卒業論文　　　　1999年度(平成11年度)

位置情報を用いたサービスの階層的検索と利用の実現

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

慶應義塾大学　環境情報学部

楠本　晶彦

卒業論文要旨　　1999年度(平成11年度)

位置情報を用いたサービスの階層的検索と利用の実現

本研究では，ネットワーク上のサービスを物理的位置情報に基づいた検索と，位置透過的な利用を提供するFLUSHシステムを提案する．

現在の計算機環境では，計算機を利用していることを強く意識する．しかし，本来計算機は道具であり，使っていることを強く意識せずに利用者の作業を支援すべきである．

本研究では，物理的位置情報を用いて，ネットワーク上のサービスを自動的に検出することを目的とする．位置情報によるサービスの自動検出を提供することで，利用者は移動を意識することなく最も近接するサービスを利用できる．

現在までに，位置情報を用いたサービスの検索機能を提供する研究は多く行われてきた．しかし，これらの研究では利用者がサービスまで移動する距離が考慮されていない．したがって，これらの研究では，最終的な選択の判断を利用者に委ね，検索の自動化を実現していない．

FLUSHシステムは空間同士の相対的な距離関係を把握している．そのため，ツリー構造を用いた検索では特定できなかった厳密なサービス間の距離関係を考慮した検索が可能となる．これにより，位置情報による検索を自動化し，利用者の負担を軽減できる．

FLUSHシステムを利用した応用例として，目的の対象の周囲に存在するディスプレイへの動画の送信や，利用者に最も近接するプリンタサービスの検出などが考えられる．

本論文では，まずネットワーク上のサービス管理と，位置情報を用いたサービス検索について論じる．次に，FLUSHシステムの設計，実装について説明し，定量的，定性的両評価を行う．本論文を通してFLUSHシステムが提供する位置情報を用いたサービスの自動検出機能の有効性を実証する．

慶應義塾大学　環境情報学部
楠本　晶彦

Hierarchical Search and Use of Service
Using Location Information

Abstract of Bachelor's Thesis

In this research, we propose the FLUSH system which provides searches for services on network and their transparent use.

Today, in the computer environment, users have to be strongly aware when using computers. But a computer is merely a tool for the realization of the user's purposes. Computer must not make users be aware of using it. In this research, the goal is auto detection of service on network using location information. Auto detection of service provides users with unconscious use of computers.

Today, many researches have proposed systems which provide searches of services on network. But in these researches, systems take no account of the distance between service and target user. The final selection of service is up to the user, which can not realize the complete automation of search.

FLUSH system takes the distance between services into account. Therefore, it can provide strict distance-based searches of services. Also provided are automation of searches and with this the user's workload is relieved.

FLUSH system realizes transmission of a movie to the display in the area surrounding the target, the auto detection of the nearest printer, and so on. In this thesis, we discuss the management of services on network and the search for them using location information. Then, we explain the design and implementations of the FLUSH system, and discuss the result of qualitative and quantitative evaluations of the system. We prove the auto detection of services using FLUSH system effective.

Akihiko Kusumoto

Faculty of Environmental Information
Keio University

目次

第1章 序論 **1**
1.1 本研究の背景 1
1.2 本研究の目的 2
1.3 本論文の構成 2

第2章 ネットワーク上のサービス管理 **3**
2.1 ネットワーク上のサービス 4
2.1.1 サービスの分類 4
2.1.2 サービスと位置情報 5
2.1.3 位置情報の座標表記と文字列表記 5
2.1.4 サービス検索と名前空間 6
2.2 Service Location Protocol(SLP) 6
2.2.1 SLPの構成要素 6
2.2.2 スコープ 7
2.2.3 SLPでの通信 7
2.2.4 SLPでのサービス検索 8
2.3 Jini 9
2.3.1 Jiniの構成要素 9
2.3.2 Jiniにおけるサービスの提供 9
2.3.3 Jiniにおけるサービスの利用 10
2.3.4 Jiniにおけるサービスの検索 11
2.4 Service Discovery Service 12
2.4.1 SDSサーバの階層的配置 12
2.4.2 SDSのセキュリティ 12
2.4.3 SDSのサービス検索 13
2.5 サービス検索の比較 13
2.6 本章のまとめ 14

第3章 位置情報を用いたサービスの検索と利用 **16**
3.1 サービスの自動検出 17
3.2 物理的位置情報によるサービス検出モデル 17

3.2.1 シナリオ1：空間内部からのサービス利用 17
3.2.2 シナリオ2：空間外部からのサービス利用 18
3.2.3 ユーザとターゲット 19
3.3 位置情報を用いた先行システム 20
3.3.1 Service Discovery Service 20
3.3.2 GLI 21
3.3.3 Ultrasonic Location System 22
3.3.4 先行システムの比較 24
3.4 用語の整理 24
3.5 本章のまとめ 25

第4章 FLUSHシステムの設計 26
4.1 設計方針 27
4.1.1 デバイス透過性 27
4.1.2 簡便性 27
4.1.3 分散性 27
4.2 FLUSHシステムの構成 27
4.2.1 サービス管理機構 28
4.2.2 位置情報管理サーバ 28
4.2.3 位置情報取得モジュール 28
4.3 想定環境 28
4.4 FLUSHシステムの動作手順 28
4.5 位置情報の設計 29
4.6 サービス管理機構の設計 31
4.6.1 サービス管理機構の構成と機能 31
4.6.2 サービス登録 32
4.6.3 サービス検索 33
4.6.4 空間同士の距離の評価 36
4.6.5 サービス管理サーバの階層化 39
4.7 位置情報取得モジュールの設計 39
4.7.1 ターゲットの情報 39
4.7.2 位置情報取得モジュールの形態 40
4.8 位置情報管理サーバの設計 40
4.8.1 位置情報管理サーバの機能 41
4.8.2 管理データ 41
4.8.3 検索 41
4.9 本章のまとめ 42

第5章 FLUSHシステムの実装 43
5.1 実装環境 44
5.1.1 既存のJiniシステムとの共存 44

5.2 サービス管理機構の実装 44
  5.2.1 LocationHelper部の実装 44
5.3 位置情報取得モジュールの実装 50
  5.3.1 地理的位置情報の取得 50
  5.3.2 サービス管理サーバの取得 50
  5.3.3 位置情報管理サーバへの送信 51
5.4 位置情報管理サーバの実装 51
  5.4.1 位置情報管理サーバの機能 51
  5.4.2 PostgreSQL 52
  5.4.3 ターゲット情報の収集 52
  5.4.4 ターゲット情報の返答 52
5.5 実装例 52
  5.5.1 サービスプロバイダ 53
  5.5.2 クライアントアプリケーション 54
5.6 本章のまとめ 55

第6章 評価と考察 58
6.1 定量的評価 59
  6.1.1 測定環境 59
  6.1.2 ツリー内検索所要時間 59
  6.1.3 距離評価所要時間 60
6.2 定性的評価 63
  6.2.1 既存システムとの親和性 63
  6.2.2 空間同士の距離関係の把握 64
  6.2.3 検索の自動化 64
  6.2.4 相対位置情報利用の有効性 64
6.3 本章のまとめ 64

第7章 終わりに 66
7.1 今後の課題 66
  7.1.1 サービスプロバイダの移動 66
  7.1.2 基本位置情報の変更 66
  7.1.3 距離の抽象化 67
  7.1.4 ユーザインタフェースの向上 67
7.2 まとめ 67

図 目 次

2.1　SLPの構成 7
2.2　SLPでのサービス登録時の属性情報例 8
2.3　SLPでのサービスリクエスト例 8
2.4　サービスの提供手順 10
2.5　サービスの利用手順 11
2.6　SDSの通信と階層化 12
2.7　サービス検索に用いるクエリ 13
2.8　検索にマッチする場合の属性情報 13
2.9　検索にマッチしない場合の属性情報 14

3.1　空間内部からのサービス利用のイメージ 18
3.2　空間外部からのサービス利用のイメージ 19
3.3　空間内のサービス 21
3.4　空間内のサービスの階層的ツリー表現 22
3.5　Active Batの空間管理 23

4.1　動作手順 29
4.2　サービスの位置情報 30
4.3　ツリー構造の例 34
4.4　サーバ内検索のフロー 35
4.5　サービスのツリー 37
4.6　κ棟の地図 38

5.1　FLUSHシステムとJini 45
5.2　Helperインタフェース 46
5.3　サービスプロバイダの位置情報 46
5.4　位置情報設定ファイルの例 49
5.5　PostgreSQLサーバへの検索要求例 53
5.6　サービスプロバイダのサンプルコード 56
5.7　クライアントアプリケーションのサープルコード 57

6.1　ツリー検索の所要時間 60
6.2　距離関係の把握先の検出 62

6.3　ツリー検索の所要時間 63

表 目 次

2.1 各システムのサービス検索の比較 15

4.1 位置情報の表記 31
4.2 位置情報の表記パターン 32
4.3 サービス検索のクエリ 33
4.4 ターゲットとサービスの位置情報管理例 34
4.5 位置情報の表記パターン 38

5.1 実装環境 44
5.2 基本位置情報の例 47
5.3 ある空間における階層距離の例 48
5.4 ターゲットに関する情報の表記例 51
5.5 位置情報管理サーバ内情報の例 52

6.1 測定環境 59
6.2 ツリー検索の所要時間 59
6.3 距離関係の把握先の検出 61
6.4 距離関係の計算 61

第 1 章

序論

1.1　本研究の背景

近年のネットワーク技術の向上と分散共有環境の登場により，ネットワーク上の計算機資源を**サービス**として提供し，それらを共有するといった概念が生まれた．具体的なサービスの例としてはネットワークプリンタや，Web サーバを始めとする各種サーバプログラムである．そして，これらサービスの共有がビジネスの世界ではイントラネットとして実用化された．

しかし，以前は手もとの計算機で用意していた機能をネットワーク上で共有することによって新たに生まれた問題もある．**サービスの管理**と**ソフトウェアの適応**といった問題がそれにあたる．

まず，サービスの管理の問題に関しては，ネットワーク上で共有するサービスの増加に伴いそれらのサービスを容易に発見し，透過的に利用する仕組みが必要となる．

また，従来ソフトウェアは単体の計算機で実行された．ソフトウェアは，個々の計算機に最適化された形で提供されれば良かった．しかし，計算機のネットワーク化に伴い，ソフトウェアは計算機と共に移動したり，ソフトウェアが状態を保持したまま計算機上を移動するといった状況が生まれた．これにより，ソフトウェアが移動先の環境に適応する必要が生じた．ソフトウェア環境への**適応**である．

ソフトウェア環境への適応には二種類あり，オペレーティングシステムや実行処理速度などの計算機の内部環境に対する適応と，実行される計算機の地理的な位置情報などの外部環境への適応である．サービスの提供はネットワーク上で行われ，それらのネットワーク上の環境も一様ではない，したがって，サービスをネットワーク上で共有するには，こういった環境への適応が不可欠である．

サービスを共有することで，サービス利用に対する簡便性が失われてはならない．サービスはあくまでも，手もとの計算機で提供されているかのような状態でネットワーク上に存在する必要がある．そのためには，サービスを適切に管理し，環境の違いをシステムが吸収する必要がある．

1.2　本研究の目的

本研究では，サービスの管理とソフトウェアの環境への適応，特に外部環境への適応を実現する．

ネットワーク上で提供されるサービスは，プリンタなどのデバイスによって提供されるサービスとWebサーバを始めとするソフトウェアによって提供されるサービスに大別できる．本研究では特にデバイスによって提供されるサービスに着目する．デバイスによって提供されるサービスは，仮想世界の情報と我々の生活する実世界の橋渡しをする．そして，実世界と仮想世界の境界では，**位置**という情報が大きな価値を持つ．Webサーバが地理的にどこに存在するかは，ユーザにとって無意味な情報である．しかし，ユーザが印刷したいのは，物理的に近接するプリンタであり，ユーザから遠く離れたプリンタへの印刷は，ほとんど意味をなさない．このように，デバイスによって提供されるサービスと位置情報には密接な関係がある．

本研究の目的は，位置情報とサービスを関連付けることで，ユーザが位置情報を明示することなく対象から物理的に近接するサービスを特定し，それらのサービスをネットワーク上で透過的に利用できる環境を構築することである．

そして，本研究の応用例として，ユーザに最も近接するプリンタを自動的に発見し，そのプリンタへ印刷を行う例や,「○○さんの近くのディスプレイ」といった指定で，対象の周囲に存在するディスプレイに動画を送信する例などが考えられる．また，本研究ではこれらの応用例を一つのモデルで実現する．

1.3　本論文の構成

本論文では，第2章においてネットワーク上のサービス管理について先行研究に触れながら考察する．第3章で外部環境への適応の一つとして，位置情報への適応を考える．第4章でFLUSHシステムの設計について，第5章で実装について述べる．そして，第6章でFLUSHシステムを評価し，第7章で本論文をまとめる．本論文では以上のような構成をとり，FLUSHシステムについて論じる．

第 2 章

ネットワーク上のサービス管理

本章では，ネットワーク上でのサービス管理について述べる．

現在，計算機環境はネットワークを通じて計算機資源を共有するという方向で発展している．そして，これらのサービスを管理し，検索する必要性が増している．

まず，ネットワーク上で提供されるサービスを管理し，検索する機構について論じる．次に，サービスを検索する際の物理的位置情報の重要性について述べる．そして，サービス検索に位置情報を用いる際の問題点を関連研究に触れながら考察する．

2.1　ネットワーク上のサービス

ネットワーク技術の向上と分散共有環境の成熟に伴い，ネットワーク上でプリンタやストレージなどの計算機資源の**共有**が可能となった．個々の計算機に接続されていた計算機資源が**サービス**としてネットワーク上に提供され，それらの計算機資源を持たない計算機から，ネットワークを通じてサービスを検索し，利用できる環境が構築されつつある．ネットワークプリンタやWorld Wide Web (WWW) なども，ネットワーク上で共有されたサービスの一例である．

サービスをネットワークで共有することのメリットとして，サービス同士の連携が可能な点が挙げられる．個々のサービスは単機能を実現するだけで，それら単機能の集合としてより複雑な機能を実現できる．例えば，WWWを実現している個々のWebサーバの仕組みは単純で，要求されたデータを返すだけである．しかし，それらのWebサーバが大規模にネットワークで接続され連携することで，現在のWWWという大規模な情報の共有環境を実現している．

また，ユーザが計算機の機能を位置透過的に利用できる点も，ネットワーク上でサービスを共有するメリットである．ネットワーク上でサービスを共有することで，ユーザの地理的位置に依存しない計算機環境を提供できる．例えばNFS (Network File) を利用することで，ユーザはネットワークを通じてどの計算機からも常に同じファイルを編集できる．

これまで，サービスとして計算機資源をネットワーク上で共有することの意義について述べてきた．しかし，ネットワーク上でサービスを共有することで生じる問題もある．それは，サービスの管理と検索である．ネットワーク上のサービスは動的に追加されたり，サービスの提供を中止したりする．また，ネットワーク上のトラブルにより，サービスが利用できなくなったりする．こうした動的な変化が絶え間なく起こるネットワーク上において，ユーザに位置透過的なサービス利用を提供するためには，個々のサービスの状態や属性をシステムが管理する必要がある．また，ネットワーク上に存在する膨大なサービスの中から，適切なサービスを検索しユーザに提供する仕組みがない限り，手もとの計算機上で動作する機能と同様な簡便性をユーザに提供できない．

2.1.1　サービスの分類

Webサーバはユーザの要求したファイルを返すというサービスを行っている．一方，ネットワークプリンタは関連するソフトウェアと連携して，プリントアウトのサービスを提供している．この二つはネットワーク上でサービスを提供していることに変わりは無い．しかし，Webサーバが提供しているサービスは仮想世界の中だけで閉じている．一方，ネットワークプリンタというサービスは，計算機の中の情報を紙という物理的媒体に変換する．

これら二つのサービスは，ソフトウェアが中心となって提供するサービスとデバイスが中心となって提供するサービスに分類できる．前者を**ソフトウェアサービス**，後

者を**デバイスサービス**と呼ぶ．そして，Webサーバや Mailサーバなどがソフトウェアサービスの例にあたり，プリンタやディスプレイなどがデバイスサービスにあたる．

2.1.2　サービスと位置情報

ネットワーク上の情報や計算機上の情報を，実世界の我々が関知するためにはそれらの情報をディスプレイによって可視化したり，プリンタによって紙という物理的媒体への変換が不可欠である．逆に，温度や物理的な物体の位置などの実世界の環境情報が計算機上に取り込まれるときも，センサを始めとするデバイスサービスが利用される．このように，デバイスサービスはネットワーク上の情報や計算機上の情報で構成される仮想世界と実世界を結びつける役割を持つ．

そして，この仮想世界と実世界の接点では常に物理的な**位置情報**が重要となる．もし，ソフトウェアサービスのように仮想世界の中だけで閉じていれば，位置という情報はあまり意味を持たない．例えば，同じコンテンツを提供するwebサーバであれば，そのWebサーバが物理的にどこにおいてあろうがサービスを利用する上で違いは無い．しかし，ユーザから遠く離れたプリンタへのプリントアウトサービスは意味を持たないし，文字入力に利用したいキーボードは，手元にあるキーボードである．また，センサによって取得される情報と，そのセンサが持つ位置情報とは不可分である．

現在，計算機がネットワーク上でどのように接続されているかは知れたとしても，計算機がどのような物理的位置関係の基に設置されているかは容易には知れない．しかし，前述したように位置情報とデバイスサービスには密接な関連性があり，これら物理的位置情報とサービスを関連付けることが重要である．そうすることで，物理的位置情報によってサービスを特定できる．

2.1.3　位置情報の座標表記と文字列表記

位置情報は，様々な表現形態による指定が考えられるが，大きく二通りに分類できる．一つは，住所のような文字列による表記であり，もう一つは，緯度，経度，高度などから構成される数値情報による表記である．この二つの表記の最大の違いは，文字列表記は人間が理解できるものであるのに対して，座標による数値情報は人間にとって理解不能な点である．

また，座標の情報は空間の絶対的位置を表現するのに適しているのに対し，文字列表記では相対的な位置関係の表現に適している．相対的な位置関係の必要性は次のような事例を考えると理解できる．例えば，ある人間の位置を座標データを基に特定できたとする．しかし，その人間が車に乗ってその場所にいるのか，あるいは歩いているのかは絶対的な座標情報だけでは特定できない．このように，対象がどのような環境に存在するかを特定するためには，絶対的な位置情報だけでなく，相対的な位置情報が必要となる．そして，これら相対的な位置情報を記述するには，座標表記ではなく文字列による表記が適している．

2.1.4 サービス検索と名前空間

文字列によって表記された位置という属性情報と，その他の属性情報の違いは，位置という情報が階層構造をなしている点である．住所を例に取れば分かるように，日本国の中に神奈川県が属し，神奈川県の中に藤沢市が属すといったように，入れ子状の階層的構造を位置情報は持っている．このように位置情報は階層的構造を持っているが，その他のサービスの属性情報のほとんどは階層的構造を持っていない．

現在までにもネットワーク上のサービスを管理し，検索機能を提供するシステムは後述するSLP[1]やJini[2]を始めとして多く存在する．しかし，その多くは属性情報に関してフラットな名前空間を提供し，階層的構造を持つ属性情報を扱いづらいものであった．

多くのシステムでフラットの名前空間を選択した理由は，多くの属性情報は階層的でなく，階層的名前空間を提供した場合に検索の柔軟性が失われる危険性があるからである．例えば，プリンタの属性情報に関して，印刷色数と印刷スピードの二つでツリー状の階層構造を作ることは無意味である．印刷色数と印刷スピードの二つ属性情報によってサービスの絞り込みを行いたいのであれば，フラットな名前空間の中に二つの属性情報を対等に配置し，それら属性情報に論理和や論理積をとることで，柔軟な検索ができる．このように，フラットな名前空間では柔軟な検索ができる一方，階層的構造を持つ属性情報を扱うには適していない．

現在までに，ネットワーク上のサービスを管理し，ユーザに検索機能を提供する研究は数多く行われてきた．次の項では，サービスを管理する仕組みとして具体的にService Location Protocol (SLP)とJini，Service Discovery Service (SDS)[3]について触れ，それらのシステムの利点及び問題点を考察する．

