修　士　論　文

ネットワーク分散キャッシュを用いた
大規模センサ情報配信システムの提案

Scalable Sensing Information Delivery
Using Network Distributed Cache

指導教官　　江崎 浩

情報理工学系研究科
電子情報学専攻

氏 名　　石田 真一

提 出 日　　2007年2月2日

概要

無線通信技術の発展と半導体技術の進展に伴いセンサデバイスの高機能化とネットワーク化が急速に進展している．これまで，ネットワークに接続されることのなかったセンサデバイスがネットワークに接続されるようになり，また特有の通信プロトコルを用いて構築運用されていた既存のセンサネットワークの IP 化も急速に進展し，センサノードをインターネットに接続してデータの収集と配信を行う傾向が広がりつつある．

本研究は，インターネットへの接続性を持つすべてのセンサ群を対象に，効率的でスケーラブルなデータの収集と配信を実現可能な情報通信基盤を構築整備し，グローバルなセンサネットワークの構築に資する基盤アーキテクチャと基盤技術の研究を目的としている．

今後もネットワークに接続されるセンサ/アクチュエータの数やユーザアプリケーション数は増加を続け，現在のデータ収集配信アーキテクチャのままでは，さらなる大規模化 (グローバル化とユビキタス化) に対応することが困難であると考えられる．膨大な数のノード間で起こるデータ要求と応答処理にともなう処理負荷の増大やネットワークにおける輻輳状態の発生，センサやそのセンサ情報の検索方法の多様化への対応など，種々のスケーラビリティに関する技術課題の解決が必要となる．

本研究では，このような要求に対応可能なデータ収集・配信のアーキテクチャを提案し，その初歩的な実装評価を行うためのエミュレーションシステムの実装を行った．提案システムでは，参加ノード同士でオーバレイネットワークを構築し，DHT を応用したセマンティクスの多様化への対応，ネットワーク上に分散配置したキャッシュを効率的に利用することでセンサデータ配信に関わる負荷分散と応答時間特性の向上を実現する．

目次

図目次

第 1 章

序論

本章では，本研究の背景を述べ，本研究で解決しようとしている問題点を明確にする．また，最後に本論文の全体構成を示す．

1.1 研究の背景

IP ネットワークの普及に伴い，これまで独自のインターフェイスでデータ通信が行われてきたセンサやそのデータ収集・配信システムも IP ネットワーク化への対応が急速に進んでいる．例としてビルオートメーションのためのセンサ群や，車・自動改札機などといった交通システムにおけるセンサ，また気象情報を収集する気象センサシステムなど，様々なセンサネットワークを挙げることができる．ひとつひとつのセンサノードが IP アドレスを持ち，インターネットを介してデータを提供し，インターネット上に存在する任意の計算機上で動作するアプリケーションもインターネットを介して任意のセンサノードからデータを取得するようなグローバルなセンサネットワークが，現実に利用できる環境が整いつつある．さらに，既存の様々なセンサーネットワークに関しても，インターネットとの相互接続により，広くそのセンサデータを多目的に活用されることが可能となる．

センサデータをインターネットを利用して広範囲に収集・配信するプロジェクトとして，Live E! Project[1] のような活動も拡大しつつある．Live E!プロジェクトでは現在，主に気象センサノードを世界各地に設置し，インターネットを通じて観測データをサーバに保存し，そこからユーザへのデータの提供を行っている．また，既存のセンサネットワークも取り込み，あらゆるセンサ群を統合したグローバルでユビキタスなセンサネットワークの基盤を構築することを目指している．

本研究は，Live E!プロジェクトの活動を通じて提起された技術課題を解決することを目的としている．今後，センサ数の増加や多種多様なアプリケーションの動作が本格化した際にも，その増大し続ける処理負荷に対応可能なシステムの提供が必須となる．しかし，センサ対アプリケーションの純粋でナイーブなエンドツーエンドの通信アーキテクチャ (現在の Live E!で運用されているような特定のサーバにデータを集中管理させる方法など) では，データ送受信や検索要求の処理負荷がサーバノードに集中し，システムの大規模化が困難になると考えられる．

すなわち，今後のセンサネットワークの普及と大規模化に対応するために，ネットワークとしての分散処理を実現することによって，各ノードの負荷を低減する新しいアーキテクチャの研究開発と導入を進めなければならない．

1.2　研究の目的

上述のように，膨大な数のノードがセンサーネットワークに接続される環境においては，クライアントサーバモデルによるシステム構築と運用では，大規模化に関する限界が存在すると考えられる．本研究では，インターネットへの接続性を持つすべてのセンサ群を対象に，効率的でスケーラブルなデータの収集と配信を実現可能な情報通信基盤を構築整備し，グローバルなセンサネットワークの構築に資する基盤アーキテクチャと基盤技術の提案と確立を目的としている．そこで，本研究では，根本的なアーキテクチャの見直しと検討を行い，ノード同士の P2P 通信を導入することによって処理負荷をネットワーク上に分散させる手法の提案と実装を目指す．センサネットワークの大規模化に関わる技術課題のうち，特に以下の問題を解決することを本研究の目的とした．

- 検索対象の多様化
- 特定のノードに対するクエリ (検索要求) の集中
- アプリケーションが送信すべきクエリ (検索要求) 数の増加
- クエリ (検索要求) に対する応答レイテンシ特性の向上

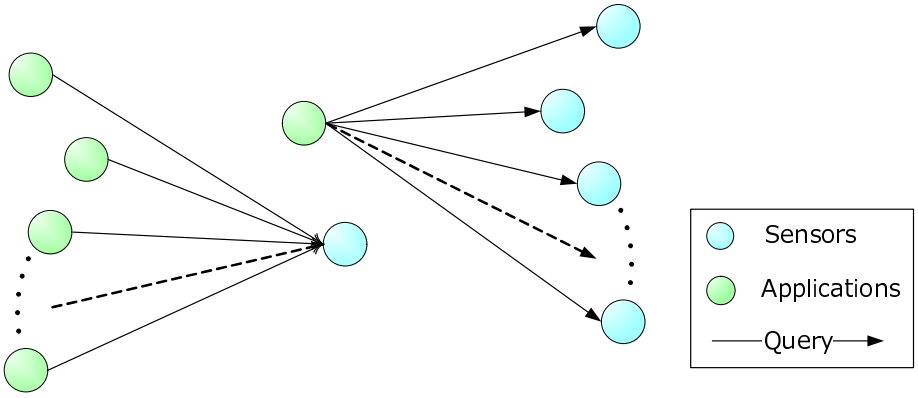

図 1.1　クエリの増加・集中

これらの問題点について，それぞれ簡単に補足する．

システムの大規模化は，その数の増大による処理負荷の増大の問題のみにではなく，多様で大量の検索対象から必要とする対象をどのようにして効率的に適切なデータを選び出すかという技術課題も含んでいる．センサもアプリケーションも拡大を続けている状況では，必要とされる検索方法を予め網羅することは不可能であり，要求に応じた柔軟な変更が容易な仕組みが求められる．

一方，負荷に関しては，アプリケーションからデータを利用されるセンサ群には偏りが存在するのが一般的傾向として観測されている．その結果，図 1.1 のように，他より多くのアプリケーションからデータを要求されるセンサが偏在してしまう．このようなノードは規模の拡大によって，より一層クエリ (検索要求) が集中し，サービス停止に追い込まれることになる．

逆にアプリケーションでは，特定の個々のセンサからデータを収集することよりも，ある一定の条件を満たすセンサ群に対してデータを要求する機会の方が多いと考えられるため，図 1.1 に見られるように送信すべきクエリ (検索要求) 数の増大が発生してしまう．

クエリ (検索要求) に対する応答レイテンシ特性は，提供するサービスが広く一般に利用されるためには重要な品質基準である．P2P による分散で多段中継されたり，上記のように，ノードにかかる負荷の増大によって発生するレイテンシの増大も深刻な問題となりうる．

このように，センサネットワークを構成するノードが高負荷の状態に陥る原因と要素としては，単純に構成ノードの全体数の増加だけでなく，処理量の偏りの発生が大きな原因/要請であることに気づく．本研究では，このような不均一性を積極的に利用することに着眼した新しいアーキテクチャの提案を行っている．

検討対象とするセンサネットワークは，先に述べた Live E!システムを想定する．すなわち，それぞれの時間間隔で定期的に観測データを提供する気象センサノード群と，気象情報を定期的に問い合わせるアプリケーションノードがインターネット上に多数散在している．また，このセンサネットワークにおいては，必ずしも厳密なリアルタイムのデータ取得要求ではないが，近リアルタイムでのセンサデータの取得を要求することを仮定する．また，アーカイブデータの提供については，本研究の範囲外とする．

1.3　本論文の構成

本論文は，以下のように構成されている．

- 第 1 章 (本章) では，大規模なセンサ情報配信システムの現状と問題について述べ，本研究の目的を明らかにする．
- 第 2 章では，提案アーキテクチャを考察する上で，参考・比較した既存のデータ収集配信システムについて，その技術的要点をまとめる．
- 第 3 章では，本研究で提案するアーキテクチャを解説し，本章で提起した技術課題の解決手法の議論を行う．
- 第 4 章では，提案手法のエミュレーションシステムの実装方法を解説し，これを用いた提案システムの動作確認と初歩的な性能評価結果を示す．
- 第 5 章では，まとめと今後の課題を述べ、本論文の結論とする．

第 2 章

既存のデータ収集配信システム

本研究で提案するアーキテクチャを考える上で比較検討の対象となった，大規模ネットワークにおけるデータの収集および配信に関するシステムについて概観する．現在も広く利用されているものから，近年研究が盛んにされるようになったものなど，その負荷分散のための既存技術について述べる．

2.1 クライアント・サーバシステム

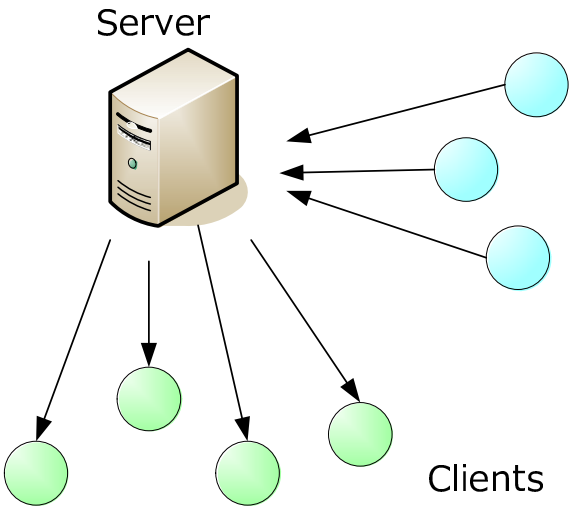

図 2.1 クライアントサーバシステム

図 2.1 のように，資源を集中管理してサービスを提供するサーバと，このサーバに対してデータや要求を送り，処理を任せている大多数のクライアントで構成されるのが，クライアントサーバシステムと呼ばれるものである．

インターネット上の多くのサービスはこのように役割がはっきりわかれた形態で提供されており，現在の Live E!で運用されているシステムもこのアーキテクチャで動作している．つまり，センサは Live E!が管理しているサーバに対して観測データを送信し，これを受信したサー

バはデータベースにそのデータを格納する．もうひとつのクライアントであるデータの利用者はサーバに対して要求を送り，サーバは要求に合致するデータをデータベースから検索して，クライアントに返信する．

このモデルでは，クライアントは既にわかっているアドレスもしくは URL に対してアクセスするだけでよく，通信相手の検索やデータベースからの検索などを自分で処理する必要がない．またデータの流れもほとんどがサーバからクライアントへのものであり，実際のインターネットの回線事情からも効率がよい．

しかし，このように特定のサーバにサービスのすべてを依存しているモデルでは，サーバやネットワークの障害時に全てのサービスが停止することとなる．また，本研究の背景のように膨大な数のクライアントがネットワークに参加するようになると，アクセス集中によるネットワーク帯域の圧迫やサーバの過負荷によって，応答時間の大幅な増大が避けられない．

解決のためには，サーバの冗長化が必要になるが，よく用いられるのは図 2.2 ような 2 つの方法がある．ひとつは，ロードバランサによって複数のサーバをまるで一台であるかのように見せるクラスタリングで，負荷を各サーバに分散したり，故障したサーバがあってもサービスを停止させることがない．もうひとつは，ネットワークの負荷を分散するために，DNS ラウンドロビンによって別ネットワークに置かれた複数のサーバのアドレスを順に返すものである．

これらの冗長化手法によってある程度の規模であれば，サービスを提供し続けることが可能であるが，急速に拡大するセンサネットワークのノード数とデータ量に対しては，十分な拡張性を実現できるものではない．また，複数のサーバ間で同期をとる必要があったり，システムの提供者が資源を追加・保守していく必要があるため管理コストが高いものとなってしまう．

