第 8 部

Mobile and Ubiquitous Computing

第 1 章

階層的手法に基づくネットワーク自動構成プロトコル

1.1 背景・目的

近年まで、計算機を取り巻くネットワークの環境は希にしか変化しなかった。そのためIP アドレスの割当や設定、(ルータの IP アドレスなどの) ネットワーク環境に関する情報の取得と設定などを手作業で行なうことは現実的であった。しかしインターネットを取り巻く状況の変化により、新たな問題が発生している。

第一に携帯型計算機の普及により、設定変更の頻度が増えた。設定作業は煩雑かつ (手動では) 誤りやすいものであり、利用者への負担となっている。第二に近い将来に実現される “Ubiquitous Computing” の環境 (身の回りの物すべてに計算機が含まれる) では、極めて多数の計算機が自律的にネットワークを構成し、ネットワーク単位で接続や切断を繰り返す。このような状況において、手動によるネットワーク設定管理は実用的でない。

第三にネットワークアドレスの枯渇 [29] と、経路情報数の急増によるネットワーク運用への支障 [14] の問題がある。未割当の IP アドレスは減少し、近い将来に枯渇する。また経路情報数が激増し、メモリの少ないルータでは処理しきれないことがある。抜本的対策として IPv6[19] が開発されているが、IPv6 へ移行するまでの対策も提案されている。

まず一般に大きなサブネットほどアドレスの利用率が低下する傾向にある。したがって適切な大きさにネットワークを分割してアドレスの利用率を向上させ、同時に複数のネットワークへの経路情報を一つに集約して経路数を減らしている [14]。これらの対策はネットワークアドレスの付け替えにより更に効果的なものとなる。しかしアドレスの付け替えの労力は大きく、また付け替えの前後を通じてインターネットとの接続を確保する体系的手法が確立されていない。これらの理由によりアドレス付け替えは敬遠される傾向にある。

これらインターネットを取り巻く状況の変化に対応した新しいネットワーク自動設定管理機構を実現することが本研究の目的である。その際に以下の 3項目に着目しつつ、特に IP アドレスという共有資源の自動割当、回収および設定方法について考察する。

1. **ネットワークの設定および管理を自動化する**

複数セグメントとルータからなる小〜中規模ネットワークで、アドレス等の資源割当や、ルータアドレス等のネットワーク環境に関する情報の設定を自動化する。

2. **ネットワークアドレスの付け替えを支援する**

   アドレスの効率的利用や経路情報の集約に必要なアドレス付け替えを支援する。

3. **ネットワークアドレスの効率的利用を実現する**

   現状では利用率の低い IP アドレスを節約し、アドレス空間を効率的に利用する。

1.2 ネットワーク設定管理手法の現状

本節では関連研究とそこで用いられているネットワーク設定管理ツールについて概観し、これらがインターネットを取り巻く状況の変化に適さなくなりつつあることを示す。

- **Reverse Address Resolution Protocol (RARP)**
  RARP[30] は、MAC アドレスをネットワークアドレスへ変換する。サーバには各ホストのアドレスを静的に登録するため、個々の計算機の設定を集中管理できる。RARP は計算機の自動設定機構の一部を実現するが、アドレスの動的割当、ネットワーク環境に関する情報の自動設定、アドレス付け替え、回収などは実現していない。RARP を拡張した Dynamic RARP [31] ではアドレスの動的割当と再利用が可能である。
- **Network Information Protocol (NIP)**
  NIP[32] ではアドレスを動的に割り当てる。サーバはクライアントに未使用と思われるアドレスの集合を通知する。その中からクライアントはアドレスを 1 つ選択し、ARP 要求を送信して未使用であることを確認する。もし使用中ならば選択し直す。NIP では、まれにアドレスの重複が起こる。NIP は計算機の自動設定と動的アドレス割当という要求を部分的に解決していて、取得できる情報には IP アドレス、ネットマスクやデフォルトルータなどがある。しかしアドレスの付け替え、回収、ネットワーク全体の自動設定などは実現していない。
- **Bootstrap Protocol (BOOTP)**
  BOOTP[33] は、IP アドレスやルータアドレスなどのネットワーク環境に関する情報を自動的に設定する。アドレス割当は静的であり、一時割当や未使用アドレスの再利用はできない。BOOTP によって IP アドレスの集中管理が実現され、また各セグメント内で共通な設定は自動化できる。ただし管理者は個々の IP アドレスを手動で割り当てなければならない。
- **Point-to-Point Protocol (PPP)**
  PPP[34] は、コネクションの確立に Link Control Protocol (LCP) を用いる。IP 用の LCP である Internet Protocol Control Protocol (IPCP)[35] では、コネクション確立時に両端の IP アドレスを設定する。IPCP は実際にアドレスを割り当てる方法は規定しておらず、通常は受け側の管理者が静的に設定する。またネットワークアドレス

の回収はコネクションが切断された時点で行なわれる。しかしネットワーク環境に関する情報の取得設定やアドレスの付け替えについては考慮されていない。

- **Dynamic Host Configuration Protocol (DHCP)**
  DHCP[36, 37, 38] は BOOTP を拡張したもので、IP アドレスの割当に 4 つの種類がある。静的に割当アドレスを設定しておくのが静的割当 (BOOTP 相当) である。サーバが割り当てるアドレスを自動的に選ぶのが動的割当 (有効期限つき) と自動割当 (無期限) である。最後の 1 つはアドレスを割り当てず、各種の設定情報だけを通知する (アドレスは手動あるいは IPCP などの外部機構で設定済の場合に用いる)。DHCP はアドレスを自動的に割り当てるため、管理者や一般ユーザの作業が軽減される。アドレスの再利用も実現しているが、回収の機構はない。またネットワーク全体の設定も考慮していない。DHCP を用いたアドレス付け替えは、文献 [39] で提案されている。

- **IPv6 Autoconfiguration**
  IPv6[19] のアドレスは 128 ビットの長さを持つため、アドレスを手動で入力することは困難である。そのため初めからアドレスの自動設定機構が組み込まれている。これには Stateless と Stateful の 2 つのモードがある。Stateful モードの自動設定機構としては DHCP を IPv6 用に拡張した DHCPv6[40] が提案されている。Stateless モード [41] では、例えば 6 オクテットの Ethernet アドレスと 10 オクテットのプレフィクスを連結することでアドレスを決定する。プレフィクスには RFC などで定義されるものと、ルータがアナウンスするものがある。どちらのモードもアドレスの付け替えについては考慮しているが、ネットワーク単位の自動設定などは考慮していない。

1.3　階層化モデルによるネットワーク自動設定機構の提案

はじめに新しいネットワーク設定管理機構の設計目標を示す。次に、これらの目的には階層モデルを用いた管理機構が適していることを明らかにし、システムの設計概要を述べる。

1.3.1　設計目標

新しいネットワーク設定管理システムを設計する際に「1. (現実のネットワークや Ubiquitous Computing で用いるために) 負荷分散を実現し、大規模性を確保する」「2. (管理者の負担を減らすために) 管理者が直接操作するサーバは少数にする」「3. (インターネットの新たな問題を解決するために必要な) 意図的なアドレス付け替えを支援する」「4. (多様な管理ポリシに柔軟に対応するために) ネットワーク管理ポリシから独立にする」「5. (DHCP などの既存プロトコルを統合できるように) モジュール性を高める」などの目標を設けた。

1.3.2 ネットワークおよび計算機に関する前提

ここで本システムが対象とするネットワークについて前提条件を定義する。これらの大半は現在の一般的な環境では満たされており、十分に現実的である。

- **単純な経路制御**
  本システムの対象ネットワークでは単純な経路制御 (default route のみ、RIP、単一自律システム内の OSPF[42] などを意味する) のみを用いている。
- **VLSM のサポート**
  効率的にアドレスを利用するためには、サブネットの大きさを自由に設定できる必要がある。ネットマスク長の異なる複数のネットワークインタフェースを適切に扱うため、全ルータは Variable Length Subnet Mask[14, 43] をサポートしているものとする。
- **サーバ群の位置的連続性**
  本システムではサーバ群が連続して設置されていなければならない。
- **単一の管理ポリシ**
  対象ネットワークは、単一の管理ポリシの元に運用されているものとする。部署ごとにネットワークを切り分け、異なるポリシで運用するようなケースは取り扱わない。
- **ルータに対する前提**
  現在のルータの多くは 1 つのネットワークインタフェースに複数のアドレスを設定できる。これにより 1 つの物理ネットワークを 2 つの論理ネットワークに分割して一方を従来通りに運用し、もう一方で本システムを運用することができる。アドレス付け替えを実現するためには全ホストでクライアントが動作しなければならないが、この方法によりクライアント機能を持たないホストを無視できる。
- **計算機に対する前提**
  ルータだけでなく全てのホストは一つのネットワークインタフェースに複数アドレスを設定することができるものとする。この条件が満たされないと、アドレス付け替えの際に古い論理通信路が切断されてしまう。しかし複数アドレスを用いて緩やかにアドレスを付け替えれば、この問題をある程度回避できる。緩やかな付け替えでは古いアドレスと新しいアドレスを同時に設定し、古いアドレスを宛先に持つパケットも受信する。ただし新しく通信を開始する場合は、新しいアドレスを始点とする。最後に古いアドレスを用いた通信が十分少なくなってから、そのアドレスを削除する。

上述のネットワークの二重運用の例を図 1.1に示す。図中の NetA が本システムを運用しているネットワーク、NetB が従来のネットワークである。NetB と NetA が重なっている部分は、本システムに移行中であることを示す。NetB と NetA は論理的には別ネットワークであるため、クライアント機能を持たないホストは NetB に接続しておけば NetA には影響を及ぼさない。また徐々にクライアント機能を持つホストを増やすことにより、本システムへの緩やかな移行が実現できる。

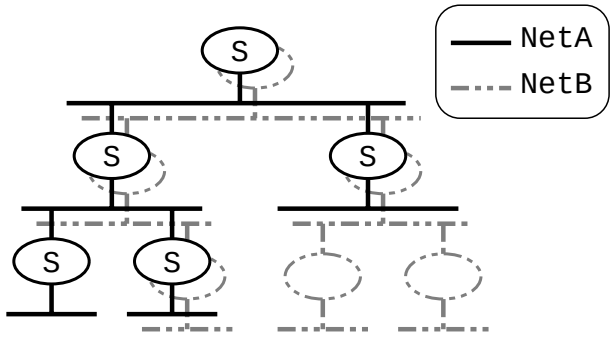

図 1.1: ネットワークの二重運用例

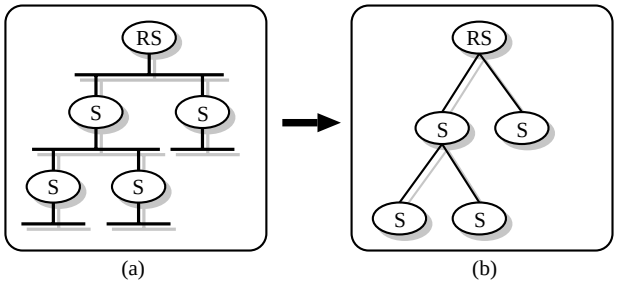

図 1.2: ネットワーク階層化の一例

1.3.3　階層構造モデルの導入

新しいネットワーク設定管理モデルでは、管理対象となるネットワーク上の全てのルータ上でサーバが動作する。したがってネットワークアドレスの付け替えにも影響を受けないし、マシンの起動時のようにアドレスを取得する前でも動作する。

本システムでは、サーバ群が自律的にネットワークトポロジに基づいた階層構造 (Spanning Tree) を構築する。階層化の例を図 1.2に示す。図 1.2(a) のようなネットワークが存在する場合、各ルータ上のサーバ群は図 1.2(b) のような階層構造を構成する。

階層構造を導入する理由は、大規模性が確保しやすいためである。またサーバはクライアントや下位サーバからの要求を処理できない場合に、上位サーバへ要求を転送する。これによって負荷分散の効果を持たせることができる。さらにネットワーク管理者が操作するサーバを階層構造のトップに位置するサーバ (木構造の根に位置するサーバのことで、以後ルートサーバと呼ぶ) だけに限定することができる。

1.4 サーバ木構成部

本システムは、サーバ木構成部、ネットワーク設定部、ホスト設定部の 3 つから構成されるが、ここでは「サーバ木構成部」について述べる。

1.4.1 サーバ木構成部の設計目標

サーバの階層構造を静的に設定すると耐故障性が低くなったり、設定を誤る可能性がある。本システムは階層構造を基に動作するため、誤った設定はシステム全体に悪影響を及ぼす。したがって自動的に階層構造を決定する機能 (サーバ木構成部) が必要である。ただしルートサーバは管理者が静的に指定する。

以下の 6項目をサーバ木構成部の設計目標とする。

1. **IP ルーティングや IP アドレスに依存しない**

   本システムでは IP アドレスが変化するため、階層構造を決定する際にも、通常の IP アドレスや IP ルーティングを利用できない。したがってこれらに依存せずに、独立に階層構造を決定しなければならない。
2. **階層構造を素早く収束させる**
3. **階層構造の変動は少なく抑える**

   収束が早くなると、逆にネットワークの些細な変動にも過剰に反応してしまう。階層構造の変化はアドレスの付け替えなどを引き起こす可能性がある。不必要なアドレスの付け替えはコストが大きいため、階層構造の変動を回避しなければならない。
4. **少ない計算量で動作する**
5. **任意のサーバの起動が遅延しても矛盾しない**

   階層構造を決定した後に、新しくサーバが追加される場合もある。そのような場合に、既に収束している階層構造には可能な限り影響を与えないようにする。
6. **簡潔な機構にして、実装を容易にする**