2.2 Service Location Protocol(SLP)

SLPはネットワーク上に存在するサービスを様々な識別子を利用して，検索する仕組みの1つである．また，SLPはTCP/IPベースの通信でサービスを検出する．

2.2.1 SLPの構成要素

SLPは，ユーザエージェント，サービスエージェント，ディレクトリエージェントの3つのエージェントによって構成されている．図2.1に，SLPの構成を示す．

ユーザエージェント

ユーザエージェントは，アプリケーションのためにサービス検出の処理を行う．

サービスエージェント

サービスエージェントは特定のサービスを求めるユーザエージェントに返答する．また，サービスエージェントはサービスに，1つ以上のディレクトリエージェントを登録する．

ディレクトリエージェント

ディレクトリエージェントは，サービスエージェントの情報を収集する．また，該当するサービスの情報を持っている場合にのみ，ユーザエージェントの照会要求に答える．

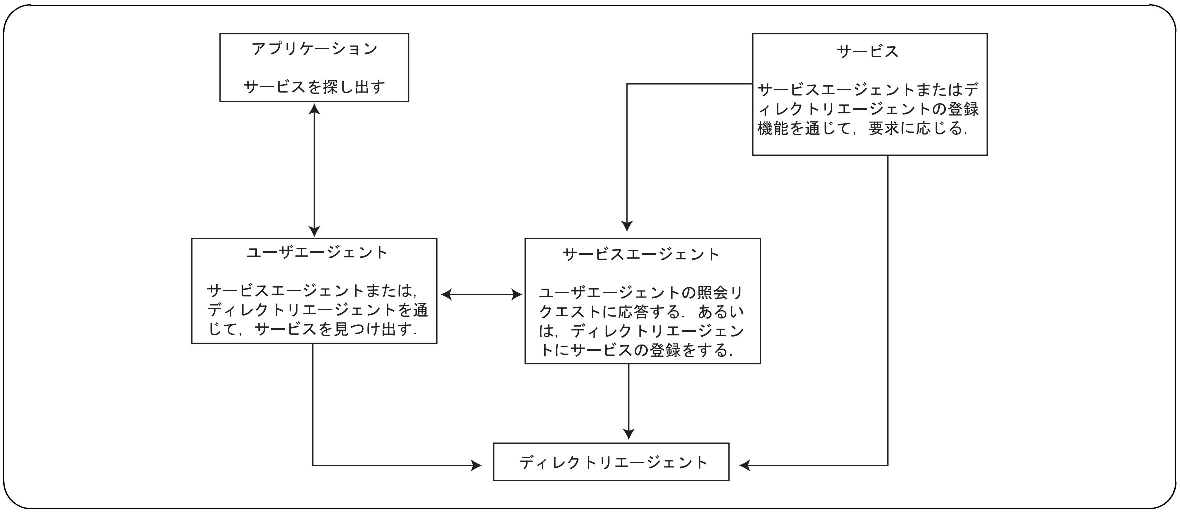
アプリケーション
サービスを探し出す
サービス
サービスエージェントまたはディレクトリエージェントの登録機能を通じて，要求に応じる．
ユーザエージェント
サービスエージェントまたは，ディレクトリエージェントを通じて，サービスを見つけ出す．
サービスエージェント
ユーザエージェントの照会リクエストに応答する．あるいは，ディレクトリエージェントにサービスの登録をする．
ディレクトリエージェント

図 2.1: SLP の構成

2.2.2 スコープ

SLP では，スコープという概念を取り入れている．スコープはサービスの集合であり，大きなネットワークには，スコープを設定しディレクトリエージェントのグループを構成する．そして，これらグループ化されたディレクトリエージェントに，サービスを登録する．こういった仕組みを導入することで，ネットワークトラフィックの軽減や，特定のディレクトリエージェントに対する負荷の集中を回避できる．

2.2.3 SLP での通信

SLP では次のような通信手段を組み合わせて通信を行うことで，ネットワーク上に存在するサービスをホスト名や IP アドレスを用いることなく指定できる．

ディレクトリエージェントディスカバリマルチキャストパケット

SLPデバイスがディレクトリエージェントを発見するために用いられる．マルチキャストアドレスの 224.0.1.35 を使用する．

ユニキャストパケット

サービスエージェントは，ディレクトリサービスが見つかると，ディレクトリサービスと直接ユニキャストパケットによって通信を開始する．

SLPジェネラルマルチキャストパケット

ユーザエージェントは，ディレクトリエージェントが存在しない場合には，直接サービスを見つけるためにSLPジェネラルマルチキャストパケットを送信する．このとき，224.0.1.22をマルチキャストアドレスとして使用する．

Internet Group Message Protocol(IGMP)マルチキャストパケット

SLPデバイスは，起動する過程でIGMPマルチキャストパケットを送信する．これによって，SLPデバイスはマルチキャストグループに加わる．

2.2.4 SLPでのサービス検索

SLPでは，図2.2に示すような記述を用いてサービスの登録を行う．図2.2の例では，DEVELOPMENTというスコープへ，12階にあるモノクロでレターサイズのプリンタをアクセス制限なしで登録した例である．

```
Lifetime:    10800
URL:         service:lpr://akihiko.ht.sfc.keio.ac.jp:515/test
Attributes:  (SCOPE=DEVELOPMENT),
             (PAPER COLOR=WHITE),
             (PAPER SIZE=LETTER),
             UNRESTRICTED_ACCESS,
             (LANGUAGE=POSTSCRIPT, HPGCL),
             (LOCATION=12 FLOOR)
```

図 2.2: SLPでのサービス登録時の属性情報例

ユーザエージェントはサービスを検索するために，以下のような表記のサービスリクエストを作成し，ディレクトリエージェントに送信する．図2.3の例は，lprプロトコルを利用でき，12階にあり12ppmの速度で出力可能なプリンタを検索するためサービスリクエストである．

```
lpr//(&  (PAGES PER MINUTE==12)
         (UNRESTRICTED_ACCESS)
         (LOCATION=12th FLOOR))/
```

図 2.3: SLPでのサービスリクエスト例

以上のような登録と検索の表記を使うことで，SLPではネットワーク上から，適切なサービスを選択できる．

2.3 Jini

Jiniは，Sun Microsystems社が提唱する分散システムで，リソースとそのユーザの連合体を形成することを目的としている．

2.3.1 Jiniの構成要素

Jiniシステムの構成要素は，以下の通りである．

- 分散システムのサービス統合に必要なインフラストラクチャ
- 信頼性の高い分散サービスの実現を支援するプログラミングモデル
- Jini連合体を形成し，各自の機能を相互に利用し合うサービス

以上のような構成要素を基に，Jiniシステムでは分散システムのすべてのコンポーネントを流動化し，オブジェクトの流動性をネットワークシステム全体に押し広げる．具体的には，ルックアップサービス，サービスプロバイダ，クライアントの3者によってJini連合体は形成される．

ルックアップサービス

ルックアップサービスは，サービスの各種情報が登録されており，サービスプロバイダとクライアントの仲介をする．ルックアップサービスも他のサービスと同様に一つのサービスとして抽象化されている．

サービスプロバイダ

サービスプロバイダは，実際にネットワーク上にサービスを提供する実体である．サービスプロバイダは，自分の情報をルックアップサービスに登録する．

クライアント

クライアントは，ネットワーク上のサービスを利用する．クライアントはネットワーク上のサービスをルックアップサービスを用いて検索し，サービスプロバイダを特定する．その後，直接サービスプロバイダと通信することで，サービスを利用する．

2.3.2 Jiniにおけるサービスの提供

Jiniシステムにおいて，サービスの提供はサービスプロバイダが自分自身の情報を登録するルックアップサービスを発見することから始まる．図2.4に，Jiniにおけるサービス提供手順を示す．

1: サービスプロバイダは自分のネットワーク上の接続位置の情報をネットワーク上にマルチキャストを用いて送信する．

2,3: ネットワーク上の接続位置情報を受け取ったルックアップサービスは，それに対して感知応答を送信する．この過程を**ディスカバリ**と呼ぶ．

4: サービスプロバイダは自分自身の情報をルックアップサービスに登録する．この作業を**ジョイン**と言う．ジョインに含まれる情報には，サービスの各種属性，サービスの型，そしてクライアントとの通信に必要なインタフェースが含まれる．

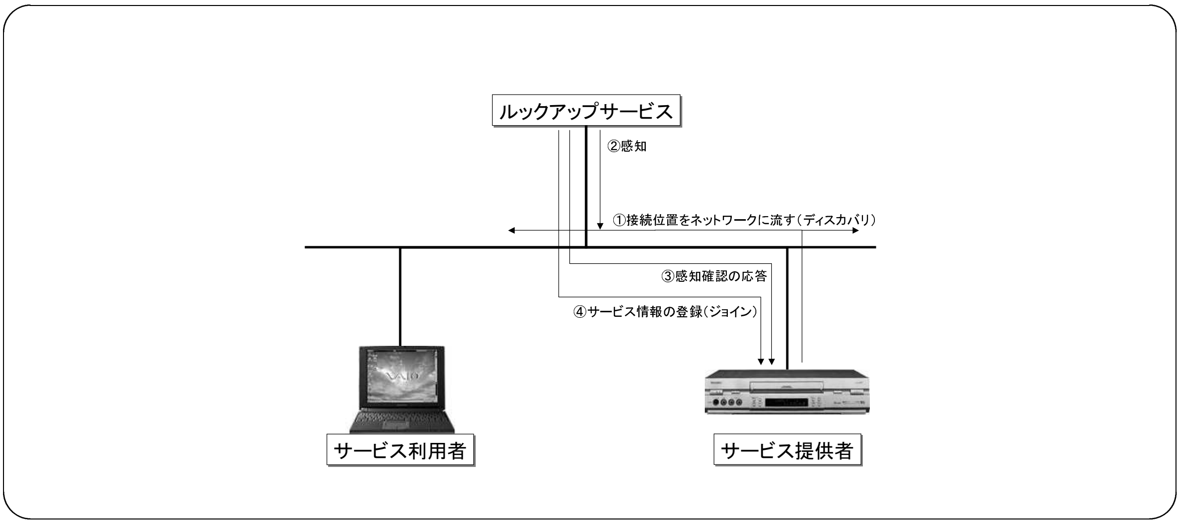
ルックアップサービス
②感知
①接続位置をネットワークに流す（ディスカバリ）
③感知確認の応答
④サービス情報の登録（ジョイン）
VAIO
サービス利用者
サービス提供者

図 2.4: サービスの提供手順

2.3.3 Jiniにおけるサービスの利用

クライアントによるサービスの検索を可能にするためにも，ルックアップサービスが必要となる．図2.5に，Jiniにおけるサービスの利用手順を示す．

1: クライアントはサービス提供の時と同様にディスカバリと呼ばれる手続きによって，ネットワーク上に存在するルックアップサービスを探し出す．

2: ルックアップサービスを用いて，属性情報を基に適切なサービスを検索する．

3: 必要に応じてサービスとの通信に使用するインタフェースをダウンロードする．

4: クライアントはサービスプロバイダと直接通信を行うことで，サービスの提供を受ける．

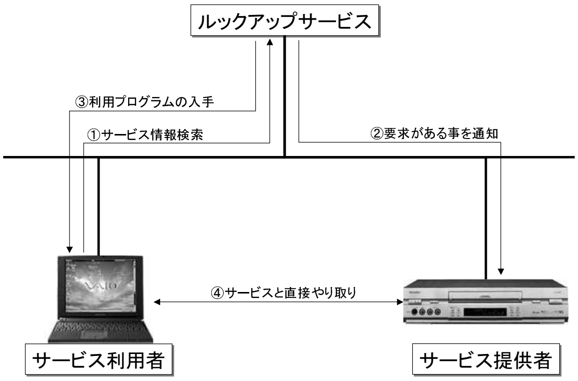
ルックアップサービス
③利用プログラムの入手
①サービス情報検索
②要求がある事を通知
④サービスと直接やり取り
サービス利用者
サービス提供者

図 2.5: サービスの利用手順

2.3.4　Jiniにおけるサービスの検索

Jiniシステムにおいて，サービスの検索はルックアップサービスを通して行われる．Jiniでは，以下のような指定方法を用いてサービスを検索できる．

サービスID

Jiniでは，サービスIDを用いることでサービスを一意に指定できる．

サービスの型

クラスだけでなくインタフェースを含め，サービスの型でサービスを指定する場合に用いる．

アトリビュート

サービスに関する様々な属性情報を指定するときに用いる．具体的な属性情報としては以下のような標準的な属性の他，サービスプロバイダが自由に属性情報を作成し，ルックアップサービスに登録できる．

基本情報： 名前，製造者，ベンダ，バージョン，モデル，シリアル番号などのサービスの基本的な属性情報．

固有情報： 標準化／自動化の限界を解決するために，Jiniでは人間が目で見て選択する方法を提供する．そのためのアイコンや説明文が含まれる属性情報．

コメント： サービスに関するコメントを定義する．

物理情報： 物理的な位置情報はLocationとAddressという2つのクラスによって提供される．

サービスの情報： プリンタの紙切れやデバイスの故障など，サービスが現在どのような状態にあるかを定義する．

2.4 Service Discovery Service

Service Discovery Service (SDS) はNinja[4]プロジェクトの一環として開発された分散共有環境を実現するシステムである．SDSは高信頼性，耐故障性，大規模分散性を特徴に掲げている．他のNinjaプロジェクト同様にJava言語で記述されており，サービスの属性情報がExtensible Markup Language (XML)[5]で記述されていて柔軟な属性情報の記述が可能である点も特徴の一つである．

2.4.1 SDSサーバの階層的配置

SDSサーバはシステム全体の中心的役割を演じている．サービスの登録や検索はこのSDSサーバを通して行われる．そして，個々のSDSサーバは階層的に配置されている．このことにより，SDSでは耐故障性や大規模分散性を実現している．あるSDSサーバにユーザが希望するサービスがない場合，階層的に配置されているネットワーク上の他のSDSサーバを利用することで，広範囲な検索を実行し，目的のサービスの発見が可能である．このようにSDSでは，サーバ間でのサービスの提供と検索が可能であり，スケーラビリティを確保している．図2.6に，SDSの構成を示す．

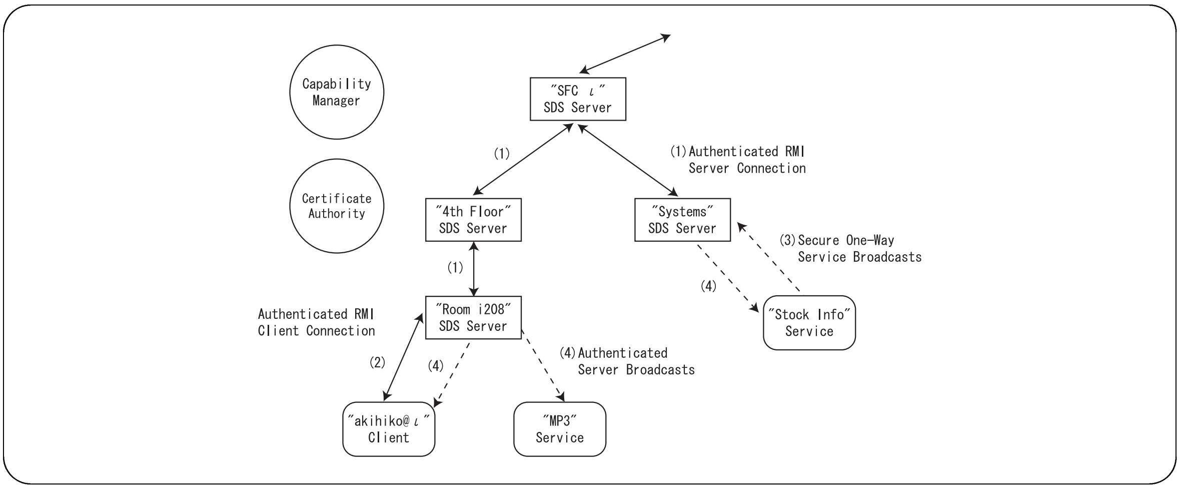
Capability Manager
Certificate Authority
"SFC ι" SDS Server
(1)
(1) Authenticated RMI Server Connection
"4th Floor" SDS Server
"Systems" SDS Server
(3) Secure One-Way Service Broadcasts
(1)
(4)
Authenticated RMI Client Connection
"Room i208" SDS Server
"Stock Info" Service
(2)
(4)
(4) Authenticated Server Broadcasts
"akihiko@ι" Client
"MP3" Service

図 2.6: SDSの通信と階層化

2.4.2 SDSのセキュリティ

SDS はデータの盗聴や改竄など外部からの不正行為を防ぐため，各コンポーネント間の通信する際，認証を行い，暗号化する．各コンポーネントは CA (Certificate Authority) を利用して，安全性が保証されているコンポーネントかを確認できる．また，SDS サーバは CM (Capability Manager) を利用して，プライベートなサービスへの Access Control List (ACL) を参照し，許可されているクライアントにのみサービス情報を送信する．各コンポーネント間の通信は公開鍵方式で暗号化される．

2.4.3 SDSのサービス検索

SDSでは，サービスの属性情報をXMLで管理しており，これにより柔軟なサービスの属性表記や検索を可能としている．また，XMLを用いることでSDSでは，サービスの属性情報を階層的な名前空間とフラットな名前空間の両方で表現できる．図2.7に，サービス検索に用いられるクエリの例をあげる．図2.7のクエリは，ポストスクリプト対応のカラープリンタの検索に用いられる．このクエリを用いて，図2.8のような属性情報を持つサービスを検索できる．しかし，図2.9に示すようなXMLによって記述された属性情報を持つプリンタは検索にマッチしない．

```
<?xml version="1.0"?>
<printcap>
    <color>yes</color>
    <postscript>yes</postscript>
</printcap>
```

図 2.7: サービス検索に用いるクエリ

```
<?xml version="1.0"?>
<!doctype printcap system
    "http://www/~akihiko/printer.dtd">
<printcap>
    <name>i208lw</name>
    <location>i208</location>
    <color>yes</color>
    <postscript>yes</postscript>
    <duplex>no</duplex>
    <rmiaddr>http://akihiko.ht/i208lw</rmiaddr>
</printcap>
```

図 2.8: 検索にマッチする場合の属性情報

2.5 サービス検索の比較

ネットワーク上でサービスの管理を実現しているシステムに関してサービス検索について比較する．名前空間に関しては，SLPとJiniはフラットの名前空間を提供して

```
<?xml version="1.0"?>
<!doctype printcap system
    "http://www/~akihiko/printer.dtd">
<printcap>
    <name>e311</name>
    <location>e311</location>
    <color>yes</color>
    <postscript>n/a</postscript>
    <duplex>yes</duplex>
    <rmiaddr>http://jin.ht/e311</rmiaddr>
</printcap>
```

図 2.9: 検索にマッチしない場合の属性情報

いるのに対し，SDSだけはXMLによって属性情報を表記しているため，フラットな名前空間中に階層的な情報を表現できる．2.1.4項で述べたように，フラットな名前空間は属性情報で論理和や論理積をとることで，柔軟な検索を提供できる．しかし，階層的構造を持つ属性情報を扱いづらいという問題点がある．

次に，サーバ間でのサービスの検索に関して比較する．Jiniではルックアップサービスも一つのサービスとして抽象化されているため，ルックアップサービスを別のルックアップサービスに登録することで，階層的なルックアップサービスの配置は可能であるが，現在のところ，それに対する具体的な取り決めは無い．SLPに関しても，同様の問題を抱えているのに対し，SDSははじめからサーバを階層的に配置し，その中でサーバ間でのサービスの検索を実現している．

また，セキュリティに関しても，JiniではACLなどの機構で一応の対応を図っているが，成りすましの問題など，セキュリティに関する不安は存在する．SDSでは，証明機構や認証機構を組み合わせる事で，セキュリティに関してもその他のシステムに対し，優位な状況である．

よって，以上をまとめるとネットワーク上のサービスを管理するシステムの中で，位置情報によってサービスを検索する点においてはSDSが優れているといえる．表2.1に，各システムが提供するサービス検索の比較を示す．

2.6 本章のまとめ

本章では，ネットワーク上のサービス管理の必要性を述べ，先行システムに関しての分析をした．また，サービスにはソフトウェアからなるサービスとデバイスによってその機能が提供されるサービスの2種類があり，デバイスによって提供されるサービスは「物理的位置」といった属性情報が特に重要となる．そして，文字列表記の位

表 2.1: 各システムのサービス検索の比較

システム名	**属性の名前空間**	**サーバ間での検索**	**セキュリティ**
SLP	フラット	△	△
Jini	フラット	△	△
SDS	フラットと階層構造の併用	○	○

置情報は，階層構造を表現できる点と人間が理解できる点において，座標による表記より優れていることを示した．そして，物理的な位置情報によってサービスを特定するためには，サービスが文字列によって表現された階層的な名前空間で位置情報と関係付けられている必要がある．

第 3 章

位置情報を用いたサービスの検索と利用

本章では，サービスを位置情報に基づいて検索する際の問題点について述べる．

まず，サービスの自動検出の必要性について述べた後，位置情報を用いたサービス検索のシナリオを挙げる．そして，位置情報による検索を提供する先行研究として，SDSとGLI，Ultrasonic Location Systemについて説明し，それぞれに関してシナリオの実現可能性について論じる．最後に，位置情報に基づいたサービスの自動検出に必要な要素を導き出す．

