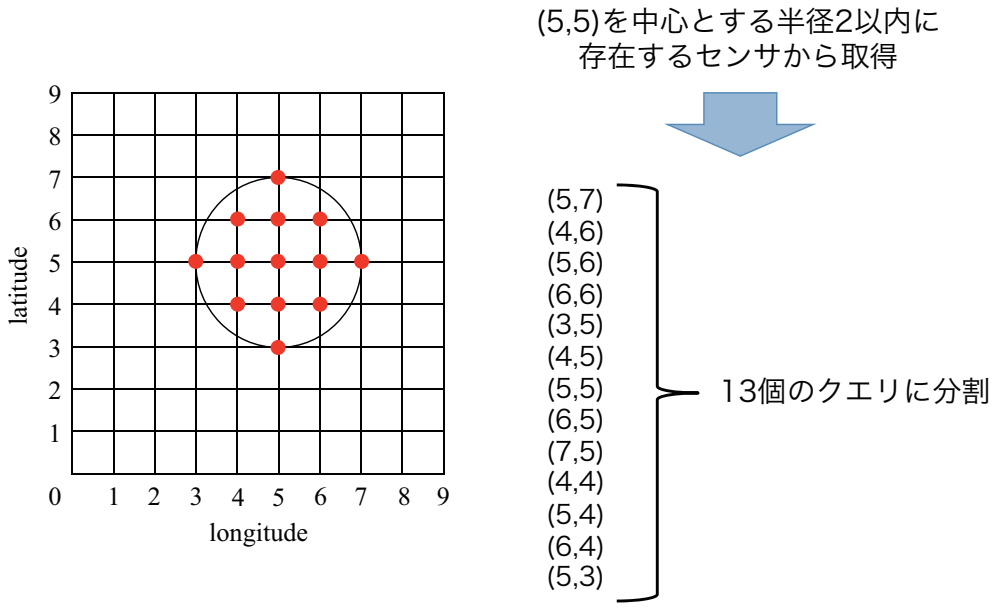


図 5.2　クエリの分割

サタイプ，マスターピアの番号の 4 次元によるハッシュ値が前データと同様であった場合に行われる．この場合，前データで取得したアドレスの Successor のアドレスを取得する．Finger Table による計算は前データの際に行ったため，計算回数は $O(1)$ である．

これらの方法により発見されたアドレスをデータ保存モジュール，データ取得モジュールに送り返す．

Listing 5.1　データ保存時の保存ピア発見

```
public class NextStorePeer {
  private TRingImpl tring;
  private HashMap<String, Node> nextNode;
  private HashMap<String, Long> restTime;
  private HashMap<String, Long> prevSP;
  private HashMap<String, Long> prevTime;
  ArrayList<Long> positions;

  public NextStorePeer(JoinData joinData) {
    this.positions = new ArrayList<Long>();
    this.nextNode = new HashMap<String, Node>();
    this.restTime = new HashMap<String, Long>();
    this.prevSP = new HashMap<String, Long>();
    this.prevTime = new HashMap<String, Long>();
    this.tring = joinData.tring;
  }

  public synchronized Node getNextStorePeer(Node node, DataForStore data,
      Key key) throws CommunicationException {
    setNextStorePeer(node, data, key);
    return nextNode.get(Arrays.toString(key.getBytes()));
  }

  private synchronized void setNextStorePeer(Node currentPeer,
      DataForStore data, Key key) throws CommunicationException {
    long SP = (data.time - data.configTime) / data.startPoint;

    if (nextNode.get(Arrays.toString(key.getBytes())) == null) {
```

```
      this.nextNode
          .put(Arrays.toString(key.getBytes()), currentPeer
              .findSuccessor(currentPeer.getNodeAddingOneID()));
      this.prevSP.put(Arrays.toString(key.getBytes()), SP);
      this.restTime.put(Arrays.toString(key.getBytes()), data.startPoint
          - data.chunk);
    } else {
      if (this.prevSP.get(Arrays.toString(key.getBytes())) != SP) {
        this.positions.add(data.latitude);
        positions.add(data.longitude);
        positions.add(data.type);
        this.positions.add(SP);
        ZorderInterface zorder = new Zorder(positions);
        byte[] SPKey = zorder.getZorder();
        Key masterPeerKey = new ByteArrayKey(SPKey);
        nextNode.put(Arrays.toString(key.getBytes()),
            tring.getResponsiblePeer(masterPeerKey));
        this.prevSP.put(Arrays.toString(key.getBytes()), SP);
      } else if (restTime.get(Arrays.toString(key.getBytes()))
          - data.chunk > 0) {
        this.nextNode.put(Arrays.toString(key.getBytes()), currentPeer
            .findSuccessor(currentPeer.getNodeAddingOneID()));
      }
      this.restTime.put(Arrays.toString(key.getBytes()), data.startPoint
          - data.chunk);
    }
  }
}
```

Listing 5.2　データ取得時の保存ピア発見

```
public class NextRetrievePeer {
  private TRingImpl tring;
  private HashMap<String, Node> nextNode;
  private HashMap<String, Long> restTime;
  private HashMap<String, Long> prevSP;
  ArrayList<Long> retrievedData;
  ArrayList<Long> positions;

  public NextRetrievePeer(TRingImpl tring) {
    this.positions = new ArrayList<Long>();
    this.nextNode = new HashMap<String, Node>();
    this.restTime = new HashMap<String, Long>();
    this.prevSP = new HashMap<String, Long>();
    this.retrievedData = new ArrayList<Long>();
    this.tring = tring;
  }

  public synchronized ArrayList<Long> getNextRetrievePeer(
    DataForRetrieve data, long latitude, long longitude,
    HashMap<String, Long> sensorInfo, Key key)
    setNextRetrievePeer(data, latitude, longitude, sensorInfo, key);
    return this.retrievedData;
  }

  private synchronized void setNextRetrievePeer(DataForRetrieve data,
      long latitude, long longitude, HashMap<String, Long> sensorInfo,
      Key key) throws CommunicationException {
    String tag = Arrays.toString(key.getBytes());
    long currentTime = data.timeFrom;
    long SPTime = sensorInfo.get("SP");
    long chunk = sensorInfo.get("chunk");
    long configTime = sensorInfo.get("configTime");
    long SP = (currentTime - configTime) / SPTime;
    long pastTime = currentTime - configTime - (SP * SPTime);
    long leftChangeTime = SPTime - pastTime;
    long leftTime = data.timeTo - currentTime;

    ArrayList<Long> positions = new ArrayList<Long>();
    positions.add(latitude);
```

```
    positions.add(longitude);
    positions.add(data.type);
    positions.add(SP);
    ZorderInterface zorder = new Zorder(positions);
    byte[] SPKey = zorder.getZorder();
    Key masterPeerKey = new ByteArrayKey(SPKey);

    nextNode.put(tag, tring.getResponsiblePeer(masterPeerKey));

    Node currentNode = nextNode.get(tag);

    while (leftTime > 0) {
      leftTime -= chunk;
      leftChangeTime -= chunk;

      if (leftChangeTime > 0) {
        this.retrievedData.addAll(tring.retrieveData(key,
            this.nextNode.get(tag)));
        this.nextNode.put(
            tag,
            this.nextNode.get(tag).findSuccessor(
        this.nextNode.get(tag).getNodeAddingOneID()));

      } else {
        leftChangeTime += SPTime;
        positions.set(3, positions.get(3) + 1);
        zorder = new Zorder(positions);
        SPKey = zorder.getZorder();
        masterPeerKey = newByteArrayKey(SPKey);
        this.nextNode.put(tag,this.nextNode.get(tag).findSuccessor(this.nextNode.get(tag).
            getNodeAddingOneID()));
        this.retrievedData.addAll(tring.retrieveData(key,this.nextNode.get(tag)));

      }
    }
  }
}
```