1.4.2 サーバ木構成部の設計

IP ルーティングや IP アドレスに依存しないために、本システムでは独自の経路制御プロトコルに基づいて階層構造を決定する。主な経路制御方式には Distance-Vector 型 (Bellman-Ford 型) と Link-State 型の 2 つがあり、一般に Link-State 型の方が収束は早い。しかし Distance-Vector 型の方が簡潔であり、また Link-State 型はネットワークの接続性がわずかに変化しただけで大量の演算を必要とする。

これらの理由から本システムでは Distance-Vector 型を採用する。ただし単純な Distance-Vector 型アルゴリズムでは収束速度が不十分であるため、Triggered Update に類似した手

法を取り入れる。Triggered Update とは、経路が変化した場合に直ちに近接ルータへ知らせる方式である。これにより階層が深くなっても、実用的な時間内でルートサーバから末端サーバまで情報が伝播する。さらに階層構造の変動を抑えるために、経路情報が安定するまで待機する。また新たにサーバが起動しても階層構造が変動しないように、Designated メッセージというものを交換する。

サーバ木構成のアルゴリズムの詳細を以下に示す。「経路表」や「経路情報」は本システム独自のものを意味し、通常の IP ルーティングのものとは無関係とする。

1. ルートサーバ (以下 RS) は初期状態で

   宛先:RS, 次ホップ:RS, ホップ数:0

   という経路表を保持している。他のサーバの経路表は初期状態では空である。
2. ルートサーバは全てのネットワークインタフェースから定期的に次の経路情報を流す:

   宛先:RS, Sender:RS, ホップ数:0, Sequence

   ただし無限に経路情報が伝播しないように、ホップ数が上限値を超えた場合には、それ以上は経路情報を流さない。またあらかじめ指定したネットワークインタフェースからも経路情報は流さない。Sequence は、経路情報のループを防ぐためにルートサーバによって管理される連続番号である。ルートサーバは経路情報を送信する度に Sequence 番号を 1 ずつ増加させる。
3. 階層構造の変動を少なくするために、ルートサーバ以外のサーバは一時経路表と本経路表の 2 つの経路表を管理する。また Triggered Update の手法に基づき、各サーバは以下のように動作する。

   (a) 現在の一時経路表に記録されているものよりも大きい Sequence 番号を持つ経路情報を受け取った場合には、一時経路表に次のような経路を記録し;

   宛先:RS, 次ホップ:Sender, ホップ数:受信ホップ数+1, Sequence

   さらに全てのネットワークインタフェースから次のような経路情報を流す:

   宛先:RS, Sender:自分, ホップ数:受信ホップ数+1, Sequence

   (b) 一時経路表と等しい Sequence の経路情報を受け取った場合、ホップ数を比較する。小さい場合には一時経路表を更新し、再度全てのネットワークインタフェースから経路情報を流す。同じホップ数で異なる次ホップの場合は、サーバの ID を比較して小さい方を経路表に登録する。ただし近接ルータには通知しない。大きい場合は、何も処理しない。ID は同じ木構造に属するサーバ群の中でユニークであれば良く、例えば Ethernet アドレスなどが利用できる。

   (c) 経路の更新は常に一時経路表で行ない、一定時間経過後に本経路表に複製する。このように Triggered Update を導入することで収束を早くするとともに、収束途中の経路が階層構造に影響しないようにする。

4. 最終的に全てのサーバが上位サーバの候補を決定し、上位サーバになることを要求する UpperRequest メッセージを送信する。上位サーバは、この時点で初めて下位サーバの存在を検知する。UpperRequest を受け取ったサーバは UpperAck を返し、下位サーバを登録する。

経路表から求められる上位サーバと現在の上位サーバが異なる状態が長期間継続した場合 (例えば上位サーバが故障したり、よりホップ数の小さな上位サーバが発見された場合) には、上位サーバを変更する必要がある。しかし上位サーバ変更は影響が大きいので、頻繁に行なうべきではない。上位サーバを変更する条件やその時期については今後の研究で検討する必要がある。

前述のようにして決定された木構造を元に、サーバは上位サーバ側 (および明示的に除外した) ネットワークインタフェースを除く、全ネットワークインタフェースを管理する。ただし複数のサーバが同一セグメントを管理することのないように、あるネットワークセグメントを管理するサーバ (Designated サーバ) を次のようにして決定する。

1. 1〜3 秒までの乱数をふり、タイマーをセットする。
2. タイムアウトするまで Designate メッセージの受信待ちをする。Designate メッセージには送信するサーバからルートサーバまでのホップ数を含める。「自身のホップ数より小さい」あるいは「同ホップ数かつ、より小さいサーバ ID」を 含む Designate メッセージを受け取った場合には、そのセグメントの管理は他のサーバに頼るものとし、これ以上の処理は行なわない。同ホップ数の場合に小さいサーバ ID を持つものを選択するのは、サーバの起動時刻に多少の遅延があっても、一定のサーバが選択されるようにするためである。
3. タイムアウトまでに Designate メッセージを受け取らなかった場合や、「自身のホップ数よりも大きい」あるいは「同ホップ数かつ、より大きいサーバ ID」を含む Designate メッセージを受け取った場合には、Designate メッセージを送信し、送信回数を 1 つ加算する。さらに N 秒後にタイマーをセットする。送信回数が M 回未満の場合には 2へ戻る (M および N の値は未定義である)。
4. タイムアウトまでに「自身のホップ数より小さい」あるいは「同ホップ数かつ、より小さいサーバ ID」を含む Designate メッセージを受け取った場合には、そのセグメントの管理を他のサーバにまかせる。受信できなければ、そのセグメントを管理する。
5. あるネットワークを管理することが決定した後に Designate メッセージを受け取った場合には、ホップ数を 0 にして送信する。これは、後からより小さな ID を持つサーバが起動しても、階層構造に変更が起きないようにするためである。

1.5 ネットワーク設定部

階層構造に基づき、ネットワークアドレスの集合の授受を行なってネットワークを設定するのがネットワーク設定部である。以後、連続したアドレスの集合をアドレスバルクと

呼び、サーバが管理するアドレスバルクの集合をアドレスプールと呼ぶ。

1.5.1　ネットワーク設定部の設計目標

設計目標は、以下の 5項目である。

1. **ネットワーク管理ポリシとプロトコルを分離する**

   多様なネットワーク管理ポリシに対応するために、プロトコルを分離して設計する。
2. **ネットワーク管理者が操作するのはルートサーバのみ**
3. **アドレス付け替えを実現**

   アドレス付け替えには意図的なものと意図しないものがある。本システムは主に意図的な付け替えを支援する。
4. **メッセージの不必要な送信を防ぐ**

   例えば割当を拒否されたアドレスバルクの割当要求を繰り返さないようにする。
5. **アドレスバルクの強制的な回収をサポートする**

   アドレス付け替えの最終段階では古いアドレスを回収する必要がある。またネットワーク管理の一環としてアドレスの回収が必要な場合もある。

1.5.2　ネットワーク設定部の設計

ポリシとプロトコルを分離するため、ここではアドレスプールの状態および状態遷移、アドレスプールへの操作、使用するメッセージなどについてのみ定義する。その際にルートサーバに対する要求が適切な下位サーバまで伝達するように状態遷移を定義し、「管理者はルートサーバだけを操作する」という目標を達成する。またアドレス付け替えを考慮し、アドレスプールに返却中という状態を定義する。不必要なメッセージの送信を防ぐために、割当や返却を拒否されたアドレス領域は記憶しておく (拒否されたことを意味する状態を設ける)。また、要求をただちに処理できない場合は処理中を意味するメッセージで応答する。返却要求メッセージには強制回収を示すフラグを設ける。

ネットワーク設定部は「上位サーバから下位サーバへのアドレスバルク割当」と「下位サーバから上位サーバへのアドレスバルク返却」の 2 つの操作を基本として構成する。つまり全てのネットワーク管理ポリシは、これらのメッセージを用いて実現される。メッセージはアドレスの割当/返却を要求する側が再送する。ネットワーク設定部で用いるメッセージの一覧を表 1.1に示す。

ネットワーク管理者が操作するのはルートサーバだけである。したがって例えば、あるアドレス領域の返却要求がルートサーバから下位サーバへ伝播していくようにしなければならない。階層が深くなると各サーバでの処理が重なって、要求の伝播にも時間がかかた

表 1.1: ネットワーク設定部で用いるメッセージの種類

名称	用途
AllocRequest	下位サーバから上位サーバへの割当要求
ReleaseRequest	上位サーバから下位サーバへの返却要求
Alloc	上位サーバから下位サーバへの割当
Release	下位サーバから上位サーバへの返却
AllocNack	下位サーバからの割当要求に対する拒否
ReleaseNack	上位サーバからの返却要求に対する拒否
AllocPending	上位サーバが AllocRequest を受信し、処理中であるという通知
ReleasePending	下位サーバが ReleaseRequest を受信し、処理中であるという通知

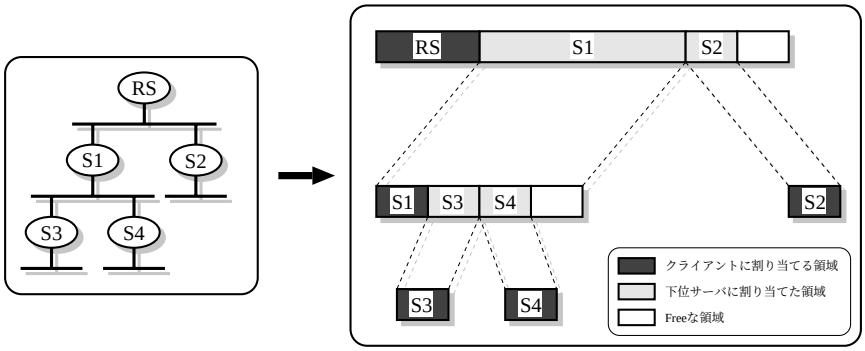

図 1.3: アドレスプールの状態例

め、例えば他のサーバの処理が終わるまで待機させる必要がある。要求を直ちに処理できないサーバは、処理中を意味する Pending メッセージを返す。

アドレス不足を判定する基準や、割当と返却のどちらを要求するか選択する基準、それらの要求を拒絶する基準などはポリシに深く関わるため、ここでは定義しない。しかし例えば下位サーバに割り当ててあるアドレスバルクが返却された場合にどういった動作を行なうかについては、ポリシとは概ね無関係である。本節の残りでは、アドレスバルクの状態と状態遷移を定義することで、ネットワーク設定部の設計とする。

アドレスバルクの状態として定義した 11 個の状態を表 1.2に概要と共に示す。注意しなければならないのは、例えばあるサーバで Allocated 状態のアドレス領域が、その下位サーバでは Free 状態などの他の状態をとる場合があるということである。つまりアドレス空間の同じ領域であっても、サーバごとに状態が異なる (図 1.3)。