3.1　サービスの自動検出

現在，計算機と人間のインタフェースはキーボードやマウスといった機器が主流である．そして，これらインタフェースによって様々な情報が計算機上に取り込まれている．しかし，これらキーボードやマウスなどの既存のインタフェースは計算機を利用していることをユーザに強く意識させる．しかし，将来の計算機は無意識のうちに，ユーザの能力を高めてくれる眼鏡のような存在となることが望ましい．そこで現在，次世代インタフェースとして，実世界指向インタフェース[6] の研究が進められている．実世界指向インタフェースとは，実世界に存在する様々なオブジェクトの状況を積極的に計算機上に取り込むことで，現実世界と情報世界の透過的な結合を実現するインタフェースである．このように，計算機環境はユーザに必要以上の作業を要求せずに，計算機を使用していることを強く意識させることのない環境へと向かっている．

一方，近年ノートPCやPDA (Personal Digital Assistants)をはじめとする携帯移動端末は劇的な小型化を遂げた．加えて携帯電話や無線LANなどの無線技術の向上により，携帯端末の高速無線通信が可能となった．このような技術的革新によって，計算機は固定されたものではなく，移動中も使用されるものへと変わりつつある．しかし，移動計算機は計算処理速度や描画能力が低く，搭載されているデバイスに制限がある．移動計算機で，画像演算を始めとする高機能な処理を行う場合や，プリンタやスキャナなどのデバイスを使用する場合，移動計算機の周辺のサービスにそれらの機能を頼ることになる．しかし，移動計算機から移動先の移動計算機を利用する場合，それらの計算機資源を移動の度に認識し直す必要がある．こういった作業はユーザに負担を強いると同時に，ユーザに計算機の利用を強く意識させることにつながる．

計算機環境が人間やデバイスなどのサービスの物理的位置情報を管理する．そして，ユーザの必要に応じて，システムが物理的位置情報を基に適切なサービスを自動的に一つに特定する．この物理的位置情報によるサービスの自動検出により，ユーザは煩雑な作業から開放され，移動を意識する必要がなくなる．

3.2　物理的位置情報によるサービス検出モデル

システムが物理的位置情報によって，ネットワーク上のサービスを自動的に検出することで，ユーザの負担が軽減される．実際に，サービスの自動検出を用いた具体例を挙げ，そのシナリオを実現するための問題点を考える．

以下に，ユーザがサービスと物理的に同じ空間に存在する場合と，サービスが存在する空間の外部からそれらを利用する場合の二つのシナリオを挙げる．

3.2.1　シナリオ１：空間内部からのサービス利用

現在のネットワークで接続されたオフィスやキャンパス環境では，ネットワーク上で利用可能なプリンタは多数存在する．しかし，印字したいプリンタは，通常同一の部屋にあるか，もしくは最も近い部屋のプリンタである．

移動計算機からその部屋のプリンタに印字する場合，通常はユーザがプリンタを明示的に指定して印字する必要がある．しかし，ユーザの「プリントアウト」という抽象的な命令を計算機環境が，「移動計算機が現在位置する部屋のプリンタへの出力」だと理解する．そして，物理的に最も近接するプリントサービスを自動的に探し出し，印刷できる．例えば，図3.1の空間Aに移動計算機が存在する場合には，空間Aのプリンタへ印刷処理を行う．移動計算機が空間Cに移動した際には，自動的に空間Cにプリントアウトが行われる．また仮に，プリンタの存在しない空間Bから印刷をする場合には，空間Bから物理的に最も近接する空間である空間Aのプリントアウトサービスを利用する．

このようなシステムを構築することで，ユーザは移動先のサービスを明示的に指定することなく使用できる．

また，位置情報を取得するデバイスを特定せずに，GPSやPHSなどの地理的位置情報取得機器や，赤外線センサなどのデバイスを柔軟に使い分け，外部環境やユーザの位置情報を特定する．これにより，ユーザは特定のデバイスに依存せずに，位置情報に基づいたサービスの自動検出が可能となる．

このように，シナリオ1ではユーザの周辺に存在するサービスを自動的に検出し，利用する．これにより，ユーザは移動を強く意識せずにサービスを利用できる．

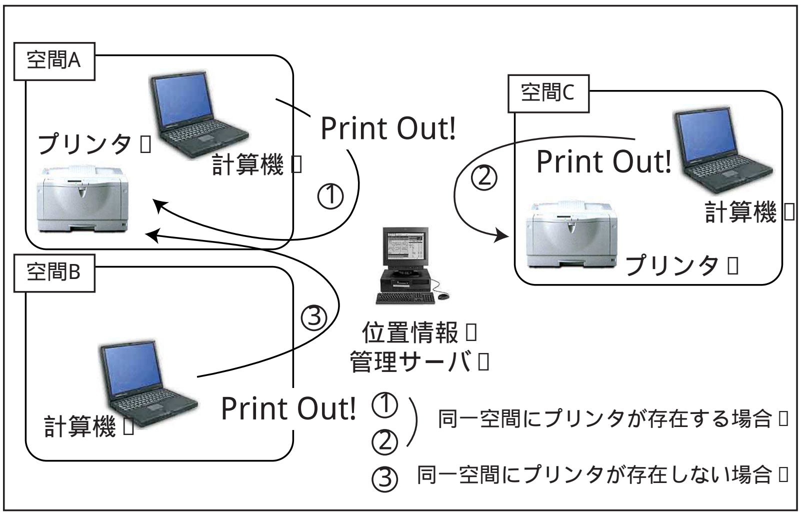
空間A
プリンタ
計算機
Print Out!
①
空間C
②
Print Out!
計算機
プリンタ
空間B
③
位置情報
管理サーバ
計算機
Print Out!
①
②
同一空間にプリンタが存在する場合
③
同一空間にプリンタが存在しない場合

図 3.1: 空間内部からのサービス利用のイメージ

3.2.2 シナリオ2：空間外部からのサービス利用

シナリオ2では，サービスがユーザとは別の空間に存在し，それらサービスを物理的位置情報によって検索し，ユーザが利用する場合を想定する．

例えば，ある特定の人の周囲のディスプレイに対し動画を送信するサービスを考える．移動計算機自体に対して，データを送信するのであれば，Mobile IPなどの既存の

技術でも可能である．しかし，移動計算機が持つ以上の計算機資源を要求される場合は，対象の環境に対しての働きかけが必要となる．ユーザと同一の空間に存在する計算機に描画処理を委任することで，移動計算機では再生不可能な動画をユーザは目にできるほか，移動計算機が用意する機能を必要最小限にできる．

ユーザは動画を送信する先を空間によって指定するのではなく，対象の名前によって指定する．例えば，図 3.2のような動画配信サービスの例では，ユーザは「××さんの周辺のディスプレイ」といった指定方法をとる．そして，ディスプレイサービスを「○○ビルの△△号室」といった指定や IP アドレスとポート番号による指定はしない．

シナリオ 2 では，ユーザがサービスを物理的位置情報やネットワーク上の接続位置なしに指定でき，それらを利用できることを想定する．

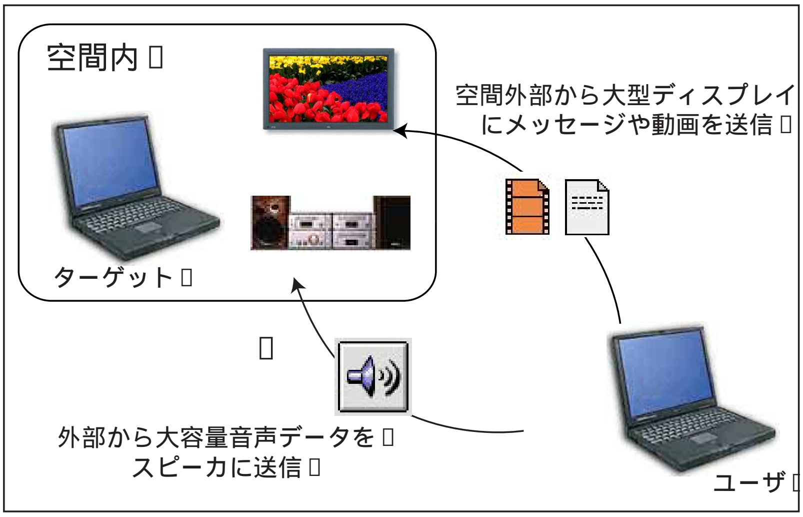

図 3.2: 空間外部からのサービス利用のイメージ

3.2.3　ユーザとターゲット

システムがネットワーク上のサービスを物理的位置情報に基づいて自動検出し，それらをユーザが利用する．こういった流れは，システムと，サービスを実際に利用するユーザ，そして，物理的位置情報による検索の始点となる対象の三つの構成要素に分けて考えられる．

システムによって検出されたサービスを利用する実体が**ユーザ**である．一方で，システムがサービスの検出に用いる物理的位置情報の始点となるのが**ターゲット**である．システムはサービスの検索をターゲットの周辺から始め，ターゲットを中心とした検索範囲を徐々に広げ，最初に発見されたサービスがターゲットに対して，最も物理的に近接すると特定する．

前述した二つのシナリオは,「同一空間内からのサービス利用」と「空間外部からのサービス利用」といった分類が可能である．シナリオ 1 では，ユーザと同一の空間内に

存在するプリンタサービスを利用する例である．そして，シナリオ2では，ユーザがサービスが存在する空間の外部から，それらを利用する例であった．

一方で，二つのシナリオをユーザとターゲットの関係に着目して分類できる．サービスを利用する側が**ユーザ**であり，位置情報によるサービス検索の基準を**ターゲット**とすると，シナリオ1はユーザとターゲットが同一な場合であり，シナリオ2はユーザとターゲットが異なる場合であるといえる．

しかし，二つのシナリオはそれぞれユーザから位置情報を隠蔽し，ターゲットの周囲のサービスを指定し，利用するという点において本質的な違いはない．したがって，ユーザとターゲットの役割を明確に分離することで，二つの異なるシナリオを一つのアプローチで解決できる．

3.3 位置情報を用いた先行システム

位置情報を活用し，物理的に近接するサービスを検索するシステムは多く存在する．これらのシステムの中からSecure Service Discovery Service (SDS), Geographical Location Information System (GLI)[7], Ultrasonic Location System (Active Bat)[8]の3つについて先行研究として触れる．

3.3.1 Service Discovery Service

2章でも述べたように，SDSでは物理的な位置関係をツリー状の階層構造で表現できる．これにより，サービスを空間ごとで指定したり，位置情報によってある程度はサービスを絞り込める．しかし，先ほど示したように，本研究では物理的に最も近接したサービスを自動的に選択し，利用できるシナリオを想定している．こういったシナリオを実現するためには，空間同士の距離関係をシステムが把握している必要がある．だが，SDSではこういった空間同士の距離関係を階層構造による抱合関係としてしか管理していない．

例えば，図3.3のように1階にAとBという部屋，2階にCとDという部屋，そして3階にEという部屋が存在するαという建物を想定する．そして，図3.3のように部屋A，C，Eに対してそれぞれ1つのサービスが存在する．このようなケースでターゲットに対して物理的に最も近接なサービスをSDSで検索する場合，この空間を図3.4のようなツリー状の階層構造を用い検索を行う．

このケースでは，作成されたツリーをターゲットから上に順に昇っていき，最初に見つかるサービスがサービスAであることから，ターゲットに最も近接するサービスはサービスAとなる．次に，仮にサービスAが存在しなかった場合を想定する．この場合でも，ツリーを用いると結果として，サービスCとサービスEの二つが検索結果として返ってくる．しかし，実際にターゲットに最も近接するサービスはサービスCである．例えば，実世界において1階にいる人が，2階と3階に存在するサービスを選択する場合，通常は2階に存在しするサービスを選択する．

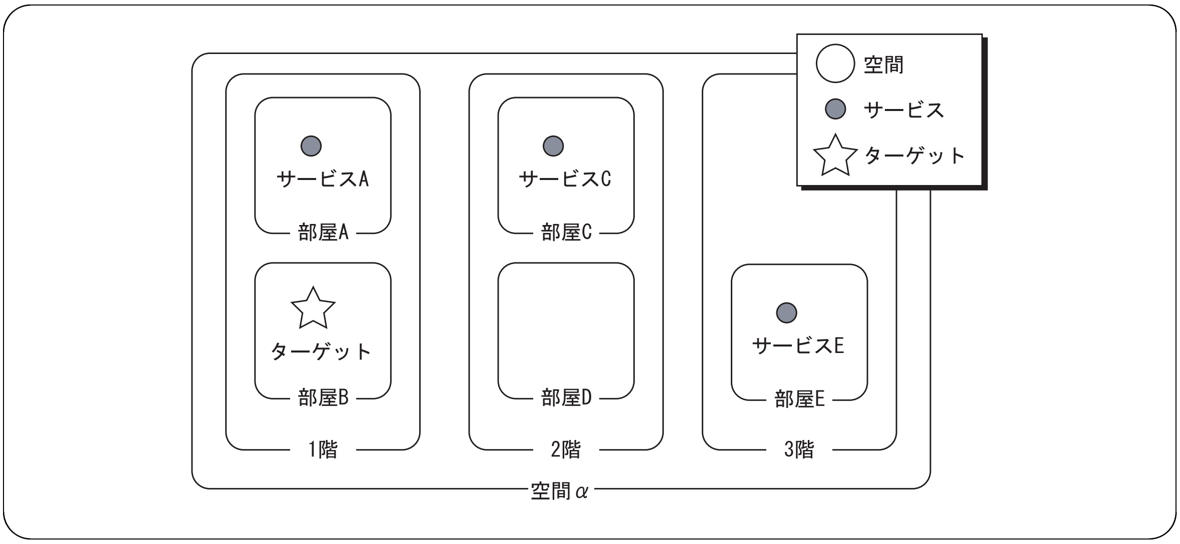

図 3.3: 空間内のサービス

同じ階層での2つ以上のサービスの位置関係を比較するためには，ツリー構造とは別に，空間同士の距離関係をシステムが把握している必要がある．先ほどの例では，1階と2階の距離と，1階と3階距離を比較できる必要がある．

SDSの問題点

SDSのように位置情報をツリーによる階層的位置関係のみで管理しているシステムでは，ターゲットに対し，どちらが近接するサービスなのかを厳密には特定できず，最終的にはユーザに選択を委ねることになる．したがって，シナリオに挙げたようにターゲットに対して，物理的に最も近接するサービスの発見をSDSによって完全に自動化するのは不可能である．

3.3.2 GLI

Geographical Location Information (GLI) システムは位置情報を直接，又は間接的に取得できるものをエンティティとしてとらえ，それらエンティティの位置情報を管理する．これによって，ネットワーク上の識別子であるIPアドレスと実世界の地理的位置情報が関連付けられる．そして，GLIシステムを使用することで，あるホストの地理的位置情報に基づいてホストを検索したり，ある地点に何があるのかを遠隔地から把握することが可能となる．

当初のGLIシステムは，単一のサーバによって地理的位置情報を管理していた．そのため，スケーラビリティに限界があった．しかしその後，様々な方向から拡張が行われ，位置情報を分散管理することで，この問題を解決した．そして，SDSの手法と同様にサーバを階層的に分散配置することが可能となった．

また，GLIシステムの特徴として，移動体の速度や移動方向などのデータをサーバ

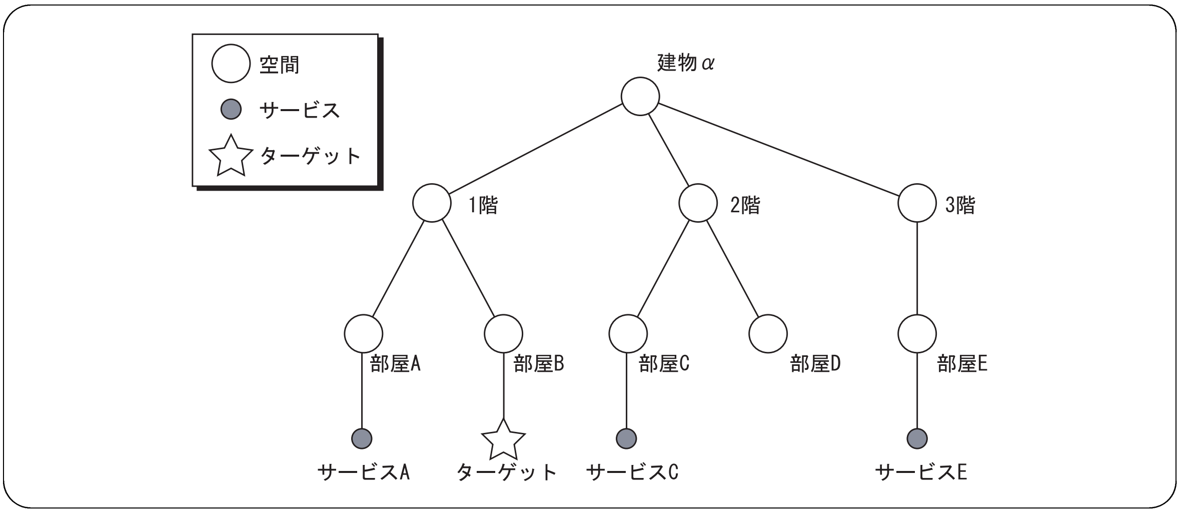

図 3.4: 空間内のサービスの階層的ツリー表現

が保持している点が上げられる．したがって，移動体の位置を捕捉するシステムとしては，非常に優れている．このことから，現在Internet Car[9] への応用も試みられている．

GLIの問題点

位置情報に基づいたサービス検索という側面からすると，GLIシステムにも問題がある．GLIシステムは空間の位置を緯度経度とドットによって区切られた文字列表記の両方を取り入れ，それらの間で相互の変換を行える．ドットによって区切られた文字列表記では位置情報の階層的管理が可能である．しかし，位置情報の階層的管理だけでは，SDSと同様に空間同士の物理的な距離関係を把握するのは困難である．また，緯度経度による距離は，空間の絶対的な距離関係を把握できる．しかし，ターゲットとサービスの間に，壁や川などの遮蔽物が存在するとき，緯度経度による絶対的な二点間の距離とユーザが体感する距離感とが必ずしも一致するわけではない．

このように，壁などの遮蔽物を考慮した空間同士の相対的距離をGLIシステムで扱うことは難しく，前述した双方のシナリオは実現できない．

3.3.3 Ultrasonic Location System

Ultrasonic Location SystemはActive Batシステムとも呼ばれ，超音波センサを用いて適応的計算機環境を実現するシステムである．Active Batシステムでは，小型の超音波発信部と室内の天井部分に埋め込まれた超音波受信部から構成される．小型の超音波発信部は，ユーザを識別するバッジの役割を果たす．ユーザは，16ビットの固有な識別子を持つそれらのバッジを持ち歩くことでサービスを受けられる．Active Batシステムでは，バッジの位置を，10cm以内の誤差で三次元の位置情報として測定可能である．Active Batでは，これらの三次元位置情報を用い，ユーザに対し位置適応的

なサービスを提供する．応用例としては，Bat Teleporting[10]がある．Bat Teleportingはバッジを持ったユーザのデスクトップの画像イメージを，移動先のディスプレイに映し出す．ユーザの移動はActive Batシステムによって感知される．

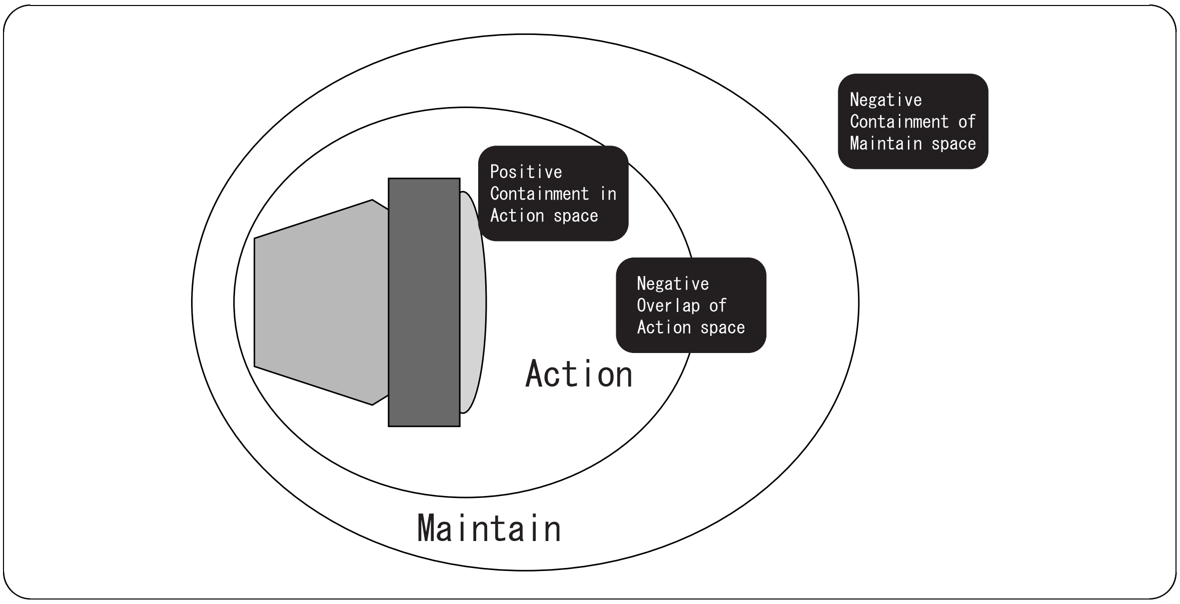
Negative
Containment of
Maintain space
Positive
Containment in
Action space
Negative
Overlap of
Action space
Action
Maintain

