ロケーションオリエンテッドな情報空間の構築 *

清水 奨 †
NTT ソフトウェア研究所　ソフトウェア技術研究部‡

概要: 携帯端末からインタネット上の情報サーバを利用する場合、利用者の物理的な位置が従来より大きな意味を持つ。しかし、これまでの情報提供は利用者の位置を特別扱いすることはなかった。本稿では、今後グローバルな情報提供と同様、ローカルな情報提供が多くなるという展望に立ち、利用者の位置をもとにローカルな情報サーバを検索するシステム (ロケーションオリエンテッドな情報空間) の構築を提案する。DNS(Domain Name System) を基に実装する。

1　はじめに

インタネットの拡大にともない、インタネット上での情報提供が活発に行なわれるようになってきている。WWW (World Wide Web) [15] に代表される情報サーバの数は指数関数的に増大しており、留まるところを知らない。

一方、半導体技術の進歩により、高性能な携帯端末が安価に手にはいるようになりつつある。携帯電話の無線ネットワークと組合わせれば、いつでもどこでもインタネット上のサービスを享受できる。

以上の背景から、携帯端末からインタネット上の情報サーバにアクセスしたいというニーズが高まると予想される。この場合、固定されたデスクトップ環境からのアクセスと異なり、携帯端末の利用者に特有の query(問い合わせ) が多くなると考えられる [16]。例えば、「近くで食事できる所は?」「一番近くで空いている駐車場は?」などである。

本稿では、このような query に迅速に答えるためのアプローチとして、位置 (ロケーション) の情報をキーに持ち、情報サーバの論理アドレスを値に持つ分散データベース (これを、ロケーションオリエンテッドな情報空間と呼ぶ) の構築を提案する。実現のための技術的課題を明らかにし、既存の分散データベース (archie, WHOIS++, DNS) の不足点を指摘する。また、DNS を基にロケーションサーバを構築する例をあげる。

以下、2 章ではここで構築しようとしているロケーションオリエンテッドな情報空間について議論し、必要とされる機能を指摘する。3 章では既存のデータベースについて述べ、不足点を議論する。4 章で DNS をベースにした実現例について述べ、5 章にまとめを置く。

2　ロケーションオリエンテッドな情報空間

本章では、構築しようとしている情報空間について議論する。議論を進める上での前提をいくつかあげ、サービスイメージを紹介し、技術的課題について述べる。

2.1　前提

2.1.1　情報サーバの形態

ロケーションをキーにした情報検索を行なうには、あるエリア[1]における全ての情報を一

*Towards building the location-oriented information space

†Susumu SHIMIZU, Email:shimizu@slab.ntt.jp
‡Software Engineering Laboratory, NTT Software Labs. 3-9-11 Midori-cho, Musashino-city, Tokyo JAPAN

[1]エリアの最適な大きさや、階層構造などについてはここでは触れない

つのサーバに集中[2]しておく方法 (集中管理型 - 図1) と、あるエリアにおける情報サーバの概要とアドレスを一つのサーバ (ロケーションサーバ) で管理し、詳細情報は提供者が個別に管理する方法 (分散管理型 - 図2) がある。

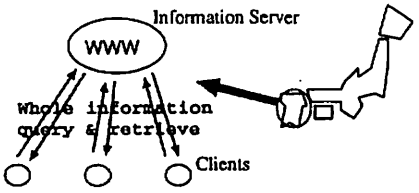

図 1: 集中管理型サーバ

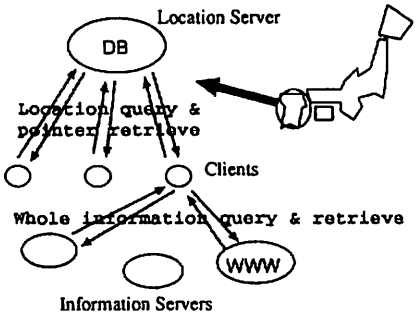

図 2: 分散管理型サーバ

前者の方法は、エリア内の情報管理を一つの組織でやるため、提供内容の変化が少ない場合にうまく機能する。例えば各地の観光案内所など、情報提供者がある程度決まっていて、内容も一度登録してしまえばその後の維持の手間があまりかからないような場合である。しかし容易に想像がつくように、エリア内の情報がダイナミックに変動する場合には管理者の負担が増すため、対処が困難である。また変更の度に管理者に要請する必要があるので、情報提供者の側にも心理的な抵抗が生ずる。