5.1.6 センサ管理モジュール

本モジュールはセンサネットワーク内のセンサ毎の緯度，経度，センサタイプ，などを管理を行うモジュールである．センサが環境に設置された際に，そのセンサの情報を所属する保存ピアへ登録する．また，センサの移動により，緯度，経度の変更が発生した際に，その情報を更新する．

センサが環境に設置を行う際，以下のようなセンサ情報登録クラスのインスタンスを用いて SensorInfo.csv に登録を行う．SensorInfo.csv に登録されるエントリは，各センサごとに，sensorId，latitude，longitude，sensorType，chunkTime，startPointTime である．センサの設置場所が変更された際は，sensorId から対象のエントリを発見し，変更を行う．この SensorInfo.csv の情報は，センサ情報保存モジュール，データ保存モジュールで利用される．以下がセンサ情報の登録，変更のメインクラスである．

5.1.7 アプリケーションモジュール

本モジュールはユーザアプリケーションに対してのデータ取得のインターフェースを提供し，ユーザアプリケーションは 5.1.1 で述べたクエリを本モジュールに送信する．本モジュールとは TCP で通信を行い．7777 番ポートを利用する．ユーザアプリケーションとのセッションが確立

し，クエリの受信が完了すると，データ取得モジュールにこのクエリをフォワードし，その結果をユーザアプリケーションが指定した方式で送り返す．この方式の指定には 4 つの方法があり，指定領域内の全データのリスト，最大値，最小値，平均値である．

5.2 まとめ

本章では，まず，ソフトウェアの構成について述べた，そして，個々のモジュールの細かな設計と実装について述べた．特に重要なアルゴリズムについては，各小節の末尾に実際のプログラムを掲載した．

第 6 章

評価

本章では，T-Ring システムの評価を行う．

6.1 評価方針

本研究では，センサデータを多次元データとして扱う際に，時間を他の属性と同様に扱っている Synapse と，センサデータの時間的特殊性を考慮した T-Ring の取得時間の比較を行う．Synapse の手法では，保存，取得の際に，データごとに Finger Table による保存ピアの探索を行わなければならない．よって，本評価では，ネットワークに参加するピア数を固定し，データの保存，取得に共通する，保存ピア探索の計算コストを計測する．次に，保存時，取得時に分けて探索に要する時間の計測を行う．

6.2 評価環境

本研究は，評価環境として，慶応義塾大学湘南藤沢キャンパス内の特別教室内に設置されている 124 台の iMac の内，任意の複数マシンを用いて行う．詳細なスペックなどについては以下の表 6.1 で示す．

表 6.1　実験環境

ホスト名	zmac000-159
本体	iMac 21.5 インチ
CPU	3.06 GHz Intel Core 2 Duo
メモリ	4 GB (2 GB X 2)
ハードディスク	499.76 GB
OS	Mac OS X 10.6.5

6.2.1 実験環境内におけるネットワークレイテンシ

本研究では，データの保存や探索に要する時間を計測するため，各保存ピア間の通信におけるレイテンシが重要になる．そこで，本実験環境におけるマシン間のレイテンシが実験にどの程度の影響を与えるのかを調査するため，事前実験として，各リージョン間の RTT を計測した．各リージョンがどこに存在するかは，図 6.1 において示す．それぞれ ICMP における Echo Message を送信し，リプライが返されるまでの時間の計測を 100 回行い，その平均値，最小値，最大値，標準偏差を表 6.2，6.3，6.4，6.5 で示す．本実験において，実験環境下では RTT が 0.400ms 以下であるという知見が得られた．本システムが用いられる実環境における RTT は 0.400ms より十分に大きいため，本評価では，実験環境内におけるネットワークレイテンシを無視する．

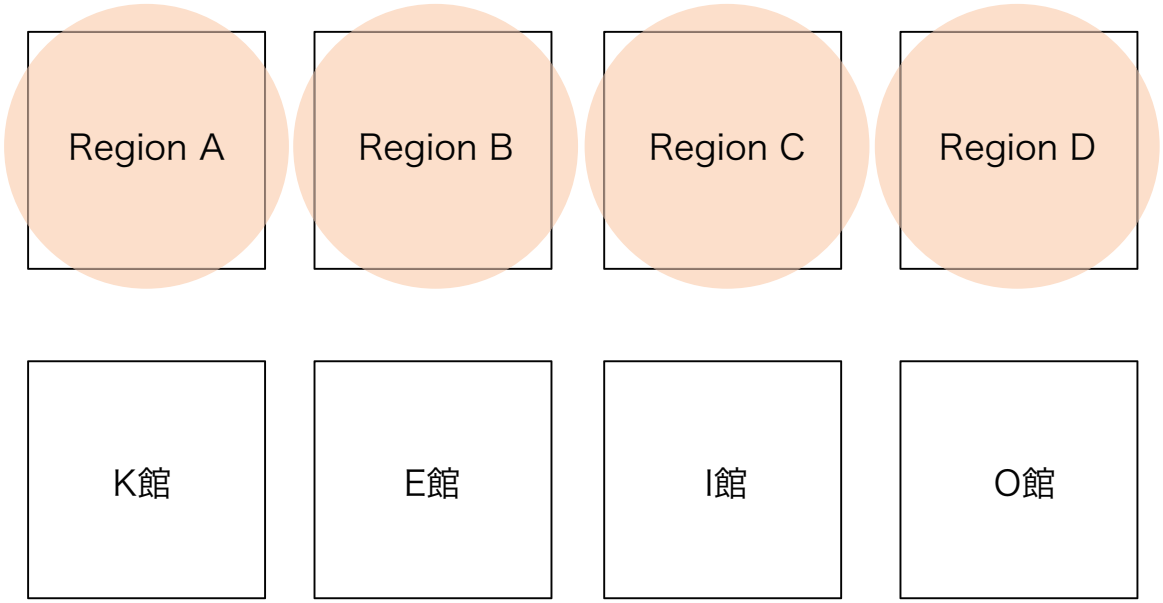


図 6.1　リージョンの関係性

表 6.2　リージョン A からの RTT

	平均値	最小値	最大値	標準偏差
リージョン A	0.234 ms	0.193 ms	0.271 ms	0.022
リージョン B	0.324 ms	0.287 ms	0.349 ms	0.015
リージョン C	0.321 ms	0.285 ms	0.347 ms	0.016
リージョン D	0.315 ms	0.277 ms	0.339 ms	0.016

6.3　保存ピア探索の計算コスト

データの保存，取得時に，Synapse では対象のデータの時間毎に保存ピアの探索を行う一方で，T-Ring は対象のセンサーの chunk，SP の値により，探索の回数が変化する．そこで，1000 個の連続したデータを保存，取得することを想定し，Synapse，T-Ring の両手法において，1 データの平均のピア間ホップ数を計測する．また T-Ring においては，各データの時間の間隔を 5，SP を 100，chunk を 10，30，50，70，90 の 5 つとする．図 6.2 はその結果である．

6.4　データ保存時の評価

データの保存時に Synapse では，データ毎に保存場所の計算を行う．その一方で，T-Ring では，ある一定の時間が経過するまでは，保存場所の変更が発生しない．そこで，ある保存ピアに

表 6.3　リージョン B からの RTT

	平均値	最小値	最大値	標準偏差
リージョン A	0.327 ms	0.288 ms	0.347 ms	0.015
リージョン B	0.238 ms	0.203 ms	0.270 ms	0.019
リージョン C	0.319 ms	0.274 ms	0.349 ms	0.018
リージョン D	0.317 ms	0.282 ms	0.347 ms	0.015