アドレスバルクの状態間の関係を詳しく状態遷移図として示したのが図 1.4である。Request

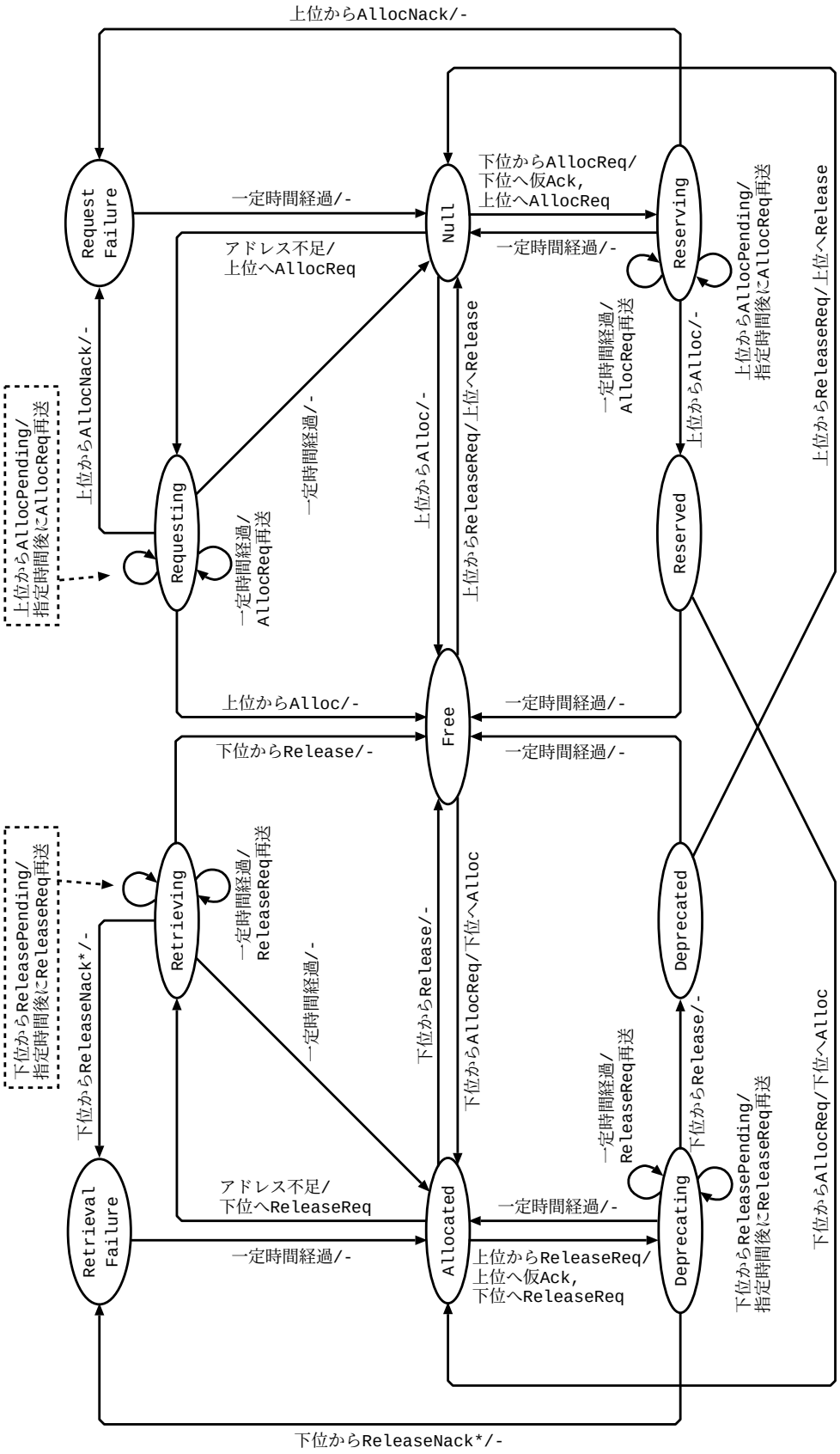

図 1.4: アドレスバルクの状態遷移

表 1.2: アドレスバルクの状態

状態名	概要
Null	空を意味する初期状態。ルートサーバ以外のサーバは初期状態で Null 状態のアドレスバルクだけを保持している。
Free	どのサーバにも割り当てられていない状態。ルートサーバは初期状態で Free 状態のアドレスバルクを保持している。
Allocated	下位サーバかローカルクライアントに割り当てている。
Requesting	上位サーバへ割当を要求中。割り当てられると Free 状態に遷移する。
Request Failure	割当要求を拒否された状態。一定時間、この領域の割当は要求できない。
Retrieving	下位サーバへ返却を要求中。返却されると Free 状態に遷移する。
Retrieval Failure	返却要求を拒否された状態。一定時間、この領域の返却は要求できない。
Reserving	下位サーバからの割当要求に応じて上位サーバへ割当を要求中。
Reserved	上位サーバから割り当てられ、下位サーバからの割当要求を待っている。一定時間内に要求がなければ Free 状態に遷移する。
Deprecating	上位サーバからの返却要求に応じて下位サーバへ返却を要求中。
Deprecated	下位サーバから返却され、上位サーバからの返却要求を待っている。一定時間内に要求がなければ Free 状態に遷移する。

Failure および Retrieval Failure の 2 つの状態はメッセージの不必要な再送を防ぐために設けた。これらは「要求に失敗した」という情報を一定時間保持するために用いられる。例えば Request Failure 状態のアドレスバルクに対して下位サーバが AllocRequest を送信して来た場合には、AllocNack を返す。また Deprecating および Deprecated 状態は、アドレス付け替えの最中のアドレスを示すためのものである。これらの状態のアドレスバルクに対して AllocRequest が届いた場合には、やはり AllocNack を返す。

最後にネットワーク設定部の動作例を以下に示す。

1. ルートサーバ RS, RS の下位にサーバ B, サーバ B の下位にサーバ C があるとする。
2. RS には下流側ネットワークインタフェースが 1 つ、B, C には上流側と下流側にネットワークインタフェースがそれぞれ 1 つずつあるとする。
3. 起動直後には RS は 133.27.0.0/16 という Free 状態のアドレスバルクを保持している。B および C のアドレスプールは空である。
4. C は下流側ネットワークで大きさ 16 のアドレスバルクが必要と推定し、B に対し AllocRequest を送る。同時に大きさ 16 で Requesting 状態のアドレスバルクを (空の) アドレスプールに追加する。
5. B は Null 状態のアドレスバルクに AllocRequest が届いたので、C に対して AllocPending を返す一方で、上位サーバ RS へ AllocRequest を送る。同時に大きさ 16 で Reserving 状態のアドレスバルクを (空の) アドレスプールに追加する。

6. B は下流側ネットワークで大きさ 32 のアドレスバルクが必要と推定し、RS に対し AllocRequest を送る。さらに大きさ 32, Requesting 状態のアドレスバルクをアドレスプールに追加する。
7. RS は大きさ 16, 32 のアドレスバルクを割り当てる Alloc メッセージを B へ送る。割り当てられた領域は RS 上では Free 状態から Allocated 状態へ遷移する。RS の下流側ネットワークにもアドレスバルクが割り当てられ、Allocated 状態となる。

1.5.3　ネットワーク設定部におけるアドレス付け替えの支援

アドレス付け替えのために、ネットワーク設定部では新しいアドレスの割当、古いアドレスの回収、経路制御の変更の 3 つを実施する必要がある。アドレス付け替え手順の概要を以下に示す。

1. 期限付きで ReleaseRequest を下位サーバに送る。
2. 受け取った下位サーバは、その領域が Free であれば直ちに上位サーバに返却する。
3. その領域を下位サーバに割り当ててある場合には、下位サーバに RelaseRequest を送る。同時に上位サーバには ReleasePending メッセージを送る。
4. その領域を自分で使用している (クライアントに割り当てている) 場合にも、上位サーバに ReleasePending メッセージを送る。同時に上位サーバへ新たなアドレスの割当を要求し、ホスト設定部を通じてクライアントのアドレス付け替えを行なう。
5. 期限が来るまでに、上位サーバへアドレスを返却する。
6. サーバは自らが使用 (クライアントへ割り当て) を開始する時に経路情報をアナウンスし始める。逆に上位サーバへアドレスを返却した時点で、アナウンスを止める。

1.6　ホスト設定部

ホスト設定部は他の部分からは独立に実現する。最初にホスト設定部の設計目標を明らかにし、次に詳細な設計を行なう。

1.6.1　ホスト設定部の設計目標

ホスト設定部の設計目標を以下に示す。

1. **複数アドレスの設定が可能**
   アドレス付け替えを実現するには複数のネットワークアドレスを一つのホストに割り当てられなければならない。

2. **アドレスの強制回収が可能**
   アドレス付け替えを実現するため、アドレスを強制的に回収する機構を持っていることが望ましい。クライアントが割当の期限切れなどにより自発的に再割当要求を送ってくるまで待つ方法は、付け替えに要する時間などの点で望ましくない。

3. **ホストがネットワークを移動したことを検知できる**
   クライアントだけで自分自身が別のネットワークへ移動したことを検知することは難しい。これはハードウェアと OS のどちらかの制約による場合が多い。本システムは OS やハードウェアに依存せずに移動を検知できるように設計する。

4. **プロトコルを簡潔にする**
   プロトコルを複雑にして高機能化するより、機能を限定してプロトコルを簡潔にする。

1.6.2 ホスト設定部の設計

サーバはネットワークに関する情報を含む Advertise メッセージを定期的に送信する。クライアントは新しい情報を含むメッセージを受信すると、アドレスの割当を要求する。情報の異なるメッセージを複数送信することで、複数アドレスがサポートできる。さらにメッセージに強制回収を示すフラグを設ける。次に Advertise メッセージは定期的に送信されるため、一定時間メッセージが受信できなければネットワークを移動したと推測できる。最後の「プロトコルを簡潔にする」という目標を達成するために、アドレスには有効期限などの質的な差を設けない。また各セグメントを管理するサーバは 1 つしか存在しないと仮定することにより、プロトコルの簡略化を目指す。

DHCP はアドレスを割り当てるために 4 回メッセージを交換する必要があった。これには「1 つのネットワークセグメントに複数のサーバが存在する場合があるため」および「サーバの割当可能なアドレスには有効期限の長さなどの質的な差が存在するため」という 2 つの理由がある。しかし本システムではどちらも考慮しなくてよいため、アドレス割当は Request と Reply の 2 つのメッセージ交換によって成立する。Reply には割当を意味する Ack と拒否を意味する Nack の 2 つがある。またクライアントへ通知される情報は、IP アドレス, ネットマスク, デフォルトルータの 3 つのみとし、それ以外の情報については Service Location Protocol[44] などを利用することを想定する。

さらに「ネットワークの移動を検知する」という目標をオペレーティングシステムに依存せずに実現するために、サーバから定期的にネットワークに関する情報を Advertise する。この Advertise メッセージが受信できなくなったり、あるいは新しい Advertise メッセージを受信した場合には、ネットワークを移動したと判定する。素早く移動を検知するためには Advertise の間隔を短くする必要がある。ただし Advertise の間隔は、ネットワークの管理者がポリシに基づいて指定できるようにする。

1.6.3　ホスト設定部におけるアドレス付け替えの支援

アドレス付け替えのために、ホスト設定部は「サーバが任意の時点でクライアントに対してアドレスを返却させる」機能を持っている必要がある。以下に本システムでの実現方法を、通常のホスト設定処理の流れと共に述べる。

1. サーバは定期的に自分の管理するネットワークの情報を Advertise する。アドレス付け替えの最中には新旧それぞれのネットワークについて Advertise する。ただし付け替え中の古いネットワークアドレスについては、それを示す Deprecating フラグおよび有効期限を Advertise メッセージに含める。
2. クライアントが過去に受信したことのない Advertise メッセージを受信した場合、Request を送ってアドレスの割当を要求する。
3. クライアントは自分の属するネットワークアドレスの Advertise を一定時間以上受信できないか、あるいは自分の属するネットワークアドレスの (Deperecating フラグと共に指定された) 有効期限が過ぎたら、そのアドレスの利用を中止する。

1.7　階層型ネットワーク自動設定機構の実装と評価

設計に基づき、BSD/OS 2.0.1 上にネットワーク自動設定機構のプロトタイプを実装した。

1.7.1　サーバ実装の構造

サーバは他のサーバが送るメッセージを待ちつつ、メッセージの送信や再送を行なう必要がある。さらに本システムは複数のアドレスバルクに対する複数の操作を並行して行なわなければならない。このような要求を実現するには「1. マルチプロセスを用い、プロセス間通信によって同期する」「2. マルチスレッドを用いて実装する」「3. サーバ自身でスケジューリングを行なう」などの方法がある。1 の方法は、アドレスプールに対する操作の排他制御を実現しなければならない。また多数のプロセスが起動するため、サーバマシンに負荷がかかる可能性がある。2 の方法を実現するには、オペレーティングシステムがマルチスレッドをサポートしていなければならず、移植性が低くなる (BSD/OS 自体もマルチスレッドをサポートしていない)。これらの理由から 3 番の方法を採用した。

サーバの移植性を高めるために、本システムはアプリケーションプログラムとして実装した。メッセージの交換には UDP を採用した (TCP は IP アドレスが設定されていないと利用できないため不適当である)。本システムはアドレスが設定される前に動作しなければならないため、通常の IP ユニキャストは使用できない。そのためブロードキャストないしマルチキャストを利用する必要がある。プロトタイプでは Berkeley Packet Filter(BPF)[45] を利用してブロードキャストする。

本システムはメッセージを受信したネットワークインタフェースを認識しなければならない。また、どのネットワークインタフェースからメッセージを送信するか指定できなけ

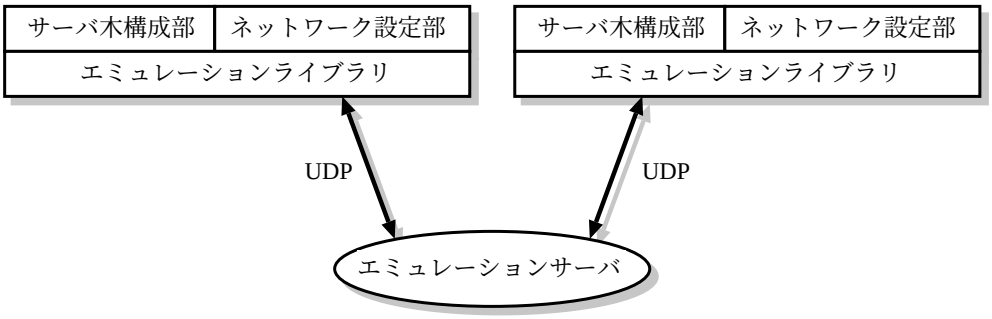

図 1.5: エミュレータの概念図

ればならない。これらは通常のソケットでは不可能だが、BPF を利用すれば可能である。BPF は BSD 系の UNIX には標準で実装されている (他のオペレーティングシステムも BPF 同様の機能を持つものが多い) ため、移植性は比較的高い。

1.7.2 Ethernet エミュレータの実装

本システムは複数のネットワークセグメントを自動的に設定管理するためのものである。したがって動作検証を行なうためには多数のネットワークと計算機が必要である。しかし現実には、そのような規模の実験環境を用意することは困難である。また仮にそのような環境が用意できたとしても、さまざまなネットワークトポロジを構築して実験、検証するには多くの時間が必要となる。したがって仮想的にネットワークを構築し、実験できる環境を構築することが望ましい。そのような環境を実現するための方法としては、本システムを全てネットワークシミュレータ上に実現するという方法がある。もう一つの方法としてデータの送受信の部分だけをエミュレートする方法がある。本論文ではエミュレータ方式を採用した。これは将来的に現実のネットワーク上にシステムを構築する場合に、データ送受信部分以外はそのまま利用できるためである。

今回実装したエミュレータは、UDP を用いた簡易なものである。エミュレータは実際にエミュレーションを行なうエミュレーションサーバと、エミュレーション機能を利用するためのライブラリ関数から構成される (図 1.5)。

エミュレータは以下のようにして動作する。

1. ネットワーク設定サーバがネットワークインタフェースを初期化する代わりにエミュレータのライブラリを呼び出す。ライブラリは設定ファイルから必要な情報を読み込み、エミュレーションサーバへ登録メッセージを送る。
2. 登録メッセージにはネットワーク設定サーバのネットワークインタフェースの MAC アドレスと接続すべきネットワークの識別番号が含まれる。エミュレータは識別番号別のリストに MAC アドレスを登録する。