一方後者の方法は、前者に比べ柔軟である。各サーバが提供する情報の管理は個々のサーバ管理者に任されているため、よりダイナミックな情報提供が可能となる。例えばスーパーがタイムサービスの安売り情報を自分のサーバで流す場合、ロケーションサーバの登録内容はそのままでよく、提供する情報自体を自由に更新できる。

以上の議論より、本稿では後者の分散管理型のサーバ形態を前提とする。

2.1.2 サーバの位置

各情報サーバ、ロケーションサーバについて、どちらも位置が固定されているということを前提にする (図3左)。

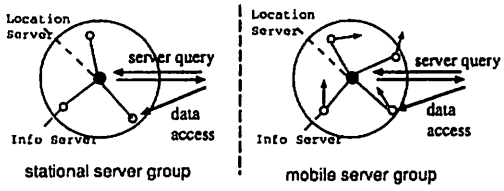

図 3: 情報サーバの種類

我々の構想では、将来的に情報サーバが移動する (図3右) 場合にも対応することを考えている。これは携帯端末が非常に進化し、どこにでもある状態 [7] となった時、ユーザ一人一人の端末が自動的に情報を発信するようになると考えられるためである。

しかし現状技術の制限 (端末技術、通信容量) により、携帯端末が大量の情報を提供することは当面先のことであると考えられることから、本稿では情報サーバは固定されているものとする。

2.1.3 ロケーションの表現

ロケーションの表現には行政区画を用いるやり方と、GPS(Global Positioning System) による絶対位置座標によるやり方が考えられ、それぞれ一長一短がある。

- 行政区画を用いる
  日本、東京、武蔵野市、緑町のような行政区画による表現は階層化されていて責任範囲が分散しており、ロケーションサーバ構築に適していると思われる。しかし旅行先など不慣れな土地では、行政区画

[2]実装上、分散データベースで構成される場合も含む

による指定方法はわかりにくい場合がある。

- GPS を用いる
  GPS の長所は、利用者が特別な操作をしなくても現在位置がわかるという点にあるが、経緯度情報だけでは責任範囲の分散が難しく、スケーラビリティに乏しい。

GPS を使う場合でも、カーナビゲーションシステムのように地図情報を持てば経緯度から行政区画名を得られるので、行政区画による query が可能である。したがって、ここでは行政区画による表現を前提とする。

2.2 サービスイメージ

ロケーションオリエンテッドな情報空間の実現イメージを図 4に示す。図左側は利用の様子を、右側は地理的な配置を示している。

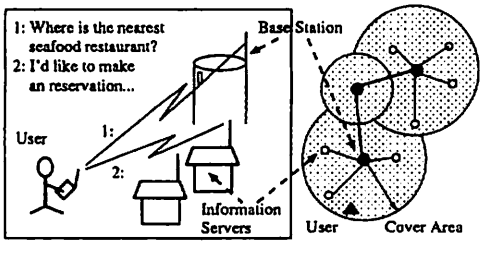

図 4: 実現イメージ

図ではレストランの予約を入れるシーンを例にあげている。はじめにロケーションサーバに、「最も近くのシーフードレストランは?」と問い合わせ、レストランが提供する情報サーバのアドレスを得る。次に、レストランの情報サーバにアクセスして予約を入れる、というストーリである。もちろん、例にあげたような携帯端末からの利用だけでなく外部ネットワークからの利用もあるだろう。

ただし、このようなイメージを描く際の社会的な前提として、次の二点がある。

- 有線、無線いずれかのネットワークのインフラが整備されており、誰でもすぐに情報サーバを運営できる環境が存在すること。
- 携帯端末が普及しており、商店や企業が個別に情報サーバを運営しようと思う程度の需要が期待されること。

これらは技術的な問題ではないので本稿では妥当性について触れることはしないが、そう遠くない将来に実現すると考えている。

2.3 技術的課題

信頼性やパフォーマンスといった基本的な特性においても十分な性能が求められるが、ここでは次の二点を取り上げる。

2.3.1 スケーラビリティ

非常に多くの情報サーバを管理するため、スケーラビリティが問題となる。 CCITT の X. 500[4] や DNS[10] に見られるような階層構造を導入したり、 whois++[3] のようなインデックスサーバが必要になるだろう。