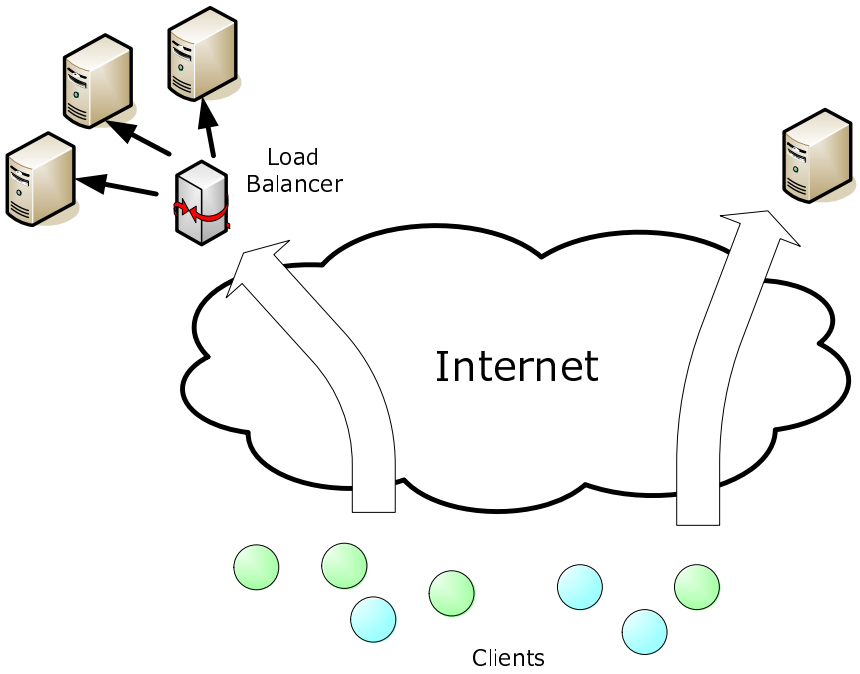

図 2.2　サーバの負荷分散

2.2　CDN

家庭にも広帯域な回線が提供されるようになり，ストリーミングによる放送や動画のオンデマンド配信といったサービスが本格的に始まったが，逆にコンテンツの提供側とってはサービスの提供が難しくなってしまった．つまり，多数ユーザのアクセス集中により，サーバの負荷が高くなり応答速度が悪化したり，ネットワークの帯域を圧迫するという問題を解決する必要が出てきた．

CDN(Content Delivery/Distribution Network) とは，コンテンツの大量配信をするためにネットワーク上の様々なポイントにコンテンツを分散保存し，ユーザがネットワーク的に近い位置にあるキャッシュを利用してもらうよう作られたネットワークである．技術的にはリバースキャッシュとミラーリングを積極的に用いた負荷分散の手法といえる．

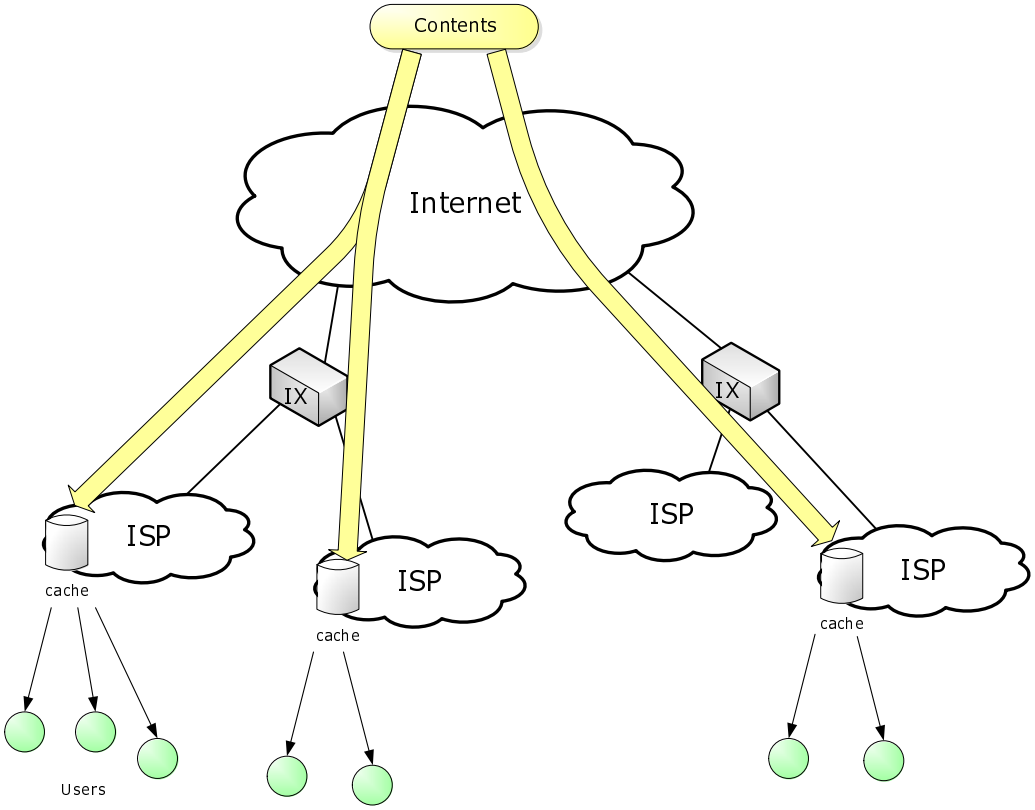

図 2.3　CDN

通常，図 2.3 のように，各 ISP などの単位でそれぞれキャッシュを保持するサロゲートサーバを置き，コンテンツ要求があるとユーザの位置から判断される適切なサロゲートに導く．ここで用いられるリクエストルーティングには DNS を利用したものや，URL の書き換え，エニキャスト [2] などが利用される．こうすることで，サーバへのアクセスを分散させ，ネットワーク全体の帯域も効率よく利用することができる．

CDN は実際のトポロジーや利用状況をを考慮して，キャッシュを配置することで，効果を発揮するシステムである．それゆえ，コンテンツ提供側も受信側も，いつどこに現れるかわからない動的なネットワークでは，キャッシュを置く位置やリクエストのルーティングの効率化が難しい．また，インターネット全体で規模が拡大するようなシステムでは，サロゲートサーバの設置・管理コストが大きくなりすぎる．

2.3　IP マルチキャスト

IP ネットワークにおける通信は主に，1 対 1 のユニキャストと 1 対全のブロードキャストが用いられている．一方，特定の受信者グループに対してのみ送信するマルチキャストという方式がある．

複数の受信者に対して送信する場合，ユニキャストを用いると図 2.4 のように受信者の数だけ同じパケットを送信する必要がある．これは送信者にとっても，途中の中継を行うルータにとっても，受信者数の増加にしたがって負荷が高くなる．

マルチキャストを用いると，登録した特定のグループを対象に，同じコンテンツをルータでコピーしながら配信するため，受信者の数がどれだけ増えても送信者はひとつのパケットを送信するだけでよい．このようにネットワーク帯域の有効活用ができるため，膨大な数のユーザに同じコンテンツを一斉送信する放送を実現するための技術として期待される．

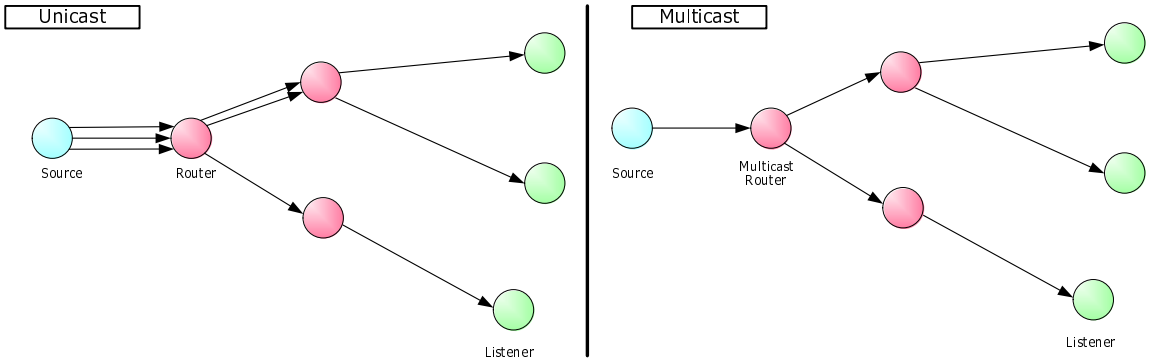

図 2.4　ユニキャスト - マルチキャスト

IP マルチキャスト技術を構成するのは主に，クラス D(224.0.0.0 - 239.255.255.255) の IP マルチキャストアドレスの割り当て，グループ管理プロトコル，マルチキャストルーティングプロトコルである．

マルチキャストアドレスは，受信を希望するトラフィックのグループを指し，このアドレスに対して送信が行われる．このうちの一部は IANA によってすでに予約されているが，それ以外の部分で動的にアドレス割り当てをするプロトコルがいくつか提案されている．

グループ管理には，IGMP(Internet Group Management Protocol) [3, 4] が利用され，これは各ホストが受信したいグループに参加するために，その IP マルチキャストアドレスを LAN 内のルータに通知するためのプロトコルである．ルータは IGMP メッセージを待ち受けつつ，

定期的に IGMP メンバシップクエリを送信して，そのグループ宛のトラフィックを受信希望しているホストがいるかどうか確認している．また，ホストはグループから明示的に外れることを通知する Leave Group を送信することもできる．

マルチキャストのルーティングプロトコルは，DVMRP[5]，MOSPF[6]，PIM-SM[7, 8] などいくつか存在し，このうち PIM-SM はもっともよく使われているプロトコルで，IP ルーティングプロトコルに依存せず，様々なユニキャストルーティングプロトコル上で動作する．

これらルーティングプロトコルの動作によって送信元から全受信者へのループのない配信ツリーが構築されるが，図 2.5 配信ツリーにはソースベースのツリーと共有ツリーがある．ソースベースのツリーは送信者と受信者との間で最適なパス，つまり送信者をルートとする SPT(Shortest Path Tree) を作る．一方，共有ツリーはランデブーポイント (RP) と呼ばれるルータを明示的に指定し，送信者からのマルチキャストパケットはすべて RP に対して送信され，RP を起点にして全受信者へ配信される．受信側も RP へ明示的に配信を要求する必要があり，このため余分なトラフィックを流すことがなく動作できる．

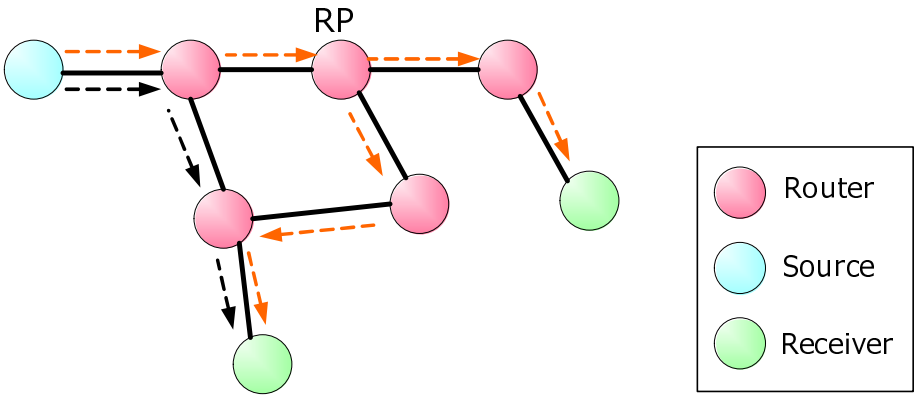

図 2.5　マルチキャストツリー

RP にトラフィックが集中する問題が考えられるが，PIM-SM では RP からの共有ツリーより最適な経路があれば，RP に Prune を送りソースベースを使って負荷を減らすようになる．しかし，ソースベースでもセンサネットワークのように送信元の数と受信者が参加するグループの数が多くなると，マルチキャストルータが保持するパスの情報が多くなってしまい大規模化の対応は難しい．

その他の問題として，経路上のすべてのルータがマルチキャストに対応している必要があり，利用環境が限られる．同報配信であるため，パケットロスが起きると抜けたデータを復旧できない．信頼性を考慮したマルチキャストプロトコルとして，SRM(Scalable Reliable Multicast Protocol)[9] や RMTP(Reliable Message Transport Protocol)[10] といった研究もされているが実際に使われる状態にない，といった問題がある．

2.4　オーバレイネットワーク

インターネットは，もともと様々なノードを相互接続した自律分散ネットワークであり，各ノードが送信する IP パケットを転送する働きを提供しているが，効率やセキュリティの理由でルータやファイアウォールによってセグメントごとに区切られて制限が存在する．しかし，ノード上で動作するアプリケーションにとっては通信相手とのつながりが重要であり，IP を意識する必要なく自由に通信したいという要求がある．そこで，図 2.6 のように IP 層を隠蔽して上位のアプリケーション層でのコネクションによって構築したものをオーバレイネットワークという．