3. 各ネットワーク設定サーバがライブラリを通じてデータを送信するとエミュレーションサーバに届く。この時データには、ネットワークの識別番号と宛先 MAC アドレスが含まれる。メッセージを受信したエミュレーションサーバは対応するリストを検索し、宛先 MAC アドレスと一致したネットワーク設定サーバにのみメッセージを転送する。もし宛先 MAC アドレスがブロードキャストの場合には全てのネットワーク設定サーバにメッセージを転送する。

ただしエミュレータと現実のネットワークには次のような差がある。

- パケットロストが実際のネットワークよりも少ないと思われる。パケットロストを実現することも容易だが、影響が少ないと判断した。
- ブロードキャストはリストに登録された順に配送される。これに対し現実のネットワークでは、ほぼ同時にブロードキャストが到着する。ただし現実の環境でも、到着したパケットがオペレーティングシステムから各プロセスに渡されるまでの時間は異なるため、これについても無視した。

1.7.3　サーバ木構成部の実装と評価

設計では述べなかった実装の詳細を以下に示す。

1. ルートサーバは 10 秒ごとに経路情報をアナウンスする。ルート以外のサーバは、経路情報メッセージを受信したら処理を行ない、自分自身の経路情報を送信する。
2. 初めて経路情報を受け取ってから、10 秒後に上位サーバの決定と登録を開始する。これは経路情報を受け取ってから 5 秒以内に一時経路表が安定し、本経路表に複製されるという仮定に基づく。
3. 上位サーバは UpperRequest を受け取ると、送信者を下位サーバとして登録し、UpperAck を返す。UpperAck を受信した下位サーバは上位サーバを登録し、Designated サーバを選定するためにメッセージの送信を開始する。
4. Designated メッセージの再送間隔 (5 秒) と再送回数 (5 回) から、少なくとも 30 秒程度経過すると Designated サーバが決定できると仮定する。30 秒後に Designated サーバを決定し、ネットワーク設定部の処理を開始する。

サーバ木構成部は Ethernet エミュレータを使って構築した仮想ネットワーク上で実験した。サーバ木構成部からは通常の BPF に対して読み書きしているように見える。図 1.6に仮想ネットワークのトポロジと、実験により構築された階層構造を示す。このようにループを含んだトポロジでも設計通りに動作することが確認できた。今後はさらに複雑なトポロジや、仮想ネットワークでない現実のネットワークで動作することを確認する必要がある。

素早い収束という目標も達成することができた。例えば仮想実験ネットワークでは経路情報は 2 秒程度で収束する。収束までの所要時間についてはネットワークのトラフィック

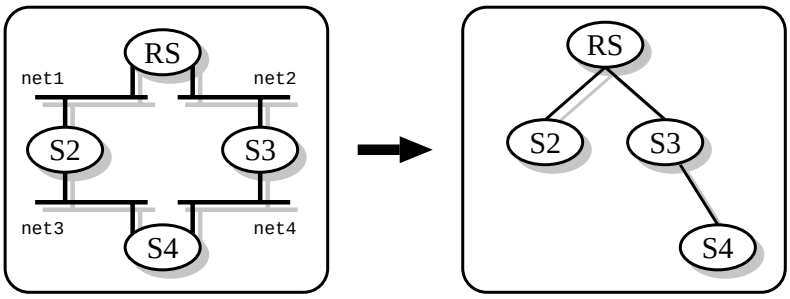

図 1.6: 仮想実験ネットワークに構築された階層構造

や計算機の性能によって異なると思われるが、今回の実装では初めて経路情報を受け取ってから 5 秒後には経路情報が安定すると仮定した。しかし条件によっては仮定が成立しないと思われるので、今後は適切な判定方法を考案しなければならない。

「階層構造の変動は少なく抑える」と「任意のサーバの起動が遅延しても矛盾しない」という目標に関して検証するために、故意に特定のサーバの起動を遅延させる実験を行なってみた。その結果、すでに確定したサーバの木構造には影響を及ぼさないことが確認できた。

簡潔な機構であるかについては評価が難しいが、サーバ木構成部の実装は約 600 行であることから達成できたといえる。計算量については定量的に評価することができなかった。これについては今後の課題である。

1.7.4 ネットワーク設定部の実装と評価

ネットワーク設定部は状態遷移図に基づいて実装した。以下に処理の概要を示す。

1. Designated サーバに決定したネットワーク設定部は、ネットワークインタフェース毎に必要なアドレスバルクの大きさを推定し、上位サーバに AllocRequest を送信する。AllocRequest の再送スケジュールも設定する。
2. 他のサーバやホスト設定部とアドレスバルクを授受する毎に、アドレスが不足していないか検証する。不足していれば上位サーバに AllocRequest を送信する。
3. 今回の実装では、自動的には下位サーバに返却を要求しない。返却要求 (ReleaseRequest) はネットワーク管理者がルートサーバに指示を送った場合にのみ送信される。

1番目に示した「必要なアドレスバルクの推測」は、起動直後だけ特別に処理する必要がある。すなわち通常、ホスト設定部はクライアントへのアドレス割当を管理しているため、必要となるアドレスの大きさも把握している。しかし起動直後には、クライアントへのアドレス割当は一切行なわれていないために、別の方法を用いなければならない。起動直後のアドレスバルクの推定するには、例えば次のような方法がある。

ネットワーク設定部は、自分の管理するネットワークインタフェースから、メッセージをブロードキャストする。クライアントは、数秒間 (ランダム) 待機してから応答を返す。サーバはクライアントからの応答の数をカウントする。これを何度か繰り返し、最大値を採用する。

今回の実装ではネットワーク設定部とホスト設定部を統合しなかったため、この方法は用いなかった。代わりにアドレスバルクの大きさの推測値として乱数を用いた。

ネットワーク設定部の設計では「ポリシとプロトコルを分離する」ことを目標とした。そのためネットワーク設定部を実装するのに必要なアルゴリズムのいくつかは定義していない。本実装では次のようなアルゴリズムを採用した。

- ある 1 つのセグメントに割り当てるアドレスの総数の 50%を使い切った時点で、アドレスが不足したと判定する。
- アドレスが不足した場合に、下位サーバに返却を要求するのではなく常に上位サーバへ割当を要求する。
- 下位サーバへ返却を要求するのは、ネットワーク管理者がルートサーバに指示した場合のみとする。自律的に各サーバが返却を要求することはない。
- アドレスが過剰でも、上位サーバから要求されない限り返却しない。

ネットワーク設定部も前述の仮想ネットワークで実験し、基本動作を検証した。その結果、各サーバがアドレスバルクを上位サーバに要求し、ルートサーバから割当が行なわれることが確認できた。またルートサーバを操作することにより、下位サーバからアドレスを回収できることを確認した。本実装では各サーバは必要なアドレスバルクの大きさを決定すると個別に割当を要求する。実験の結果、この方法では上位–下位関係にあるサーバの間でもアドレスプールが不連続になってしまうことがわかった。したがって起動時に下流にあるサーバからの割当要求を集計して、上位サーバへ割当要求を送る方が望ましい。

次にポリシとプロトコルの分離という目標は達成した。ただし両者の完全な分離は難しく、あるポリシを実現するのに必要な機能がプロトコルに含まれていない可能性がある。したがって多様なポリシに対応できるようにプロトコルを拡張することが今後の課題である。アドレス付け替えについてもネットワーク設定部での支援機構は動作することを確認した。ただしサーバがアルゴリズムに基づいて自律的に行なう「意図的でない」アドレス付け替えに関しては動作確認していない。

以上のようにネットワーク設定部の基本動作を確認できたが、ネットワーク設定ポリシを実現するためのアルゴリズムの設計が不適切であると、サーバ群全体が機能しない。したがって今後はアドレス分散管理アルゴリズムについて検証する必要がある。

1.7.5　ホスト設定部の実装と評価

ホスト設定部は BPF を用いて実装したが、実装を簡潔にするために物理メディアは Ethernet に限定した。ホスト設定サーバも独自のスケジューラを用いている。またホスト設定

部はネットワーク設定部と統合せずに単独のプロセスとして実装した。理由は「1. ホスト設定部の過負荷がネットワーク設定部に影響を与えないようにするため」「2. デバッグなどのメンテナンスを容易にするため」「3. 既存のホスト設定部と組み合わせて動作を確認し、モジュール性の高さを実証するため」などである。

ホスト設定サーバは「Advertise メッセージの定期送信」と「クライアントへのアドレス割当」および「(ネットワーク設定部からホスト設定部を制御するための) ネットワーク設定サーバとの通信」の 3 つの機能を提供する。ただし今回はネットワーク設定サーバとの通信機能については実装を行なわなかった。これについては今後の課題とし、今回は簡単なユーザインタフェースでホスト設定サーバを制御した。

アドレスの割当アルゴリズムはポリシに関わるため、ホスト設定部の設計では論じなかった。本実装では以下の仮アルゴリズムに基づいて割当を行なう。

- 希望するアドレスが指定されている場合、そのアドレスが
  - そのクライアントに割り当てられていて、かつ有効期限内ならば Ack を返す。
  - 他のクライアントに割り当てられておらず、かつ Deprecating の印がついていなければ、そのアドレスを Ack で割り当てる。
  - 他のクライアントに割り当てられているか、あるいは Deprecating の印がついている場合は Nack を返す。
- 希望するアドレスが指定されていない場合は、他のクライアントに割り当てられておらず、かつ Deprecating でないアドレスを選択し割り当てる。

クライアントの処理の概要を以下に示す。

1. 初期状態ではアドレスおよびネットワーク情報のリストは空である。
2. 新しい Advertise メッセージを受信したらネットワーク情報のリストに加え、さらにカウンタを Nに初期化する。以後、適切な Advertise メッセージを受信したら counter を Nに再初期化する。
3. 数秒後 (ランダム) に Request を送る。Ack を受け取ったらアドレスのリストに加え、さらに割当期限の延長要求をスケジュールする。Nack を受け取ったら、ネットワーク情報のリストから削除し、しばらく待機してから 2に戻る。
4. 延長の要求に失敗した場合はネットワーク情報とアドレスのリストから削除し、一定時間後に 2に戻る。
5. 定期的にカウンタを減らしていくようにスケジュールを設定する。もし一定時間以上、Advertise メッセージを受け取れない場合には、カウンタが 0 になる。この場合ネットワーク情報とアドレスのリストから削除し、2に戻る。
6. Deprecating フラグの設定された Advertise を受信したら、返却期限を取り出す。返却期限が来たら、アドレスを解放する。

ホスト設定部については実際のネットワークを用いて動作確認を行なった。その結果、複数アドレスの設定は達成した。同時に、単一ホストのアドレスの付け替えを緩やかに実現できることを確認した。すなわち古いアドレスでの通信を維持しつつ、新しいアドレスを用いた通信が開始できることを確認した。

Advertise メッセージを用いたホスト設定は、その簡潔さにも関わらずうまく動作することが確認できた。単一ホストのアドレス強制回収も実現することができた。強制回収に応じない誤ったクライアントも、ルータ側で経路情報のアナウンスを中止されると外部とは通信できなくなるため、他への影響を小さくすることができる。「ネットワークを移動したことを検知する」ことも確認できた。ただし実際に移動してから、それを検知するまでに数十秒を必要とする。通常の LAN では移動の頻度がそれほど高くないので問題ないが、無線 LAN で用いる場合には問題がある。

1.8　まとめ

本論文ではネットワークの自律構成およびアドレス付け替えの支援機構を実現するための手法として階層化モデルを提案した。この階層化モデルに基づいて実装したネットワーク自動設定機構は、計算機単体の設定のみならずネットワーク単位での設定を可能にする。このようなネットワークの自律構成、ネットワーク単位での自動設定およびアドレス付け替えを実現する機構は前例がなく、その点で本研究の意義は大きい。

本システムはサーバ木構成部、ネットワーク設定部、ホスト設定部の 3 つから構成されている。サーバ木構成部には Triggered Update の手法を取り入れた Distance-Vector 型経路制御アルゴリズムを適用し、階層構造の素早い構築を実現した。ネットワーク設定部はさまざまなネットワーク管理を実現できるようにポリシ独立に設計・実装し、エミュレータにより構成される仮想実験ネットワーク上で実験を行なった。実験の結果、サーバ木構成部とネットワーク設定部の動作を確認した。またホスト設定部はサーバがネットワーク情報を定期的に配布するという手法を用い、この手法がアドレスの付け替えや回収およびネットワークの移動を検知するのに有効であることを実証した。

今回の実装では検証できなかった事項で、比較的容易に解決あるいは実現できるものを以下に示す。

- 実際のネットワークにおけるサーバ木構成部およびネットワーク設定部の動作確認が必要である。本論文ではエミュレータによる仮想ネットワークで動作を確認したが、現実のネットワークでも問題が起きないことを検証しなければならない。
- ホスト設定部のモジュール性を高め、既存のホスト設定機構を利用できるようにするという設計目標を実現し、評価する。
- これら全てを統合した完全な 1 つのシステムを、実際のネットワークにおいて試験運用しなければならない。ただしネットワーク管理アルゴリズムについては比較的単純なものを用いる。

その他にも「サーバ木構成に必要なトラフィックおよび演算量の評価」「意図的でないアドレス付け替えの発生頻度の測定」や「アドレスバルクの強制的回収に要する時間の評価」などを行なう必要がある。

長期的な課題として、第一にさまざまなネットワーク管理アルゴリズムを実装し、それらの性能を比較評価する必要がある。例えばアドレスの利用効率だけを重視するアルゴリズム、経路が集約可能になることを重視するアルゴリズム、アドレス付け替えをなるべく行なわないアルゴリズムなどを実装し、長所と短所を比較する。また大規模ネットワークにおける運用実験を行ない、トラフィック解析やアドレス付け替えの所要時間などの性能評価を行なう必要がある。