表 6.4　リージョン C からの RTT

	平均値	最小値	最大値	標準偏差
リージョン A	0.325 ms	0.277 ms	0.359 ms	0.016
リージョン B	0.312 ms	0.279 ms	0.342 ms	0.015
リージョン C	0.252 ms	0.192 ms	0.379 ms	0.022
リージョン D	0.319 ms	0.283 ms	0.343 ms	0.015

表 6.5　リージョン D からの RTT

	平均値	最小値	最大値	標準偏差
リージョン A	0.296 ms	0.240 ms	0.727 ms	0.047
リージョン B	0.295 ms	0.259 ms	0.374 ms	0.014
リージョン C	0.252 ms	0.192 ms	0.379 ms	0.022
リージョン D	0.291 ms	0.239 ms	0.335 ms	0.013

100 個のデータを保存するクエリを送り，それらの保存が完了するまでの時間を計測する．本評価では，保存されるデータの設定を chunk=30，SP=100，各データの時間の間隔を 5 として実験を行った．また，ネットワークの環境については，ネットワークに参加している保存ピアの数（ネットワークサイズ= NS）を 10，50，100 の 3 種類，各ピア間の RTT を 5，10，50，1000 の 4 種類とした．図 6.3，6.4，6.5，6.6 は RTT ごとにまとめた実験結果である．

6.5　データ取得時の評価

保存時と同様に，データの取得時に Synapse では，取得するセンサデータごとに保存場所の探索を行い，T-Ring ではある一定の時間が経過するまで，取得先の変更は発生しない．そこで，ある特定の領域に対して，500 単位時間のデータの取得を行うクエリを送り，データが全て返されるまでの時間を計測する．本評価では，データはすべて 5 単位時間ごとに保存されている環境を想定

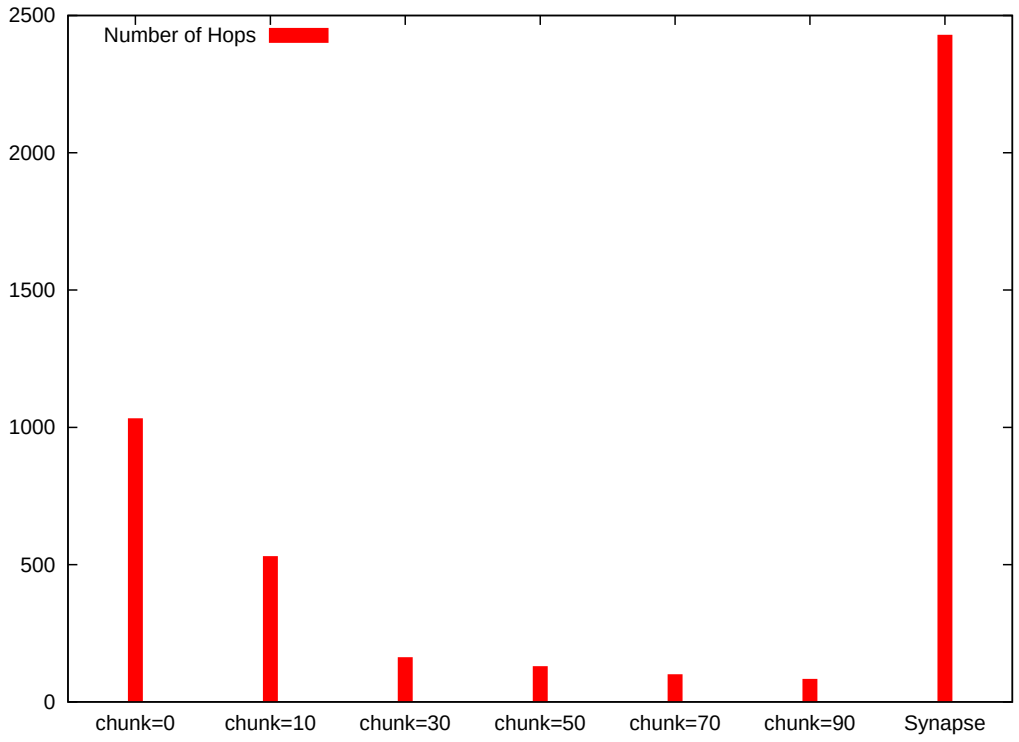


図 6.2　ホップ数

する．よって，500 単位時間分のデータは，100 個のデータを取得することとなる．クエリの地理的範囲は半径 10 の広さを持ったものとする．また，ネットワークに関しては，保存時と同様に，ネットワークに参加している保存ピアの数（ネットワークサイズ＝ NS）を 10，50，100 の 3 種類，各ピア間の RTT を 5，10，50，1000 の 4 種類とした．図 6.7，6.8，6.9，6.10 は RTT ごとにまとめた実験結果である．

6.6　考察

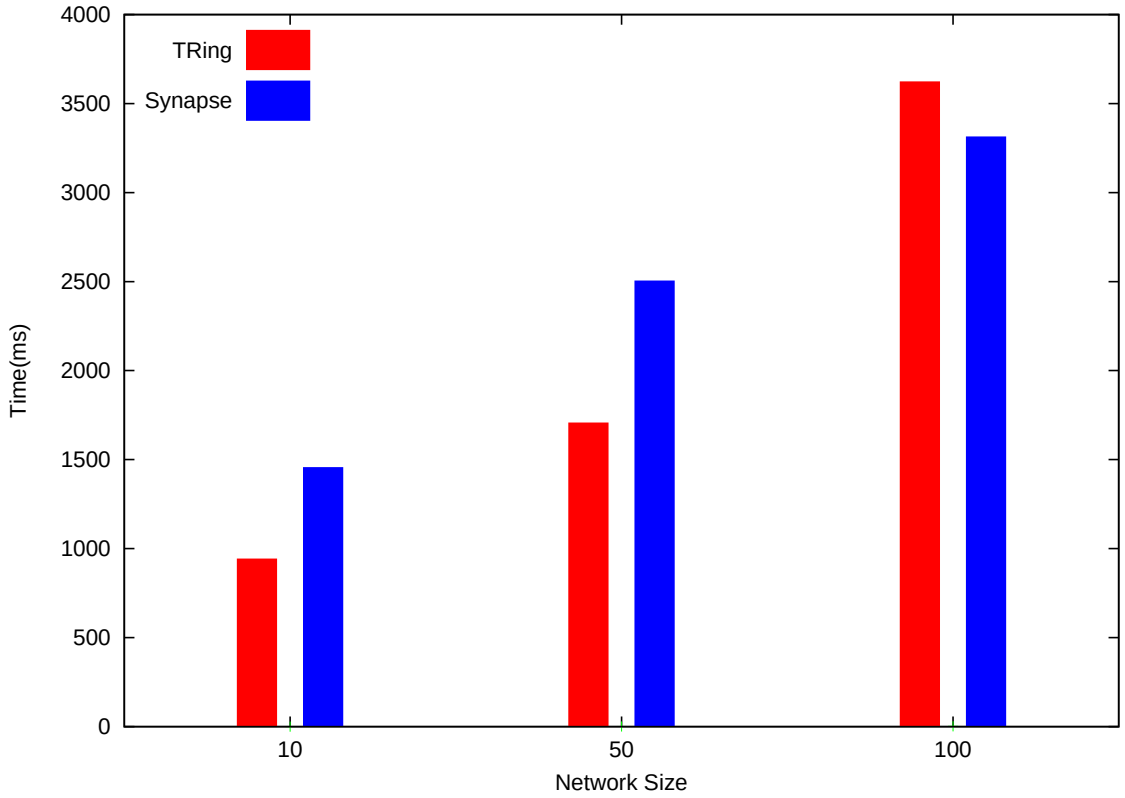
本節では，保存ピア探索の計算コスト，保存，取得それぞれの実験の結果に関しての考察を行う．