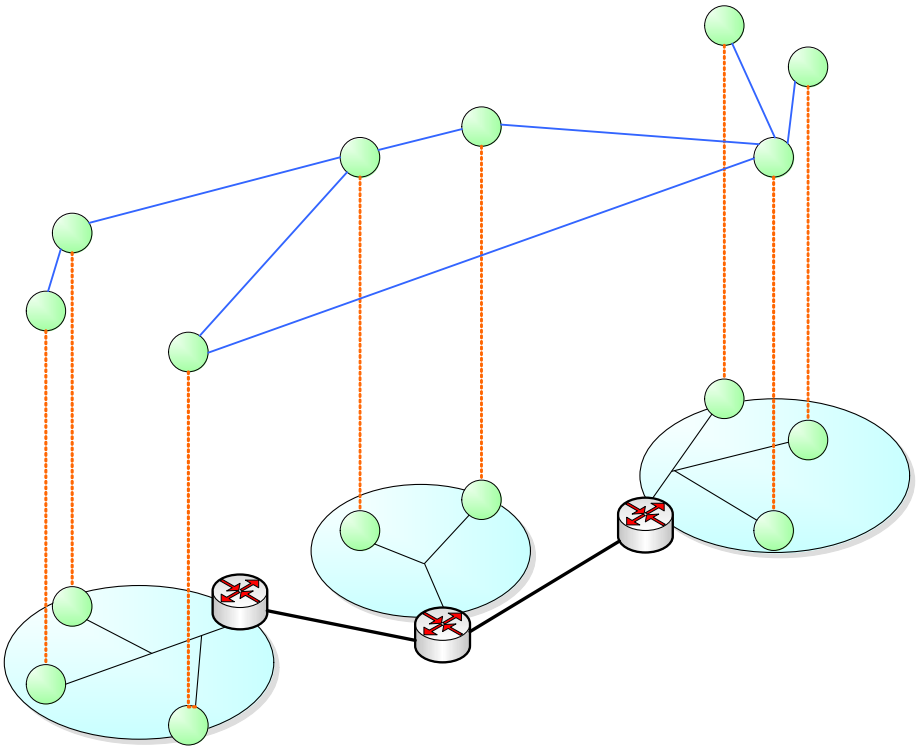

図 2.6　オーバレイネットワーク

このようなオーバレイネットワークの中で，近年注目を集めているのが，P2P ネットワークと呼ばれるものである．P2P ネットワークにおいて参加ノードは対等なもの (ピア) として動作し，前節までのシステムのように，はっきりと役割がわかれていない．つまり，各ピアはサービスを受けるだけでなく，提供者の側にもなりうる．しかし，現実にはそれぞれのピアごとにネットワーク環境や処理能力、記憶容量などに違いがあり，すべてのピアをまったく同等に扱うのは効率的ではない．そのためノードごとの差異を考慮して，ネットワークを構築する必要が出てくる．

このような特徴のために，ノードにかかる負荷は分散され，クライアントサーバシステムに

比べて，システムの急拡大にも耐えられると考えられている．一部が停止しても他のノードがその役割を引き継ぎ可用性が高いサービスとして，ファイル共有ソフトの技術として広く使われることとなった．一方，全体のノードや情報を一元管理したり，同期したりといった部分は苦手とする．

現在のところ，様々な P2P ネットワークがコンテンツの分散配信に利用されているが [11]，それらを特徴づけるポイントには，ピアの識別，情報検索，ピア間でのデータ転送などがある．

中央サーバが存在しない P2P ネットワークにおいて，情報の検索はピア同士が互いに聞きあって，自律的にその所在を発見する．ピアの発見やデータ転送をするにも，IP ネットワークに依存しないオーバレイネットワークでは，アプリケーション層で張られた仮想的なリンク上で，どのようにデータを転送するかといった，ルーティングの機構を自身で持たなくてはならない．目的のピアと通信を直接するのか，中継をいれて間接的にするのかという違いもある．そのためには，オーバレイネットワーク上で通信相手ノードを一意に指定する ID の割当てかたや，トポロジーの構成に工夫が必要となる．

オーバレイネットワークは大きく分けて 2 つに分類される．それぞれ Structured(構造化) オーバレイネットワークと Unstructured(非構造化) オーバレイネットワークと呼ばれており，以下に具体的な例をいくつか提示した．

- Unstructured Overlay Network
  - Gnutella
  - Freenet
  - Winny
- Structured Overlay Network
  - CAN
  - Chord
  - Pastry
  - Tapestry

Unstructured オーバレイでは，どのノードとピアを張るかに関して，制限がない．そのため，コンテンツやノードの発見には，擬似ブロードキャストやフラッディングといった，ピアを張っているノードに聞いてまわりバケツリレー式に転送する手法を使うこととなる．これは効率が悪く，遅延の大きさや，ループ防止のために TTL で検索範囲をある程度に制限する必要があるために目的の情報を発見できないことがある，といったことが問題である．一方，検索自体は複雑で柔軟な条件で行うことができるといった長所もある．

Structured オーバレイでは逆に，後述する DHT などを用いることによって，トポロジーの制約がある．検索はノードやデータが構造的に配置されいてるため，大規模化しても破綻することなく発見が可能となる．そのかわり，Unstructured オーバレイのように複雑な条件での検索は難しく，部分文字列や一定範囲での検索について研究が盛んに行われている．

次節からは Unstructured，Structured オーバレイそれぞれについて，代表的になアプリケー

ションの動作とともに，大規模な情報の分散配信手法をまとめる．

2.4.1　Winny

サーバを介せずユーザのコンピュータだけで情報交換を可能にした Gnutella や匿名性を高めた Freenet などの，P2P ファイル共有ソフトの発展とともに，Unstructured オーバレイは注目されるようになった．その中で，Winny[12] は Freenet[13, 14] を参考に，ファイル共有ソフトとして階層構造とキャッシュを利用して大規模運用に耐えうるものを目指した．

Freenet は匿名性が高いということで注目されたが，そのポイントは情報を直接でなく，間接的に多段中継することであった．図 2.7 のように，ネットワークに公開する情報を細かく分割し，暗号化することで，中継ノードがその断片を受け取っても中身がわからないようになっている．この断片が複数ノードの間で次々と拡散していくため，第一発信者を特定することができない．しかし，規模が大きくなるとフラッディングによるデータ配信では全体に拡散させることは難しく，そのデータを受信したいノードのところへ届けられない可能性がある．

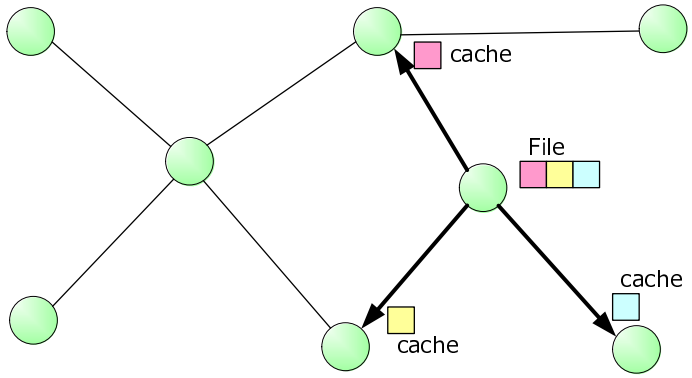

図 2.7　Freenet での断片化ファイル拡散

Winny では，匿名性・効率性を追求する P2P ネットワークには，Freenet の暗号化や情報の拡散は効率を犠牲にしすぎていると考え，多段中継，つまりプロキシ技術に注目して設計された．また，Freenet のようにファイルの断片を接続しているピアに対して，自発的にプッシュすることがなく，プル型でファイルが拡散していくため，大きな負荷をかけることなく，効率をよくしている．

Winny の中継方式が特徴的な部分を 2 つ述べる．

ひとつはファイル自体のかわりに，その要約情報であるサイズの小さなキーをネットワーク内に拡散しておき，これがファイルを検索するためのインデックスとして機能する．拡散したキーには TTL が設定されており，一定時間が経過すると削除されるため，ファイルをもつノードが離脱していればキーはネットワーク上から消滅するが，どこかに存在すれば定期的にキーは運ばれてくることになる．このキーを見つけたユーザは，そこに含まれている情報によってファイルを保持しているノードを知ることができる．さらに，定期的な拡散を待つだけでなく，検索クエリを次々と隣接ノードに転送してもらうことで，条件にあうキーを探すこともできる．

見つかれば検索元まで逆にたどって届けるが，その中継ノードにもキーのコピーを置いていくことで，より広くキーを拡散している．

もうひとつは，ファイルが転送されたノードから新たにキーを作成して公開するだけでなく，転送経路上の中継ノードにも一定確率でキャッシュを残していくことである．キーを転送していく過程において，中継ノードが一定確率でそのキーに書かれたファイルの宛先を自分に変更して転送することがある．検索して，このキーを受け取ったノードは書き換えられた情報をもとに，データ転送要求を行うが，実際にはそこにデータはない．自ノードにファイルがなければ，自分が保持しているキーを利用して元のノードからファイルを得て，同時にそれを転送する．この仕組みによって，頻繁に要求されるファイルほど，多くのノード上にキャッシュされることになり，ネットワーク全体の配信効率が高くなる．

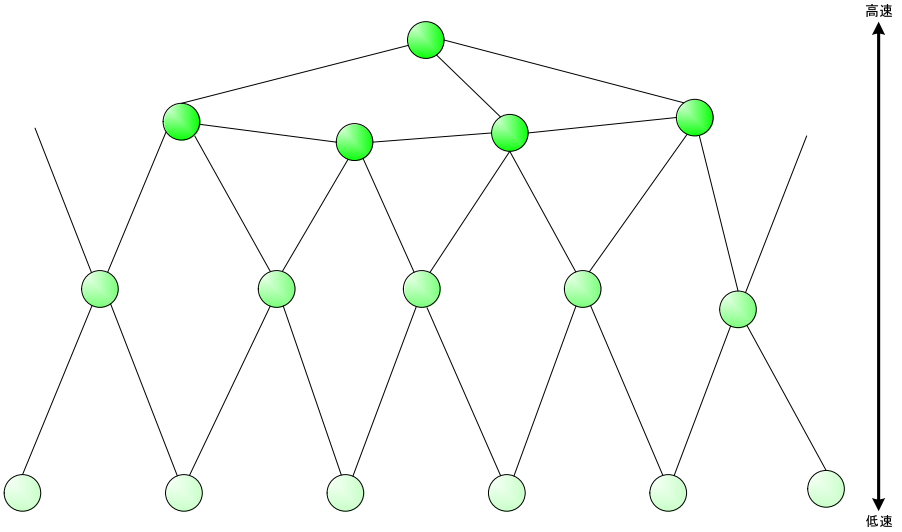

図 2.8　Winny ネットワーク

Winny では大規模化に耐えるために，ネットワークのトポロジーに関しても以下のような 2 つの特徴がある．ひとつは，図 2.8 のように，高速な回線に接続しているノードほど上流に配置し，低速なものは下流に配置している．上流に配置されるノードの方が，他ノードよりもサービス提供能力が高いと見なし，より多くのキーやファイルキャッシュが集まるようにして検索や配送効率を高めるためである．上流には次々とキーが流入してくるため，古いものは短期間で捨てられ，絶えず新鮮な人気の高いキーが保存される．このため上流に検索クエリを送信することで検索のヒット率が高くなる．

上流にキーやキャッシュを集めることで検索効率は確かに改善するが，それでも規模が拡大するにつれて検索可能な範囲にキーが存在するヒット率の低下は避けられない．Winny では検索の嗜好性が似ているノードが，なるべく近くに配置されるようにネットワークを構築するクラスタリングという概念が導入されている．傾向が似たノードが近くに集まることで，検索クエリがヒットするまでの転送ホップ数が少なくできる．

2.4.2　DHT

DHT(Distributed Hash Table，分散ハッシュテーブル) は，構造化オーバレイネットワークを構築する際の技術のひとつとして利用されている．これは，ある情報のハッシュ値 (キー) とそれに対応するデータのペア (ハッシュテーブル) を複数ノード上で分散管理し，各ノードは一部の情報だけから協調動作することで目的とするデータを比較的少ないステップ数で探索することができるようなルーティングの仕組みを提供する．DHT の特徴をまとめると，構成ノードだけで自律分散的にネットワークを構築でき，一部ノードが頻繁に参加・離脱を繰り返しても動作しつづけるだけの耐性を持っていて，参加ノードの数が増えても効率的な処理を可能とするだけの拡張性をもったシステムであると言える．

DHT では，アルゴリズムによるが 128bit や 160bit のハッシュ空間を構成ノード間で分割し，各ノードは自分に割り当てられた空間にマッピングされるキーと値の組を保持する．つまり，ハッシュ関数には MD5 や SHA-1 など多数存在するが，DHT 内で同一ハッシュ関数を用いて，ノードの ID と格納するデータのキーを得ることで，この 2 つを同一空間上にマッピングすることになる．