第 2 章

移動するネットワークのための透過的な通信機構

2.1 背景・目的

近年のコンピュータの小型軽量化は著しく、利用者が自分の作業環境を携帯することが可能になってきている。さらにカード型の通信用機器の普及により、移動した先でイーサネットに接続する、もしくはデジタル携帯電話等を利用して移動しながらインターネット等を利用して作業する、いわゆるモーバイルコンピューティングが現実のものとなりつつある。

残念ながら現在のインターネットの設計においては接続されるシステムがネットワーク環境内を移動するということはまったく考慮されていない。そのためインターネット・プロトコルそれ自体だけでなく、これを利用する多くのアプリケーションは、通信相手が常にインターネットに接続されていることを前提としたものとなっている。

しかし移動するシステムが接続するネットワークは、必ずしも同じではない。つまり、インターネットと接続する際に使う識別子、すなわち IP アドレスが常に同じであるとはかぎらない。通常 IP アドレスには、世界中で一意に識別できる名前がつけられているが、移動時の IP アドレスの変化により、それまでの IP アドレスと名前との整合性が失われてしまう。また通信中に IP アドレスが変わることにより、今までおこなっていた通信は継続できなくなってしまう。

こういった問題を解決するために、インターネット上でのホストの移動に対応するためのプロトコルがいくつか開発され、実験がおこなわれている。WIDE プロジェクトでも VIP(Virtual Internet Protocol) 方式 [46] を開発し実験が続けられている。

しかし将来のモーバイルコンピューティングを考慮した場合、現在のようにホストを単独で携帯しながらインターネットに接続することができるというだけでは不十分であり、今後登場することが予想されるさまざまな利用方法や利用形態への対応は困難であると考えられる。よっていくつかのホストが LAN(Local Area Network) を構成し、インターネットへの接続先を次々に変更しながら移動する、いわば移動ネットワークという形態も考慮する必要がある。移動ネットワークが実現した場合、以下にあげるような応用が考えられる。

1. 自動車や飛行機などの交通機関内部に構築された LAN から、効率よくインターネットへアクセスする

2. 複数の接続先 (ネットワークプロバイダ等) を切り替えて、家庭内 LAN 内部のホストからインターネットへ直接アクセスする

3. 移動プロトコルに対応していない通常のホストであっても、移動ネットワーク内部からインターネットへアクセスすることにより、間接的に移動を実現できる

これらの応用について具体的に説明する。将来的には自動車や飛行機などにも LAN が構築されることが予想される。当然ながら移動中は無線通信媒体を利用してインターネットへアクセスするという形態にならざるを得ない。しかし無線通信媒体は一般的に帯域が狭いため、自動車の場合は停車中に、飛行機の場合は空港での停止時などに、帯域の広い光ファイバーによるインターネットへのアクセスに切り替えて、キャッシュの更新等をおこなう必要がある。当然切り替わったことは、利用者には意識させない必要がある。

次に家庭内 LAN を複数の接続先を切り替えてインターネットにアクセスすることを考える。この場合、家庭内 LAN 自体が物理的に移動しているわけではないが、接続先を切り替えるということはネットワークトポロジ的に見た場合、ネットワークが移動したことと等価であるとみなせる。この場合、現在おこなっている通信を継続しながら、複数のプロバイダを切り替えるという機構が実現できる。

また移動プロトコルに対応していないホストであっても、移動ネットワークを介して、間接的に移動に対応することが可能になる。これは既存のオペレーティングシステムやネットワークソフトウェアを変更せずに、不完全ながらも移動を実現できるため、一般への普及ということを考えた場合、単独の移動ホストの実現に比べて大きな利点になる。

2.2 移動するネットワークの問題点

インターネットプロトコルの仕様上ではネットワーク単位の移動は可能ではある。しかしそのためには現在の運用形態では許されていないさまざまな条件を満たすことが必要になる。本節では、まず移動ネットワークを定義し、インターネット上での移動ネットワークの実現が不可能である理由について解説する。

2.2.1 移動ネットワークの定義

本節での移動ネットワークとは、

自身の構成を保ったままインターネット上を移動しても、内部のホストとインターネット上のホストとの通信は透過的に継続できる機構を持つネットワーク

と定める。ここでの “通信を透過的に継続する” とは、移動ネットワーク内部のホストとインターネット上のホストの双方において、互いに移動したことをまったく意識せずに、現在おこなっている通信を継続することを示す。また移動ネットワークとインターネットを結ぶルータを**モーバイルルータ** (以下 MR) と呼ぶ。

またネットワークに接続されている状態においては、移動ネットワークは以下にあげる条件を満たさなければならない。

- 移動ネットワーク内部のホストは移動対応でなくても通信をおこなえる。
- 移動ネットワーク内部のホストが移動対応であれば、単独で離脱して別のネットワークに接続しても通信を継続できる。
- 移動ネットワーク内部に他の組織の移動ホストが接続しても、その移動ホストは通信を継続できる。

移動ネットワークは、用途や条件に応じてさまざまな構成がとられ、その構成や接続形態は限定されるべきではない。しかし本研究では移動ネットワークに以下のような制限を設ける。

- 単一の物理セグメントからなる。
- 移動ネットワークとインターネット間のルータとなるホスト (**MR**) が 1 台だけ存在する。すなわち移動ネットワーク内部のホストとインターネット上のホストの通信は必ず **MR** を経由する。
- 移動ネットワーク内部に、他の組織の移動ネットワークが入れ子状に接続してはならない
- 移動ネットワーク内部の通信媒体と、**MR** とインターネットを接続する通信媒体は、イーサネットに限られる。

本研究での移動ネットワークは図 2.1に示すような形態を仮定している。

2.2.2　アドレスからみた移動ネットワークの問題点

まず移動ネットワークのアドレスが、ある Class C もしくは Class B の一部である場合を考える。移動ネットワークが新規接続をした後、MR が移動ネットワークの経路情報をアナウンスし、それが瞬時にインターネット上へ伝播すると仮定すれば、理論上は問題なく通信をおこなうことができる。しかし以下に列挙する理由によりそれは現実的ではない。

経路情報の伝播遅延

インターネット上の AS 内部に動的に経路情報を伝えるためのプロトコルには、古くは RIP [28]、最近では OSPF [25] が広く用いられている。30 秒に一回すべての経路情報を交換する RIP では経路情報がインターネット上に伝播するのにかなりの時間がかかることは想像に難くないが、OSPF ではそれに比べるとかなり短時間で経路は収束するはずである。

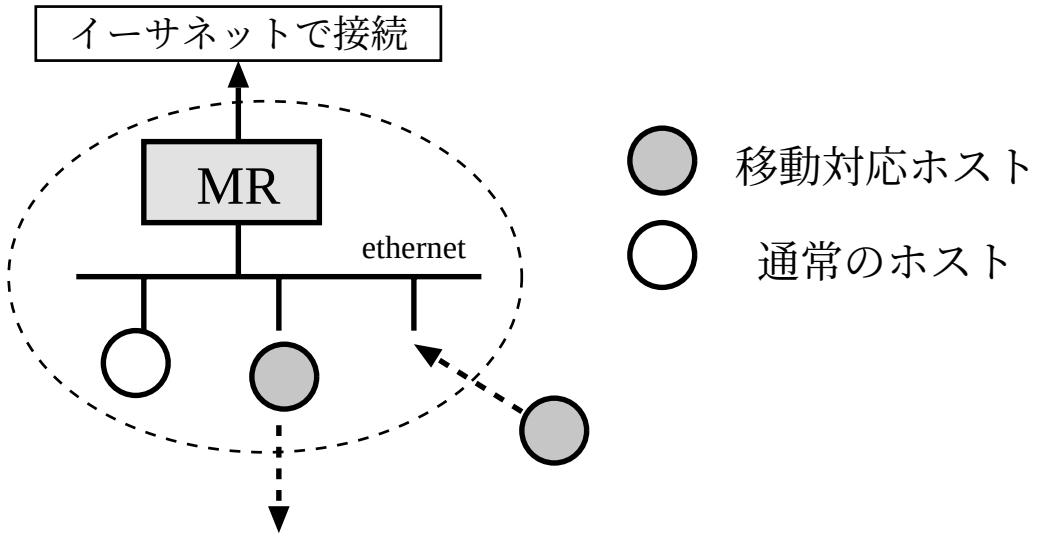

図 2.1: 本稿での移動ネットワークの構成

しかし現実には経路情報が完全に伝播するまでにはある程度の時間がかかり、その間は古い経路情報に従って、以前に接続していた場所にパケットが送られるため、通信が中断してしまう。ネットワークが移動中の時間も含めると、上位層のプロトコルにおいてタイムアウトが発生し、それまでおこなっていた通信が終了してしまう可能性が高い。

IP アドレスの枯渇

インターネットの普及とともに、IP アドレスの枯渇問題が憂慮されている。こういった状況では、各国の NIC(Network Information Center) は新規接続組織へのアドレスの割当を優先するであろう。そのため移動ネットワークのためだけに新規にアドレスを取得できる可能性は、ほとんどないといっても過言ではない。

ここであえて “Class C” のアドレスを新規に取得と限定したのは、現在ではよほど特別な事情がない限りは、Class B のアドレスはもはや割当が認められていないからである [29]。

2.2.3 経路制御からみた移動ネットワークの問題点

現状の経路制御は、経路情報の集約という観点から、さまざまな工夫をおこなっているが、それが移動ネットワークの実現に対して大きな制約となっている。

CIDR 未対応の機器の存在

ある Class B のアドレスの一部が移動ネットワークのアドレスである場合、旧来のクラ

スの概念に基づく経路表では、サブネットは連結していなければならないという物理的な制約により移動は困難である。しかし最近では CIDR [14] の普及により、可変長ネットマスク (VLSM) に対応した経路表を持つホストやルータが増えてきているが、OSPF などの VLSM を前提とした経路制御プロトコルに対応していない古いルータや、RIP にしか対応していない廉価版のルータも未だに数多く運用されている。すなわちすべてのルータやホストが CIDR をおこなうために、VLSM に対応した経路表を持って、OSPF を利用して経路情報を交換しているわけではない。

経路の集約による制限

さらに前節で述べたように、インターネットでは経路情報の増大が問題になっており、プロバイダ単位でアドレスブロックを割り当て、BGP4 [26] を用いて自律システム (AS) 間の経路情報を集約して交換している。そのため本来所属している AS とは異なる AS に移動した場合、その経路情報は伝播することはできない。これと同様な運用例で、ある組織がプロバイダを変更した場合にこのような状況が考えられるが、その際は新たなプロバイダの管理するアドレスブロックからネットワークアドレスを割り当ててもらい、IP アドレスの付け直しをおこなうことが必須条件となっている。この BGP4 による経路情報を集約する運用により、もし移動ネットワークがプロバイダ管理のアドレスブロックからアドレスを割り当てられた場合、他の AS に接続して経路情報をアナウンスすることは不可能であり、もし複数の AS を跨って移動する場合は、どのプロバイダにも割り当てられていないアドレスを使用する必要があるが、移動ネットワークが大量に存在した場合に、世界的に経路情報を増やすことになり運用上好ましくない。

静的経路を用いた運用

また OSPF 等は設定が複雑なため、外部への経路が一つしか存在しない末端のネットワークでは、静的にデフォルトの経路を設定している場合が多い。そのような運用をしているネットワークに移動ネットワークが接続した場合、経路をアナウンスしても、その情報を受け取って伝播させることが不可能であるため、結果的に経路情報が外部に伝わらないことになる。

2.2.4　セキュリティからみた移動ネットワークの問題点

インターネットが発展するにつれてセキュリティの重要性が喧伝されているが、移動ネットワークはセキュリティの観点からみた場合さまざまな問題がある。

OSPF の認証の問題

まず MR が経路情報を OSPF でアナウンスする場合を考える。現在の OSPF ではパスワードによる簡単な認証をおこなっている。そのため新規接続先のネットワークの管理者

と事前に調整し、MR の管理者はそのパスワードをあらかじめ知っていなければならない。さもなければ移動ネットワークの経路をアナウンスできないからである。しかし接続される側のネットワークの管理者の立場からすると、自組織の OSPF のパスワードを教えるなどということは、とうてい受け入れられる条件ではない。

発信アドレス偽造攻撃への対策

ある Class B のアドレスをもった組織があり、そのアドレスの一部が移動ネットワークにつけられていると仮定する。CIDR の普及によりクラスの概念がなくなったため、移動ネットワークが組織から離れた場所で経路情報をアナウンスすることは可能ではある。

しかし同一の AS 内での移動の場合で、たとえパスワードの問題が解決できたとしても、接続先のネットワークのルータがセキュリティ対策のため、経路情報のフィルタリング、すなわち指定したルータ以外からの経路情報は受け取らないという設定をしている場合が多いため、MR のアナウンスする経路情報は隣接ルータに伝播しない可能性が高い。

このフィルタリングをおこなわない場合、自組織のアドレスの一部が組織外からアナウンスされても受け入れてしまう。そうした場合、悪意を持つ攻撃者が侵入するためにパケットの発信アドレスを偽造して、その組織内のホストになりすますということが可能になってしまう。よってもし自組織のアドレスの一部が移動ネットワークとなって移動先から経路情報をアナウンスした場合、その経路情報が悪意を持つ攻撃者の侵入手段か、移動ネットワークであるかの判別はなんらかの認証機構がなければ不可能である。

2.3 移動ネットワークの設計

前述したように、現在の通常のインターネットの運用の枠組では移動ネットワークを実現することは不可能である。よって既存の移動ホスト対応プロトコルを基にして移動ネットワークを実現することにする。今回は VIP 方式を用いて以下に示す 3 種類の方式で移動ネットワークを設計した。