6.6.1　保存ピア探索の計算コストにおける考察

Synapse では，保存される全てのデータに対して保存ピアの計算が行われる．つまり，ネットワークサイズ＝ N，データの数＝ D，ホップ数＝ H とすると理論値では，

$$H = D \cdot \log 2N \tag{6.1}$$

となる．(6.1) における $\log 2N$ は，Chord の平均探索ホップ数である．それに対して T-Ring では，Synapse での所与の変数に加え，chunk の時間＝ C，SP の時間=S，保存されるデータの時間



	NS=10	NS=50	NS=100
T-Ring	944 ms	1708 ms	3625 ms
Synapse	1458 ms	2506 ms	3315 ms

図 6.3　保存：RTT=5

間隔=IT，保存されるデータの総時間=AT と，時間的な変数を加えて考えると，理論値は，

$$D = \frac{AT}{IT} \tag{6.2}$$

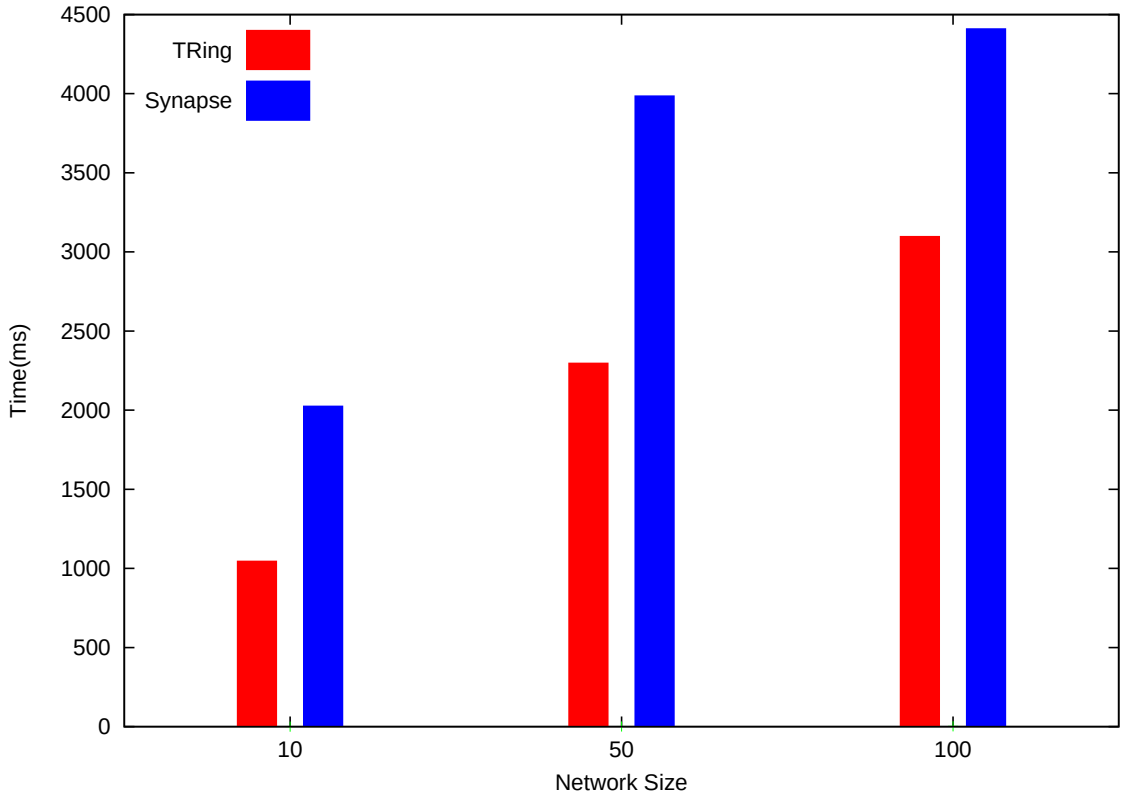
$$H = \frac{AT}{C} - \frac{AT}{S} + \frac{AT}{S} \cdot \log 2N \tag{6.3}$$

となり，(6.2)(6.3) から，AT を消去すると，

$$H = \frac{IT \cdot D\{S - C(1 - \log 2N)\}}{C \cdot S} \tag{6.4}$$

(6.4) となる．(6.1)(6.4) から，Synapse と T-Ring におけるホップ数（それぞれ SynapseH，T-RingH) の割合を計算すると，

$$\frac{SynapseH}{TRingH} = \frac{1}{IT} \cdot \frac{S \cdot C \cdot \log 2N}{S + C(\log 2N - 1)} \tag{6.5}$$



	NS=10	NS=50	NS=100
T-Ring	1049 ms	2300 ms	3101 ms
Synapse	2029 ms	3989 ms	4414 ms

図 6.4　保存：RTT=10

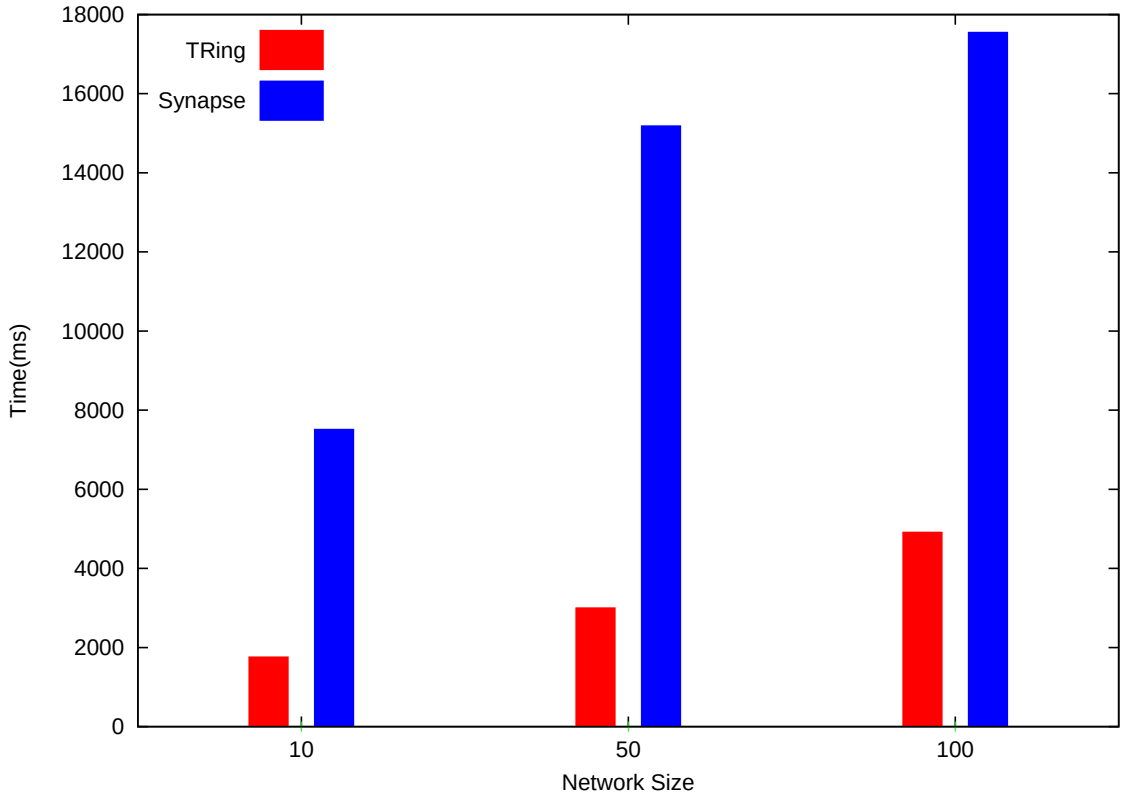
ここから，chunk と SP の関係性について考えると，

$$C \cdot S \cdot \log 2N - S + C(\log 2N - 1) = C \cdot \log 2N \cdot (S + 1) - (S + C) \tag{6.6}$$