検索はキーを宛先としてルーティングされ，ハッシュ空間上でそのキーが存在する空間を受け持つノードに向かって転送される．これは，より近いノードほどその付近について詳細な情報を持っているために，繰り返し転送することで，いつかは最も近いノードにたどり着くようになっているからである．こうして最も近いノードに格納されたデータを発見することができる．非構造化オーバレイネットワークの検索ではフラッディングであったり，前節の Winny のようにコンテンツの要約情報である小さなキーを流しておいて，それを探すようにしたものであったため，目的のデータが発見できる保証はなかった．DHT ではノードとデータが構造化されてトポロジーを生成しているため，確実に発見できる．そのかわり，DHT そのままではフラッディングでの検索のようなキーワード検索等には対応が難しく，完全一致での検索が基本となっている．

これまでに様々な DHT アルゴリズムが発表されており，代表的なものには，CAN[15], Chord[16], Pastry[17], Tapestry[18]，Kademlia[19] などがある．ここでは，その中でもしくみが比較的わかりやすい Chord を例に，DHT の動作をまとめてみる．

Chord では，m ビットのハッシュ空間 (2^m の剰余系) で図 2.9 のように円状のトポロジーを形成する．各ノードはそれぞれルーティングを行うためのフィンガーテーブルを保持しており，最低限自分の次のノードがわかっていれば，順番に問い合わせていくことで，すべてのノードを探索することが可能である．しかし，この場合 $O(N)$ のホップが必要になるため，実際は図 2.10 のように 2^k ずつ離れたノードへの経路をフィンガーテーブルに保持している．このとき，例としてノード 0 からノード 5 へのルーティングは，すぐ次のノードへの経路しか持っていなかった場合は 5 ホップ必要であるが，Chord の実際のフィンガーテーブルを利用すると，2 ホップで到達可能となる．つまり全ノード数 N に対して，$O(log(N))$ で到達可能ということになり，大規模化に耐えうる効率のよいルーティングを実現している．一方，データの検索や登

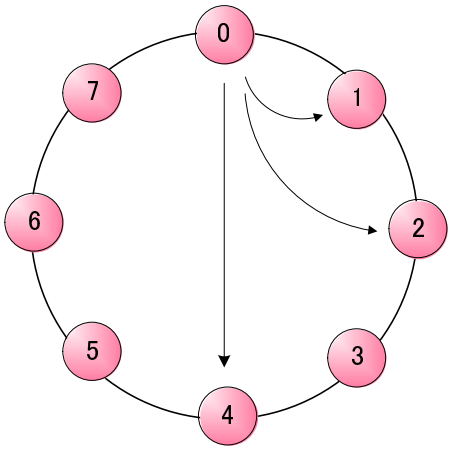

図 2.9　Chord:保有ルート情報

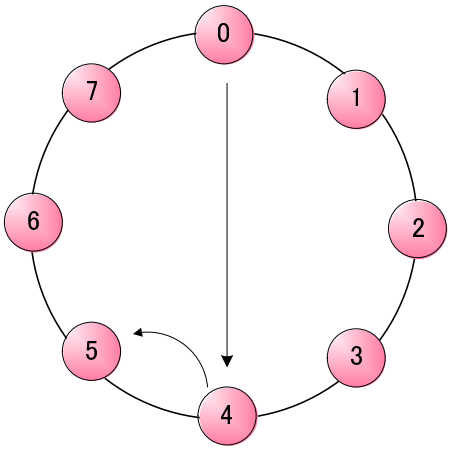

図 2.10　Chord:ノード 0 から 5 へのルート

録はキーに対応するノード，該当するノードがなければハッシュ空間上の次のノードまでルーティングされて，そのノードと通信することで実現できる．

また，ノードの参加についても述べる．新規参加ノードはハッシュ空間上で自分の次に来るノードとなりうるノードを，DHT 上のノードに検索依頼して，その情報を自分の次のノードとして登録する．各ノードは定期的に，自分の次にくるノードを確認や，フィンガーテーブルの更新を行って，不整合を解決しノードの参加・離脱に対応している．

2.4.3　オーバレイマルチキャスト

オーバレイマルチキャストは，アプリケーションレイヤマルチキャスト (ALM) とも呼ばれ，IP マルチキャストの機能をアプリケーションレイヤで実現するものとして登場した．IP マルチキャストではルータが行っていた処理を，エンドホスト同士がアプリケーション層で確立したリンクを用いて行うため IP ネットワークの制限を受けることなく導入が容易である．IP 層から見るとエンドホスト同士のユニキャストをアプリケーションが制御しているだけのものであるが，その手法は数多く存在しており特性もさまざまである．

トポロジーを構成する方法の違いから次の 2 つに分類されることが多く，最初にメッシュ型のコントロールトポロジーを構成してからマルチキャストツリーを生成する Mesh-first 型，逆に最初にノード同士でツリーを構成する Tree-first 型がある．前者では Scribe[20], SplitStream[21], Narada[22], Scattercast[23] などが，後者では Yoid[24], ALMI[25], HMTP[26]，Overcast[27] などが代表的なものである．

この中でも特に，前節で述べた DHT の上でオーバレイマルチキャストネットワークを構成し，DHT の持つ自律分散，耐障害性，拡張性を利用した P2P マルチキャストは注目されており研究がさかんである．本節では，Pastry 上に構築されたオーバレイマルチキャストである Scribe について簡単に解説する．

Scribe のネットワークはすべて Pastry に参加しているノードで構成され，グループの新規作成や参加，マルチキャストツリーの生成に Pastry を利用している．誰でも自由にグループを

作成することができ，グループはユニークな groupID で識別される．作成するには，groupID をキーとして CREATE メッセージをルーティングしてもらう．Pastry はこのメッセージを groupID と最も数値的に近い nodeID を持つノードに届けてくれて，これがグループのランデブーポイント (RP) として機能し，マルチキャストツリーのルートとなる．

Scribe でマルチキャストツリーの一部となっているノードは forwarder と呼ばれ，各 forwarder はツリー上の子に関する情報 (IP アドレスと nodeID) を children table に保持している．Scribe ノードがあるグループに参加する際は，groupID をキーとして JOIN メッセージを Pastry によって RP まで転送してもらう．このとき経路上の各ノードでは自分がそのグループの forwarder であるかどうかを確認し，そうであれば children table に追加し，まだ forwarder でなければ，このグループのエントリを作成して送信元ノードを子として登録する．こうして forwarder となったノードは JOIN メッセージを RP にむかう次のノードに転送する．

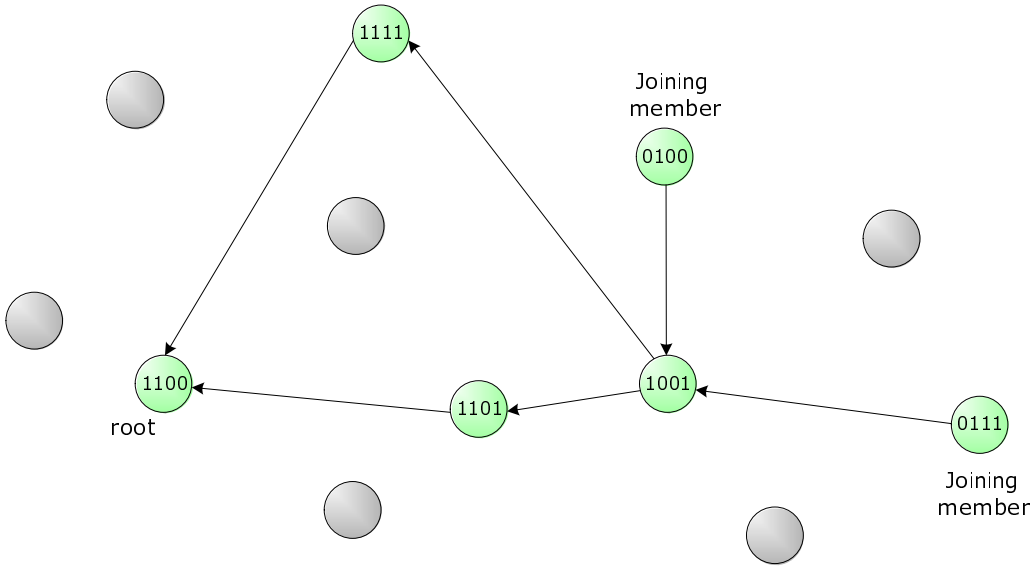

図 2.11　Scribe のメンバ管理

図 2.11 は，グループ参加のメカニズムを表したものである．ノードに書かれた数字はそれぞれの nodeID であり，簡単のため各桁は $2^b (b = 1)$ で，各プレフィックスは 1bit ずつとなる．ここでは，groupID が 1100 であり RP がちょうど同じ nodeID をもつノードとなっているグループを想定する．nodeID が 0111 であるノードがこのグループに参加しようと JOIN メッセージを送ると，Pastry によって 1001, 1101 の順に転送され，目的の 1100 である RP に到達する．ノード 1001,1101 がグループ 1100 の forwarder でなかった場合は，0111 の参加によってこの 2 つのノードが forwarder となり children table に子ノードのエントリを追加することになる．このあと新たにノード 0100 がこのグループに参加しようとすると，JOIN メッセージは先ほどと同様にまず 1001 へ転送される．今回はノード 1001 がすでにこのグループの forwarder であるため，0100 を children table に追加するだけで，JOIN の処理は完了できる．

マルチキャストツリーを維持するためには，forwarder が急に離脱した場合の復旧方法を用

意する必要がある．forwarder になっているノードは，定期的に自分の子ノードにメッセージを送っており，子ノードはメッセージが止まったら親ノードが離脱したと見なして JOIN メッセージを送りツリーを復旧する．図 2.11 において，1101 が離脱した場合を考えてみる．ノード 1001 は JOIN メッセージをルートに向かって送信すると，今度は別のルートでノード 1111 に到達し，1001 を子ノードとして記録すると，必要があれば自分も JOIN メッセージをルートに送信し，children table を変更する．

第 3 章

提案手法

本章では，第 1 章で提起した技術課題の解決を達成するために，本研究で提案したアーキテクチャについて，その要素技術と解決法の解説をする．

3.1 要求事項

インターネットを基盤としたセンサネットワークの大規模化によって，そのデータ収集・配信システムで起こりうる問題のうち，本研究で解決の対象とするものを第 1 章で述べた．本節ではその他の要求も含め，それぞれの問題点について，その要求事項をまとめる．

検索対象の多様化

本研究で対象としているのは，あらゆるセンサ群を統合したグローバルなセンサネットワーク基盤であるため，センサノードやアプリケーションの数の増大のみならず，センサデバイスや観測データの種類，検索要求に利用されるパラメタなどのメタ情報 (セマンティクス) の拡大も予想される．これらを現段階ですべて網羅することは不可能であり，要求に応じて随時変更や追加を行うことは避けられない．その際にシステムに対して何らかの変更・更新手続きや制限が必要とされない柔軟な構造を持つことが期待される．

送信クエリ (検索要求) 数

アプリケーションが特定のセンサノードに対して，明示的にその ID を直接指定してデータ要求を送信する場合，センサネットワークの規模拡大によるセンサノード数の増加は，アプリケーションにとって負荷とはなりえない．しかし，一般に利用頻度の高い検索要求は，属性・条件指定による範囲検索であり，特定のセンサノードに対してのみ行われるものではない．そのため，センサネットワークの拡大は多くのアプリケーションにとって，膨大なクエリ送信を強いることになりうる．

受信クエリ (検索要求) 数

センサネットワーク内でのユーザアプリケーション数が増加すると，ネットワーク全体でのデータ検索要求も同様に増加するが，その増加した検索要求が全てのノードに均等に分散するとは限らない．むしろ，データを利用されるセンサ群には大きな偏りが存在するのが一般的傾向として観測される．そのため，アプリケーション数増加によって膨大な検索要求に対応するノードがいる一方で，変わらず低負荷なノードが存在することになる．検索要求が集中した場合に，ネットワーク内のノード間で分散協調して，受信クエリ数の平準化を図る必要性がある．

応答レイテンシ特性

ユーザからのデータ要求に対する応答レイテンシ特性は，広くサービスを利用してもらうためには重要な品質基準である．負荷分散が実現できたとしても応答レイテンシが犠牲になっていては，一般からの利用を想定するサービスとしては許容することはできない．規模の大小や検索要求の頻度に関わらず，一定の時間内に応答可能なシステムデザインが必要となる．