スケーラビリティはグローバルな規模の情報検索システムを考えた場合必ず生じる問題であり、上記のどのシステムでも解決を試みている。我々の興味はロケーションをキーにしたサーバ検索にあるが、この点に特化した独自の解法を探索するのは得策でないと考え、汎用の検索システムで研究された機構をできるだけ利用する立場をとる。

2.3.2 情報サーバの自動管理

爆発的な普及を見せている WWW サーバから学べることの一つに、情報サーバの新設・更新・廃止といった情報空間のダイナミズムの激しさがある。このように動きの激しい情報サーバのロケーションサーバを作るためには、管理をできるだけ容易にすることが求められる。少なくとも情報サーバの登録、削除などのデータ管理は自動化することが望ましい。

3 既存データベースの活用

本章では、インタネット上の既存分散データベースを紹介し、ロケーションサーバ構築のベースとするための不足点を議論する。

3.1 archie

archie はインタネットのリソース探索を容易にするためのディレクトリサービスを提供するものである [1]。スクリプトによる自動的なデータ収集が大きな特長であり、現在は主に anonymous FTP サイトのファイル情報の提供に使われている。利用者は検索したいファイル名を archie サーバに入力し、archie サーバは入力された名前を含むか、一致するファイルがどの FTP サーバのどのディレクトリに存在するかを答える。

archie の構造を図 5 に示す。

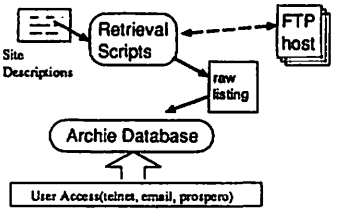

図 5: archie の構造

archie は管理すべき FTP サーバが非常に多くなった場合でも、パワフルなサーバマシンが利用できれば問題はないという思想であり [1]、スケーラビリティの問題には実質触れていない。しかし、FTP サーバのデータを集中して持つ必要があるため、ロケーションサーバ構築に用いるのは困難であると思われる。

また、あらかじめ anonymous FTP サービスをしているホストを手動登録しなくてはいけないため、FTP サーバの数が流動的な場合は管理の手間が増大する。

3.2 whois++

whois++[3] はインタネットにおけるホワイトページ[3]である whois[6] のデータモデルに拡張を加え、分散インデックスサービスを可能にしようとするものである。

実際のサーバの他に、インデックスサーバを用意し、サーバの情報を抽出して保管しておく。インデックスサーバの構造を図 6 にあげる。ツリー構造ではなくメッシュ構造である点が従来と異なる。また、概念的なレイヤ (top level, first level...Nth level) をもつ。

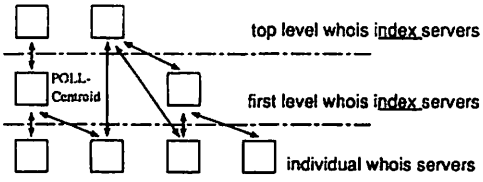

図 6: WHOIS++ のインデックスメッシュ

ドラフト [3] には情報サーバが増えた時のスケーラビリティの問題をどう回避するかについて具体的な記述がない。ロケーションサーバの構築に利用するためには、スケーラビリティを確保するため、インデックスサーバが抱える情報サーバの数を抑える必要があるだろう。また、各情報サーバ、インデックスサーバにはあらかじめユニークな名前と ID を与えておく必要があり、自動登録の機能もドラフトの段階では未定義なため、流動的な情報サーバの管理には向かない可能性がある。

whois++ は広域インデックスサービスのプラットフォームとなる可能性を秘めている。例えば加嶋らは whois++ の機構を応用して、archie サーバをリストに持つデータベースへの応用を試みている [17]。

3.3 DNS

DNS (Domain Name System)[10] はホスト名から IP アドレスやメール配送経路を知るために用いられているシステムで、ドメインによるツリー状の階層構造 (図 7) と、下位のドメインへの権限委譲[4]を特長としている。

もともとマシン名から IP アドレスを知るための機構であるため、現在の実装でも扱えるデータ型は限られており、例えばアドレス、別名、マシンアーキテクチャ、メール中継マシンなどである [9]。しかしプロトコルは柔軟にできており容易に拡張が可能である。例えば