図 3.5: Active Batの空間管理

Bat Teleporingでは，図3.5のようにAction SpaceとMaintain Spaceと呼ばれる空間を持つ．ユーザのAction SpaceとMaintain Spaceへの出入りがきっかけとなり，自動的に画像がディスプレイに表示される．ユーザはその空間に対する出入りするだけで，実際に計算機を操作することなく，移動先のディスプレイに自分のディスプレイ環境を再現できる．

超音波センサという分解能の高い位置情報取得デバイスを用い，ユーザの位置情報を三次元で扱える点などは，他のシステムに比べ屋内で位置適応的アプリケーションを構築するのに適している．

Active Batの問題点

Active Batは超音波センサというデバイスに特化している．したがって，このシステムをGPSなどの超音波センサ以外のデバイスを用いて利用空間を屋外にまで，広げる事は困難である．つまりActive Batは，様々なデバイスにおいて取得された位置情報を統合的に扱えない．

また，Bat Teleporingではディスプレイといったのサービスの周りに空間を設定し，バッジを持ったターゲットがその空間に出入りするとシステムが感知する．それに対し，サービスを検索する範囲を徐々にターゲットを中心にして拡大していき，最初に発見されたサービスが選択される．この2つの相違は，空間の中心がサービスなのかターゲットなのかといった点にある．空間の中心がサービスである場合，イベント駆

動型のアプリケーションは実現しやすいが，ユーザが能動的にサービスを検索するアプリケーションは実現しにくい．したがって，Active Batはターゲットに物理的に最も近接するサービスを検索するシナリオを実現するには適していない．

また，前述のSDSとGLIと同様に，空間同士の距離関係がユーザが体感する距離感を反映してない．ユーザが体感する距離感を反映させるためには，空間同士の距離関係を何らかの重み付けをもって表現しなくてはならない．

3.3.4 先行システムの比較

SDSやGLI，そしてActive Batでは，それぞれ位置情報に基づいてサービス検索機能を提供している．しかし，どのシステムにおいても空間同士の相対的位置情報を管理していない．したがって，同じ階層に存在する二つ以上のサービスが検索結果に含まれていた場合，その中のでターゲットに対する物理的な距離関係に基づいた厳密なサービスの絞込みは不可能である．

このような状況下では，最終的にサービスの選択はユーザに委ねられてしまう．ユーザの手を介することで，ユーザに対して計算機の利用を強く意識させ，システムの自動化を妨げる．

以上のように，前述したシステムではターゲットに対して物理的に最も近接するサービスを自動的に一つに特定することは不可能である．そして，物理的位置情報を基に，ターゲットに対して近接するサービスを検索するためには，システムが空間同士の距離関係の把握を実現する必要がある．

3.4 用語の整理

本章の最後に，想定するシナリオで使った用語や，後の章でのFLUSHシステムの設計や実装の説明で用いる用語を整理する．

サービス

ネットワークプリンタによるプリントアウトサービスなどのデバイスによって提供される計算機資源とWebサーバを始めとする各種サーバプログラムをまとめて**サービス**と呼ぶ．サービスはネットワーク上で提供され，クライアントによって利用される．

- デバイス
  - センサ（温度センサ，照度センサなど）
  - 家電 (照明機器，スピーカなど)
  - その他各種デバイス (ネットワークプリンタなど)
- ソフトウェア
  - 各種サーバ (Webサーバ，ファイルサーバ)

    - エージェントなど

ユーザ

実際にサービスを検索し，利用する実体である．

ターゲット

ターゲットは，位置情報によるサービス検索の基準となる．ユーザはターゲットの周囲のサービスを検索し，そのサービスを利用する．本システムでは，ターゲットとユーザが同一の場合と異なる場合を同一の枠組みでとらえる．

同一空間

同一空間とは，部屋，フロア，ビル，組織の敷地，町，市などの何らかのまとまりがある空間を指す．

近接サービス

ターゲットから物理的に最も近接する空間に存在するサービスを**近接サービス**と呼ぶ．

空間同士の距離

あるまとまりを持った空間と空間の間の距離を指す．**空間同士の距離**は座標計算によって求まる絶対的数値で表現されるのではなく，壁や川などの遮蔽物を考慮した相対的距離関係によって表現される．

3.5 本章のまとめ

既存のキーボードやマウスなどのヒューマンインタフェースは，ユーザに対して計算機を利用していることを強く意識させる．しかし，現在，計算機環境は極力の自動化を図ることで，ユーザの負担を軽減する方向へ向かっている．

物理的位置情報を積極的に計算機環境に取り込み，管理する．それによって，ユーザが移動を意識することなく，周辺環境のサービスを検出できる．こういった機構を自動化することで，ユーザに対する負担を軽減し，計算機の利用を強く意識させない計算機環境を構築できる．

しかし，既存のSDSやGLI，そしてActive Batシステムなどでは，サービスが存在する空間同士の相対的距離関係をシステムが管理していない．これにより，前述のシステムでは，ターゲットに物理的に最も近接するサービスを自動的に検出することは不可能である．

そして，システムが空間同士の相対的距離関係を把握することで，サービスを位置情報に基づき自動検出できる．そうすることでユーザは，移動を意識せずにサービスを透過的に利用できる．

第 4 章

FLUSH システムの設計

本章では，位置情報に基づいたサービスの自動検出を実現するFLUSH システムの設計について述べる．まず，FLUSH システムの設計方針を明確にした後に，システムの動作手順を説明する．

また，FLUSH システムは**サービス管理機構**と**位置情報管理サーバ**，**位置情報取得モジュール**から構成される．

そして，位置情報の記述方式や空間同士の評価方法について触れながらサービス管理機構と位置情報管理サーバ，位置情報取得モジュールの設計について述べる．

4.1　設計方針

FLUSHシステムは，ネットワーク上に散在するサービスを，地理的位置情報により検索し，利用できる環境の構築を目的とする．プログラマが最小限の労力でシステム上にサービスを追加でき，それらをユーザが容易に利用するために，FLUSHシステムはデバイス透過性，簡便性，分散性の３つを重視した設計を行った．

4.1.1　デバイス透過性

ネットワーク上には時計サービスのように，位置情報による検索が意味を持たないサービスも存在する．そこで，FLUSHシステムを利用したサービスとそれ以外のサービスとの共存を確保することで，柔軟なシステムの構築を実現できる．

また，FLUSHシステムでは位置情報取得デバイスを限定しない位置情報の表記を用いる．これにより，デバイス透過性を実現する．

4.1.2　簡便性

FLUSHシステムは二つの簡便性を提供する．

一つ目は，プログラマへの簡便性を提供する．位置情報を用いた階層的な検索を提供するために，既存のJiniサービスのコードに多くの変更を要求することは，プログラマへの負担を大きくする．そこでFLUSHシステムでは，少量のコード変更で位置情報の階層的検索機能を追加可能とする．

二つ目は，ユーザへの簡便性を提供する．位置情報による階層的な検索機構を提供する一方で，ユーザに対して座標の入力や候補一覧の中からの選択を要求すると，システムの自動化が阻害され利便性が薄れる．そこで，FLUSHシステムでは，ターゲットに最も近接するサービスを自動的に一つ特定する．

4.1.3　分散性

情報家電と呼ばれるネットワークに接続可能な家電製品が出現した．これにより今後，ネットワークを利用した各種サービスの増加が予想される．したがって，FLUSHシステムでもスケーラビリティを確保するためにサービスの分散管理が必要となる．また，システム全体の耐故障性の向上のためにも，単一モジュールの故障がシステム全体に影響を及ぼさないように分散配置する．

4.2　FLUSHシステムの構成

FLUSHシステムはサービス管理機構と位置情報管理サーバ，位置情報取得モジュールから構成される．

4.2.1 サービス管理機構

サービス管理機構が，実際にサービスを位置情報により階層的に管理する．そしてサービスの登録，検索，利用は，サービス管理機構を通じて行われる．これにより，ターゲットから物理的に最も近接するサービスを特定できる．

また，サービス管理機構は，空間同士の相対的距離関係も把握している．したがって，サービスの階層的属性情報だけでは特定できないサービスを絞り込める．

サービス管理を行う実体は，**サービス管理サーバ**である．このサービス管理サーバが連携してサービス管理機構を実現する．

4.2.2 位置情報管理サーバ

位置情報管理サーバは位置情報取得モジュールから送られてくるターゲットの位置情報を蓄積し，管理する．クライアントアプリケーションはターゲットの位置情報をこの位置情報管理サーバから取得可能である．

4.2.3 位置情報取得モジュール

位置情報取得モジュールは，GPSやPHSなどのデバイスから実際に位置情報を取得する．取得した位置情報は，決められたフォーマットに統一され，位置情報管理サーバに送られる．

4.3 想定環境

FLUSHシステムは，Jiniシステムを用いて実装される．現在のJiniはJava Virtual Machine (JVM)が搭載され，10Mbps以上の帯域を持つ環境を想定して設計されている．したがってFLUSHシステムでも，10Mbps以上でTCP/IP通信が可能であり，JVMを搭載した計算機環境を想定環境とする．

また，FLUSHシステムでは大学のキャンパスや企業のビル程度の規模を想定環境として加える．後述のスケーラビリティや検索速度の評価においても，この環境で行う．

4.4 FLUSHシステムの動作手順

FLUSHシステムの動作手順を以下に示す．また図4.1にFLUSHシステムの構成と動作手順を示す．

1: ターゲットは自分の物理的位置情報を送信する位置情報管理サーバを一つ持つ．これをターゲットの**ホーム位置情報管理サーバ**と呼ぶ．そしてターゲットは位置情報取得モジュールを通じて取得した位置情報をホーム位置情報管理サーバへ定期的に送信する．

2: クライアントアプリケーションはターゲットのホーム位置情報管理サーバに対し，ターゲットの位置情報検索要求を送信する．

3: クライアントアプリケーションの検索要求に対して，ホーム位置情報管理サーバは，ターゲットの位置情報とターゲットの近隣に存在するサービス管理サーバのIPアドレスを返信する．

4: 取得したIPアドレスを基に特定したサービス管理サーバに対して，クライアントアプリケーションはターゲットに最も物理的に近接するサービスの検索要求を出す．

5: サービス管理機構のサーバからの返答として，特定されたサービスの情報を得る．

6: クライアントアプリケーションは，サービスと直接通信を行うことでサービスの提供を受ける．

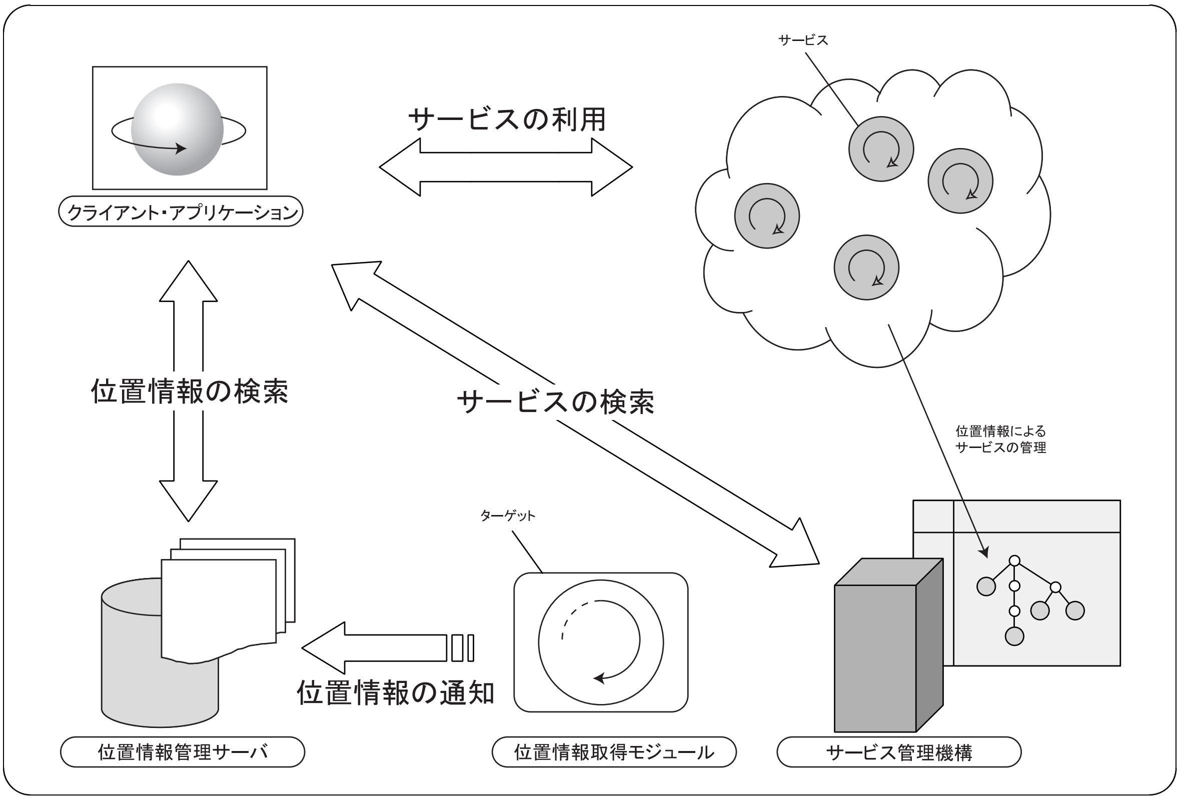
サービス
サービスの利用
クライアント・アプリケーション
位置情報の検索
サービスの検索
位置情報による
サービスの管理
ターゲット
位置情報の通知
位置情報管理サーバ
位置情報取得モジュール
サービス管理機構

図 4.1: 動作手順

4.5　位置情報の設計

位置情報は，相対位置情報と基本位置情報の２つで構成される．それら２つを合わせて絶対位置情報と呼ぶ．また，個々の位置情報はドットで区切られた文字列表記を

用いる．階層はドットで区切られ，左から右に向かって空間の粒度は拡大する．図 4.2 に，相対位置情報と基本位置情報の関係を示す.

相対位置情報 LocationHelper 部からの相対的位置情報．個々のサービスはこの表記の位置情報で登録される．

基本位置情報 LocationHelper 部の絶対的な位置情報．サーバ間検索に利用される．LocationHelper 部同士の位置関係は，階層的に上の LocationHelper 部が管理する．

絶対位置情報 上の２つの表記を合わせた表記で，サービスの検索に用いられる．

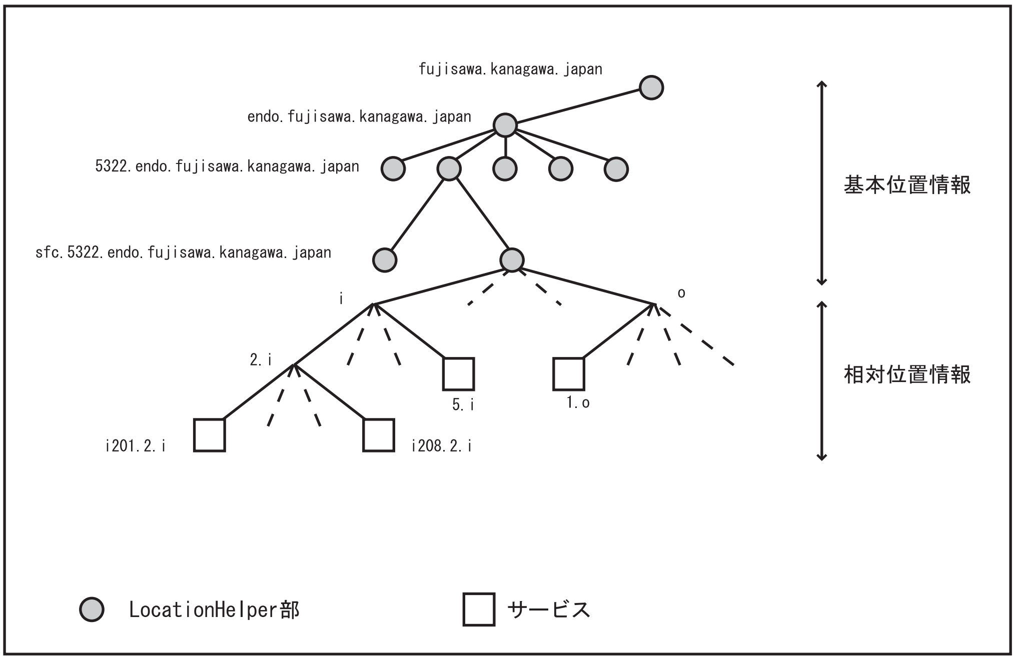
fujisawa.kanagawa.japan
endo.fujisawa.kanagawa.japan
5322.endo.fujisawa.kanagawa.japan
sfc.5322.endo.fujisawa.kanagawa.japan
i
o
2.i
5.i
1.o
i201.2.i
i208.2.i
基本位置情報
相対位置情報
LocationHelper部
サービス

図 4.2: サービスの位置情報

ネットワーク上のサービスを管理するシステムで，しばしば議論される問題として名称の統一の問題がある．

例えば，printer というサービス属性を持つプリントアウトのサービスに対して，print というサービス属性で検索を行った場合，printer の属性を持つサービスは検索結果の中には入ってこない．そして，位置情報の記述においても同様の問題が存在する．i208.2.i.sfc.5322.endo.fujisawa.kanagawa.japan と i208.2.i.keio.5322.endo.fujisawa.kanagawa.japan は同じ地点を指し示しているにも関わらず，異なる位置情報として登録される．

こういった，サービスプロバイダ側の名称不一致の問題を軽減するために，FLUSH システムでは，相対位置情報と基本位置情報を組み合わせるという方式をとった．これにより広範囲に及ぶ位置情報である絶対的位置は，LocationHelper 部を起動するシ

ステム管理者だけが把握しておけば良い．そして，個々のサービスプロバイダは，そのLocationHelper部に対する相対的位置だけを指定することで，サービス登録を行える．

神奈川県藤沢市遠藤 5322 湘南藤沢キャンパス内の ι 棟2階にある ι 208という部屋のに存在するサービスを，湘南藤沢キャンパスを管理するサービス管理サーバに対して登録する場合，表4.1のような表記をとる．

表 4.1: 位置情報の表記

i208.2.i	sfc.5322.endo.fujisawa.kanagawa.japan
相対位置情報	基本位置情報
i208.2.i.sfc.5322.endo.fujisawa.kanagawa.japan	
絶対位置情報	

4.6 サービス管理機構の設計

サービス管理機構の構成を述べた後，個々の機能の設計について述べる．

4.6.1 サービス管理機構の構成と機能

サービス管理サーバが複数接続されてサービス管理機構を構成する．そして，サービス管理サーバはJiniのLookupサービスとLocationHelper部から構成される．

Lookupサービス
LocationHelper部と連携し，サービスの登録と検索を行う．Jiniが提供している標準のLookupサービスが利用可能である．

LocationHelper部
LocationHelper部は，一つのサービスとしてLookupサービスに登録される．また，LocationHelper部は，サービス管理サーバを階層的に管理する．これにより，サービス管理サーバ間の関連付けや，サービス管理サーバ間の距離関係の評価を行いサービス管理サーバ間での検索を実現する．また，LocationHelper部は，JiniのHelper Serviceとして実装する．Helper Serviceについては5.3節で述べる．

サービス管理機構が提供する機能は，大きく分けてサービスの登録と検索である．さらにサービス検索はサーバ内検索とサーバ間検索に分けられる．

サービスの登録
サービスは，位置情報を含む属性情報と共に登録される．

サービスの検索

位置情報やその他の属性情報を基にサービスの検索は行われる．

サーバ内検索

単一のサーバ内だけで，サービスを検索する．

サーバ間検索

複数のサーバにまたがったサービスの検索をする．

次項でサービス管理機構の機能について述べる．

4.6.2 サービス登録

FLUSHシステムでは，位置情報取得デバイスで取得可能な位置情報の粒度に左右されない柔軟な位置情報表記が可能である．位置情報取得デバイスごとに取得可能な位置情報の粒度は変わってくる．また，サービスの範囲はサービスごとに異なる．例えば，プリンタサービスであれば通常はフロア単位でサービスが提供され，ディスプレイサービスなら部屋単位のサービス提供が一般的である．したがって，サービス提供の際も，表4.2のような表記パターンの中から適切な表記を選択する必要がある．