耐障害性

センサ群の中には，特に高い信頼性を要求するものも少なくない．つまりセンサノードからのデータを確実にすべて収集できるような要求が考えられる．これまでに，そのようなセンサ群が運用されてきた独自ネットワークとは異なり，インターネットの基本はベストエフォートによる通信であり，負荷軽減のためにセンサネットワーク全体での分散処理を行うと，そのような厳しい要求を満たすのは難しい．とはいえ，そのセンサ群の管理者のノード (センサおよびアプリケーションサーバ) がすべて正常稼働しているにも関わらず，他のノードの不備によって影響を受けることは回避したい．いわゆる Single Point of Failure への対策が必要とされ，分散システムとは別の冗長化も考慮した設計が必要となる．

対象としない要求

実際のセンサネットワーク展開時に期待されるであろう要求は数多くあるが，本研究においては対象の範囲外とした要求について，ここで簡単に言及する．

センサが提供する観測データは，長期間にわたってアーカイブしたいものもある．記録の保存目的に留まらず，過去との比較が重要視されるような種類のデータも多く存在するであろうが，アーカイブについては本システムでは提供することは想定しない．必要とするユーザは本システムを利用して，データを受け取り，自分でアーカイブしてもらうことを考えている．

もうひとつはセキュリティの問題である．オーバレイネットワークで自律分散したシステムを構築することにあたっては，セキュリティ上の問題は無視できない重要なテーマとして研究が活発にされている．DOS アタック，データの改竄，なりすましなどによって被る影響は甚大であり，実運用上は対応策が必須となるが，本研究においては各ノードが悪意を持った振舞いをすることは想定しない．

3.2　対象とするネットワークの特徴

本研究で対象とするネットワークは，膨大な数のセンサデバイスとアプリケーションで構築された，インターネット上に展開するグローバルなセンサネットワークである．これは現在急速な拡大を進めている新しいネットワークであり，既存のデータ配信システムとはいくつかの点で異なった特徴を持っている．ここでは，提案アーキテクチャを設計するにあたり，大きく影響を受けることとなった対象ネットワークの特徴について述べる．

セマンティクスの流動性

あらゆるセンサ群を取り込むんだグローバルなセンサネットワーク基盤の構築を計画しているため，ネットワークに取り込まれるセンサの種類は日々変化し続けることになる．また，対象となるセンサの数・種類が変化することは，アプリケーションにとっても検索の方法が絶えず変化し続けるという特徴があることを意味する．

非同期性

データを生成するセンサは，それぞれが独自の時間間隔で観測データをネットワークに対して送信する．一方で，アプリケーションもセンサデータの利用方法は様々であり，各々が目的に見合った間隔でデータの取得ができればよい．細かい時間間隔でデータを要求し，生成されるすべてのデータを収集しようとするアプリケーションもあれば，長い間隔でデータの傾向が掴めればよいとするものまで，このクエリ送信頻度は幅広く偏りが大きい．このネットワークでは，センサのタイミングに合わせることなく，アプリケーションの自由な要求を受け入れることが期待される．

データフロー

センサに対して多くのアプリケーションがデータを要求し，アプリケーションは多くのセンサを対象にデータを取得する，全体としてマルチポイントツーマルチポイントの通信であると言える．

検索・取得対象となるデータは，小さなデータユニットが一定の時間間隔で生成されるものであり，厳密なリアルタイムではないが，近リアルタイムでのデータ取得要求が仮定されている．また，データの流れはセンサデバイスからアプリケーションへの片方向のみであり，同報性は要求されず，アプリケーションごとに自由なタイミングでのデータ取得が行われる．

データのアーカイブについてはシステムとして考慮しておらず，必要であればアプリケーション側での対応を期待する．つまり取得できるのは，ノード中にキャッシュされた範囲のみであるが，ノードに保存されるセンサデータの情報量は，センサ ID が 160bit のハッシュ値，観測データ 4 バイト，タイムスタンプ 8 バイト，その他のオプションが追加されたとしても，たかだか 1 レコードあたり数百バイト未満である．現在の標準的な計算機が持つストレージが数十 GByte 以上であることを考えると，数日程度のキャッシュを保持することは可能である．

3.3　提案アーキテクチャの概要

第 2 章において，既存のデータ収集配信システムについて解説し，数多くの技術と解決策が存在することがわかった．しかし，それぞれの技術は想定するコンテンツやユーザの特徴が異なっており，その条件において効果が認められるものである．すなわち，条件が異なるネットワークに適応しても，期待する負荷分散や応答レイテンシ特性の向上が得られるとは限らない．

本研究で対象とするネットワークは 3.2 節で述べたような様々な特徴を持っているが，まだこのようなネットワーク自体も普及するに至っていないため，現状では対応した情報配信システムというのは考えられていない．そこで，本研究では新規に配信アーキテクチャを提案し，その配信メカニズムに合わせて，データ収集の方法も含めすべてをデザインすることとなった．

提案アーキテクチャの主な構成を図 3.1 に示す．

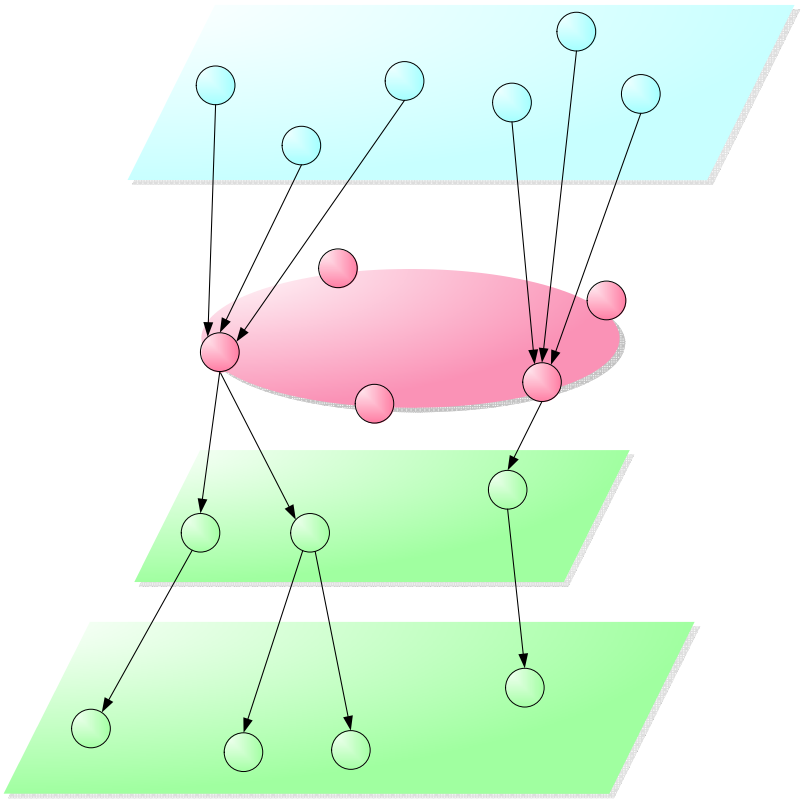

図 3.1　提案モデルの概略

3.3.1 構成

すべてのノードはインターネットに接続され，IP 到達性があるものとする．それらのノード間で自律分散してセンサデータの収集・配信が行われるオーバレイネットワークを形成している．特徴的なのは，センサノードとアプリケーションノードの中間に，DHT を用いた構造化オーバレイネットワークが存在していることである．このネットワークに属するノードを本システムではルートノードと呼んでいる．

ルートノード

ルートノードは，ごく一部のシステム維持のために用意されているものを除き，すべて参加アプリケーションノードがその役割を兼任するもの可能性のあるものである．つまり，図 3.1 においては別々に描かれているものの，論理的に 2 つのノードとしての役割を果たしている 1 ノード，ということになる．

ただし，全ノードではなく資源に余裕のある一部のノードに処理を分担してもらうことを実現可能なアーキテクチャの設計を行った．処理能力の低いノードがデータフローの途中に存在することで，逆に全体の性能を低下させるボトルネックになってしまうことを避けるためである．

センサからのデータを分散させるため，DHT を利用して root ノードがオーバレイネットワークを構成する．センサ ID のハッシュ値から，データ送信先ノードを決めるため，図 3.2 のように，一度に大量のセンサ群がネットワークに取り込まれた場合でもわずかなセンサ ID の違いから，まったく別の場所へデータをルーティングする．このように全てのルートノード間でほぼ均等になるようにセンサからのアクセス，およびそのデータキャッシュが分散することになる．

また，本システムではハッシュ前の key を階層構造を利用したものにすることで，センサ ID だけでなく住所や緯度経度といった様々なプロファイルに対応して，目的のデータを見つけることができる．このようにハッシュを階層的にする関連研究として PHT[28] があるが，この仕組みを利用して柔軟なセンサ情報の検索を可能にする．ルートノードはこのようなプロファイル情報の格納も行い，この階層化を利用することで，膨大な数のセンサデータを集約することにも寄与している．

センサノード

インターネットの基本は P2P であり，今回の P2P システムを設計する上でも，すべてが対等なノードとして参加し，平等に役割を分散させたい．しかし，一般にセンサと呼ばれるノードは通常のアプリケーションが動作している PC に比べて，演算処理においても記憶容量においても著しく性能が劣っている．そのようなノードに P2P アーキテクチャの一部として，処理を分散させることは全体での負荷軽減に役立つとは考えづらい．そのため，本システムではセンサノードだけはシステム全体の分散化への利用を考えない．センサは自分のタイミングにお

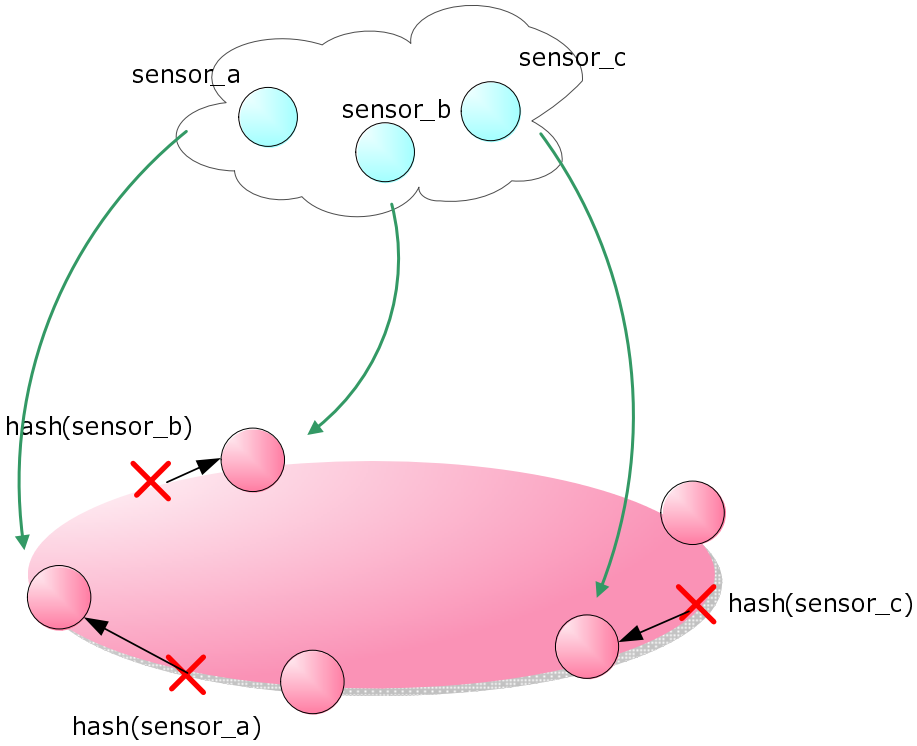

図 3.2　センサデータのルーティング

いて，その観測データを DHT を構成したルートノードのいずれかに送信することで，センサ ID のハッシュ値によって決定されるルートノード上にキャッシュを作ることができる．

アプリケーションノード

本システムを利用してセンサからのデータを取得しようとするノードである．ルートノードのいずれかに対して，データ検索要求を送信することで，センサからのデータを得ることになる．

詳細は後述するが，本システムではルートノードを配信ツリーのルートとして，アプリケーションノードもセンサデータ配信のために機能する．自身が受け取ったセンサデータをキャッシュしておき，他のアプリケーションノードからの要求に応じて配信することで，データ配信の負荷をシステム内のノードで分散していることになる．

提案システムにおけるデータの取得方法は，アプリケーション自身から自発的にデータ要求を行う Pull 型のアーキテクチャをとっている．第 2 章で述べたような，オーバレイマルチキャスト技術によって，多数のユーザに対して効率的にデータ配信をすることは可能であるが，これらのシステムは Push 型でリアルタイムデータを配信するものであり，本研究で対象としているネットワークの要求を満たさない．

3.3.2　キャッシュ

データを受け取ったノードは，受信データを一定時間キャッシュすることとする．最初にセンサからデータを受け取るルートノードだけでなく，アプリケーションノードにおいても他ノードからのデータ要求リクエストがあれば，該当するデータを返すように設計する．こうして，ネットワーク上の複数のノードにキャッシュを分散させることで，データ提供可能なポイントを多く確保し，特定のノードのみに負荷が集中することを防止する．