[3]個人向け電話帳、人名からメールアドレスや電話番号などを検索するサービス

[4]つまり、図 7 で slab に属すマシンは slab が責任を持って管理する

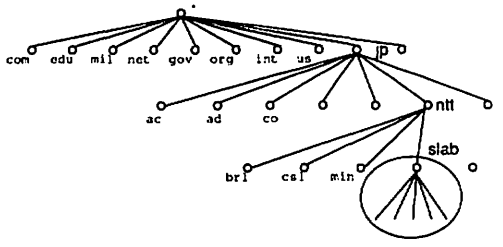

図 7: DNS のドメインツリー

[11] ではネットワークアドレスや TCP ポートの検索にも使えるようなデータレコードの記法を提案している。また [2] や [13] では新しいアドレス空間やネットワーク上の別のデータベースなどに対応するために新規のデータレコードの追加を提案している。

また、[12] ではデータレコードのうち TXT (任意のテキストを記録するレコード) に変数と値の組を入れ、新しいレコードタイプを追加することなく多目的に利用することを提案している。

以上のように、様々な拡張が提案されており、ロケーションサーバ構築のベースとしては現段階で最も使い易いプラットフォームである。

前章であげた要求のうち、スケーラビリティについては、320 万を超えるホストと、46,000 ものドメイン [8] の繋がった Internet で実際に利用されているという実績がある。

しかし、データの自動的な管理についてはこれまで考慮されていない。

4 構築実験

この章では前章までの議論を基に、ロケーションサーバの構築例を議論する。サーバのトポロジや現在研究されているデータベースなど、どのようなインフラが整備され普及するかに影響を受けると思われるため、1st step として既存の仕組みの範囲で実装を試みる。

4.1 実装のための前提

実験を行なうに当たり、次の前提を置く。

既存の枠組を利用

ロケーションサーバ構築のために全く新しい枠組を構築することはせず、既存の枠組の範囲内で実装する。理由は、次の通り。

- すぐに実装可能なレベルで実際に動作させることにより、本構想の評価を先行して固めることができる。
- 従来から蓄積されたノウハウを活用できるため、他のシステム管理者に実験の協力を仰ぎやすい。

DNS ベースで

前述の方針に従い、ベースには DNS システムを用いる。理由を次にあげる。

- whois++ がまだドラフト段階であるのに対し、全世界で稼働中。
- primary/secondary の設定、キャッシュ機構など優れた基本性能を持っている。
- 今なお発展中とはいえ開発から 10 年以上経っており、運用のノウハウを含めほぼ確立された技術である。
- 権限委譲のポリシーがロケーションサーバ構築に適しており、スケーラビリティの問題を現実的に回避できる。

4.2 DNS による実現

ここでは DNS をどう利用するかについて議論する。

4.2.1 ドメインの構成

2.1.3でロケーション表記に行政区画を用いるという前提を置いたので、ドメインも行政区画にしたがって構成する。

JPNIC のドメイン割り当て表 [5] を見ると、各都道府県と政令指定都市に対して既にドメインが割り当てられている[5]。

ロケーションサーバ構築にこうした既存の地域ドメインを活用するというのは自然な考えである。新規にロケーションドメインを導

[5] 米国でも、sacramento.ca.us や ci.sunnyvale.ca.us というような地域ドメインがある

入して構築するよりもコンセンサスが取りやすい。

現在地域ドメインを用いている組織は非常に少ないため、これらのネームサーバはかなり上位のマシンが兼ねているが、実際に情報サービスが盛んになってきたら個別にネームサーバを運用すれば良い。

例として、tokyo.jp. の下に musashino.tokyo.jp. を作るとする。このドメインの管理権限を ls.slab.ntt.jp というマシンが持つ場合の様子を図8に示す。

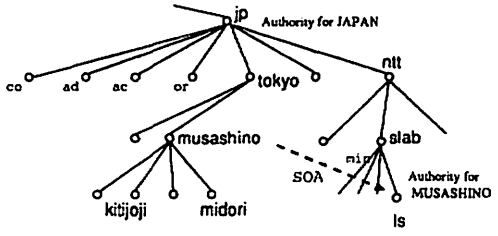

図 8: 既存の地域ドメインの利用

ここで、各ドメインの SOA(Start of Authority, ドメインの管理権限) をどのように選択するかという問題があるが、運用上の問題であるとしてここでは触れない[6]。