2.3.1 VIP 方式の概要

まずここでは VIP 方式の概要を解説する。

VIP 方式は VIP 層と IP 層の 2 つのサブレイヤのアドレスの組合せで動作する。VIP アドレスは計算機の識別子を表し、IP アドレスはネットワーク上の位置情報を表している。

移動計算機は移動先のネットワークで DHCP [36] 等を用いて一時的に IP アドレスを取得し、それを CAMT パケットによりホームルータに通知する。ホームルータの AMT には VIP アドレスと IP アドレスの組合せが登録される。さらに通知のパケットが通過した、途中にある VIP 対応のルータにも同じ AMT のエントリがキャッシュされる。

通常の計算機と移動計算機の通信を考えると、移動計算機の識別子宛、すなわち VIP ア

ドレス宛のパケットは経路情報に従ってホームルータへ向かう。途中に VIP 対応のルータがあった場合、VIP アドレスを鍵として AMT を検索する。AMT にその VIP アドレスと IP アドレスの組合せがキャッシュされていた場合、その移動計算機宛のパケットはアドレス変換されて、現在接続している地点、すなわち動的に取得した IP アドレス宛てに転送される。もし途中にある VIP 対応のルータに AMT エントリのキャッシュがなくても、移動計算機の経路情報をアナウンスしているホームルータまではパケットが届き、ホームルータには必ず AMT エントリが存在するので、そこでアドレス変換されて転送される。

2.3.2　VIP を用いた移動ネットワークの基本方針

VIP を用いて移動ネットワークを実現する場合、いくつかのおおまかな方針が考えられる。基本的には以下の 3 種類の方式があげられるが、それぞれの方針における特徴を対比させ、表 2.1にまとめた。

アドレス全付け替え方式 VIP のモデルに基づいて、移動先で動的にアドレスブロックを取得して、移動ネットワーク内部のアドレスを全部付け替える。

アドレス無変更—1 方式 移動ネットワーク内部のアドレスは変更しないが、移動先でアドレスブロックを取得してその経路情報をアナウンスする。

アドレス無変更—2 方式 移動ネットワーク内部のアドレスは変更せず、さらにアドレスブロックの取得と経路情報のアナウンスもおこなわない。

この表を用いて VIP を基にした方式を検討した場合、「アドレス全付け替え」方式は、移動ネットワーク内部のホストが全て VIP 対応でなければならないという制約があり、またネットワークが移動した場合には移動ネットワーク内部のホストはアドレスを付け替えなければならないことがわかる。また「アドレス全付け替え」方式と「アドレス無変更—1」方式の両方式ともネットワークアドレス、すなわちアドレスブロックを取得できるように DHCP を改造しなければならない。またこの両方式は取得したネットワークアドレスの経路情報をアナウンスする必要があるが、セキュリティの観点から、通常のネットワークの運用においては特定のルータ以外からの経路情報は受け取らない設定をおこなっている場合が多い。よって MR が接続して後、ネットワークアドレスを取得して経路情報をアナウンスしてもそれが伝播しない可能性が非常に高い。しかもこの問題は移動ネットワーク側で解決できる問題ではない。しかし「アドレス無変更—2」方式では、MR に単一の IP アドレスが必要なだけで、こういった問題は発生しないため、最も実現の可能性が高いことがわかる。よって本研究では「アドレス無変更—2」方式に基づいた実装をおこなうことにした。

表 2.1: 移動ネットワークの基本方針の比較

	全付け替え	無変更—1	無変更—2
移動ネットワーク内部のホストの種類	VIP 対応	通常	通常
動的なアドレスブロックの割り当て	必要	必要	不必要
取得したアドレスの経路のアナウンス	必要	必要	不必要
ネットワーク移動時の移動ホストへの通知	アドレスの再付け替え	通知のみ	通知のみ
移動後定常運用になるまでの時間	長い	普通	短い
基にしたプロトコルのモデルとの整合性	良い	普通	悪い
通常の IP のアーキテクチャとの整合性	良い	良い	良い

2.3.3 VIP 方式を基にした移動ネットワーク

以下に VIP 方式を用いた設計の議論のための例として、移動ネットワークは 133.138.194.0/24 というアドレスを持ち、そのホームルータは **HR1** とし、移動して 133.4.34.0/27 というネットワークに新規接続する場合を考える。またこの移動ネットワークに接続して来る、他の組織の移動ホストを **X** とし、**X** のホームルータを **HR2** とする。ネットワークが移動する前、すなわち初期状態を図 2.2に示す。これ以降あげられる移動ホストはすべて VIP アドレスで識別される。

ここで解説する方式は、前述したようにアドレスブロックを割り当てるという DHCP の拡張を必要とせず、MR が経路情報をアナウンスする必要もないために、もっとも実現が容易である。

通常の動作

ネットワークが移動した後の様子を図 2.3に示す。

1. モーバイルルータ **MR** が DHCP サーバ **DHCPS** から自分用の IP アドレスとして 133.4.34.10 を取得する。

2. **MR** には移動ネットワーク用の AMT エントリが生成される (図 2.4参照)。

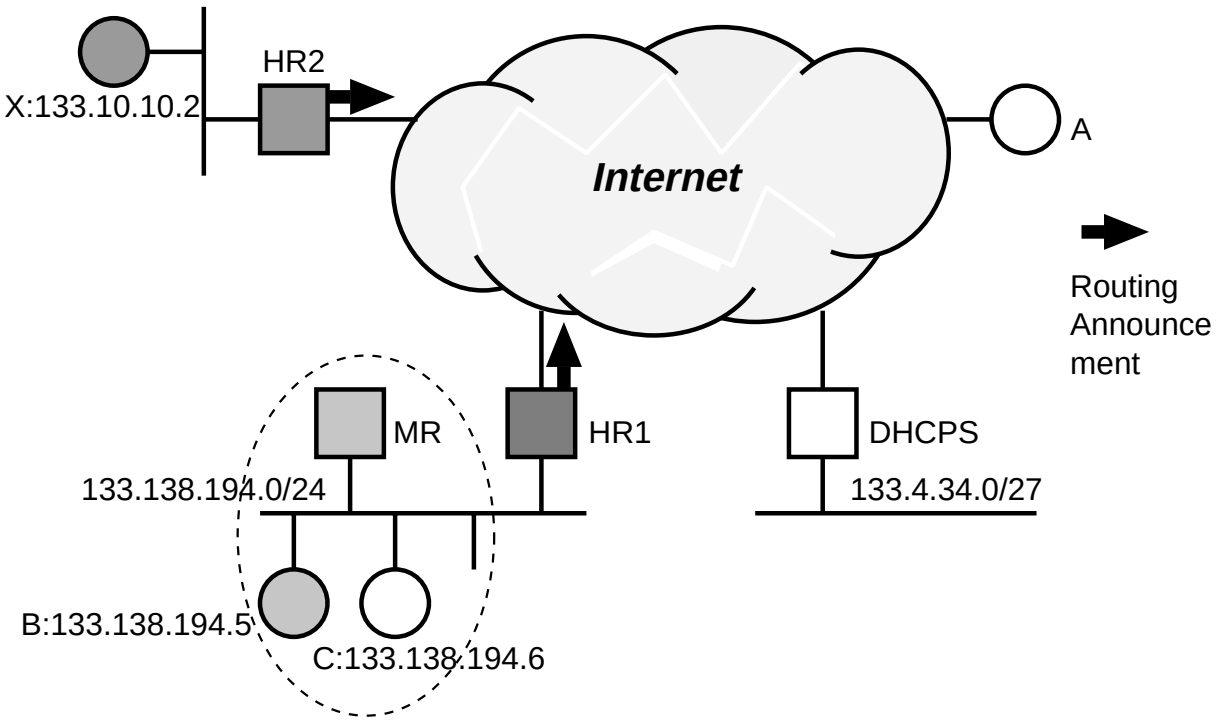

図 2.2: 移動前の初期状態

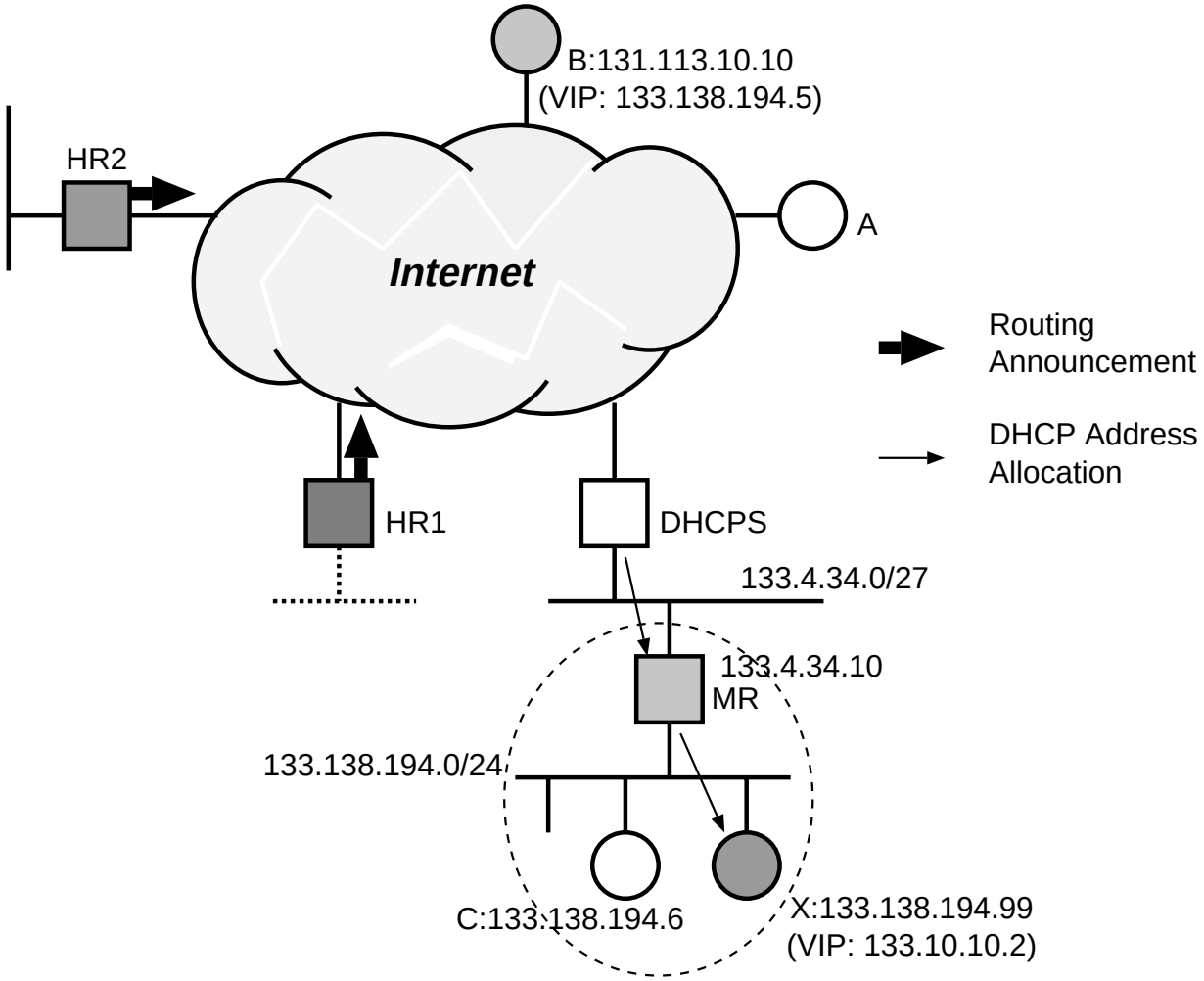

図 2.3: 「VIP 方式アドレス無変更—2」における移動後の状態

3. ホームルータ **HR1** の AMT には **MR** からの CAMT パケットにより、図 2.5に示すネットワーク対ホストというエントリが生成される。

4. 移動ネットワーク内部のホスト宛のパケットは、**HR1** でまずアドレス変換され、**MR** の AMT によって元に戻され、通常の経路制御に基づき転送される。

AMT		
133.138.194.0/24	→	133.138.194.0/24

図 2.4: MR の AMT—1

まずインターネット上の通常のホスト **A** と、移動ネットワーク内部の 133.138.194.6 という IP アドレスを持った VIP 対応でないホスト **C** との通信を考える。**A** から送出された **C** 宛のパケットは、経路情報に従い **HR1** に到達する。**HR1** では図 2.5の AMT エントリにより、宛先 IP アドレスが 133.4.34.10 に変更され **MR** に到達する。**MR** では図 2.4 の AMT エントリにより、再び宛先 IP アドレスが 133.138.194.6 に戻されて、通常の経路情報に基づいて最終的に **C** に配送される。この通信のパケットヘッダの変化を図 2.6に示す。

AMT		
133.138.194.0/24	→	133.4.34.10/32

図 2.5: HR1 の AMT—1

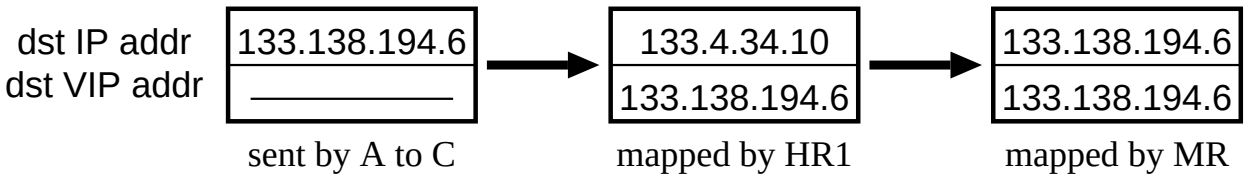

図 2.6: VIP: パケットヘッダの変化—1

ここで移動ネットワーク内部の移動対応のホスト **B**(VIP アドレス 133.138.194.5) が移動ネットワークを離脱して、他の組織のネットワークに接続され、一時的に 131.113.10.10 という IP アドレスを取得し、自身のホームルータである HR1 に CAMT パケット送信した場合を考える。**HR1** の AMT は図 2.7に示すようになり、最長一致法というアルゴリズ