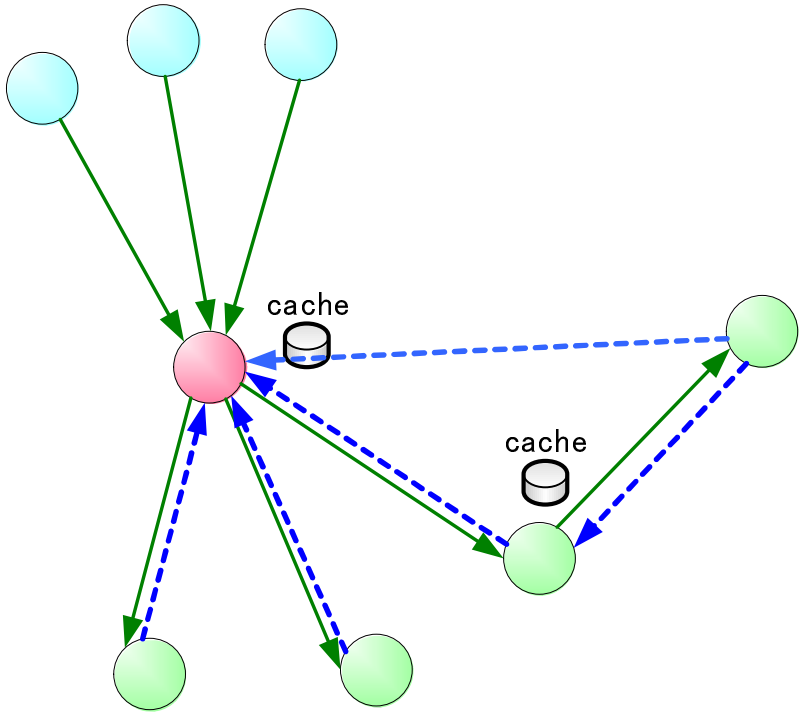

図 3.3　Max Degree = 3 での Delegation

実際にアプリケーションノードに保存されたキャッシュが使われるようになるまでの動きを解説する．

1. アプリケーションノードはルートノードのいずれかに対してクエリ送信
2. 対象となるデータを持つルートノードへリダイレクトされる
3. ルートノードに接続している子ノードの数が Max Degree 以下なら，そのままルートノードに対してクエリを送信
4. ルートノードに接続している子ノード数が Max Degree に達していた場合，その子ノードのいずれかにリダイレクトされ，次回からはリダイレクト先のノードに対してクエリを送信する
5. 以下，子ノードに対してもルートノードと同様，Max Degree に達するとさらにその子ノードへリダイレクトが繰り返される

このような動作を繰り返し，あるクエリに対応するデータに対して，ツリー状に分散してデータをキャッシュし，負荷分散を実現させる．

この手順で単純にアクセスに来た順番でツリーを作成すると，各ノードに対して定期的に送信してくるアプリケーションノードの数は一定未満に抑えることが可能である．しかし，キャッシュはユーザに近い場所にあるほど，その効果を発揮するものであり，アプリケーションのクエリ送信間隔がツリーの親子で短いものが子に来ていると，図 3.4 のようにクエリを送信する度にキャッシュミスが発生することになってしまう．キャッシュミスの多発はそのノードの応答レイテンシ増大に限らず，ミスの発生から再帰的にクエリが上流に送信されることで，ネットワーク全体の効率が悪化してしまう．

この解決手法については次節で述べる．

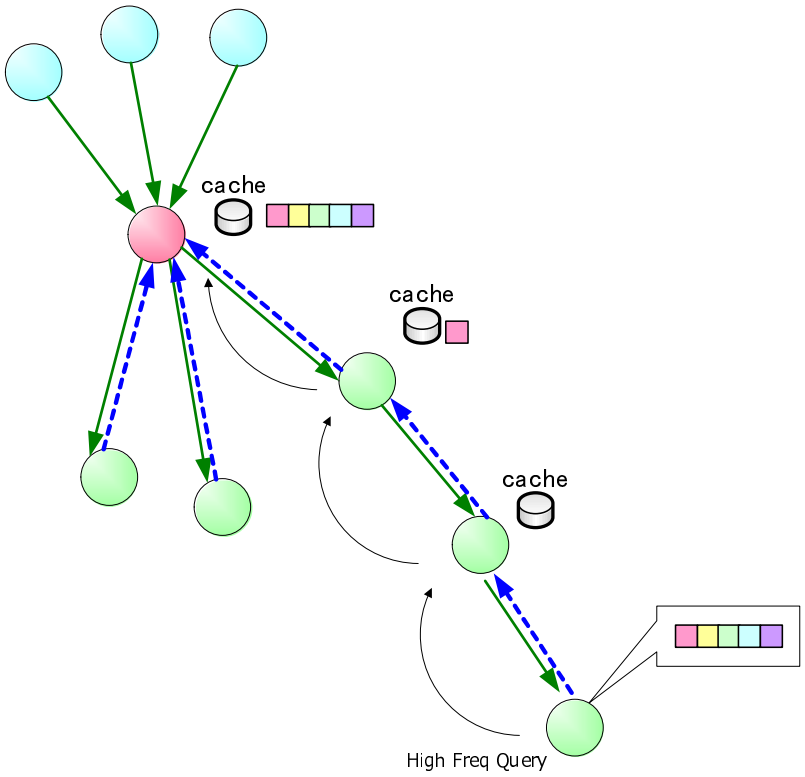

図 3.4　クエリ頻度の差によるキャッシュミス

あまりに頻繁に長期間クエリを送信し続けるアプリケーションノードに対しては，閾値を超えたらデータを push してしまうリバースキャッシュを行う方法も考えられる．

キャッシュを用いた配信ツリーは，利用者が多く頻繁にアクセスされるデータにとっては，クエリの集中を低減する有効な方法である．一方で，逆に利用頻度が低いデータに関しては，キャッシュを用いてネットワーク内に分散させるより，直接 (本システムにおいては最初にセンサからデータを受けとるルートノードに) クエリを送信するほうが効率がいい．提案アーキテクチャでは DHT 内のどのルートノードから接続しても，そのような最初に作られたキャッシュへルーティングされるため，この性質をうまく解決している．

3.3.3　クエリ頻度を考慮した配信

本研究で対象としているネットワークの大きな特徴として，各アプリケーションがデータ検索要求を発行するタイミング，時間間隔は自由であり，大きな偏りが存在しているということはすでに述べた．前節ではこの頻度の差があるために起こるキャッシュミスの問題を指摘したが，ここではその解決手法について解説する．

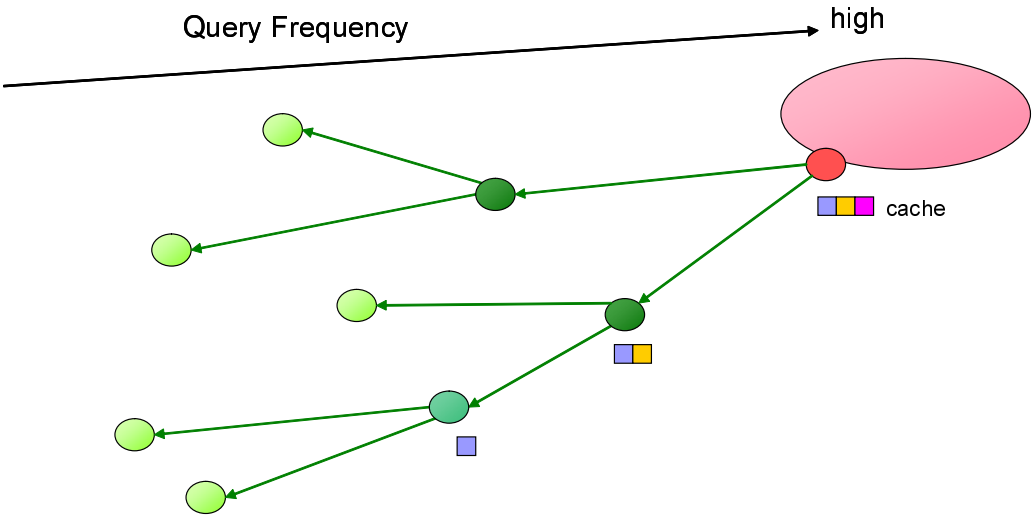

図 3.5　クエリ頻度を考慮したツリー

提案システムでは，逆にこの頻度の偏りを積極的に活用する手法をとっている．ルートから形成される配信ツリーでは，クエリの頻度が高いものほどツリーの上流に配置されるようにしておく．こうすることで，頻繁にクエリを送信するアプリケーションノードには下流のノードからアクセスが来るたびに，すでに新しいキャッシュが保存されていることになる．このように配信ツリーを形成すると，どのノードにとってもそのすぐ親のノードにキャッシュが存在している確率が高く，ネットワーク内のクエリの数を抑えられるとともに，応答レイテンシ特性に関しても，規模に関わらず 1 ホップ先との通信レイテンシしかかからない．

3.3.4　プロファイル

センサノードはネットワーク内でグローバルユニークなセンサ ID を持ち，この ID によって検索することが可能である．しかし，ユーザの利用形態の実態は，個別のセンサ ID を指定してデータ要求を行うことは多くない．もっとも期待される方法は，場所や種類，管理者といった属性値，プロファイル情報によってセンサノードの検索を行うものである．

現在想定しているプロファイルの例を表 3.1 に挙げてみる．

現在必要になっているプロファイルに対応していたとしても，今後もセンサの数・種類ともに増え続けるネットワークにおいては，新たなプロファイル項目が必要になってくることが十

表 3.1　プロファイル項目

location	bunkyo-ku.tokyo.jp	場所
latitude	35.4	緯度
atitude	139.46	経度
owner	live-e.org	所有者
type	temperature	センサ種類

分に考えられる．現在のように，クライアントサーバシステムで，サーバ上のデータベースに格納していると，最初に RDBMS のスキーマ決定をしているところで，プロファイルの種類が決まっている．ここでは，一箇所に情報が集まっていて扱いやすいにも関わらず，プロファイルの変更は簡単に行えるものではない．

提案アーキテクチャでは，この問題に対し，新たなプロファイル追加・変更に対して既存システムへの変更が不要であり，プロファイルの種類が数多くなっても負荷とはならない手法を提供している．

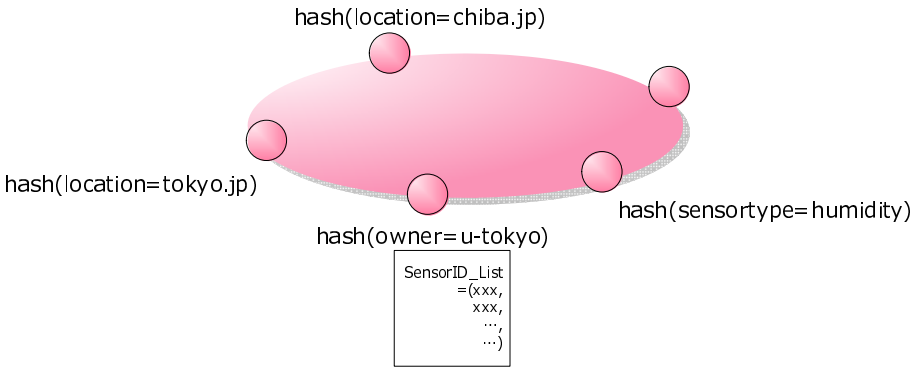

図 3.6　DHT にマップしたプロファイル

プロファイル情報はすべてルートノード上に保存される．センサは自分のプロファイルを登録する際は，プロファイル情報そのものの文字列と自分のセンサ ID を DHT 上のルートノードに送信すればよい．DHT では与えられたキーをもとにハッシュ空間上に一様にマッピングしてくれるため，センサ ID とプロファイル情報を示した文字列は，そのセマンティクスを隠蔽され，同様に扱うことが可能である．このため，既存のシステムに一切変更を加えることなく，センサノードやアプリケーションが新たなプロファイルを追加することが可能になる．また，ハッシュの特性により，同種のプロファイルが拡大してもルートノード内では一様に分散される．

3.3.5　階層化プロファイル

提案システムでは，前節で解説した DHT 上のプロファイルを図 3.7 のように階層化して利用している．プロファイルの親ノードが子ノードにどのようなプロファイルがあるか保持しておき，上流からより広い範囲で統合した検索をかけたり，通常 DHT では難しいとされてきた，数値での範囲検索にも対応する．

この手法もプロファイルの文字列を階層化したフォーマットで (例えば FQDN のように) 作っておくだけで，既存のノードに変更を加える必要なく実現できる．また，あるプロファイルのツリーにおいて，それぞれの階層にあたる管理ノードは，そのプロファイルに相当する文字列のハッシュによって分散して，対応するセンサ ID のリストを保持するため，負荷の点でも問題ない．