表 4.2: 位置情報の表記パターン

相対位置情報	基本位置情報	サービスの粒度
i208.2.i	sfc.5322.endo.fujisawa.kanagawa.japan	部屋単位
2.i	sfc.5322.endo.fujisawa.kanagawa.japan	フロア単位
i	sfc.5322.endo.fujisawa.kanagawa.japan	ビル単位

サービス登録の手順

以下に，サービス管理サーバへのサービス登録の手順を示す．

1：Lookupサービスを取得

サービス検索要求をする場合，まず始めにJiniのディスカバリプロトコルを使用してネットワーク上のLookupサービスを取得する．

2：サービス提供先の指定

サービスの提供先をグループ化して，指定できる．例えば，STUDENTグループやFACULTYグループを指定することで，サービスを提供する対象を分割できる．また，図4.2の表記パターンを基に，サービスを提供する空間の粒度を指定する．具体的には，ビル単位や組織単位など，様々な粒度でサービスを指定する．

3：サービスの属性登録

物理的位置情報やサービスの管理者などの属性情報をLookupサービスに登録する．

4.6.3 サービス検索

前述したように，サービス検索にはサーバ内検索とサーバ間検索がある．**サーバ内検索**は，単一のサービス管理サーバ内を，位置情報を基にターゲットに物理的に最も近接するサービスを取得する．**サーバ間検索**は，検索が必要なサービス管理サーバにサーバ内検索を依頼し，それらの結果を集計，比較することで複数のサービス管理サーバにまたがってのサービス検索が可能となる．

サーバ内検索

検索はツリーの下から行い，徐々に上の階層へと検索を進めていき，サービスが見つかった時点で検索を終了する．検索の際には何階層上まで検索するのかを指定しする．その階層までに該当するサービスが見つからない場合は，検索を終了する．

表4.3のクエリの例では，左から4番目の空間まで検索を行う．したがって，サーバ内でsfc.5322.endo.fujisawa.kanagawa.japanの部分が一致しているサービスを検索する．

また，サーバ内検索を行うサービス管理サーバはIPマルチキャストを用いて特定される．したがって，サーバ内検索の範囲はIPマルチキャストが到達できる範囲を上限とする．

表 4.3: サービス検索のクエリ

絶対位置情報	検索階層
i208.2.i.sfc.5322.endo.fujisawa.kanagawa.japan	4

次に，図4.3のような空間を例にとり，FLUSHシステムのサーバ内検索に用いられるツリー検索アルゴリズムについて説明する．

図4.3のようなサービスをツリー上で検索する場合を考える．FLUSHシステムでは，サービスの存在する空間の位置情報を表4.4のように管理している．これらサービスの中からターゲットに近接するものを検索する場合，まず表4.4の中から，J.F.B.Aというターゲットの位置情報に一致するものを検索する．しかし，この位置情報と一致するサービスは無い．次に表4.4の中から，F.B.Aの部分まで一致するサービスを探す．これに一致するサービス存在しない．そして，I.D.B.AとL.G.B.Aの位置情報を持つサービスはF.Bの部分までの位置情報がターゲットと一致する．これによって，ターゲットに近接するサービスが二つ得られる．この二つのサービスとターゲットの距離を評価することで，ターゲットに近接するサービスを一つに特定する．

また，サーバ内検索は図4.4で示すフローチャートに沿って行われる．

サーバ間検索

サーバ間検索が行われるケースは二つある．一つは，検索範囲が基本位置情報の部分まで及ぶケースであり，もう一つは，同じ空間を管理するサービス管理サーバが二つ

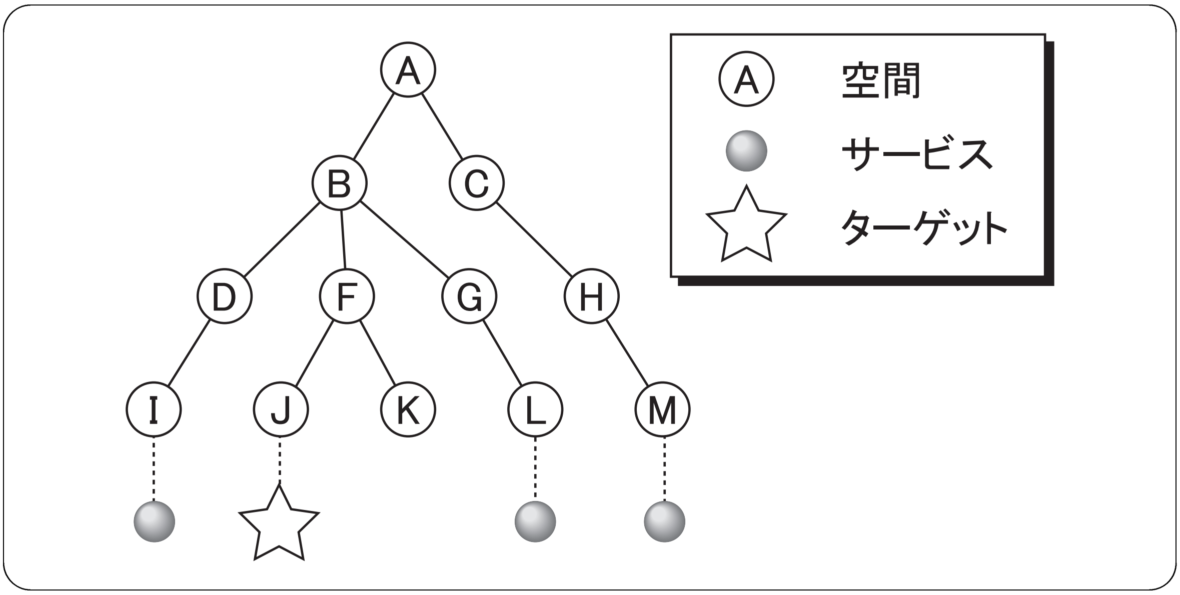

図 4.3: ツリー構造の例

表 4.4: ターゲットとサービスの位置情報管理例

ターゲット	J.F.B.A
サービス A	I.D.B.A
サービス B	L.G.B.A
サービス C	M.H.C.A

以上存在する場合である．例えば，同じ建物の中に IP マルチキャストを通さないルータに区切られたネットワークが二つ以上存在する場合がある．この場合は，一つのサービス管理サーバでターゲットに対する物理的に最も近接するサービスが特定できたとしても，そのサービスが検索空間内で最も近接するサービスとは断定できない．

例えば，IP マルチキャストを通さないルータに区切られたネットワークが二つ存在し，そのどちらか一方にプリンタが接続されている部屋がある．その部屋の中から近接するサービスの検索行う場合，両方のネットワークに対して検索を行わない限り，その部屋のプリンタを発見できない．このような状況に対応するために，ある空間に区切られたネットワークが複数存在する場合はそのすべてに対して検索を行わなければならない．

サーバ間検索は，個々のサーバで行われた，サーバ内検索の結果を集計する．各サーバから持ち寄られた候補を比較することで，検索範囲の中で最もターゲットに対して近接するサービスが得られる．

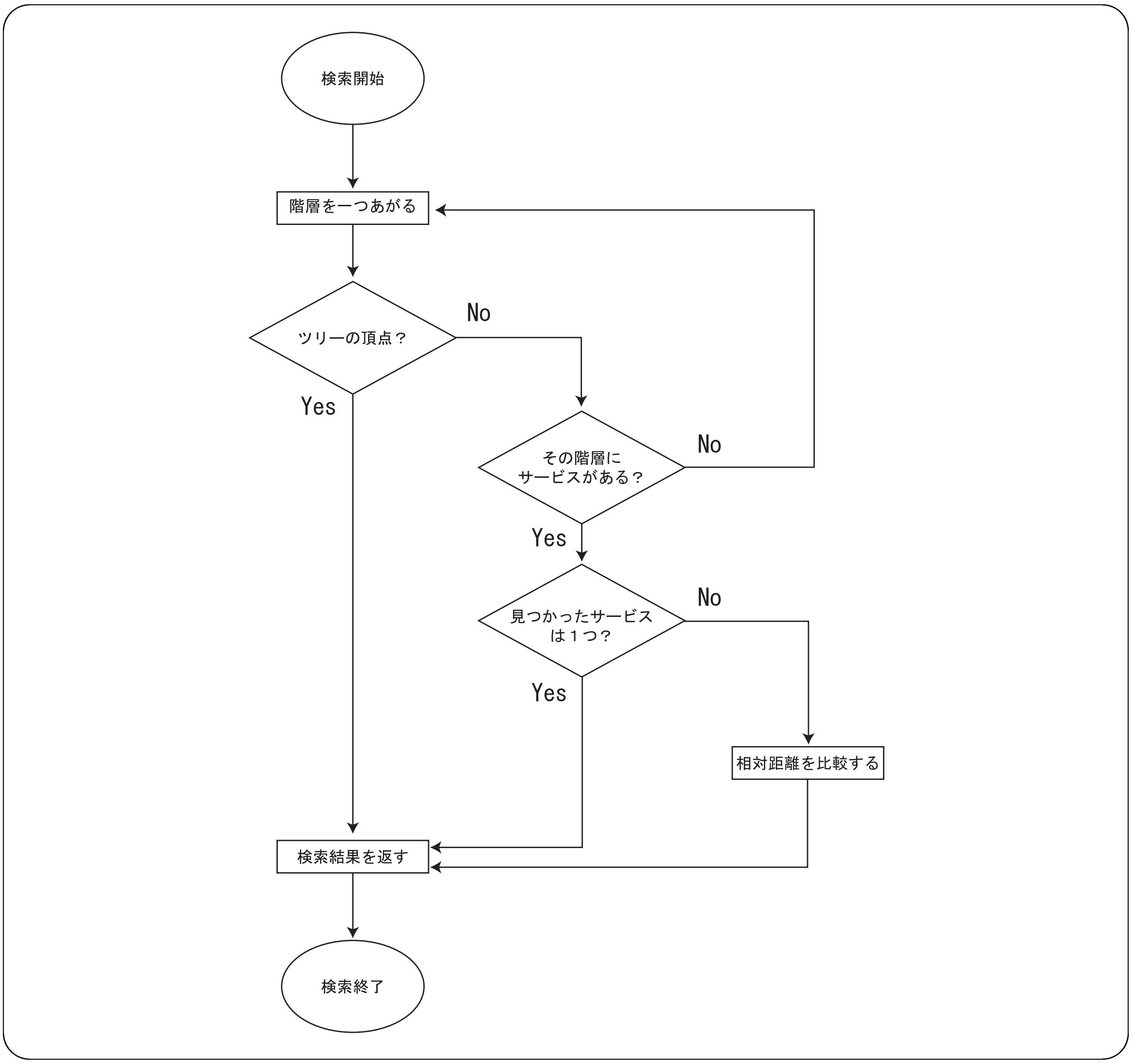
検索開始
階層を一つあがる
ツリーの頂点？
No
Yes
その階層に
サービスがある？
No
Yes
見つかったサービス
は１つ？
No
Yes
相対距離を比較する
検索結果を返す
検索終了

図 4.4: サーバ内検索のフロー

距離評価の必要性

サーバ内検索とサーバ間検索の両方の検索において空間同士の距離の評価を行う．FLUSH システムでは，該当するサービスが同じ階層に二つ以上あった場合，それらのサービスをターゲットに対する距離関係で評価し，ターゲットに最も近接するサービスを特定しなければならない．どちらのサービスがターゲットに物理的に近接するサービスかを決定するためには，FLUSH システムが空間同士の距離関係を把握している必要がある．空間同士の距離関係の把握については，1.5.5 項で述べる．

サービス検索の手順

1：Lookupサービスを取得

サービス検索をする場合，まず初めにJiniのディスカバリプロトコルを使用してネットワーク上のLookupサービスを取得する．

2：LocationHelperサービスを利用

取得したLookupサービスを利用して，Lookupサービスに登録されているLocationHelper サービスを取得する．

3：基本位置情報を取得

LocationHelperサービスを使用して，基本位置情報を取得する．

4：位置情報検索クエリの作成

ターゲットの絶対位置情報と検索粒度によって，位置情報検索クエリを作成する．検索が基本位置情報の部分に及ぶ場合には，LocationHelperで構成されるツリーを用いて，該当する位置情報管理サーバを取得する．

5：サービスのサーバ内検索

位置情報検索クエリとその他の属性情報をセットにし，該当するサービス管理サーバよってサーバ内検索が行われ，各サービス管理サーバから候補が返される．

6：サービスのサーバ間検索

複数のサービス管理サーバに要求した場合，それらの候補を比較することで，最も近接するサービスを一つ特定する．

4.6.4　空間同士の距離の評価

FLUSHシステムでは，同じ階層で2つ以上のサービスが発見された場合，その中で最もターゲットに近いサービスを利用する．そのためには，サービスごとの距離を評価する必要がある．

距離の評価方式にも，直線距離を基にした評価と移動距離を考慮した評価の二種類が考えられる．直線距離を用いた評価では，壁や川などの遮蔽物の存在を考慮しにくい．遮蔽物を加味した評価を行うには，人の移動を考慮する必要がある．

そこで，FLUSHシステムでは空間ごとに相対的な重み付けを行い．それらを評価する事で，人がサービスまで移動する距離をサービス検索に反映させる．空間はツリーを構成しているほか，図4.5のように空間同士の相対的な距離関係を持っている．例えば，湘南藤沢キャンパスのκ棟1階を例にとってみる．

FLUSHシステムにおいて図4.6のような地図で表現される空間同士の位置関係は表4.5のような相対値に基づき重み付けできる．図4.6のκ 12にユーザが存在し，ユーザに対して近接するプリンタを検索する状況を考える．そして，κ 11とκ 13にプリンタが存在する場合，表4.5によるとκ 11とκ 12の重みは1であるのに対し，κ 12とκ 13

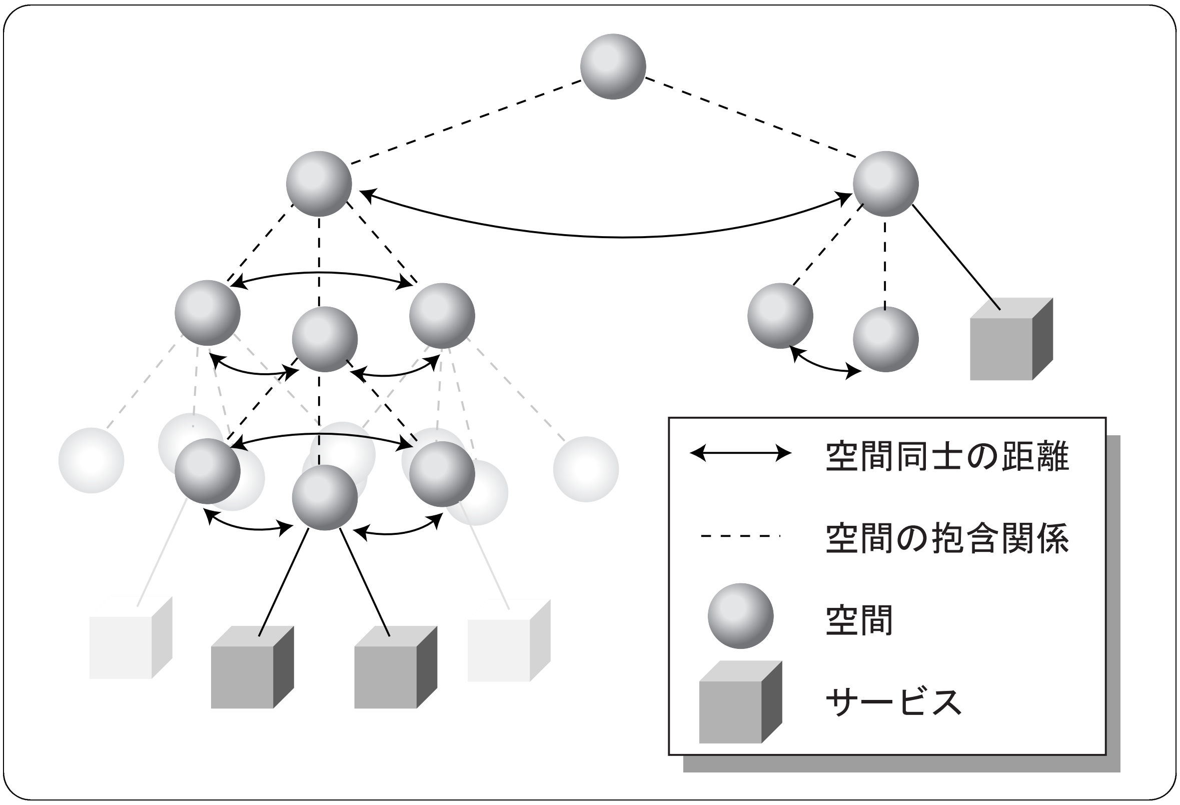

図 4.5: サービスのツリー

の重みは2である．したがって，FLUSHシステムは自動的にκ 11のプリンタに印刷が行われる．

表4.5の相対的評価データは，サービス管理サーバが保持している．サービス管理サーバは，登録されているサービスが存在する範囲の相対的評価データを管理している．

図4.3のサービスの検索を例にとり，空間同士の距離評価のアルゴリズムについて述べる．

図4.3の例では，ツリー構造によってターゲットに近接するサービスがI.D.B.AとL.G.B.Aの位置情報を持つ二つである事がわかる．しかし，ツリー構造だけでは，これ以上の位置関係の把握は不可能である．したがって，FLUSHシステムでは空間同士の距離関係をシステムが把握しており，これによってサービスを評価することで，更なる絞込みが可能となる．

FLUSHシステムでは，それぞれの候補とターゲットの距離を計算し，それらを比較する事でサービスを一つに特定する．例えば，図4.3の例であれば，I.D.B.Aのサービスとターゲットとの距離を計算したものと，L.G.B.Aを位置情報として持つサービスとターゲットとの距離を比較する．そして，距離関係の重み付けが小さいほうを

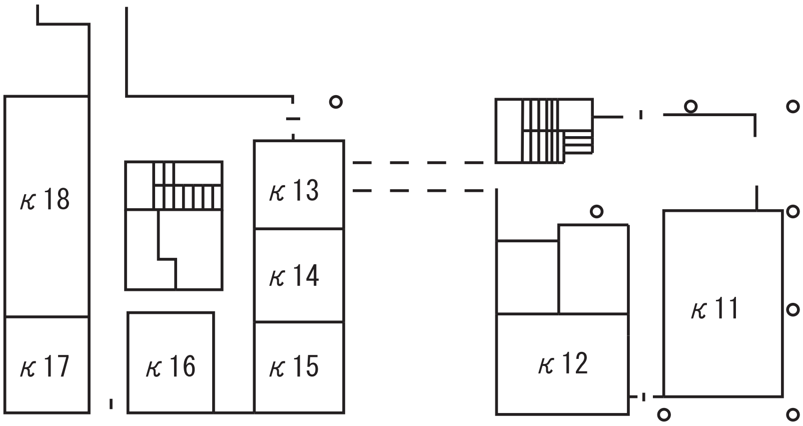

図 4.6: κ 棟の地図

表 4.5: 位置情報の表記パターン

	κ 11	κ 12	κ 13	κ 14	κ 15	κ 16	κ 17	κ 18
κ 11	-	1	3	4	5	6	7	4
κ 12	1	-	2	3	4	4	4	4
κ 13	3	2	-	1	2	3	4	2
κ 14	4	3	1	-	1	2	3	3
κ 15	5	4	2	1	-	1	2	2
κ 16	6	4	3	2	1	-	1	1
κ 17	7	4	4	3	2	1	-	1
κ 18	4	4	2	3	2	1	1	-

ターゲットに最も近接するサービスであると特定する．ある空間はツリー構造の下の空間の距離関係を把握している．図4.3の場合，空間Aは空間Bと空間Cの距離関係を把握しいる．そして，距離関係の評価は二つの過程に分けられる．まず，どの空間が比較対象の距離関係を把握しているのかを特定する．そして，その空間に対して比較対象の距離関係を検索する．例えば，I.D.B.Aを位置情報として持つサービスとJ.F.B.Aの位置情報を持つターゲットの距離関係を計算する場合，空間Dと空間Fの距離関係を計算すればよい．そして，この場合，空間Dと空間Fの距離関係は空間Bが把握している．そして，空間Bが持つ距離関係の中から空間Dと空間Fの距離関係を検索する事で，距離関係の重み付けが得られる．

4.6.5 サービス管理サーバの階層化

前述のように，サービス管理サーバ間でのサービス検索を実現するために，サービス管理サーバは階層的に配置する．このことで，サービスの数の増加に伴う過剰な負荷を分散させることが可能となる．LocationHelper部によってサービス管理サーバは階層的位置情報にマッピングされる．また，サービス管理サーバ同士の距離関係は，それらの上に存在するLocationHelper部が把握する．