4.2.2 サーバの構成

サーバの構成をどのようにするかという点で、選択肢が二つある。

- ネームサーバとロケーションサーバを兼用する方法
- ネームサーバとは別にロケーションサーバを運用する方法

後者の方法を取ると、ロケーションサーバを知るためにネームサーバに

```
musashino.tokyo.jp.  IN TXT "LS=ls.ntt.jp"
```

のようなレコード[7]を持たせる必要があるが、次にあげる理由から後者の方法をとる。

- ロケーションサーバのデータファイル (ゾーンファイル) は自動更新の実験対象となるため、安全を考えて他の情報と切りわけた方がよい。
- ロケーションサーバをネームサーバと別にしておくことで、ネーム空間とロケーション空間を別々に運営できる。将来、ロケーションサーバで管理すべき情報サーバが増えてもネームサーバと別に運営されていれば管理しやすい。

4.2.3 ゾーンファイル

前述の方針により、実際のロケーションデータは、ロケーションサーバのゾーンファイル (DNS のデータファイル) に記述する。記述法は、TXT レコードを活用する [12] の方法を採用する。これは、既存の実装に対する変更点が最も少ないとの判断による。

検索の最小単位 (町など) をキーとして持つことにし、同じドメインのマシンと区別するために、l. をつける。例えば東京都武蔵野市緑町ならば、musashino.tokyo.jp. ドメインのゾーンファイルに l.midori というキーを用意して次のように記述する[8]。

```
; musashino zone file
musashino.tokyo.jp. IN SOA ls.slab.ntt.jp.
    shimizu.slab.ntt.jp. (
    9409121600 10800 3600 3600000 86400 )
ls.slab.ntt.jp. IN  A   163.138.21.10
l.midori IN TXT "hospital=http://midori-iin"
l.midori IN TXT "fastfood=http://midorimac"
```

TXT フィールドの属性名 (イコールの左) には hostpital, fastfood など情報提供者の属性を入れ、属性値 (イコールの右) に情報サーバのプロトコル名、IP アドレス又はドメイン表記アドレスを URL 形式 [14] で入れておく。これにより、query の流れは図9のようになる。ただし、ロケーションサーバは client にとって既知とする。

[6] ドメイン配布と同様 JPNIC が行なうのがいいのだろうか?

[7] [12] による方法

[8] これは暫定案であり、細かい点については今後変更される可能性がある。

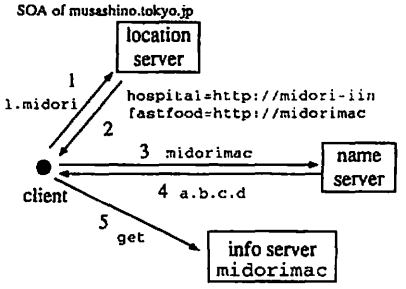

図 9: query の流れ

4.2.4 情報サーバ管理

DNS には自動的にゾーンファイルを管理する機構がないので、DNS とは別のプログラムにゾーンファイルの管理と named[9]への reload (再読み込み) を行なわせる (図 10)。

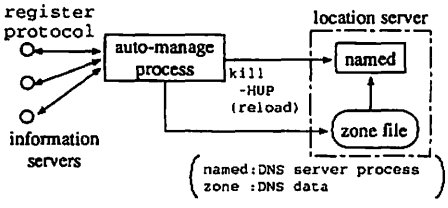

図 10: ゾーンファイルの自動管理

図 10中、"register protocol" とは情報サーバとロケーションサーバ間での通信プロトコルである。これは、例えば次のようになるだろう。

1. 登録
   - 情報サーバから登録プロセスに「追加」のシグナルとともにサーバ属性、プロトコル：サーバ名が送られる。
   - ロケーションサーバから ack が返る
2. メンテナンス
   登録プロセスは、管理している情報サーバに定期的にポーリングをかける。反応がない情報サーバがあったら、そのサーバに対応するレコードをコメントアウトし、「停止中」マークをつける。
3. 削除
   情報サーバから登録プロセスに「削除」のシグナルが送られてくるか、一定期間以上「停止中」だった情報サーバがある場合は、それに該当するレコードをデータファイルから削除する。