と $C > S > 0$，$N > 0$ から，SP の値が大きくなる場合，chunk の値が SP の値に近づく場合，(6.5) の値が大きくなる．また，同様に IT の値が小さくなるほど，(6.5) の値が大きくなる．

ここから，実験結果を考察すると，実験結果は，理論値とほぼ相違ないことが分かる．よって，今回は chunk，SP，データの保存間隔を以上で述べた通りに行ったが，chunk=999，SP=1000，データの保存間隔=1 にした場合，Synapse のホップ数に限りなく近づく．

この保存ピア探索の計算コストは，データの保存，取得に大きく関係する．次節からこの関係性について考察する．



	NS=10	NS=50	NS=100
T-Ring	1049 ms	2300 ms	3101 ms
Synapse	2029 ms	3989 ms	4414 ms

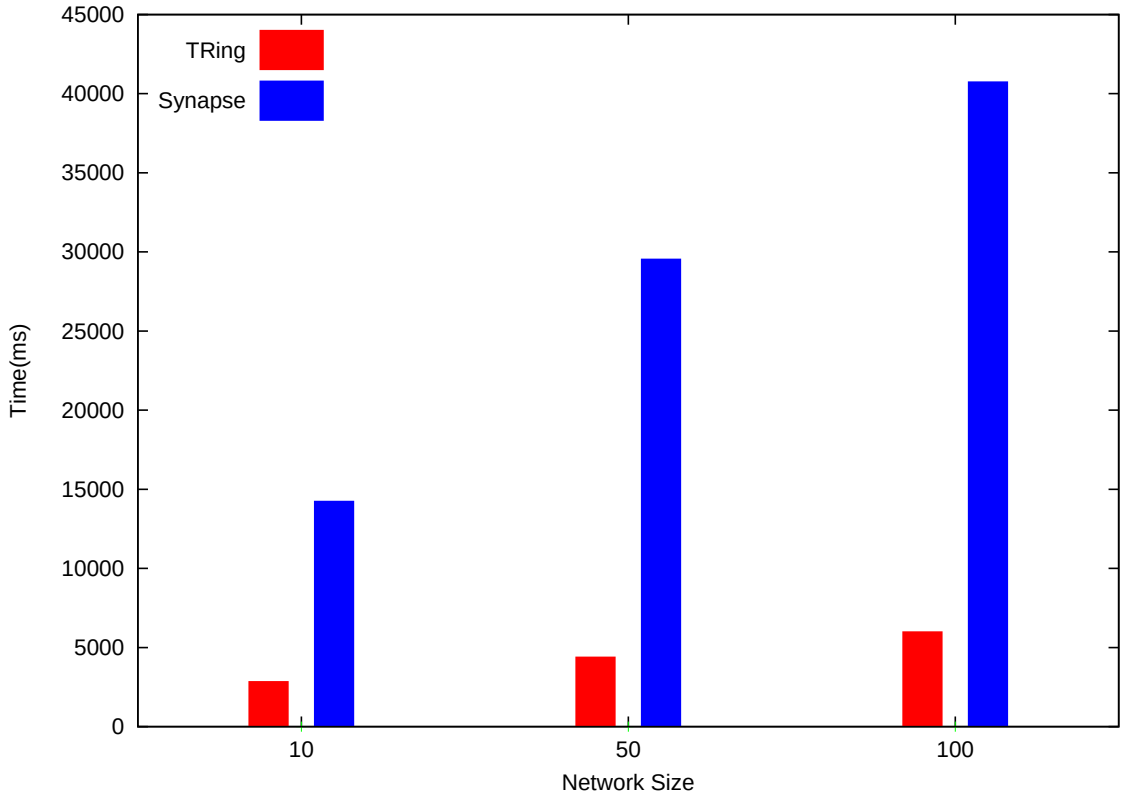
図 6.5　保存：RTT=50

6.6.2　データ保存における考察

前節において，保存ピア探索の計算コストについての考察を行ったが，Synapse と T-Ring のデータの保存にかかる時間の割合は，ネットワークレイテンシ（RTT）=RTT とすると，

$$\frac{SynapseH}{TRingH} \cdot RTT \cdot D \tag{6.7}$$

となる，Synapse と T-Ring を比較した際，RTT と D は同一の環境で実験をした場合，値の違いはないので，$\frac{SynapseH}{TRingH}$ において，取得時間は決定する．ここから，実験結果を考察すると，T-Ring における取得時間が理論値以上であることが伺える．これは，センサデータの保存を行う前段階のセンサ情報の取得による影響であると考えられる．本システムでは，センサデータがセンサノードから保存ピアに送られた際，そのセンサデータの時間とそのセンサノードの設置時間から，対象の



	NS=10	NS=50	NS=100
T-Ring	1049 ms	2300 ms	3101 ms
Synapse	2029 ms	3989 ms	4414 ms

図 6.6　保存：RTT=100

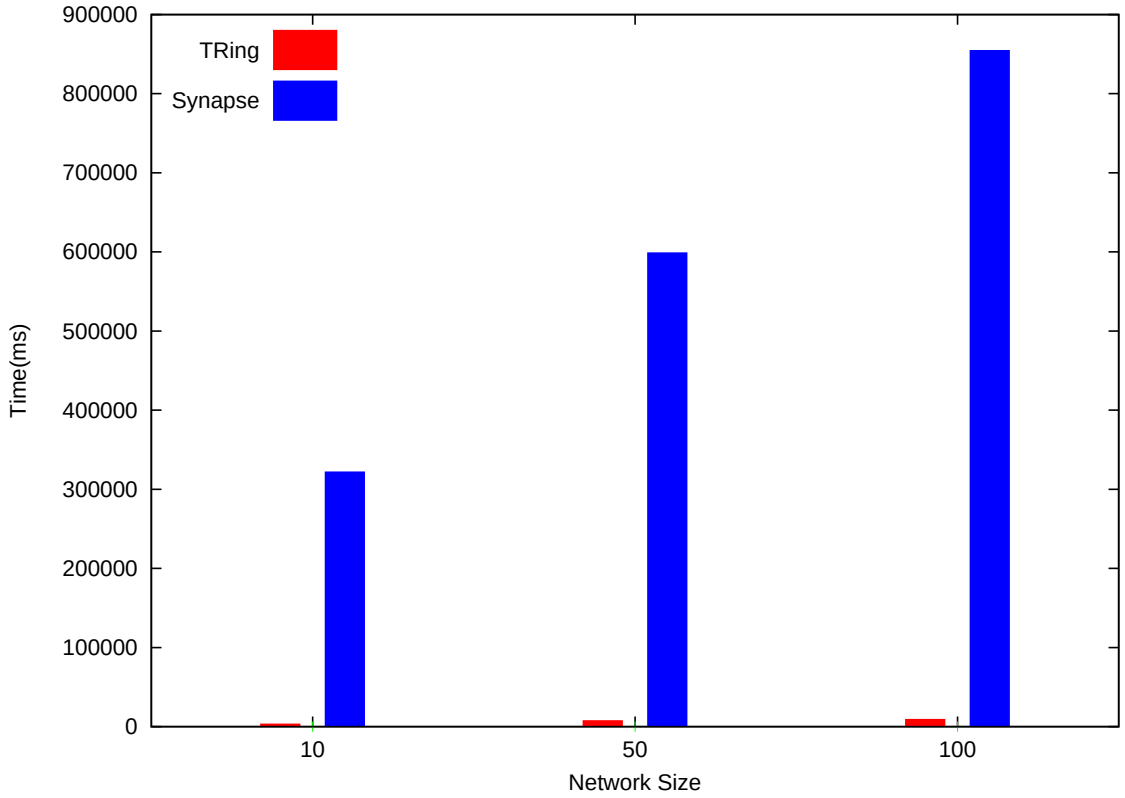
データの保存先が次の保存ピアに移行するのか，移行する際，異なるマスターノードに移行するのかを計算する．この計算に時間を要する分，T-Ring が理論値と異なっていると考えられる．