4.7 位置情報取得モジュールの設計

位置情報取得モジュールの機能は，大きく分けて二つある．一つはターゲット情報の収集，もう一つはターゲット情報の位置情報管理サーバへの送信である．

4.7.1 ターゲットの情報

ターゲットの名称

ターゲットの名称は，後述の位置情報管理サーバにおいて一意である必要がある．ターゲットの名称はターゲット自体の名前（大抵の場合ターゲットは人間である）と通知すべき位置情報管理サーバのホスト名で構成される．具体的には以下のような表記をとる．

```
akihiko@hostA.ht.sfc.keio.ac.jp
```

ターゲットの地理的位置情報

基本的に，ターゲットは位置情報取得デバイスを持つ．ターゲットが固定されている場合は，ファイルによる設定も可能である．GPSやPHSなどの位置情報を取得するデバイスによって，位置情報の表現方法は異なる．また，同じGPSでも，メーカや製品によって位置情報の表現方法が異なる．FLUSHシステムでは，各デバイスから取得された位置情報は，この位置情報取得モジュールでドットで区切られた表記に統一され，位置情報管理サーバに通知される．これにより，位置情報取得機器を限定しないシステムを実現する．

ターゲットが利用可能なLookupサーバのアドレス

位置情報取得モジュールは，ターゲットがIPマルチキャストの届く範囲に存在するサービス管理サーバを見つけ，そのアドレスを取得する．このアドレスを基に，ユーザはターゲットに物理的に最も近接するサービスの検索を，サービス管理サーバに要求する．

取得デバイス名

位置情報はドットで区切られた文字列表現に統一され，位置情報管理サーバに通知される．しかし，位置情報取得デバイスにより，取得できる位置情報の粒度が

異なる．それらの粒度を表現する目安として，位置情報をどのようなデバイスを用いて取得したかをターゲットは情報として持つ．

位置情報取得モジュールは以上のような情報を取得し，これらを位置情報管理サーバに通知する．

4.7.2 位置情報取得モジュールの形態

位置情報取得モジュールには，二通りの形態が存在する．ターゲット自身が位置情報を取得する形態とターゲットの周辺の環境がターゲットの位置を捕捉する形態である．言いかえると，位置情報管理モジュールがターゲットと共に移動する形態とターゲットの外部に位置情報管理モジュールが存在する形態である．

例えば，GPSやPHSによる位置情報の取得や無線LANの基地局を利用した位置情報の取得はターゲット自身が位置情報を取得する形態に分類できる．ターゲットがICカードを差すカードリーダの位置によって位置情報を取得したり，ターゲットを追尾するカメラによって位置を捕捉する場合では，ターゲットを取り巻く環境がターゲットの位置情報を把握しているといえる．

このように，位置情報をターゲット自身が取得する場合を位置情報の**能動的取得**と呼ぶ．また，ターゲットの外部環境がターゲットの位置情報を把握する場合を位置情報の**受動的取得**と呼ぶ．

- 位置情報の能動的取得の例
  - GPS
  - PHS
  - 無線LAN
- 位置情報の受動的取得の例
  - ICカード
  - 追尾カメラ

FLUSHシステムでは，位置情報の能動的取得と受動的取得の両方の形態をサポートする．

4.8 位置情報管理サーバの設計

位置情報管理サーバの機能について述べた後に，位置情報管理サーバで管理されるデータとそれらの検索を方法について述べる．

4.8.1 位置情報管理サーバの機能

サービス管理サーバが，サービスの物理的な位置情報を管理するのに対して，位置情報管理サーバはターゲットの位置情報を管理する．

位置情報管理サーバは，位置情報取得モジュールから送られてくるターゲットの情報を蓄積し，管理する．また，クライアントアプリケーションからのターゲットの情報検索要求に，ターゲットの情報を返答する．

クライアントアプリケーションは，取得した情報に基づきターゲットの周辺に存在するサービス管理サーバへ，ターゲットの位置情報をキーに検索を行う．

前述の様に，個々のターゲットはターゲット情報を送信するホーム位置情報管理サーバを一つ持つ．ターゲットが位置情報管理モジュールを通じて取得した位置情報と，その他のターゲットに関する情報を定期的にもしくは必要に応じて，ホーム位置情報管理サーバへ通知する．

4.8.2 管理データ

位置情報管理サーバ内では，ターゲットの位置情報をドットで区切られた階層的表現方式で管理している．また，ターゲットの位置情報にへ以下の情報を付加して管理する．

- ターゲットの名前
- ターゲットの地理的位置情報
- ターゲットが現在存在する空間のサービス管理サーバへのポインタ
  - IPアドレス
  - ポート番号
  - グループ名
- 位置情報取得に用いられたデバイス名
  - GPS
  - PHSなど

4.8.3 検索

検索は，クライアントアプリケーションから位置情報管理サーバにへ行われる．位置情報管理サーバはターゲットの名前での検索を行い，ターゲット情報をクライアントアプリケーションに通知する．

4.9　本章のまとめ

本章では，FLUSHシステムの設計について述べた．FLUSHシステムはサービス管理機構と位置情報取得モジュール，位置情報管理サーバの三つから構成される．そして，サービス管理機構は複数のサービス管理サーバから成る．

サービス管理機構はサービスの登録や検索機能を提供する．サービス検索には単一のサービス管理サーバ内の検索を行うサーバ内検索と，複数のサービス管理サーバ間で検索を行うサーバ間検索に分けられる．そして，サービス管理サーバでは，空間同士の距離関係を把握しているので，自動的にターゲットに最も物理的に近接するサービスを特定できる．

位置情報取得モジュールはターゲットの地理的位置情報を取得する．そして，位置情報管理サーバでは，位置情報取得モジュールによって取得されたターゲットの情報を管理し，検索に応答する．

第 5 章

FLUSHシステムの実装

本章では，第4章で述べた設計をもとにFLUSHシステムの実装について述べる．

FLUSHシステムの実装環境やJiniとの親和性について述べる．また，個々のコードに触れながら，**サービス管理機構**と**位置情報管理サーバ**，**位置情報取得モジュール**の実装について述べる．章末のサンプルコードを通じて，サービスプロバイダとクライアントアプリケーションの実装方法について説明する．

5.1 実装環境

第4章でも述べたように，FLUSHシステムはサービス管理機構と位置情報管理サーバ，位置情報取得モジュールの3つから構成される．

また，FLUSHシステムの実装はJava言語を用いて行った．Java言語を使用することで，マルチプラットホーム性を確保した．コンパイラは，JDKバージョン1.2を使用した．実装環境は，表5.1に示す．

表 5.1: 実装環境

項目	**実装環境**
ハードウェア	Ultra-Enterprise 3000
オペレーティングシステム	SunOS 5.7
言語	JDK-1.2fcs-O
ライブラリ	Jini1.0.1

5.1.1 既存のJiniシステムとの共存

Jiniでは，ネットワーク上でサービスを提供するサービスプロバイダとそれを利用するユーザアプリケーションの集合体を**フェデレーション**と呼ぶ．Jiniシステムでは，このフェデレーション内でサービスを享受し合うことを想定している．FLUSHシステムでは，既存のJiniシステムとの共存関係を重視した設計を行った．つまり，既存のJiniのフェデレーションの中で，FLUSHサービスの提供と利用を可能とし，両者の相互互換性を確保した．これにより，既存のJiniシステムでもFLUSHサービスが利用でき，一方でFLUSHに対応しないサービスもFLUSHシステムの中で利用可能である．図5.1にFLUSHシステムとJiniシステムの関係を示す．

5.2 サービス管理機構の実装

サービス管理機構は，LocationHelper部とJiniのLookupサービスから構成される．

5.2.1 LocationHelper部の実装

LocationHelper部もJiniの一つのサービスとして実装する．LocationHelper部はJiniのHelper Service[11]として実装する．サービスとして提供される特定の機能をカプセル化したインタフェース，又はその集合体がHelper Serviceと定義される．そして，Helper ServiceはJiniのサービスの一つとしてLookupサービスに登録され，リモートホストで実行されるメソッドを持つ．

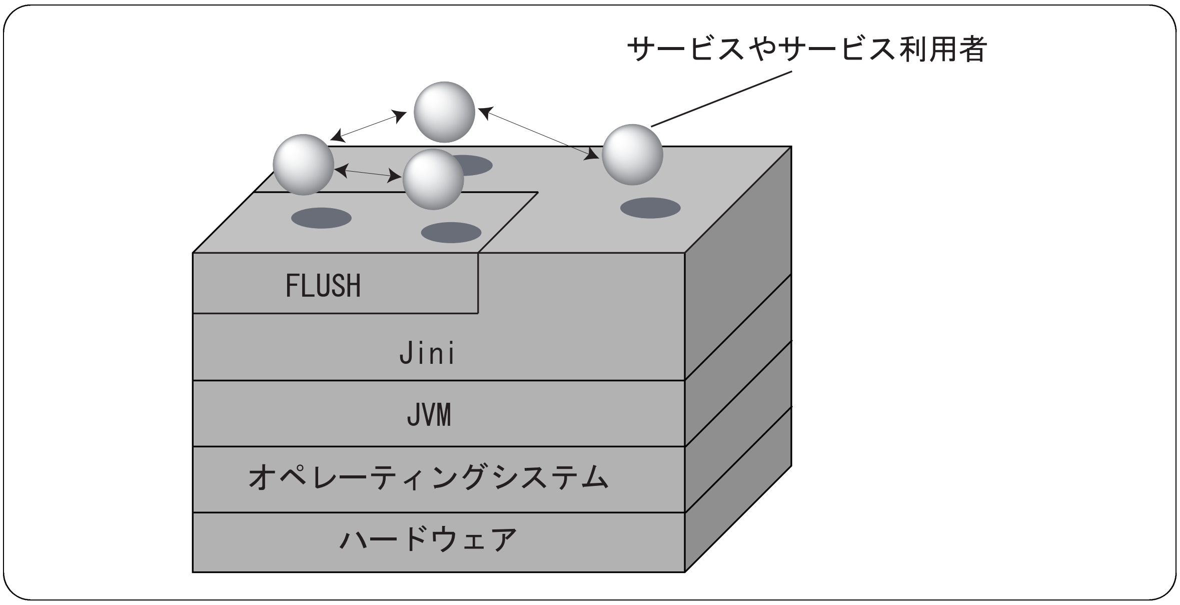

図 5.1: FLUSHシステムとJini

FLUSHシステムの位置情報による階層的検索機能を利用するためには，Lookupサービスごとに一つ以上のLocationHelperが登録されている必要がある．以下にLocation Helper部が提供する機能をあげる．

- Lookupサービスの基本位置情報を管理
- LookupサービスのIPアドレスとポート番号を管理
- 管理する空間の距離関係を評価
- Lookupサービスの位置情報を階層的にマッピング

LocationHelper部は，クライアントアプリケーションやサービスプロバイダの要求を受け，階層的な位置情報の検索機能をLookupサービスに提供する．また，Lookupサービスを階層的に結び付け，Lookupサービス越しの検索を実現する．このため，LocationHelper部は階層的に配置される．あるLocationHelper部に対して，階層的に上位にあるLocationHelperを**親LocationHelper**，階層的に下位にあるものを**子LocationHelper**と呼ぶ．

LocationHelper部はHelperインタフェースを実装する．また，FLUSHシステムのサービスプロバイダ，クライアントアプリケーションもHelperインタフェースを実装する．そして，LocationHelper部の機能はこのインタフェースを通して提供される．そして，LocationHelper部とサービスプロバイダやクライアントアプリケーションの通信はRMI (Remote Method Invocation)を用いて行われる．Helperインターフェースは，図5.2に示す．Helperインタフェースの具体的メッソドについて後述する．

次にLocationHelperが提供する個々の機能について述べる．

```
package jp.ac.keio.sfc.ht.akihiko;

import java.rmi.Remote;
import java.rmi.RemoteException;
import java.util.Vector;

import jp.ac.keio.sfc.ht.akihiko.LookupAddress;
import jp.ac.keio.sfc.ht.akihiko.CompareResult;

public interface Helper extends Remote{
  public void  setBaseLocation(LocationBaseLocation li)
    throws RemoteException;
  public String getBaseLocation() throws RemoteException;
  public void setLookupAddress(String base,LookupAddress la)
    throws RemoteException;
  public Vector getLookupAddresses(String baseLocation)
    throws RemoteException;
  public void setHelperAddress(String baseLocation,HelperAddress ha)
    throws RemoteException;
  public HelperAddress[] getHelperAddresses(String baseLocation)
    throws RemoteException;
  public CompareResult compareLocation(String location1,String location2)
    throws RemoteException;
}
```

図 5.2: Helperインタフェース

Lookupサービスの基本位置情報の管理機能

LocationHelper部はサービス管理サーバの基本位置情報を管理する．サービスの登録はこの基本位置情報に対しての相対位置情報を用いて行われる．一つのサービス管理サーバにおいて，登録されているサービスの位置情報に対して右端から最長一致する位置情報が，そのサービス管理サーバの基本位置情報となる．

```
  o.sfc.5322.endo.fujisawa.kanagawa.japan
2.i.sfc.5322.endo.fujisawa.kanagawa.japan
3.i.sfc.5322.endo.fujisawa.kanagawa.japan
```

図 5.3: サービスプロバイダの位置情報

例えば，図5.3のような位置情報を持つ空間を管理するLookupサービスは，表5.2のような基本位置情報を持つ．

基本位置情報は，LocationHelperがLookupサービスに登録される時に，管理者によってパラメータとして入力される．必要に応じて，以下のメソッドを利用して，変更が可能である．また，サービスプロバイダやクライアントアプリケーションからの

表 5.2: 基本位置情報の例

`sfc.5322.endo.fujisawa.kanagawa.japan`

基本位置情報の問い合わせも，以下のメソッドによって行われる．

基本位置情報を登録する際の引数や検索の際の返り値にはLocationInformationインスタンスが用いられる．LocationInformationクラスは，位置情報をピリオドで区切られた文字列で管理するほか，位置情報の階層の深さなどの情報も管理する．

```
public void  setBaseLocation(LocationBaseLocation li)
  throws RemoteException;
public String getBaseLocation() throws RemoteException;
```

LookupサービスのIPアドレスとポート番号の管理機能

LocationHelper部は，登録されているLookupサービスのIPアドレスとポート番号を管理している．これは，クライアントアプリケーションがLookupサービスとユニキャストで通信する場合に用いられる．通常のクライアントアプリケーションは，ネットワーク的に近接するLookupサービスをIPマルチキャストによって探し出す．しかし，遠隔に存在するサービス管理サーバとの通信はユニキャストを用いる．

IPマルチキャスト通信だけを用いると，IPマルチキャストを通さないルータが存在する場合，ルータの先のサービスプロバイダやサービス管理サーバと通信ができない．しかし，ユニキャストを用いることで，ルータの機能に左右されずにスケーラビリティを確保できる．

Lookupサービスの階層的な関連付けは，LocationHelper部によって行われる．

```
public void setLookupAddress(String base,LookupAddress la)
  throws RemoteException;
public Vector getLookupAddresses(String baseLocation)
  throws RemoteException;
```

具体的なLookupサービスのIPアドレスとポート番号の登録と問い合わせは，上のの二つのメソッドを用いて行われる．LookupサービスのIPアドレスとポート番号は，サービス管理サーバの基本位置情報に関連づけられている．この関連付けは，LocationHelper 部によって階層的に管理されている．これにより，クライアントアプリケーションは検索範囲に適合する基本位置情報を持つLookupサービスを指定できる．そして，Lookupサービスに登録されているLocationHelperサービスを利用できる．つまり，ある空間的範囲に存在する複数のサービス管理サーバに対して位置情報による階層的検索を要求できる．

空間の距離関係の評価機能

空間の距離関係の評価機能は，LocationHelper部が提供する機能の中で最も重要である．第3章で述べたように，位置情報によってサービスを階層的に管理するだけでは，ターゲットに物理的に近接するサービスを一つ特定にすることが困難である．同じ階層に二つ以上のサービスが見つかった場合に，どちらのサービスがよりターゲットに近接なサービスであるかを判定する場合，空間同士の距離関係をシステムが把握している必要がある．あるLookupサービスが管理する空間内，つまり基本位置情報で表現される空間内で，個々の空間同士の距離関係を評価するのがLocationHelper部である．LocationHelper部間の距離関係，言いかえると基本位置情報同士の空間的距離は，それらLocationHelper部の親LocationHelperが管理する．これにより，LocationHelper部は自分の管理する空間の距離関係だけを関知すれば良い．

空間同士の距離関係は，相対的な重み付けで行われる．空間同士の距離関係は，**階層距離**と**相対距離**の二つによって評価される．

表 5.3: ある空間における階層距離の例

対　象	**位 置 情 報**	**階層距離**
ターゲット	i208.2.i.sfc.5322.endo.fujisawa.kanagawa.japan	-
サービスA	i203.2.i.sfc.5322.endo.fujisawa.kanagawa.japan	8
サービスB	o308.3.o.sfc.5322.endo.fujisawa.kanagawa.japan	6

階層距離とは，ターゲットとサービスがツリーの上からどの階層までマッチしたかを示す．表5.3に階層距離の例を示す．通常，階層的に管理される情報をツリーの下から検索する場合，検索結果が発見され次第検索は終了するので，階層距離を考慮する必要はない．しかし，FLUSHシステムでは，あるサービス管理サーバに対してツリーを下から検索し，最初に見つかったサービスがターゲットに最も近接するサービスとは断定できない．

通常，JiniではIPマルチキャストを用いてLookupサービスの発見やサービスの管理が行われている．IPマルチキャストに対応しないルータで区切られたネットワークを有する空間では，その区切られたネットワークごとにLookupサービスが必要となる．そのような空間では，一つのLookupサービスの中で，ツリー上でターゲットに対して近接するサービスがその空間で物理的に最も近接するサービスであるとは言えない．

個々のLocationHelperで最もターゲットに近接するサービスの候補を一つずつをあげる．その後それら候補の中で階層距離を評価し，階層距離が一致している場合のみ，相対距離の評価を行う．

個々のサービス管理サーバでのサーバ内検索は図4.4のフローチャートのように行われる．サーバ内検索はツリーの下から行い，サービスが見つかった時点で，そのLocationHelperでの検索は終了する．同じ階層で，二つ以上のサービスが見つかった場合のみ，空間同士の距離評価を行う．

空間同士の距離評価は，ツリーの同じ階層のみで行われる．空間同士の相対的位置関係は，一対一対応の表で表現できる．例えば，図4.6の地図で示したような空間は表4.5のようになる．

LocationHelper部の起動時に相対距離を記述した設定ファイルを読み込む．この設定ファイルは，図5.4のフォーマットで記述され，LocationHelper部内で二次元配列に変換され，空間同士の相対的距離関係の評価に用いられる．先ほどの空間を設定ファイルのフォーマットに変換すると以下のようになる．

```
[1.k]
#コメント
#κ棟　1階
(k11,k12,k13,k14,k15,k16,k17,k18)
0,1,3,4,5,6,7,4,
1,0,2,3,4,4,4,4,
3,2,0,1,2,3,4,2,
4,3,1,0,1,2,3,3,
5,4,2,1,0,1,2,2,
6,4,3,2,1,0,1,1,
7,4,4,3,2,1,0,1,
4,4,2,3,2,1,1,0,
```

図 5.4: 位置情報設定ファイルの例

ファイルから取り込まれた空間同士の相対的距離関係はLocationHelperの以下のメソッドで取得可能である．このメソッドの返り値はCompareResultのインスタンスである．このインスタンスは階層距離と相対距離の値を含んでいる．

```
public CompareResult compareLocation(String location1,String location2)
  throws RemoteException;
```

Lookupサービスの階層的マッピング

LocationHelper部は位置情報に基づいて，各LocationHelper部の階層的な位置関係を管理している．また前項で述べたように，LocationHelper部はLookupサービスのアドレスを管理している．これにより，LocationHelper部は位置情報を鍵として階層的にLookupサーバを指定できる．このような機構は，クライアントがRMIを用いてLocationHelper部内のメソッドを遠隔ホストから呼び出すことで実現する．

```
public void setHelperAddress(String baseLocation,HelperAddress ha)
  throws RemoteException;
public HelperAddress[] getHelperAddresses(String baseLocation)
  throws RemoteException;
```

これらのメソッドを用いてLocationHelper部のツリーの上下関係が構築される．また，getHelperAddress()を用いることで，LocationHelper部で構成されたツリーの中から，検索の必要のあるLocationHelper部のIPアドレスを特定できる．