[9]DNS のサーバデーモンプログラム

5 まとめ

ローカルな情報を提供する情報サーバのインデックスを構築する手法について議論し、DNS をベースに実装する方法を述べた。

インタネット上の情報空間は膨張し続けており、次第に求める情報がどこにあるのか探しにくくなってきている。多くの研究者がこの問題にとり組んでいるが、本稿で我々がとったのは情報空間をロケーションに基づいて絞り込んでしまおうという単純なアプローチである。

このアプローチは、携帯端末の利用者など、従来の端末に比べローカルな情報に対する需要を強く持つと思われる層に利用者像を限定しているため、汎用インデックスを作りあげるアプローチに比べ有効範囲が狭い欠点があるが、現在の分散データベースの仕組みにわずかに手を加えるだけで実装できるという利点がある。

今後は、議論した方法を実装し、運用実験を通じて本アプローチの有効性を評価する予定である。

謝辞

本研究を進めるにあたり議論して頂いた NTT ソフトウェア研究所サービスソフトウェア方式研究グループ、平川 豊グループリーダ及びグループ員の皆様に感謝します。

参考文献

[1] A. Emtage and P. Deutsch. archie - An Electronic Directory Service for the Internet. In *USENIX/Winter*, pp. 93-110, 1992.

[2] B. Manning. DNS NSAP RRs. *RFC1348, ftp://ds.internic.net/rfc*, Jul. 1992.

[3] C. Weider, J. Fullton and S. Spero. Architecture of the WHOIS++ Index Service. *Internet Draft, ftp://nic.merit.edu /documents/internet-drafts /draft-ietf-wnils-whois-03.txt*, Jul. 1994.

[4] ITU. BLUE BOOK Volume VIII, Fascicle VIII 6,8(データ通信網 X シリーズ 勧告その 5). (財) 日本 ITU 協会, 1988.

[5] JPNIC. Allocated Domains in JP. *JPNIC DB*, ftp://ftp.nic.ad.jp /pub /jpnic /domain-list.txt 1994.

[6] K. Harrenstien, M. Stahl and E. Feinler. NICNAME/WHOIS. *RFC954, ftp://ds.internic.net/rfc*, Oct. 1985.

[7] M. ワイザー. 21 世紀のコンピューター. 日経サイエンス, pp. 60-70, Nov. 1991.

[8] NIC.MERIT.EDU. *ftp://nic.merit.edu /nsfnet /statistics /history.hosts.* available via anon-FTP, 1994.

[9] P. Albitz and Cricket Liu. *DNS and BIND.* O'Reilly & Associates, 1992.

[10] P. Mockapetris. Domain names - implementation and specification. *RFC1035, STD13, ftp://ds.internic.net/rfc*, Nov. 1987.

[11] P. Mockapetris. DNS encoding of network names and other types. *RFC1101, ftp://ds.internic.net/rfc*, Apr. 1989.

[12] R. Rosenbaum. Using the Domain Name System To Store Arbitrary String Attributes. *RFC1464, ftp://ds.internic.net /rfc*, May 1993.

[13] R. Ullman, P. Mockapetris, L. Mamakos, C. Everhart. New DNS RR Definitions. *RFC1183, ftp://ds.internic.net/rfc*, Oct. 1990.

[14] T. Berners-Lee, L. Masinter, M. McCahill. Uniform Resource Locators (URL). *ftp://nic.nordu.net/ internet-drafts/ draft-ietf-uri-url-07.txt*, Sep. 1994.

[15] Tim Berners-Lee, Robert Cailliau et al. World-Wide Web: The Information Universe. *Electronic Networking: Research, Applications and Policy*, Vol. 1, No. 2,, Spring 1992.

[16] Tomasz Imielinski and B. R. Badrinath. Data Management for Mobile Computing. *SIGMOD RECORD*, Vol. 22, No. 1, pp. 34-39, Mar. 1993.

[17] 加嶋 啓章, 平原 正樹. 分散インデックスサービスによる広域検索の効率向上とその応用事例. 第一回 JAIN CONSORTIUM Symposium 論文集, pp. 39-44. JAIN Consortium, 1994.