6.6.3　データの取得における考察

データの取得においても，保存時と同様に考え，クエリの検索範囲の半径=R とすると，Synapse と T-Ring のデータの保存にかかる時間の割合は，

$$\frac{SynapseH}{TRingH} \cdot RTT \cdot D \cdot \pi r^2 \tag{6.8}$$

となる．保存時と比較すると，対象範囲内の全ての座標を計算するため，πr^2 倍の差が発生することとなる．このことが，実験結果において，Synapse と T-Ring の時間の差が発生した論拠であると考えられる．

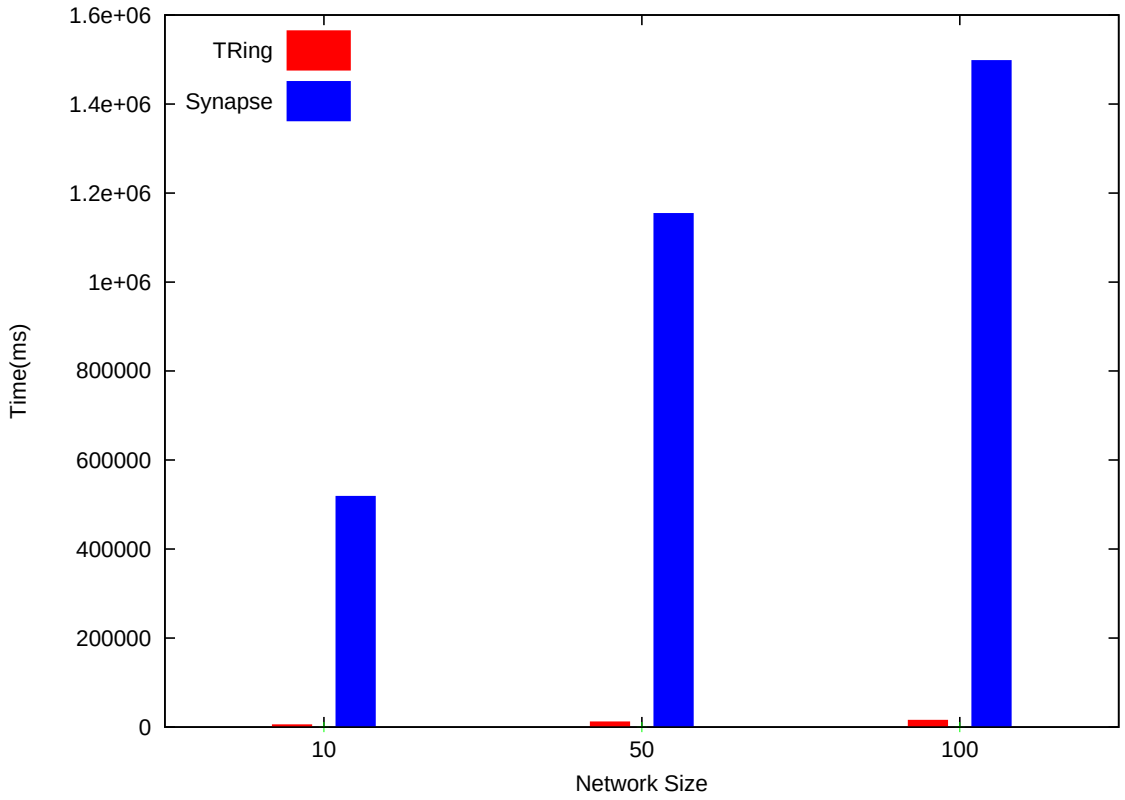


	NS=10	NS=50	NS=100
T-Ring	3884 ms	8098 ms	9823 ms
Synapse	322615 ms	599557 ms	855187 ms

図 6.7　取得：RTT=5

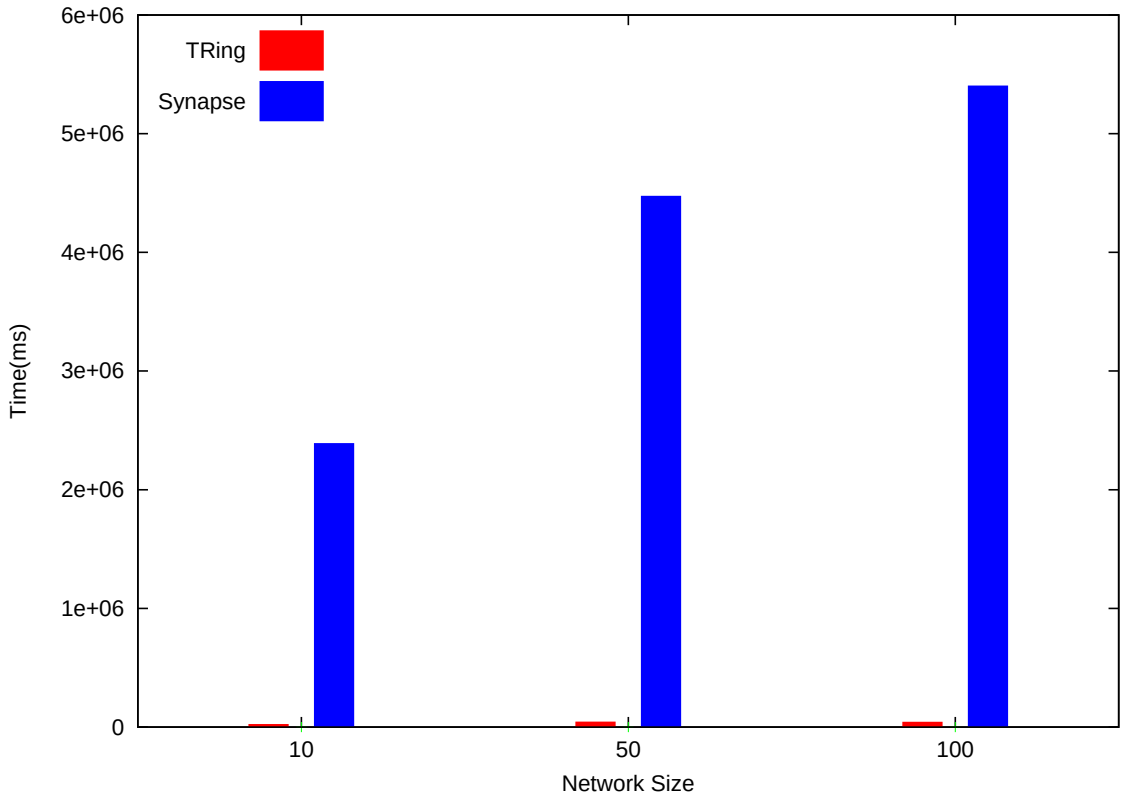
6.7　まとめ

本章では，まず，本研究に評価方針，実験環境を述べた．次に，実験環境内における実験への影響を調査するため，それぞれのリージョンから ICMP における，Message Request を送信し，リプライが返されるまでの時間（RTT）を計測した．その結果，RTT は 0.400ms 以内であり，十分に無視できることを述べた．次いで，保存ピア探索における計算コスト，データの保存に要する時間，データの取得に要する時間の観点から評価を行った．その結果から，考察を行い，理論値と実際の値に齟齬が存在することを明らかにし，その違いは，T-Ring において，保存や取得の対象となるデータの時間情報の取得を行う際に要する時間により生じていることに言及した．