ムに基づく検索により VIP アドレス (133.138.194.5) 宛のパケットは、正しく IP アドレス (131.113.10.10) にアドレス変換され転送される。なおパケットヘッダの変化は図 2.8 のようになる。

AMT		
133.138.194.0/24	→	133.4.34.10/32
133.138.194.5/32	→	131.113.10.10/32

図 2.7: HR1 の AMT—2

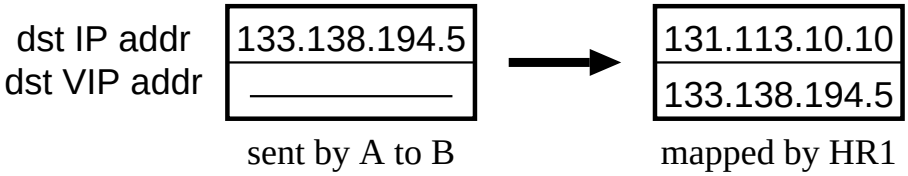

図 2.8: VIP: パケットヘッダの変化—2

他の組織の移動ホストが接続

他の組織の移動ホスト **X**(VIP アドレス 133.10.10.2) が接続されると、**X** は DHCP を用いて 133.138.194.99 という IP アドレスを一時的に取得する。CAMT パケットにより **HR2** には図 2.9 に示す AMT エントリが生成される。

AMT		
133.10.10.2/32	→	133.138.194.99/32

図 2.9: HR2 の AMT—1

しかしこのような AMT が作成されても、実際には通信はおこなえない。以下で順を追って通信の様子をみることにする。

通常のホストが移動ホスト **X**(VIP アドレス 133.10.10.2) 宛のパケットを送信すると、経路情報に従い **HR2** に到達する。**HR2** で図 2.9の AMT に基づいてアドレス変換がなされ宛先 IP アドレスが 133.138.194.99 へ変換される。このパケットは経路情報に従い今度は **HR1** に到達するが、**HR1** の AMT には 133.10.10.2 というエントリは見つからないため、アドレス変換は生じず経路情報に従ってパケットを転送しようとする。しかし移動

ネットワークが移動して **HR1** に接続されていない場合は、そのパケットは **HR1** で破棄されることになり、結果的に通信はおこなえない。この場合のパケットヘッダの変化を図2.10に示す。

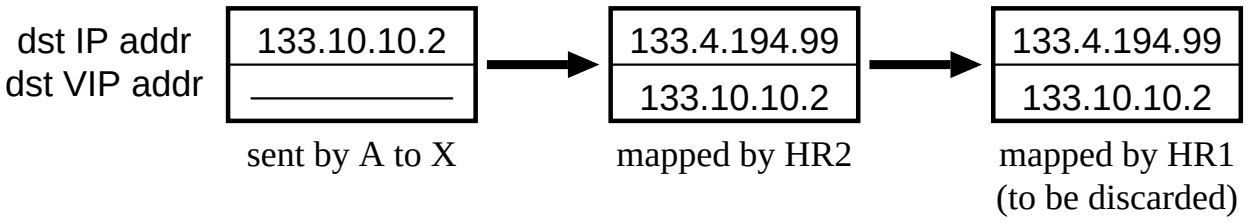

図 2.10: VIP: パケットヘッダの変化—3

以上のことから移動ネットワークに移動ホストが接続した場合には、その移動ホスト宛のパケットは、移動ネットワークが接続されている地点、すなわち **MR** へ直接転送されなければならないことがわかる。

2.3.4　仮想ネットワークと物理ネットワーク

このような要求を満たすために、新たに**仮想ネットワーク**と**物理ネットワーク**という概念を導入する。**仮想ネットワーク**は識別子としてのネットワーク、**物理ネットワーク**は位置情報としてのネットワークをそれぞれあらわす。この例の場合、仮想ネットワークは移動ネットワークの識別子である `133.138.194.0/24` で、物理ネットワークは位置情報、すなわち現在割り当てられている `133.4.34.10/32` である。

そこで CAMT パケットの仕様を変更して、仮想ネットワークアドレス (以降 Vnet addr) と物理ネットワークアドレス (以降 Pnet addr) も付加することにする。移動しない通常のネットワークに接続された場合は当然 $\mathbf{VnetAddr} = \mathbf{PnetAddr}$ であるため、このような概念は無用である。しかし移動ネットワークに接続した場合の移動ホスト **X** が送信する CAMT パケットは、図2.11のようになる。

VIP addr	133.10.10.2/32
IP addr	133.138.194.99/32
Vnet addr	133.138.194.0/24
Pnet addr	133.4.34.10/32

図 2.11: 移動ホスト X からの CAMT パケット

前節で定義した移動ネットワークの性質上、移動ホストが接続して、ホームルータに通

知する CAMT パケットは、必ず **MR** を通過する。**MR** は自分の管理している移動ネットワークのアドレスが Vnet addr である場合には、この CAMT パケットから図 2.12の下段にあるように、通常の場合と同じく VIP アドレスと IP アドレスの組合せという AMT エントリを作成する。

AMT		
133.138.194.0/24	→	133.138.194.0/24
133.10.10.2/32	→	133.138.194.99/32

図 2.12: MR の AMT—2

しかし **X** の HR である **HR2** では、受信した CAMT パケットから Vnet addr と Pnet addr と IP アドレスから図 2.13に示す AMT エントリを作成する。

AMT		
133.10.10.2/32	→	133.4.34.10/32

図 2.13: HR2 の AMT—2

よって通常のホストから **X**(VIP アドレス 133.10.10.2) へのパケットは、まず経路情報に従い **HR2** へ向かう。図 2.13 に示す AMT エントリにより、宛先IP アドレスが 133.4.34.10 に変換され、経路情報に基づいて **MR** へ転送される。**MR** では図 2.12の AMT から、宛先アドレスが 133.138.194.99 に変換され、通常の経路情報に従って最終的には **X** に到達する。この場合のパケットヘッダの変化を図 2.14に示す。

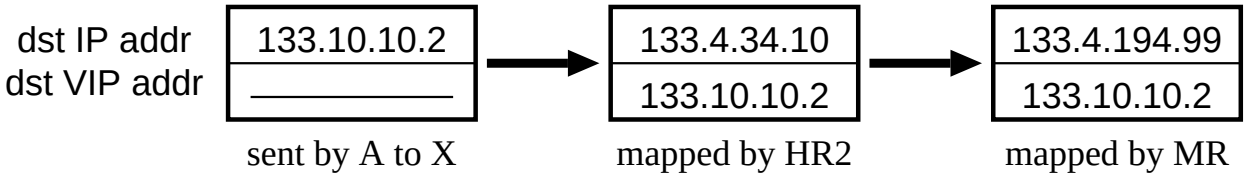

図 2.14: VIP: パケットヘッダの変化—4

2.3.5 本方式における問題点

他の VIP を用いた移動ネットワークの実現方法にあった、アドレスバルクの取得とその取得したネットワークアドレスの経路情報を伝播させるという大きな問題は回避できたが、

本方式にも以下にあげる二つの問題点があげられる。

ネットワークの移動の通知方法

まず物理ネットワークのアドレスを、他の組織から移動してきたホストに通知する方法がないことがあげられる。つまり移動ホストは、移動ネットワークの接続時に、MR が DHCP から割り当てられた IP アドレス、すなわち物理ネットワークのアドレスを知る必要がある。

現実的な解決方法として考えられるのは、移動ホストは接続時に DHCP によってアドレスを取得するので、同時にオプションとして物理ネットワークのアドレスも取得できるように DHCP を改造すれば解決する。当然割り当てられた物理ネットワークのアドレスを知っている MR が、移動ネットワーク内部の DHCP サーバとなることが望ましい。

しかしここでこの移動ネットワークがさらに移動した場合を考える。移動ネットワークが移動すると、MR は接続先で新規に IP アドレス、すなわち物理ネットワークのアドレスを再び割り当ててもらう。移動ネットワーク内部にある移動ホストは、物理ネットワークのアドレスが変更された場合、それを検知してあらたに CAMT パケットをホームルータに通知しなおさないと、移動ネットワークが直前につながっていた地点へパケットが転送され続けることになる。しかし現在の DHCP 枠組には、サーバからクライアントに対して、強制的にパラメータの変更を通知するという機構はない。

モデルとしての整合性

まずネットワーク対ホストという多対一の組合せの対応は、識別子と位置情報を、一対一の組合せに対応させるという VIP のモデルにそぐわないことがあげられる。移動ネットワーク用にアドレスブロックを取得する場合は、一旦アドレス変換が生じたパケットの宛先を見た場合に、移動ネットワーク内部のどのホストへのパケットかを判別することが可能であったが、この方式だと移動ネットワーク内部のホストへのすべてのパケットの宛先 IP アドレスが、MR が取得した IP アドレスに変換されてしまう。

2.4　移動ネットワークの実装

前節で述べた設計にしたがって、移動ホスト対応プロトコルである VIP を拡張し、移動ネットワーク対応プロトコルを実現した。この実装の特徴としては、移動ネットワークの接続を受け入れる側に特別な対応を強いる必要がないことがあげられる。そのため移動ネットワークを構成するのに必要なエンティティのみを実装すればよいため、移動ネットワークの実現が最も容易である。

2.4.1 プラットフォーム

今回の実装は、VIP バージョン 1 を基にして行なった。VIP バージョン 1 は、VIP 対応のルータであれば CAMT パケットを転送する際に、認証を行なわず無条件に AMT エントリを作成してしまう。また実験のためのプラットフォームとしては、IBM/PC 互換機上で動作する *BSD/OS 2.0.1* を選択した。

しかし VIP の場合 CAMT パケットを偽造することにより、不正な AMT エントリが作成され、移動ホストになりすますことができてしまう。よってあるホストが使用している VIP アドレスが、本当にそのホストのものなのか確認しなければならない。現在の VIP バージョン 2 [47] では、keyed MD5 と RSA を用いた移動ホストの認証機構が提案されている。これを用いることにより不正な AMT エントリの作成要求や削除要求に対して、認証を行なうことが可能になり、なりすましを防止できることになる。

2.4.2 既存の VIP からの変更点

移動ネットワークに対応するための、既存の VIP に対する主な変更点を以下にあげる。

- AMT エントリにネットワーク対ホストという対応を実現させるために、ネットマスクの概念を導入した。
- 移動ホストが移動ネットワークに接続した場合のために、仮想ネットワークと物理ネットワークという概念を新しく導入し、そのアドレスを通知するために、コントロールパケットを変更した。
- 移動ネットワーク内部は通常の経路制御が行なわれているため、変換されたアドレスを元に戻すためのダミー AMT エントリという概念を導入した。

2.4.3 AMT エントリの拡張

図 2.15に新たに拡張した AMT エントリのフォーマットを示す。変更点としては、VIP アドレスと IP アドレスのそれぞれに対するネットマスクのエントリが増えただけである。

ここで AMT エントリの検索について考えてみる。移動ネットワーク内部のホストが、単独で移動している場合には、そのホスト自身のエントリと、移動ネットワークのエントリの両方に合致してしまうが、実際にはホスト自身のエントリの方を選択しなければならない。よって検索のアルゴリズムには最長一致法を用い、目的のアドレスに合致するもののうち、ビット数のもっとも多いエントリを採用する必要がある。これは CIDR を用いた経路表の検索方法とまったく同じである。

このような検索に用いられている方法は、目的アドレスのビットパターンによってツリー状の表を形成するというものである。有名なものとして 4.3BSD Reno に実装された *Radix Tree* [48] があげられる。しかし *Radix Tree* は不連続なネットマスクをサポートするために、

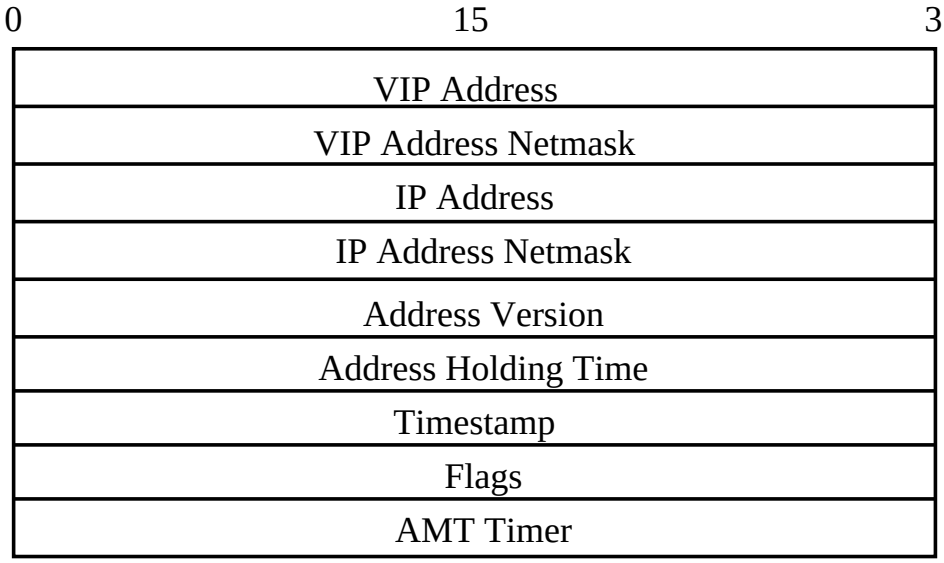

図 2.15: 拡張された AMT エントリ

ツリーの管理などが複雑であり、それに伴いソースコードの可読性も低い。さらに最近の CIDR を基本とした経路制御においては、不連続なネットマスクのサポートは不必要であるため、*Radix Tree* はオーバースペックでもある。このような現状から WIDE プロジェクトにおいて開発されたのが、*Radish* である。*Radish* はネットマスクは連続しているという前提に基づいた、ツリー状の経路表で、アルゴリズム等がかなり単純化されており、またソースコードの可読性も高い。