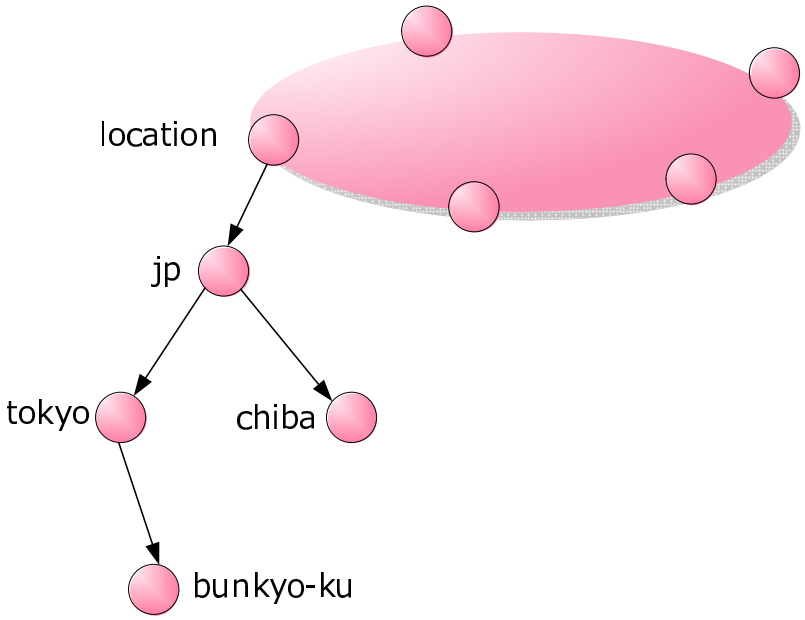

図 3.7　階層化プロファイル

ルートノードの形成に DHT を用いていることの有効性をここでは大きく確認することができる．上述のように，管理ノードの偏りをなくして均等に負荷を分散させることができ，可変長のセマンティクスを固定長のハッシュ値に変換することで，ネットワーク内のすべての検索を同じメカニズムですべて実現することが可能となる．

DNS では大規模なミラーリングやキャッシュ利用が必要となっていることからもわかるように，通常はツリー構造をした検索手法はルートに負荷がかかることが問題になる．図 3.8 のように，上位から順にたどっていく過程や，上位ノードの障害により，それより下位のツリーが利用不能に陥るといった問題は，このシステムでは起こり得ない．ここでは文字列のハッシュ値によってマッピングされているため，直接目的の管理ノードに到達することができるからで

あり，つまり $O(1)$ で目的のノードを発見できるというハッシュの特徴がそのまま生きている．

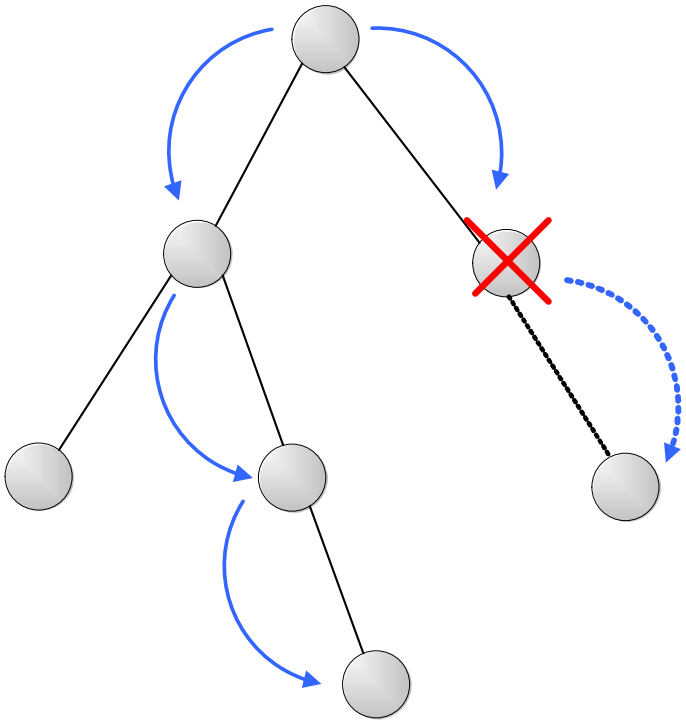

図 3.8　通常の階層化ツリー

3.3.6　送信クエリ数の低減

センサ数の増大にともなって，アプリケーションが送信すべき検索要求の数が増大する問題があることは既に述べた．本節では，その解決手法について述べる．

アプリケーションがデータ収集する対象が多数の場合とは，どのような状況か考えてみる．直接個別のセンサ ID によって，つまり計算機にとって負荷となりうるだけの多数のセンサ ID を個別に入力して，情報を収集しようと状況は考え難い．多数のクエリを送信する状況とは，プロファイル情報によるクエリであると，仮定することで解決の方法を探った．

提案システムでは，センサからのデータはそのセンサ ID のハッシュ値によってルートノード内で均等に分散されるという利点を述べたが，ここでは必要とするキャッシュが分散されてしまっていることが問題のひとつとして考えられる．そのため，図 3.9 のように，プロファイルによって関連づけられているが，それとは無関係に散らばっているキャッシュが，どこかにある程度集約されていると解決につながるのではないか，と考えられる．

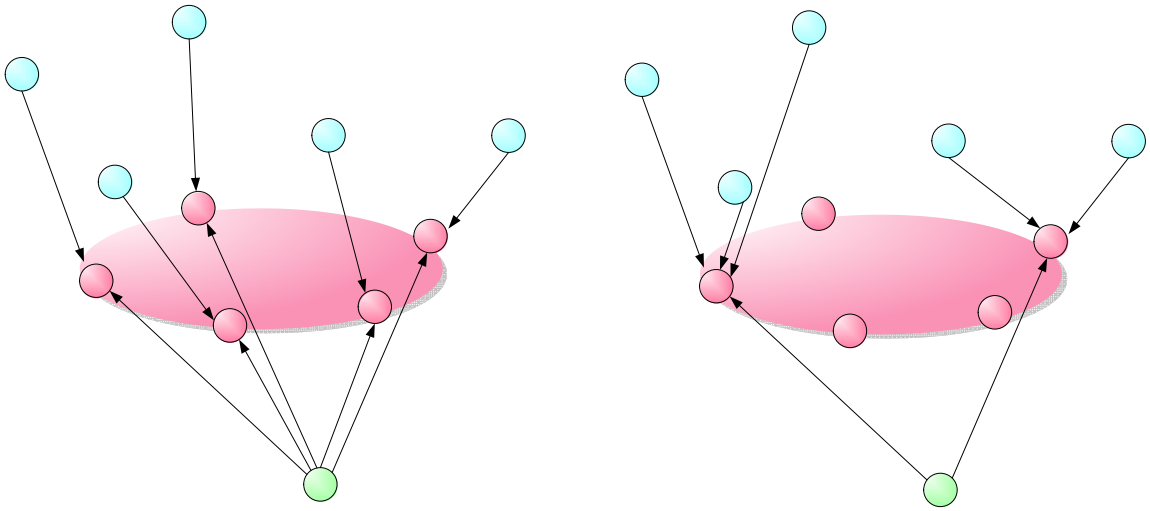

図 3.9　Aggregation of sensor cache

そこで提案アーキテクチャでは，階層化プロファイルを利用することでセンサデータのキャッシュを階層的に集約する手法を考えた．プロファイルには，該当するセンサ ID のリストがあり，これが大きくなるのが問題の原因でもある．そこで，センサ ID リストの増大にともない，各ルートノードごとの閾値で，図 3.10 リストを分割し他のルートノードに移譲してしまう．移譲されたルートノードには一定数のセンサ ID が任されており，それらのデータを予め取得してキャッシュするようにする．リストの増大に合わせて，この集約を繰り返すことで，アプリケーションノードが送信すべきクエリの数を低減することが可能となる．

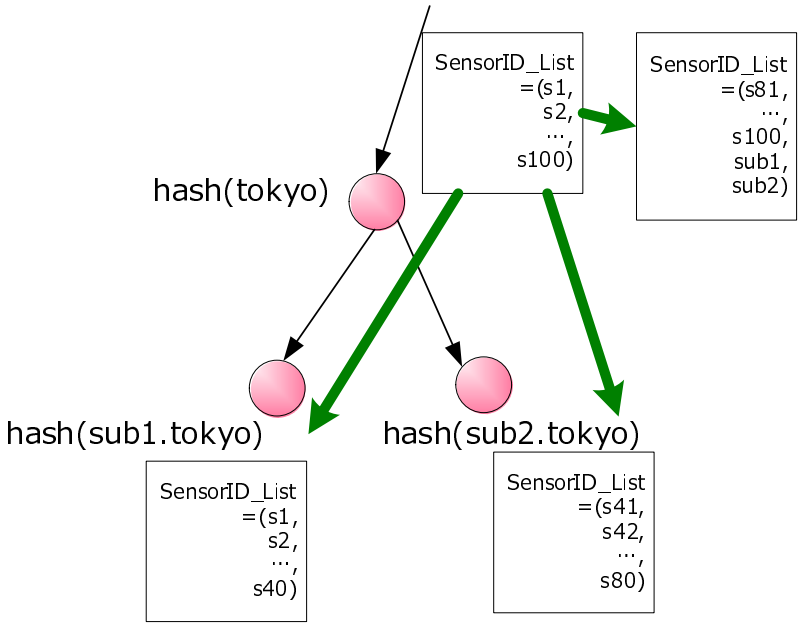

図 3.10　Aggregation of SensorID List

第 4 章

実装

本章では実装の詳細について解説し，どのような環境・条件で実験を行うかの計画を示す．

4.1 実装環境

各ノードを Java のプログラムとしてエミュレーションする．Java SE 5 を用い，ルートノードが形成するオーバレイネットワークの構築にはオーバレイ構築ツールキットとして公開されている，Overlay Weaver [29] を利用している．各ノードはそれぞれシェルを通してコマンド操作により制御する．

4.2 センサノード

センサノードは定期的に計測結果を送信する．
また，起動時に自身のプロファイル情報を登録する．

4.2.1 Bootstrap

センサが本システムに参加し，データを提供するためには，ルートノードのうちどれか 1 ノード，その IP アドレスとポート番号がわかればよい．耐障害性を高めるために，以下のように 3 つの発見手法を順に試す．

1. 前回接続時のノード情報キャッシュを利用
2. DNS による名前解決: 管理者が用意したルートノードをラウンドロビンで返す
3. 固定 IP アドレスをプログラム中に保持

ブートイメージの更新

センサは数が多く，また複雑な動作は期待できないため，管理が難しい．起動時に，または定期的に指定したホストにアクセスし，新しいイメージがあれば自己更新を行う．機能の変更や，新しいプロファイルを適用が低コストで可能になる．

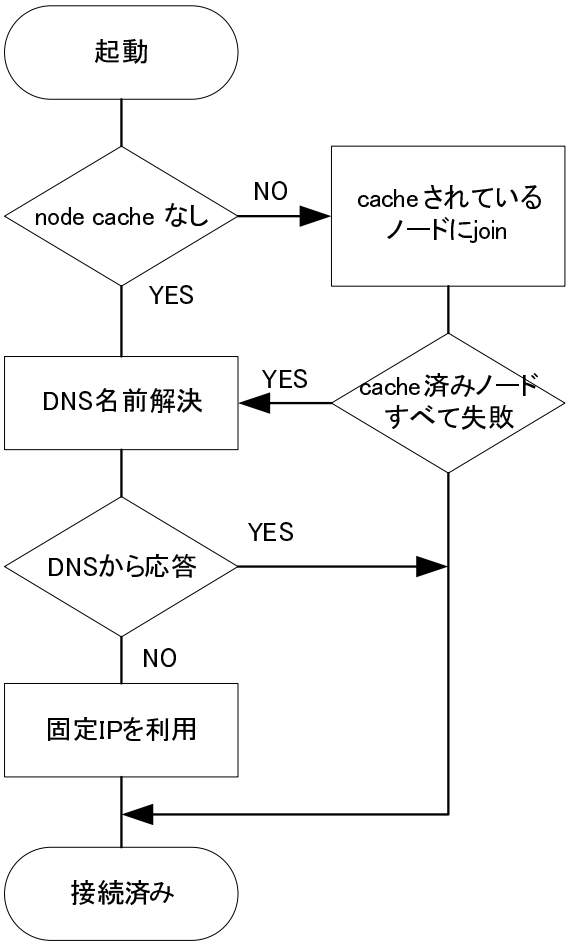

図 4.1　Bootstrap の流れ

4.2.2　測定データの送信

ルートノードに対して，以下の XML-RPC を利用して測定データを定期的に送信する．送信データには，自身のセンサ ID，測定値，タイムスタンプが含まれる．

- put(sensorID, value, timestamp)