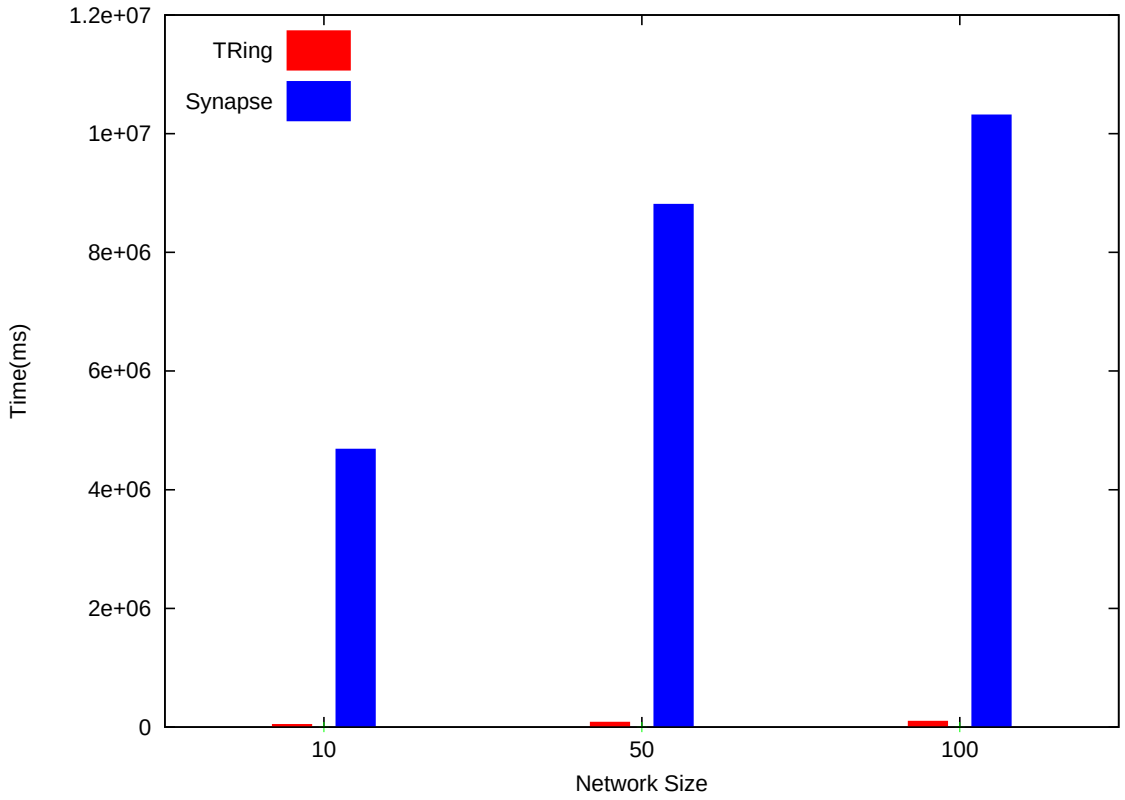
	NS=10	NS=50	NS=100
T-Ring	6247 ms	12683 ms	16249 ms
Synapse	519541 ms	1155219 ms	1498550 ms

図 6.8　取得：RTT=10



	NS=10	NS=50	NS=100
T-Ring	25730 ms	46740 ms	45068 ms
Synapse	2392797 ms	4477108 ms	5406327 ms

図 6.9　取得：RTT=50



	NS=10	NS=50	NS=100
T-Ring	51478 ms	90249 ms	104476 ms
Synapse	4691619 ms	8816386 ms	10322459 ms

図 6.10　取得：RTT=100

第 7 章

結論

本章では，T-Ring システムの結論を述べる．

7.1 今後の課題と展望

本研究では，センサデータの時間的特殊性に着目することにより，保存ピア探索における計算コスト，データ保存に要する時間，データ取得に要する時間について，既存の手法と比較し，それぞれ良い知見が得られた．しかし，データの取得に要する時間に関しては，依然として，課題が残る．既存手法との比較では，時間短縮に成功したが，半径 10 の領域から 100 個のデータの取得を行うに際し，時間数万 ms の時間を要している．想定するシナリオに挙げたように，ユーザがスマートフォンから情報を取得し，リアルタイムにアプリケーションに反映させることを斟酌すると，この所要時間では到底満足できない．よって，今後は，このデータの取得に着目した研究が必要となる．これに関する指針として，センサデータ管理のマルチレイヤ化がある．現行の T-Ring システムは，全てのデータを生データのまま保存している．しかし，現実に取得されるデータは，10:00, 11:00 など切りの良い時間から取得される可能性が高いと考え得る．また，値の平均や最大，最小などが利用されるケースも存在する．このように，個々のセンサデータについての特徴は存在しないながらも，ある一定のデータが集約されることにより，特定の意味を有する 1 つのデータになることは十分に考えられる．これらの，データの集約による特徴ごとに，マルチレイヤで管理することにより，ユーザのクエリに対する対象データ数の削減に寄与すると考えられる．

また，本研究のメインフォーカスとして挙げてはいないが，近年，Cyber-Physical Systems と呼ばれる，実空間情報を情報空間に取り込むことにより，新たな価値を提供することを目指す研究分野が注目されている．Cyber-Physical Systems では，人間や物の行動や動きを逐次記録する手法が用いられることがある．この記録データは，センサデータと同様に時間に伴って増大する．T-Ring はこのようなデータを管理する手法としても用いることが可能である．

7.2 本論文のまとめ

本研究は，従来の多次元データ管理手法がセンサデータにおける時間のような，パラメータに特徴のあるデータを対象とした手法が存在しないことに着目した．次いで，センサデータの時間的特殊性に着目した T-Ring システムの提案を行った．このシステムの評価を行った結果，既存のシステムと比べ，データ保存，取得において，高速化を実現した．しかし，データの取得に関しては，リアルタイムアプリケーションに利用するに耐えうる所要時間ではないので，最終章において，今後の課題として取り組むべきであることに言及した．

謝辞

本論文の執筆にあたり，親身になって丁寧にご指導して頂きました，慶應義塾大学環境情報学部教授徳田英幸博士に深く感謝致します．また，また，貴重なご助言を頂きました應義塾大学環境情報学部准教授高汐一紀博士，慶応義塾大学環境情報学部専任講師中澤仁博士，慶応義塾大学環境情報学部米澤拓郎特任助教，慶応義塾大学政策・メディア研究科伊藤友隆氏に深く感謝致します．

慶應義塾大学徳田研究室の諸先輩方には折に触れ貴重なご助言を頂き，また多くの議論の時間を割いて頂きました．特に，政策・メディア研究科博士課程生天目直哉氏には，本研究に対し，多くの時間を割いて頂きました．ここに多大なる感謝と尊敬の意を表します．

また，数少ない同学年として，研究活動に切磋琢磨した，伊藤瑛氏，小鷲麻奈美氏，グエンミンザン氏，KMSF，Link 研究グループにおいて，研究活動だけではなく，公私に関わらず，親しく接していただいただいた井村和博氏，鈴木幸大氏，皆川昇子氏，高木慎介氏，宇佐美真之介氏，豊田智也氏，荻野メリッサ氏を始めとした諸先輩，後輩方，研究で疲弊している中，いつも強引に徹夜でのサッカーゲームに誘ってくれて，疲れを増幅させながらも精神衛生を整え続けてくれた二神直也氏，阿部寛氏に深く感謝し，謝辞と致します．

2013 年 1 月 22 日

寺山淳基

参考文献

[1] Yin Chen Andreas Terzis Andong Zhan, Marcus Chang. Accurate caloric expenditure of bicyclists using cellphones. November 2012.