5.3　位置情報取得モジュールの実装

位置情報取得モジュールは，大きく分けて，3つの部分からなる．

- デバイスによる地理的位置情報の取得
- ターゲットの周辺のサービス管理サーバの取得
- ターゲットの情報の位置情報管理サーバへの送信

第4章で述べたように，ターゲットの位置情報はターゲット自身によって取得される位置情報の能動的取得とターゲットを取り巻く環境によって取得される位置情報の受動的取得の二通りが存在する．

この取得方法の違いによって，位置情報取得モジュールの実行場所が変わる．ターゲットが能動的に位置情報を取得する場合，位置情報取得モージュールはターゲットが持つ移動計算機上で実行される．一方，ターゲットが位置情報を受動的に取得する場合，位置情報取得モジュールはターゲット自身ではなく，ターゲットの周辺環境に存在する．ICカードによってターゲットの位置情報を把握する場合では，位置情報取得モジュールはカードリーダのつながる計算機上に存在する．しかし，ターゲットの地理的位置情報とサービス管理サーバのネットワーク上の接続位置が位置情報管理サーバに送信されるまでの処理は，取得の方法に無関係である．

5.3.1　地理的位置情報の取得

位置情報をターゲットが能動的に取得するケースでは，ターゲット自体が位置情報取得デバイスを持つ．具体的には，GPSやPHSなどの機器によって地理的位置情報が取得される．そして，この情報を取得する多くの部分の実装は，デバイスに依存する．よって，最終的にドットで区切られた文字列表記の位置情報に変換され位置情報管理サーバに通知されたことだけが保証されれば良い．

5.3.2　サービス管理サーバの取得

サービス管理サーバの取得は，以下の流れで実現される．

1：Discoveryプロトコルによる Lookupサービスの取得 位置情報取得モジュールは，Jiniの Discoveryプロトコルに従って，Lookupサービスを発見する．具体的には，IPマルチキャストを使いネットワーク上に位置情報取得モジュールのIPアドレスが入ったパケットを送信する．このパケットを受信したLookupサービスは応答メッセージを位置情報取得モジュールに送信する．以上のようなやり取りを通じて，位置情報取得モジュールはLookupサービスの接続位置を感知する．

2：LocationHelperサービスの取得 位置情報取得モジュールは，取得したLookupサービスにLocationHelper サービスの検索要求を出し，LocationHelperサービスを発見する．

3：サービス管理サーバのアドレスの取得 LocationHelperサービスの以下のメソッドを呼ぶことで，位置情報取得モジュールは，ターゲットからIPマルチキャストの届く範囲にあるサービス管理サーバのネットワーク上の接続位置を把握する．

```
public Vector getLookupAddresses(LocationInformation li)
      throws RemoteException;
```

5.3.3 位置情報管理サーバへの送信

以上の手続きを経てターゲットの情報は取得される．具体的には表5.4のようなターゲット情報が存在する．これらは位置情報管理サーバに送信される．実際のターゲット情報の位置情報管理サーバへの送信方法は，5.4.3で述べる．

表 5.4: ターゲットに関する情報の表記例

項目	使用例
ターゲットの名前	akihiko@hostA.ht.sfc.keio.ac.jp
ターゲットの地理的位置情報	i.sfc.5322.endo.fujisawa.kanagawa.japan
サービス管理サーバの接続位置	hostB.ht.sfc.keio.ac.jp:12345
デバイス名	`GPS_Eathmate`

5.4 位置情報管理サーバの実装

5.4.1 位置情報管理サーバの機能

位置情報管理サーバは，以下の二つの機能を実現する．

- ターゲット情報の収集

- ターゲット情報の返答

5.4.2 PostgreSQL

位置情報管理サーバ本体はリレーショナルデータベースである．位置情報取得モジュールやクライアントアプリケーションの位置情報管理サーバとの通信は，JDBC[12]を通して行われる．JDBCはJava言語で記述されたアプリケーションがリレーショナルデータベースにアクセスするために用意されたJavaのクラスの集合である．

今回の実装ではリレーショナルデータベースとしてPostgreSQL を用いた．

5.4.3 ターゲット情報の収集

位置情報取得モジュールによって取得されたターゲットの情報は以下のメソッドを呼ぶことによって，位置情報管理サーバ内に格納される．

```
public int upDate(String name,Taget info) throws SQLException{}
```

また，位置情報管理サーバは表5.4.3のような形態でターゲットの情報を格納している．

表 5.5: 位置情報管理サーバ内情報の例

Name	Location	Address	Port	Group	Device
akihiko	sfc.5322.endo.fujisawa.kanagawa.japan	reach.ht.sfc.keio.ac.jp	12345	student	Earthmate
kusumoto	7.7.7.nishishinjuku.shinjuku.tokyo.japan	pinfu.kusumoto.org	34565	faculty	PHS_personal
-	-	-	-	-	-

5.4.4 ターゲット情報の返答

位置情報管理サーバはクライアントアプリケーションのターゲット情報検索要求に，ターゲット情報を返答する．ターゲット情報の検索は，ターゲットの名前を鍵として行われる．

そして，クライアントアプリケーションがPostgreSQLサーバ内のターゲット情報を名前によって検索する例を図5.5に示す.

5.5 実装例

実際のFLUSHシステム上では，サービスプロバイダの提供したサービスをクライアントアプリケーションが検索し，利用する．この節では，実際のサービスプロバイ

```
    try{
      //ホスト名とグループ名を指定してPostgreSQLサーバへの
      //コネクションを確立する.
      LocationPostGres sql
= new LocationPostGres("reach.ht.sfc.keio.ac.jp","student");

      //名前を指定しターゲットの位置情報を取得する.
      String location = sql.getLocationByName("akihiko");

    } catch(Exception ex) {
      System.err.println("Exception caught.\n"+ex);
    }
```

図 5.5: PostgreSQLサーバへの検索要求例

ダとクライアントアプリケーションの実装例を通して，サービスの登録と検索，利用の方法を説明する．

5.5.1 サービスプロバイダ

以下のソースコードは，既存のJiniのサービスプロバイダのプログラムに変更を加え，FLUSHシステムに対応するように修正した．

変更が加えられた部分は28～29行目と31～32行目そして，45～46行目である．28～29行目にかけて，ネットワーク上のLookupサービスからLocationHelperサービスを取得する．そして，取得したLocationHelperサービスを利用してLookupサービスの基本位置情報を得る．31～32行目にかけて，取得した基本位置情報を引数として与えられたサービスプロバイダのLookupサービスへの相対位置情報を結合し，絶対位置情報を作成する．そして，45行目で位置情報のテンプレートを作成した後，既存の属性情報を合わせたLocationTemplateインスタンスを作成する．以上がFLUSHシステム固有の処理である．

そして，既存のJiniサービスプロバイダと同様に，属性とクラスの型，サービスIDなどの情報を含んだServiceItemという形に結合された後，51行目でサービスがLookupサービスに登録される．

```
public Helper getHelperService(ServiceRegistrar regs)
```

このメソッドはnet.jini.core.lookup.ServiceRegistrarを引数にとり，このServiceRegistrarインスタンスからLookupサービスを取り出す．その後，取り出したLookupサー

ビスに登録されているLocationHelperサービスを取得する．

```
public String getBaseLocation()
```

getHelperService()メソッドで取り出したLocationHelperのメソッドで，Lookupサービスの基本位置情報を取得する．

```
public String append(String relativeLocation, String baseLocation);
```

文字列型で表された位置情報を引数に取り，二つの位置情報を連結する．このメソッドを用いて，相対位置情報と基本位置情報から絶対位置情報を作成する．

```
public Entry[]  getTemplate(Entry[] attribute);
```

net.jini.core.entry.Entryの配列を引数にとる．そして，それらの属性情報に階層的位置情報を付加したものを返す。このメソッドにより，既存のフラットな名前空間で表現される属性情報と階層的位置情報を併せ持った属性情報でサービスを登録できる．

5.5.2 クライアントアプリケーション

クライアントアプリケーションも既存のJiniクライアントのコードを3箇所修正するだけで，FLUSHシステムに対応できる．

図5.7に示すように28～29行目でLocationHelperサービスを取得する．取得したLocationHelperサービスと連携して位置情報によるサービスの階層的検索を行う．ソースコードの34行目が，LocatoinHelperサービスへの検索依頼の窓口になるLocationRegistrarの作成過程である．そして，39行目において，実際に検索要求をLocationHelperに送る．そして，返ってきたサービスのメソッドをリモートで呼んでいる部分が40行目の処理である．

以下に示すLocationRegistrarクラスのメソッドがLocatoinHelperサービスと連携し，ターゲットに物理的に最も近接するサービスをネットワーク上から取得してくる．

public LocationRegistrar LocationRegistrar(Helper helper,String location,int depth);

LocationRegistrarインスタンスを生成するコンストラクタを上に示す．第一引数に

検索を依頼するLocationHelperサービスを指定する．第二引数にターゲットの位置情報を絶対位置情報で指定する．この際，位置情報取得モジュールから取得できる位置情報が相対位置情報のみであった場合は，LocationHelperから基本位置情報を取得し，絶対位置情報を作成する必要がある．そして，第三引数では検索階層を指定する．このような情報を基にLocationHelperとの検索の窓口の役割を果たすLocationRegistrarを作成する．

```
public Object lookup(ServiceTemplate tmpl)
```

上に示したlookupメソッドが実際の位置情報に基づいたサービスの階層的検索を行うメソッドである．位置情報以外の属性情報を引数に取り，この属性にマッチして，ターゲットに近接するサービスを一つ返す．サーバ間検索，つまり複数のLookupサービスにまたがった検索は，コンストラクタで指定した位置情報と検索範囲を基に，必要に応じて行われる．

5.6 本章のまとめ

本章では，FLUSHシステムの実装について述べた．FLUSHシステムはサービス管理機構と位置情報取得モジュール，位置情報管理サーバの三つから構成される．サービス管理機構でのサービス登録やサービスの位置情報による検索の方法や実装について説明した．また，位置情報取得モジュールと位置情報管理サーバの機能や実装方法について述べた．そして，サービスプロバイダとクライアントアプリケーションの実装例を通して，FLUSHシステム内のサービス提供と物理的位置に基づいたサービスの検索方法について説明した．

本章で述べた実装によって，FLUSHシステムではクライアントアプリケーションはターゲットの位置情報と検索範囲を指定するだけで，ターゲットに物理的に最も近接するサービスを一つ特定できる．

```
public class EntrySample extends UnicastRemoteObject
  implements Service, DiscoveryListener, Serializable {
    ...
    public Job jobs()  throws RemoteException {
      ...
    }
    public static void main(String argv[]){
      ...
      thisObj = new EntrySample();
      lookupDiscovery = new LookupDiscovery(groups);
      lookupDiscovery.addDiscoveryListener( thisObj );
      ...
    }
    public void discovered(DiscoveryEvent evnt) {
      thisObj . new registerThread(evnt).start();
    }
    public void discarded(DiscoveryEvent evnt) {
      ServiceRegistrar[] regs = evnt.getRegistrars();
    }
    public class registerThread extends Thread {
      ...
      public void run() {
        ServiceRegistrar[] regs = evnt.getRegistrars();
        StringTool st = new StringTool();

        for(int i=0; i<regs.length;i++){

          GetHelper gt = new GetHelper();                // 変更箇所
          Helper helper = gt.getHelperService(regs[i]);  // 変更箇所
          try{
            String base =  helper.getBaseLocation();     // 変更箇所
            location = st.append(argv[1],base);          // 変更箇所

          }catch(RemoteException e){
            System.err.print(e);
          }

          Entry[] attrs = new Entry{
            new ServiceInfo("EntryService","KEIO","KMSF",
                             "1.0","akihiko",""),
             new BasicServiceType("EntrySample");
             new LowerLocation(location);};

           LocationTemplate lt =
             new LocationTemplate(location);              // 変更箇所
           attrLocation = lt.getTemplate(attrs);          // 変更箇所

           ServiceItem sit = new ServiceItem(null,thisObj,
                                             attrLocaion);
           try {
             sre = regs[i].register( sit, Lease.FOREVER);
```

図 5.6: サービスプロバイダのサンプルコード

```
public class GetService implements DiscoveryListener {
  ...
    public static void main(String argv[]){
      ...
      location = argv[0];
      depth = Integer.parseInt(argv[1]);
      ...
      thisObj = new GetService();
      lookupDiscovery.addDiscoveryListener( thisObj );
      ...
    }

  public void discovered(DiscoveryEvent evnt) {
    thisObj . new lookupThread(evnt).start();
  }

  public void discarded(DiscoveryEvent evnt) {
    ...
  }

  public class lookupThread extends Thread {
    ...
    LookupAddress[] la;
    public void run() {
      ServiceRegistrar[] regs = evnt.getRegistrars();
      for(int i=0; i<regs.length;i++){

        GetHelper gt = new GetHelper();                  //変更箇所
        Helper helper = gt.getHelperService(regs[i]);     //変更箇所

        try{
          ...
          LocationRegistrar lr
          = new LocationRegistrar(helper,location,depth); //変更箇所

          ServiceTemplate tmpl
            = new ServiceTemplate(null,classes,null);

          Service service = (Service)lr.lookup(tmpl);     //変更箇所
          if(service!=null) value = service.jobs();
```

図 5.7: クライアントアプリケーションのサープルコード

第 6 章

評価と考察

本章では，FLUSH システムが定量的評価と定性的評価を行う．まず，サーバ内でのサービス検索処理に要する時間の最大値を測定し，諸条件との相関関係を導く．そして，サービス検索時間の増加を制限できることを述べる．その後 FLUSH システムの定性的評価を行い，システムの有用性について述べる．

6.1 定量的評価

本節では，サーバ内検索の所要時間に関する定量的評価を行った．サーバ内検索はツリーを用いた検索と検索結果の距離比較からなるため，定量的評価もツリーの検索と空間同士の距離評価の両方について行った．

6.1.1 測定環境

表6.1に示すような環境の基で定量的評価を行った．

表 6.1: 測定環境

筐体	**CPU**	**メモリ**	**OS**
Sun Ultra 30	UltraSPARC-II 248 MHz	128 MB	SunOS 5.6

6.1.2 ツリー内検索所要時間

FLUSHシステムでは，4.6.3項に示したように検索を複数回繰り返す方法でツリー内のサービス検索を行う．ここで1回の検索に要する時間をt_0とし，ツリーの深さをd，検索準備のオーバーヘッドをaとする．1回ごとの検索はハッシュ関数を基に求められるので，本システムではt_0の値は一定とみなせる．したがって，ツリー検索に要する時間のベストケースとワーストケースは以下の式によって表現できる．

$$t_0 + a \quad \leq \quad t \quad \leq \quad t_0 \cdot d + a \tag{6.1}$$

式(6.1)の左辺は，1回目で検索が成功する場合である．また，右辺はツリーの頂点まで検索が行われる場合である．

表6.2に，実際にツリー検索に要する時間を測定した値を示す．表6.2より，検索する階層が深くなるに従って所要時間が増加することが分かる．表6.2を基に作成したグラフを図6.1に示す．

表 6.2: ツリー検索の所要時間

検索階層	1	2	3	4	5	6	7	8	9	10
所要時間（ミリ秒）	122.01	134.31	151.67	165.18	181.24	194.20	204.56	222.38	236.43	249.03

図6.1において，横軸は検索する階層数を示し，縦軸はワーストケースの所要時間を示す．また，点線で示される直線は，測定結果より算出した線形回帰直線である．この線形回帰直線を式6.2に示す．

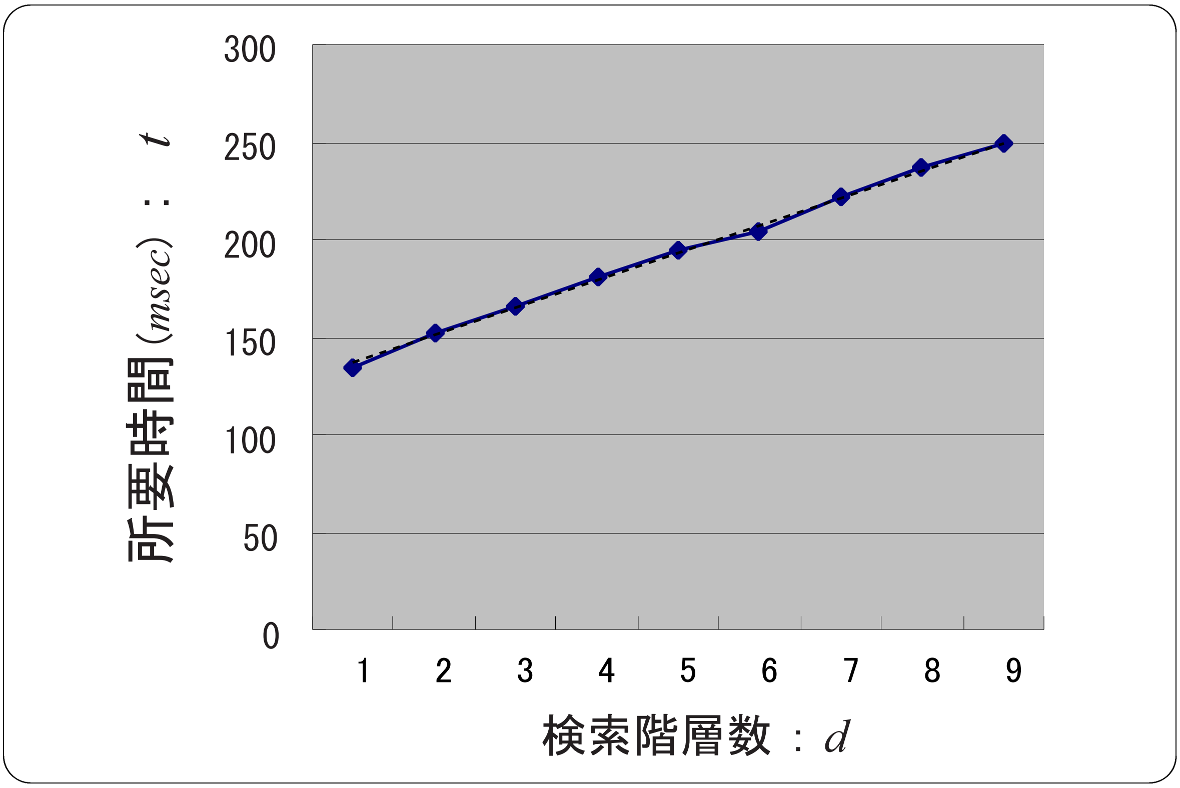

図 6.1: ツリー検索の所要時間

$$t = 14.18 \cdot d + 122.32 \tag{6.2}$$

式 6.2 より，一回の検索に要する時間 t_0 が 14.18 ミリ秒であり，検索の前処理によって生じるオーバヘッドを示す a の値が 122.32 ミリ秒であることが分かる．そして，Jini の Lookup サービスで同じ個数のサービスを登録し，文字列による完全一致検索を行った場合，1 回の検索時間は 14.55 ミリ秒である．このことから，FLUSH システムにおいて，1 回の検索時間は Jini の Lookup サービスの検索処理時間が大部分を占めていると言える．

そして，個々の検索においてはハッシュ関数を基にサービスの検索を行っているため，検索時間は登録サービス数に影響を受けない．そして，式 6.2 からも分かるように検索処理時間のワーストケースは，ツリーを構成する階層数に比例する．しかし，4.5.2 項で述べたように，FLUSH システムは相対位置情報と基本位置情報の二つに位置情報を分割して表記する．サーバ内検索では，相対位置情報によってツリーが構成される．したがって，位置情報の分割表記を行うことによって，過度な階層の増大を防げる．

6.1.3　距離評価所要時間

4.6.4 節に示したように，距離関係の評価は二つの過程に分けられる．まず，どの空間が比較対象の距離関係を把握しているのかを特定する．そして，その空間に比較対

象の距離関係を検索する．

測定もこの二つの過程に分けて行った．まず，距離関係の把握先の検出についての測定である．空間同士の距離関係はツリーの節の部分が把握している．図4.3の場合，空間Aと空間B，空間Fが把握している．これらをツリー上から見つけ出す作業を測定した．測定は，ある個数の空間からツリー上を検索したときの，ワーストケースをそれぞれ測定した．その結果を表6.3に示す．

表 6.3: 距離関係の把握先の検出

空間の数	10^2	10^3	10^4	10^5	10^6
所要時間（ミリ秒）	0.60	2.45	20.83	210.42	2022.71

また，表6.3を基に作成したグラフを図6.2に示す．横軸は空間の個数nを示し，縦軸は所要時間tを示す．また，点線で示される直線は，測定結果より算出した線形回帰直線である．

$$t = 0.002 \cdot n + 2.30 \tag{6.3}$$

式6.3に示す．このことから，ツリー上から空間の距離関係を把握している節を検出する時間のワーストケースは，ツリー上の節の数に比例する事がわかる．しかし，サーバ内検索と同様に位置情報を分割表記することで，あるサーバ内に存在する空間の個数を分割できる．これにより，ツリー上の節の数は制限できる．