4.2.3　プロファイル情報

通常，起動時にセンサ自身のプロファイル情報を登録する．測定データと同様に XML-RPC でルートノードに送信する．センサ ID のフィールドに，”profile-name=value”の形でそのまま送ればよい．システムとしては，ハッシュされた値によってルーティングするだけなので，基本的なデータの処理や流れに変更は必要ない．

- put_profile(profile_info, sensorID)

4.3 ルートノード

DHT に参加するノードである．センサからのデータが DHT 内でルーティングされ受信する．アプリケーションからのクエリを受け付けて，適切なルートノードに Redirect させる．

4.3.1 Bootstrap

一般にはアプリケーションノードとして参加し，そののちにルートノードとしての機能を持つようになる．ルートノードになるには，

- joinOverlay(host)

4.3.2 delegation

ルートノードで受け付けているクエリの数が，そのノードに取って高負荷になり始めたら，そのクエリに対する応答を他のノードに移譲するためリダイレクトメッセージを返す．

4.3.3 プロファイルの管理

プロファイル情報のハッシュ値をキーとし，データとして SensorID が渡される．センサデータと扱いは同様．

4.4 アプリケーションノード

センサから収集されたデータを利用するアプリケーションが動作するノード．一般のユーザはこの少なくとも機能を持つ．最初はアプリケーションの機能のみだが，上流の負荷に応じて，自身もキャッシュからデータを提供する側にまわる．

4.4.1 Bootstrap

ルートノードのうちどれか 1 ノードの IP アドレスとポート番号がわかればよい．センサノードでの処理と同様．

4.4.2 データの取得

ルートノードに対して XML-RPC で get メソッドを呼ぶ．キーは，センサ ID を用いても，プロファイルを用いても同じである．

表 4.1　実験 PC

	PC1	PC2	PC3	PC4	PC5
CPU	Pentium D 920	Pentium 4	Pentium 4	Pentium 4	Pemtium M
RAM	2GB	1GB	1GB	1GB	2GB
OS	Linux	Linux	Windows XP	Linux	Windows

- get_all(key): キーに該当するキャッシュにあるすべてのデータを取得
- get_current(key): キーに該当する最新のキャッシュを取得
- get(key, from_when): キーに該当し，指定時間以降のデータを取得

4.4.3　delegation

自身で指定した Degree(最大子ノード数) を越える接続があった場合，すでに接続済みの子ノードへのリダイレクトメッセージを返す．

4.5　実験環境

5 台の PC を用いて，それぞれ複数ノードをエミュレーションできるよう準備した．このうち，PC1 から PC3 までの 3 台，PC4 と PC5 の 2 台はそれぞれインターネットを通して接続可能な別々のネットワーク上に存在しており，図 4.2 のようになっている．それぞれの PC のスペックは表 4.1 である．

4.6　実験方法

各計算機で複数のノードをエミュレーションする．通常の UDP パケットが利用され，インターネットを介しても各ノード間の通信は変わらない．

動作するノードの数を増やしながら，それに対する応答時間や処理クエリ数を測定する．

評価すべき項目は，

- 参加ノード間での負荷分散がなされているか．
- 一部のノードに負荷が集中していないか．

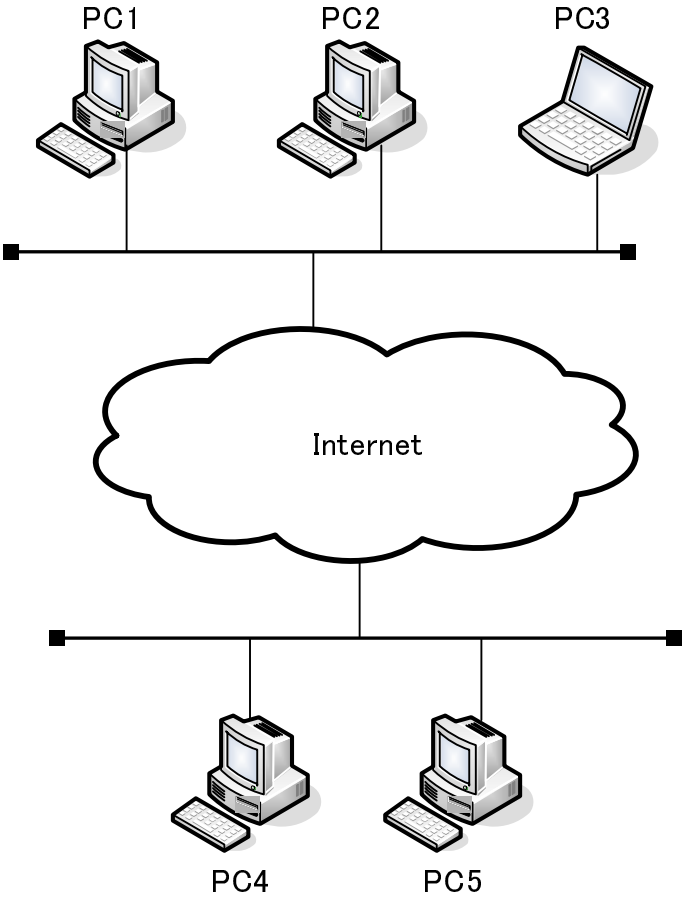

図 4.2　実験環境

第 5 章

結論

本章では，ここまで述べてきた本研究の内容をまとめ，また今後の課題について述べ，本論文の結論とする．

5.1 まとめ

本論文では，広域に展開する大規模センサネットワークがインターネット上に構築されたことで発生する問題点を明らかにし，そのネットワークの特徴から新たなデータ配信アーキテクチャの必要性を示した．提案アーキテクチャでは構成ノードがオーバレイネットワークを形成し，DHT やキャッシュを用いつつ，対象ネットワークの特徴を積極的に活用することで大規模化に対応可能なシステムを設計した．その中で種類，数，時間のそれぞれに関してスケールフリーであることを実現できるものとなっている．

5.2 今後の課題

今後の課題は，本論文で提案したアーキテクチャの全機能の実装を完成させ，大規模化したときの動作を確認する．また，このアーキテクチャによって実際に，応答レイテンシ特性やノードへの処理負荷が低減できたか，定量的な評価を行う必要がある．

謝辞

本研究を進めるにあたり，終始的確なご指導・アドバイスを頂いた東京大学大学院情報理工学系研究科教授 江崎浩博士に深く感謝いたします．

経験豊かな先輩として数多くの助言をいただいた，助手の山本成一氏，博士課程の岡部宣夫氏，吉田薫氏，藤田祥氏に感謝いたします．

同輩として，研究の苦楽をともにした澤村正氏，中島亮氏，賈洪光氏に感謝いたします．研究室でともに過ごした Guillaume Valadon 氏，山口龍太郎氏，安本直史氏，田中陽介氏，王智勇氏，落合秀也氏，神谷誠氏，Sergio Carrilho 氏，大口諒氏，坂本裕介氏，杉山哲弘氏，姜鵬氏に感謝いたします．事務処理など研究室の環境をサポートしていただいた秘書の高橋富美氏，田坂佳苗氏に感謝いたします．

最後に，このような研究の機会を経済的に援助してくださった両親，および研究生活を様々な形で支えていただいた多くの方々に感謝いたします．

参考文献

[1] “Live E! Project,” http://www.live-e.org/.

[2] C. Partridge, T. Mendez, and W. Milliken, “RFC 1546: Host Anycasting Service,” , Nov 1993.

[3] S.E. Deering, “RFC 1112: Host extensions for IP multicasting,” , Aug. 1989.

[4] W. Fenner, “RFC 2236: Internet Group Management Protocol, Version 2,” , Nov. 1997.

[5] D. Waitzman, C. Partridge, and S.E. Deering, “RFC 1075: Distance Vector Multicast Routing Protocol,” , Nov. 1988.

[6] J. Moy, “RFC 1584: Multicast Extensions to OSPF,” , Mar. 1994.

[7] D. Estrin, D. Farinacci, A. Helmy, D. Thaler, S. Deering, M. Handley, V. Jacobson, C. Liu, P. Sharma, and L. Wei, “RFC 2362: Protocol Independent Multicast-Sparse Mode (PIM-SM): Protocol Specification,” , Jun. 1998.

[8] S. Deering, D. Estrin, D. Farinacci, C. Liu, and L. Wei, “The PIM architecture for wide-area multicast routing,” IEEE/ACM Transactions on Networking (TON), vol.4, no.2, pp.153–162, 1996.

[9] S. Floyd, C. Liu, S. McCanne, and L. Zhang, “A reliable multicast framework for light-weight sessions and application level framing,” IEEE/ACM Transactions on Networking (TON), vol.5, no.6, pp.784–803, 1997.

[10] S. Paul, K. Sabnani, J. Lin, and S. Bhattacharyya, “RMTP: Reliable multicast transport protocol,” Selected Areas in Communications, IEEE Journal on, vol.15, no.3, pp.407–421, 1997.

[11] S. Androutsellis-Theotokis, and D. Spinellis, “A survey of peer-to-peer content distribution technologies,” ACM Computing Surveys (CSUR), vol.36, no.4, pp.335–371, 2004.

[12] 金子勇，Winny の技術，アスキー，Oct. 2005.

[13] I. Clarke, O. Sandberg, B. Wiley, and T. Hong, “Freenet: A distributed anonymous information storage and retrieval system,” Workshop on Design Issues in Anonymity and Unobservability, vol.320, 2000.

[14] I. Clarke, et al., “A distributed decentralised information storage and retrieval system,”

, 1999.

[15] S. Ratsanamy, P. Francis, M. Handley, and R. Karp, "A Scalable Content-Addressable Network," ACM SIGCOMM Conference, pp.161–172, Aug. 2001.

[16] I. Stoica, R. Morris, D. Karger, M. Kaashoek, and H. Balakrishnan, "Chord: A scalable peer-to-peer lookup service for internet applications," Proceedings of the 2001 SIGCOMM conference, vol.31, no.4, pp.149–160, 2001.

[17] A. Rowstron, and P. Druschel, "Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems," IFIP/ACM International Conference on Distributed Systems Platforms (Middleware), vol.11, pp.329–350, Nov. 2001.

[18] B. Zhao, J. Kubiatowicz, and A. Joseph, "Tapestry: An Infrastructure for Fault-tolerant Wide-area Location and Routing," Computer, vol.74, Apr. 2001.

[19] P. Maymounkov, and D. Mazieres, "Kademlia: A peer-to-peer information system based on the XOR metric," Proceedings of the 1st International Workshop on Peer-to-Peer Systems (IPTPS'02), vol.258, p.263, 2002.

[20] M. Castro, P. Druschel, A.m. Kermarrec, and A. Rowstron, "Scribe: a large-scale and decentralized application-level multicast infrastructure," Selected Areas in Communications, IEEE Journal on, vol.20, no.8, pp.1489–1499, 2002.

[21] M. Castro, P. Druschel, A. Kermarrec, A. Nandi, A. Rowstron, and A. Singh, "SplitStream: high-bandwidth multicast in cooperative environments," Proceedings of the nineteenth ACM symposium on Operating systems principles, pp.298–313, 2003.

[22] Y. Chu, S. Rao, and H. Zhang, "A case for end system multicast," ACM SIGMETRICS Performance Evaluation Review, vol.28, no.1, pp.1–12, 2000.

[23] Y. Chawathe, et al., Scattercast: an architecture for internet broadcast distribution as an infrastructure service, Ph.D thesis, University of California, Berkeley, Spring 2000., Dec. 2000.

[24] P. Francis, "Yoid: Extending the Internet Multicast Architecture," (Unpublished paper), http://www.aciri.org/yoid/docs/index.html, Apr. 2000.

[25] D. Pendarakis, S. Shi, D. Verma, and M. Waldvogel, "ALMI: An Application Level Multicast Infrastructure," Proceedings of the 3rd USNIX Symposium on Internet Technologies and Systems (USITS '01), pp.49–60, Mar. 2001.

[26] B. Zhang, S. Jamin, and L. Zhang, "Host multicast: a framework for delivering multicast to end users," INFOCOM 2002. Twenty-First Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE, vol.3, 2002.

[27] J. Jannotti, D. Gifford, K. Johnson, M. Kaashoek, and J. O'Toole, "Overcast: Reliable multicasting with an overlay network," Proc. of the 4th Usenix Symp. on OSDI 2000, Oct. 2000.

[28] S. Ramabhadran, J. Hellerstein, S. Ratnasamy, and S. Shenker, "Prefix Hash Tree: An

indexing data structure over distributed hash tables," Proceedings of the 23rd ACM Symposium on Principles of Distributed Computing.

[29] 首藤一幸，田中良夫，関口智嗣，“オーバレイ構築ツールキット Overlay Weaver,” 情報処理学会 論文誌: コンピューティングシステム，vol.47，pp.358–367，Sep. 2006.

発表文献

[1] 石田真一，江崎浩，“オーバレイマルチキャスト技術を用いた大規模センサ情報配信システムの検討”，信学総大 B-7-100，Mar. 2007