[2] Matthew Kay, Eun Kyoung Choe, Jesse Shepherd, Benjamin Greenstein, Nathaniel Watson, Sunny Consolvo, and Julie A. Kientz. Lullaby: a capture & access system for understanding the sleep environment. In *Proceedings of the 2012 ACM Conference on Ubiquitous Computing*, UbiComp '12, pp. 226–234, New York, NY, USA, 2012. ACM.

[3] Cosm Ltd. `https://cosm.com/`.

[4] Apple Inc. `http://www.apple.com/`.

[5] SAMSUNG ELECTRONICS CO. LTD. `http://www.samsung.com/`.

[6] 株式会社 MM 総研. `http://www.m2ri.jp/`.

[7] Haowen Chan and Adrian Perrig. Security and privacy in sensor networks. Vol. 36, pp. 103–105, Los Alamitos, CA, USA, October 2003. IEEE Computer Society Press.

[8] Wenbo He, Xue Liu, Hoang Nguyen, Klara Nahrstedt, and Tarek Abdelzaher. Pda: privacypreserving data aggregation in wireless sensor networks. In *in: Proceedings of the IEEE Infocom2007*, pp. 2045–2053, 2007.

[9] Claude Castelluccia, Einar Mykletun, and Gene Tsudik. E.cient aggregation of encrypted data in wireless sensor networks. In *Proceedings of the The Second Annual International Conference on Mobile and Ubiquitous Systems: Networking and Services*, MOBIQUITOUS '05, pp. 109–117, Washington, DC, USA, 2005. IEEE Computer Society.

[10] Alastair R. Beresford and Frank Stajano. Location privacy in pervasive computing. Vol. 2, pp. 46–55, Piscataway, NJ, USA, January 2003. IEEE Educational Activities Department.

[11] Volker Gaede and Oliver Günther. Multidimensional access methods. Vol. 30, pp. 170–231, New York, NY, USA, June 1998. ACM.

[12] Roger Weber, Hans-Jörg Schek, and Stephen Blott. A quantitative analysis and performance study for similarity-search methods in high-dimensional spaces. In *Proceedings of the 24rd International Conference on Very Large Data Bases*, VLDB '98, pp. 194–205, San Francisco, CA, USA, 1998. Morgan Kaufmann Publishers Inc.

[13] Ibrahim Kamel and Christos Faloutsos. Hilbert r-tree: An improved r-tree using fractals.

In *Proceedings of the 20th International Conference on Very Large Data Bases*, VLDB '94, pp. 500–509, San Francisco, CA, USA, 1994. Morgan Kaufmann Publishers Inc.

[14] Sylvia Ratnasamy, Brad Karp, Li Yin, Fang Yu, Deborah Estrin, Ramesh Govindan, and Scott Shenker. Ght: a geographic hash table for data-centric storage. In *Proceedings of the 1st ACM international workshop on Wireless sensor networks and applications*, WSNA '02, pp. 78–87, New York, NY, USA, 2002. ACM.

[15] Ion Stoica, Robert Morris, David Karger, M. Frans Kaashoek, and Hari Balakrishnan. Chord: A scalable peer-to-peer lookup service for internet applications. In *Proceedings of the 2001 conference on Applications, technologies, architectures, and protocols for computer communications*, SIGCOMM '01, pp. 149–160, New York, NY, USA, 2001. ACM.

[16] Sylvia Ratnasamy, Paul Francis, Mark Handley, Richard Karp, and Scott Shenker. A scalable content-addressable network. In *Proceedings of the 2001 conference on Applications, technologies, architectures, and protocols for computer communications*, SIGCOMM '01, pp. 161–172, New York, NY, USA, 2001. ACM.

[17] James Aspnes and Gauri Shah. Skip graphs. In *Proceedings of the fourteenth annual ACM-SIAM symposium on Discrete algorithms*, SODA '03, pp. 384–393, Philadelphia, PA, USA, 2003. Society for Industrial and Applied Mathematics.

[18] Chong Zhang, Weidong Xiao, Daquan Tang, and Jiuyang Tang. P2p-based multidimensional indexing methods: A survey. Vol. 84, pp. 2348–2362, New York, NY, USA, December 2011. Elsevier Science Inc.

[19] Akamai Technologies. `http://www.akamai.co.jp/`.

[20] Meeyoung Cha, Alan Mislove, and Krishna P. Gummadi. A measurement-driven analysis of information propagation in the flickr social network. In *Proceedings of the 18th international conference on World wide web*, WWW '09, pp. 721–730, New York, NY, USA, 2009. ACM.

[21] Yahoo! Inc. `http://www.flickr.com/`.

[22] Salvatore Scellato, Cecilia Mascolo, Mirco Musolesi, and Jon Crowcroft. Track globally, deliver locally: improving content delivery networks by tracking geographic social cascades. In *Proceedings of the 20th international conference on World wide web*, WWW '11, pp. 457–466, New York, NY, USA, 2011. ACM.

[23] Twitter Inc. `https://twitter.com/`.

[24] Facebook. `http://www.facebook.com/`.

[25] Lu-An Tang, Xiao Yu, Sangkyum Kim, Jiawei Han, Wen-Chih Peng, Yizhou Sun, Hector Gonzalez, and Sebastian Seith. Multidimensional analysis of atypical events in cyber-physical data. In *Proceedings of the 2012 IEEE 28th International Conference on Data Engineering*, ICDE '12, pp. 1025–1036, Washington, DC, USA, 2012. IEEE Computer Society.

[26] Craig Labovitz, Scott Iekel-Johnson, Danny McPherson, Jon Oberheide, and Farnam Jahanian. Internet inter-domain traffic. In *Proceedings of the ACM SIGCOMM 2010 conference*, SIGCOMM '10, pp. 75–86, New York, NY, USA, 2010. ACM.

[27] Gregor Maier, Anja Feldmann, Vern Paxson, and Mark Allman. On dominant characteristics of residential broadband internet traffic. In *Proceedings of the 9th ACM SIGCOMM conference on Internet measurement conference*, IMC '09, pp. 90–102, New York, NY, USA, 2009. ACM.

[28] John S. Otto, Mario A. Sánchez, David R. Choffnes, Fabián E. Bustamante, and Georgos Siganos. On blind mice and the elephant: understanding the network impact of a large distributed system. In *Proceedings of the ACM SIGCOMM 2011 conference*, SIGCOMM '11, pp. 110–121, New York, NY, USA, 2011. ACM.

[29] Junki Terayama, Jin Nakazawa, and Hideyuki Tokuda. Dht-based sensor data management for geographical range query. In *Proceedings of the 2012 ACM Conference on Ubiquitous Computing*, UbiComp '12, pp. 623–624, New York, NY, USA, 2012. ACM.

[30] Jonathan K. Lawder and Peter J. H. King. Using space-filling curves for multi-dimensional indexing. In *Proceedings of the 17th British National Conferenc on Databases: Advances in Databases*, BNCOD 17, pp. 20–35, London, UK, UK, 2000. Springer-Verlag.

[31] Verena Kantere, Spiros Skiadopoulos, and Timos Sellis. Storing and indexing spatial data in p2p systems. Vol. 21, pp. 287–300, Piscataway, NJ, USA, February 2009. IEEE Educational Activities Department.