次に，特定した節から二つの空間の比較を行う．測定はあるツリー上の節がn個の空間の距離関係を把握している場合の検索にかかる時間のワーストケースを所要時間tとして測定した．その結果を表6.4に示す．

表 6.4: 距離関係の計算

空間の数	10^2	10^3	10^4	10^5	10^6
所要時間（ミリ秒）	0.86	6.48	48.41	599.56	6012.88

$$t = 0.006 \cdot n - 3.28 \tag{6.4}$$

表6.4を基に作成したグラフを図に示す．点線で示される直線は結果を基に算出した線形回帰直線である．その式を式6.4に示す．このことから，ツリー上のある節から二空間の距離関係を計算する処理時間のワーストケースも節が管理する空間の個数に比例することが分かる．そして，大抵の場合，ある節が管理する空間の個数は数個から数百程度である．通常，一つのビル内に存在するフロアは高々百程度であり，数百

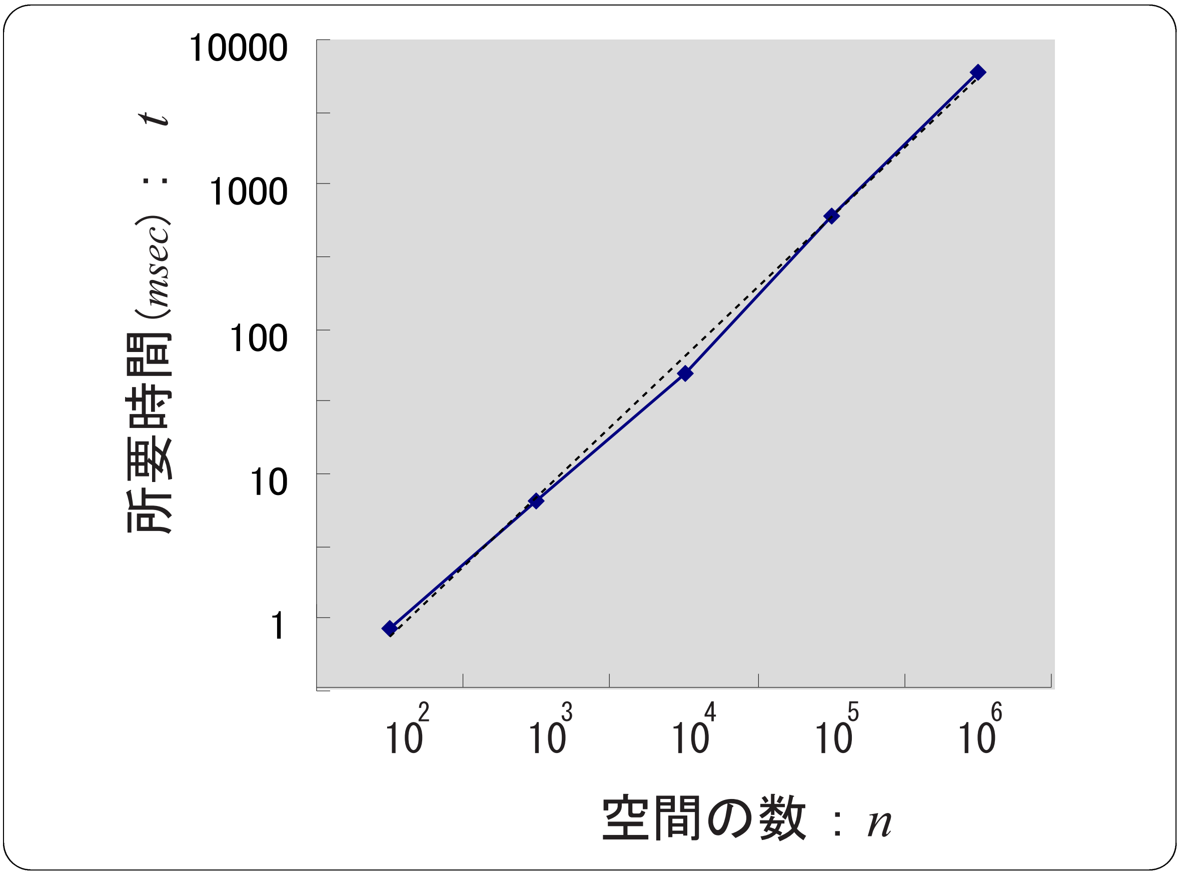

図 6.2: 距離関係の把握先の検出

の部屋が存在するフロアがある可能性は低い．たとえ，そのような状況が存在しても，節の数を増やし，分割して管理することで対応が可能である．

そして，以上のことから，一階層に m 個の空間が存在し，n 個の節からなるツリーからサービスとターゲットの距離関係を計算する時間のワーストケースは式6.5で表現できる．

$$t = (0.002 \cdot n + 2.30) + (0.006 \cdot m - 3.28) = 0.002 \cdot n + 0.006 \cdot m - 0.98 \qquad (6.5)$$

そして，以上の測定結果からサービス検索にかかる時間は，空間の数と空間が構成する階層の深さに比例する．だが，本システムでは相対位置情報と基本位置情報による位置情報の分割表記をとっている．これによって，一つのサービス管理サーバに存在する階層数と空間の数を分散させることができる．したがって，階層数と空間の数を抑えることができ，単一サーバ内の検索時間の増大も防げる．

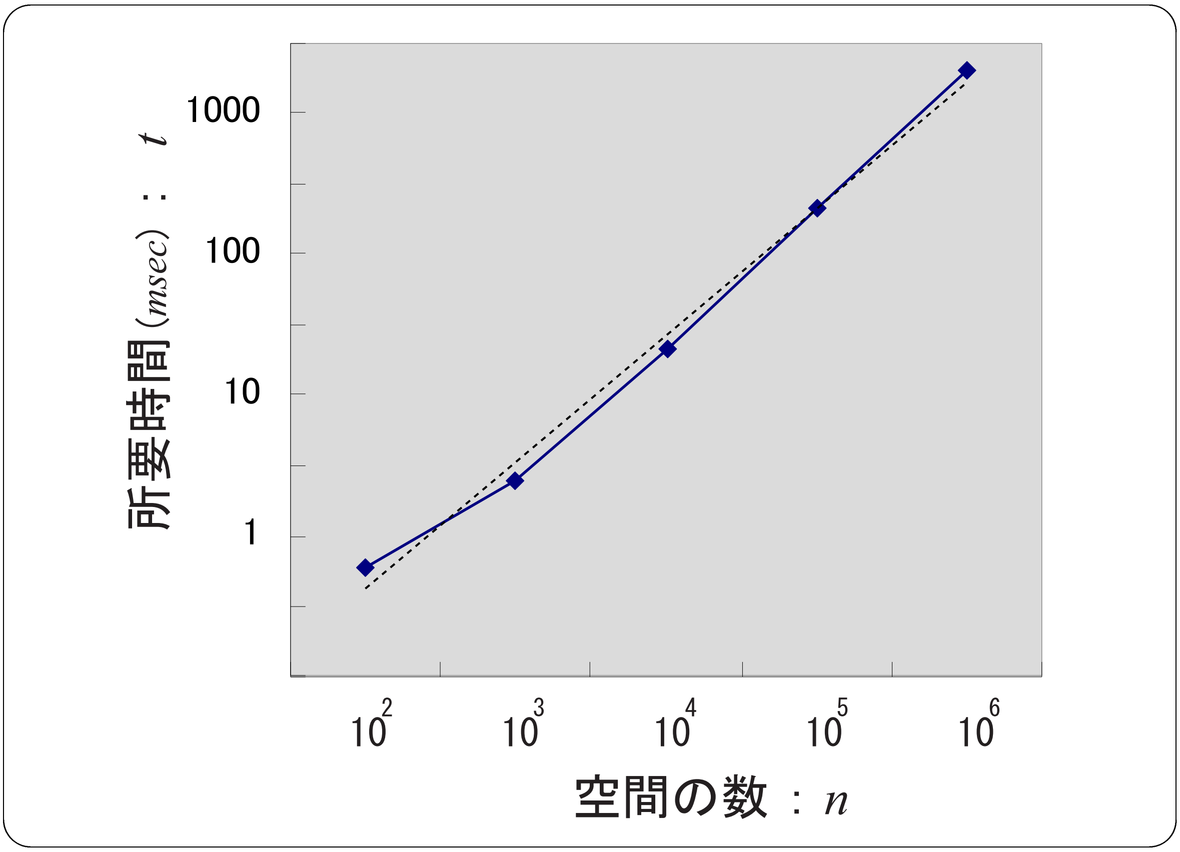

図 6.3: ツリー検索の所要時間

6.2 定性的評価

この節では，FLUSHシステムの定性的評価を行い，システムの有用性について論じる．

6.2.1 既存システムとの親和性

FLUSHシステムは，既存のJiniシステムに対して，サービスの階層的検索と空間同士の相対的距離関係の評価を付加した．しかも，既存のJiniシステムとの親和性を確保し，FLUSH未対応のシステムにFLUSHのサービスを登録できる．また，逆に階層的位置情報を属性として持たない既存のJiniサービスもFLUSH システムのサービス管理サーバに登録可能である．したがって，FLUSHシステムは既存のJiniシステムとの相互乗り入れを実現したシステムといえる．

また，既存のJiniのサービスプロバイダとクライアントに対して，最小限の変更を加えるだけで，FLUSHシステムが提供する位置情報によるサービスの階層的検索を提供できる．具体的には，サービスプロバイダとクライアントの両方にFLUSHのコードを3ヶ所挿入するだけである．これは，プログラマにとって，過剰な負担にはならない

といえる．

6.2.2　空間同士の距離関係の把握

二点間の直線距離によって求められたターゲットとサービスの距離と，ターゲットのサービスまでの移動距離とは必ずしも一致しない．サービスを位置情報によって検索する際に重要となるのは，単純な二点間距離ではなくターゲットがサービスまでの移動距離である．FLUSH システムでは，サービスが存在する空間同士の距離関係を相対的に重み付けしている．したがって，空間と空間の間に壁などの遮蔽物が存在した場合でも，ターゲットのサービスまでの移動距離を基にサービスの検索が可能である．

6.2.3　検索の自動化

FLUSH システムでは，位置情報に基づき，ターゲットに物理的に最も近接するサービスを検索する．ユーザは「ターゲットに近接するサービス」といった指定をするだけで，FLUSH システムが，自動的にターゲットに物理的に最も近接するサービスを検索し，ユーザに返す．これにより，ユーザは，位置情報によるサービスの検索ができるが，実際に位置情報を意識する必要はない．

位置情報による検索を提供する多くのシステムは，位置情報が完全に一致していないと検索できなかったり，最終的に候補の中からユーザに選択を要求したりする．しかし，FLUSH システムでは，前述のようにシステムが空間同士の距離関係を把握しているため，ユーザから検索要求を受けその返答をユーザに返すまで，ユーザの手を一切借りることなく，一つのサービスを特定できる．

6.2.4　相対位置情報利用の有効性

FLUSH システムでは，サービスプロバイダが持つ位置情報を相対位置情報と基本位置情報の二つに分割した．これにより，サービスプロバイダを登録する際には，サービス管理サーバの相対位置情報だけを意識すれば良く，サービスを特定する際に生じる名称の不統一といった問題を軽減する．一方で，システムが持つ基本位置情報と相対位置情報から絶対位置情報を作成することで，検索のスケーラビリティを確保した．

6.3　本章のまとめ

本章では，定量的評価と定性的評価について述べた．

定量的評価に関しては，サービス検索の所要時間について定量的評価を行った．単一のサービス管理サーバ内のサービス検索時間を測定し，その結果から検索条件と所要時間の因果関係を導いた．その結果，階層的に管理される位置情報の階層数と検索に要する時間の最大値が比例することが分かった．また，サービスが存在する空間の数と検索処理時間の最大値にも比例関係が存在する．本システムでは，相対位置情報

と基本位置情報による位置情報の分割表記をとっている．これによって，一つのサービス管理サーバに存在する階層数と空間の数を分散させることができる．したがって，階層数と空間の数を抑えることができ，単一サーバ内の検索時間の増大も防げる．

定性的評価においては，サービスの自動検出を中心に定性的評価を行った．FLUSHシステムでは，位置情報をツリー構造によって管理するだけでは特定できない空間同士の相対的な距離関係を把握している．したがって，ユーザに候補の中からの選択を要求したり，位置情報が完全に一致している場合のみの検索が成功する，といったことはない．また，ターゲットの位置情報をユーザが明示せずに，ターゲットの空間に近接するサービスを検索できる．以上のことから，FLUSHシステムでは既存のシステムには無い，位置情報によるサービス検索の柔軟性を確保し，それらの自動化を実現したといえる．

第 7 章

終わりに

7.1 今後の課題

この節では，現在のFLUSHシステムが抱える問題点について述べる．そして，その問題点に対する解決策について述べながら，今後のFLUSHシステムの拡張の方向性を示す.

7.1.1 サービスプロバイダの移動

今回，FLUSHシステムの実装においては，ターゲットやユーザの移動に対しては，システムが感知し，自動的にターゲットに最も近接するサービスを特定できた．しかし，サービスプロバイダがサービスを提供しながら空間を移動する場合は現在の実装では対応できない．例えば，移動計算機のディスプレイがサービスを提供しながら，別の部屋へ移動した場合，移動後のディスプレイの位置情報をサービス管理機構に反映できない.

位置適応的なシステムを構築するためには，サービスプロバイダの移動に対応する機構が必要である.

7.1.2 基本位置情報の変更

現在，位置情報は基本位置情報と相対位置情報に分けて管理されており，サービスプロバイダはサービス管理サーバの相対位置情報だけを関知すればよい．つまり，グローバルな位置情報はサービス管理サーバだけで扱われている.

このような機構をとった事によりサービス管理サーバを含む空間全体が移動した場合，サービス管理サーバの基本位置情報を変更するだけで，移動先でサービスを提供できる.

例えば，バスや船などの内部にサービス管理サーバとサービスプロバイダが存在し，移動する場合，船上のサービスが移動先の港でサービスを提供し，検索機能を提供できる.

このような機構をとることでFLUSHシステムは，サービス管理サーバとサービスプロバイダの相対的な位置関係が保持されたまま空間が移動し，移動先でサービスを提供し，物理的位置情報によるサービス検索を提供できる．

このように動的にサービス管理サーバとそのサーバが管理するサービスが，空間と共に移動する場合を今後はサポートする．

7.1.3　距離の抽象化

FLUSHシステムでは，同じ階層に二つ以上のサービスが存在する場合にのみ空間同士の距離関係を評価する．位置情報によって作られた階層的ツリー構造で，あるターゲットに対して同じ階層に存在するサービスは，あるターゲットに対して階層的に一つ上の階層に存在するサービスより必ず物理的に近接したサービスとして認識される．

しかし，実世界において同じ階に存在するプリンタより，一つ下の階のプリンタの方が近いケースが存在する．このように，現在は階層的ツリー構造の横の重み付けより，階層的な上下の重み付けの方を先に評価する設計になっている．しかし，実世界には前述するような問題が存在するため，今後階層的な上下の重み付けと横の重み付けを一つのパラメータで表現できるように抽象化する．また，階段や扉などの空間への入り口をツリー上で表現する．

7.1.4　ユーザインタフェースの向上

現在のFLUSHシステムでは，ドットで区切られた位置情報の表記を用いている．この様な表記はシステムがサービスを管理する点においては優れている．しかし，人間が実際にこの表記を用いてサービスを登録することは，必ずしも容易とは言えない．そこで，サービスを登録する人が位置情報の文字列表記を用いずにサービス登録が行えるようなGUI(Graphical User Interface)を構築する．これによって，ディスプレイ内に表示された地図を用い直感的なサービス登録が可能となる．

7.2　まとめ

本論文では，まずネットワーク上で提供されるサービスを管理し，検索する際の物理的位置情報の有用性について論じた．同時に，ネットワーク上のサービスを使用していることをユーザに対して，強く意識させないためには，システムがサービスを自動的に検出する必要がある．

ネットワーク上のサービス管理や検索を実現する先行システムとして，Jini, SLP, SDSについて比較した．そして，位置情報は他のサービスに関する属性情報とは異なり，階層的構造を成すことについて述べた．物理的位置情報によってサービス検索を行う場合，属性情報に階層的名前空間が必要である．

また，物理的位置情報を利用した先行システムとして，SDS, GLI, Active Batについて考察した．先行システムでは，サービス同士の距離関係をツリー構造でしか管理

していない．より厳密なサービスの特定を行う場合は，ユーザの手によってそれらのサービスの中から選択をする必要がある．そして，より厳密なサービスを自動的に検出するためには，システムが空間同士の相対的な距離関係を把握している必要がある．空間同士の距離は，二点間の直線距離と移動を考慮した距離の二つが考えらる．ネットワーク上で位置情報によってサービス検索を行う場合，壁などの遮蔽物を考慮し，サービスまでの直線距離より移動距離が重要である．

以上のような問題点を解決するために，FLUSHシステムを提案した．また，サービスの利用者をユーザ，位置情報による検索の基準となるものをターゲットと定義した．そして，FLUSHシステムでは，ユーザがターゲットの位置情報を明示することなく，ターゲットに物理的に近接するサービスの特定することを目的とする．そのために，FLUSHシステムはサービスの存在する空間の相対的な距離関係を把握している．したがって，ツリーの階層構造だけでは特定できないサービス間のより厳密な距離関係を表現でき，ターゲットからサービスへの移動距離をサービス検索に反映できる．そして，サービスを管理するサーバを階層的に配置することでサーバ越しの位置情報によるサービス検索を提供した．

以上のように，FLUSHシステムでは，位置情報を明示することなく，ターゲットに最も近接するサービスを自動的に一つ特定する機構を実現した．

FLUSHシステムには，様々な応用例が考えられる．例えば，外部のユーザから，常にターゲットに近接するプリンタに出力することで，ターゲットの移動を追尾するFAXサービスを実現できる．また，将来的に網膜カメラのなどの技術が向上し，ユーザがデバイスを持たずに自身の位置情報を特定できれば，ユーザは何も手にせずに必要な機能を環境から検索し，利用できる．こうすることで，ユーザは，計算機を携帯することなく，同様の機能を外部から享受できる．

今後，空間の入り口などの抽象化し，より厳密な空間同士の距離関係の記述を可能にする．また，サービス提供者が移動した場合でも，移動先の位置情報をシステムに反映させる機能を追加する．こうした機能を追加することで，FLUSHシステムを基に，ユーザがユーザやサービスの移動を意識することなく，近接するサービスを利用する環境が構築可能となる．

謝辞

本研究の機会を与えて下さり，絶えず丁寧なご指導を賜りました，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，寝食を共にし，多方面に及び相談にのっていただいた慶應義塾大学徳田中澤仁氏に感謝致します.

そして，坪井孝弘氏，大越匡氏，岩本健嗣氏，永田智大氏，藤村浩士氏，牧野聡氏をはじめとする徳田研究室の諸先輩方に感謝します.

そして，研究室で本論文の執筆活動をした徳田研究室4年の皆さんに，この場を借りてお礼致します．また，卒論執筆の後押しをしてくれた徳田研究室の2，3年生にもお礼致します.

参考文献

[1] Veizades, J., Guttman, E., Perkins, C. and Kaplan, S.: Service Location Protocol (1997).

[2] Sun Microsystems, Inc.: Jini Architecture Specification (1998). http://www.javasoft.com/products/jini/specs/jini-spec.pdf.

[3] Czerwinski, S. E., Zhao, B. Y., Hodes, T. D., D.Joseph, A. and Katz, R. H.: An Architecture for a Secure Service Discovery Service, , pp. 24–35 (1999).

[4] Team, T. N.: The Ninja Project. http://ninja.cs.berkeley.edu.

[5] World Wide Web Consortium: Extensible Markup Language(XML) 1.0 (1999).

[6] 暦本純一: 実世界指向インタフェースの研究動向, コンピュータソフトウエア, Vol. 13, No. 3, pp. 4–18 (1996).

[7] watanabe, Y., shionozaki, A., Teraoka, F. and Murai, J.: The Desibh and Implementation of the Geographical Location Information System, (1996).

[8] Harter, A., Hopper, A., Steggles, P., Ward, A. and Webster, P.: The Anatomy of a Context-Aware Application, , pp. 59–68 (1999).

[9] UEHARA, K., TATEOKA, T., WATANABE, Y., SUNAHARA, H., NAKAMURA, O. and MURAI, J.: A Network Architecture for Continuous Mobility, , pp. 224–269 (1998).

[10] T, R., Q, S.-F., K, W. and A, H.: Virtual Network Computing, , Vol. 2, No. 1, pp. 59–68 (1999).

[11] Sun Microsystems Inc.: Jini Technology Helper Utilities And Services Specification (1999).

[12] Sun Microsystems Inc.: JDBC API TUTORIAL AND REFERENCE, SECOND EDITION.