今回は AMT にネットマスクの概念が導入されたが、不連続なネットマスクをサポートする必要性はまったくないため、AMT は構造を大幅に変更し、*Radish* を採用する予定であった。しかし動作の確認を最優先したため、今回の実装では AMT を固定長の配列で管理し、最も単純な線形検索による方法を一時的に採用した。当然近い将来には *Radish* を利用したものに改良する予定である。

2.4.4　コントロールパケットの変更

AMT エントリにネットマスクの概念を導入するため、当然ながら AMT の作成や削除を行なうためのコントロールパケットのフォーマットにもネットマスクのフィールドを追加する。図 2.16に新しいコントロールパケットのフォーマットを示す。

従来の VIP のコントロールパケットと比較すると、Target VIP Address と Traget IP Address にそれぞれネットマスクの情報が増えていることがわかる。

移動ネットワークに他の組織の移動ホストが接続してきた場合に対応するために、新たに仮想ネットワークと物理ネットワークの概念を導入すると述べた。しかし図 2.16にはそ

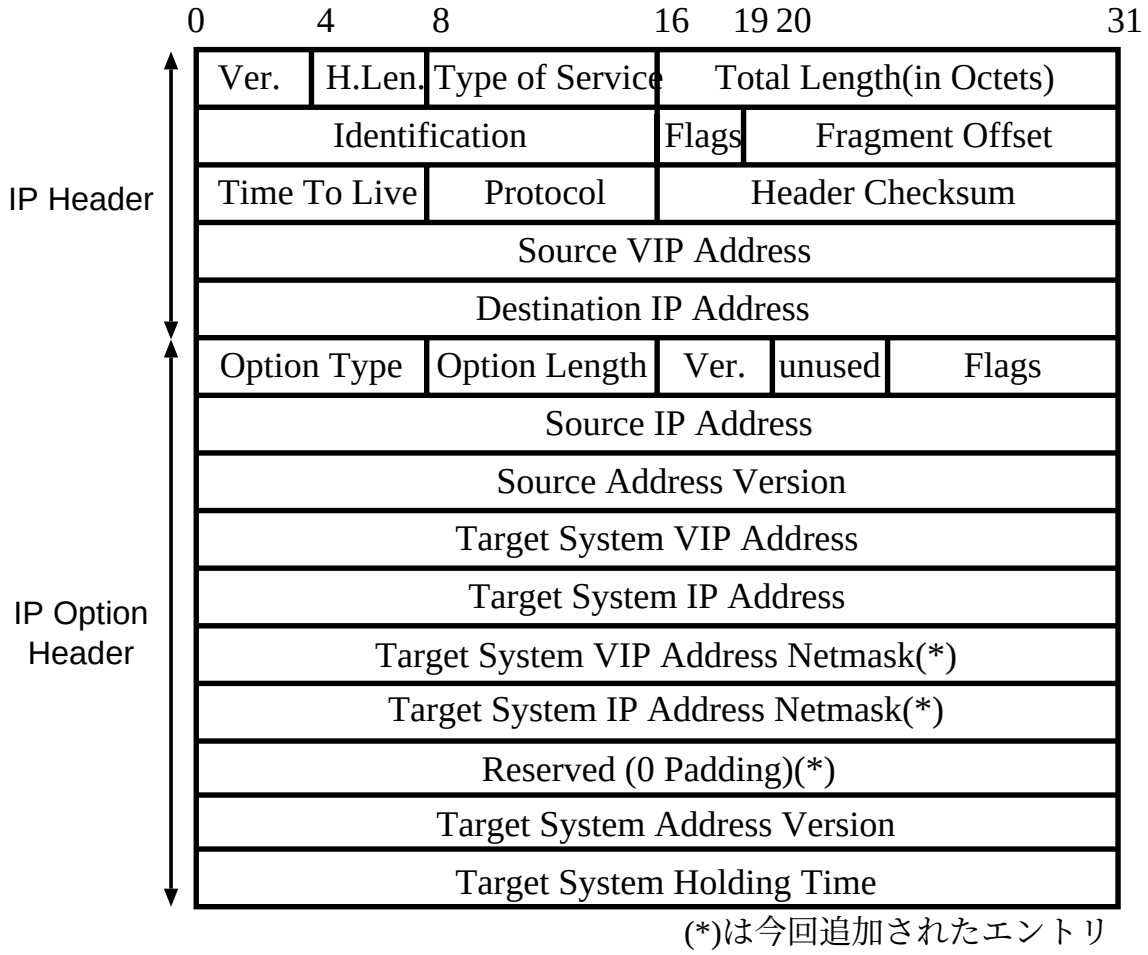

図 2.16: 新しいコントロールパケット—1

のフィールドが存在しない。これは VIP が IP オプションを用いて実装されていることに由来する制限で、IP オプションに含めることができるデータは最大でも 40 オクテットまでである。すなわちコントロールパケットには必要なすべての情報を含めることが不可能である。

そこで、MR が移動した場合の CAMT パケットと、移動ネットワーク内部に他の組織の移動ホストが接続した場合の CAMT パケットでは、必要となる情報がそれぞれ異なることに着目し、コントロールパケットを 2 種類用意することにした。この 2 種類は “**Flags**” のフィールドの flag によって判別する。移動ホストがあるネットワークに接続した場合、もし移動ネットワークであった場合、MR のアドレス、すなわち物理ネットワークアドレスを HR に通知しなければ、その移動ホストは通信が継続できない。よって移動ホストは、DHCP 等を用いて、接続したネットワークが移動ネットワークか通常のネットワークかを判別し、どちらの形式の CAMT パケットを HR に送信すれば良いのかを判断しなければならない。図 2.17に移動ネットワークに接続した場合のコントロールパケットを示す。

この図 2.17のコントロールパケットは、移動ホストが移動ネットワークに接続した場合のみ使用される。よって “**Target VIP Address**” と “**Target IP Address**” の組合せは、ホスト対ホストになるため、それぞれのネットマスクの情報は不必要である。また “**Virtual Network Address**” のフィールドが無く、“**Virtual Network Netmask**” しか存在していないのは、“**Target IP Address**” と “**Virtual Network Netmask**” から “**Virtual Network Address**” が簡単に算出できるためである。

また “**Physical Network Address**” は、MR が接続先で割り当てられたアドレスであり、そのネットマスク情報である “**Physical Network Netmask**” フィールドは本来必要のないものであるが、将来的にアドレスブロックを割り当てることが可能になった場合のために、“**Physical Network Netmask**” フィールドは予約されている。しかし現状では常に `255.255.255.255` という値が設定される。

なおデータパケットに関しては、まったく変更を行なっておらず、今回基にした VIP バージョン 1 と完全な互換性がある。

2.4.5　ダミー AMT エントリ

MR において移動ネットワーク内部のホスト宛のパケットは、宛先 IP アドレスを元の状態に戻して、通常の経路制御に従って配送する必要がある。よって宛先 IP アドレスを元に戻すためのダミー AMT エントリが必要になる。図 2.18に `133.138.194.0/24` というアドレスの移動ネットワークの MR 上のダミー AMT エントリを示す。

このダミー AMT エントリはコントロールパケットで生成や削除されないように、特別なフラグを立てている。MR のダミー AMT エントリは、MR の管理者がコマンドなどから手動で設定する必要がある。

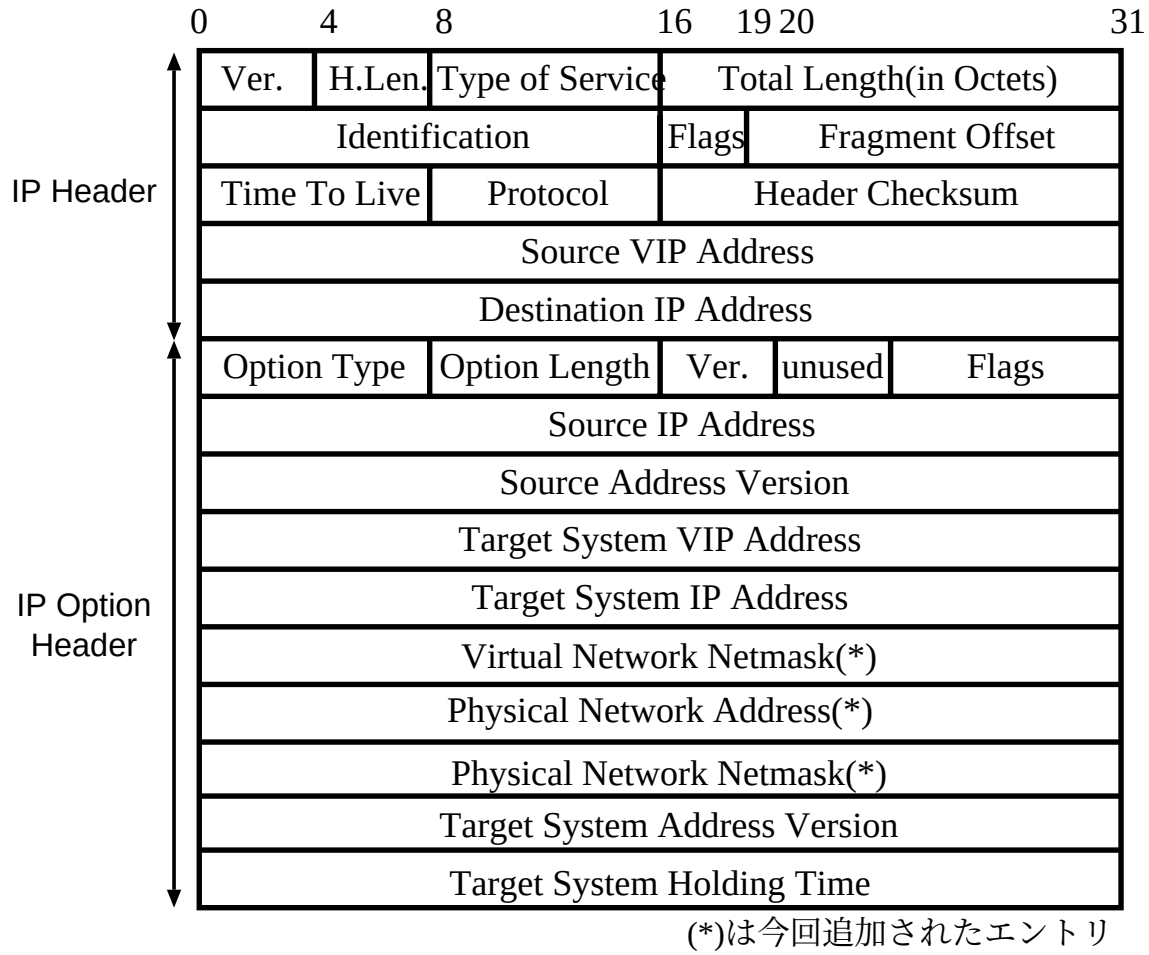

図 2.17: 新しいコントロールパケット—2

AMT		
133.138.194.0/24	→	133.138.194.0/24

図 2.18: MR のダミー AMT の例

2.5　評価と考察

今回の実装を実際に簡単な構成のネットワークを構築して実験し、前述した移動ネットワークの条件に対し、どの程度対応できているのかを、表 2.2にまとめてみた。。

表 2.2: 実装状況のまとめ

内部のホストは移動対応でなくとも通信をおこなえる	○
ホストが移動対応であれば単独で離脱しても通信を継続できる	○
内部に他の組織の移動ホストが接続しても通信を継続できる	△ (実験が不十分)
移動ネットワークの規模が大きくなっても対応できる	× (未検証)
移動ネットワークはさまざまな方法でインターネットに接続できる	× (未検証)

2.5.1　考察

今回の実験では移動時のアドレス変更をはじめとする各種設定は、すべて手動でおこなった。今回の実装とは直接の関係はないが、モーバイルコンピューティングにおいて各種パラメータを取得し、自動的に設定をおこなう DHCP のような機構の必要性を強く認識した。

もし移動ホストが移動ネットワークに接続した場合には、CAMT パケットに物理ネットワークアドレス (MR のアドレス) が必要である。現在の実験環境のように小規模な場合や、MR の管理者が近くにいる場合は、物理ネットワークアドレスを尋ねて、人間がそれを設定することは可能である。しかし手動での設定は煩雑であり、将来的に飛行機内などの比較的大規模な LAN に接続する場合などでは、管理者に尋ねて設定するという方法は現実的ではない。また現時点では、移動ネットワークは単一の物理的セグメントという構成であるが、将来的には複数のセグメントから構成され、大規模化することが予想される。

よって将来の大規模化した移動ネットワークを運用するためには、物理ネットワークアドレスをはじめとするさまざまなパラメータを自動的に取得する、ないしは配布されたパラメータの変更を動的に通知するなどの、現行の DHCP の枠組に因われない柔軟な機構が必要である。

しかし実際の利用においては、移動ホストは移動して別のネットワークに接続した場合、必ず DHCP 等によって自身の IP アドレスを取得する必要がある。よって新規に何らかの方法を開発して、移動ネットワークであるかどうかの判断と物理ネットワークアドレスを取得するよりは、現行の DHCP を拡張して同時にそれらのパラメータも取得できるように改造するのが効率的である。

また移動ネットワークを実際に運用する場合は、今回想定したイーサネットによる接続よりも、電話回線経由による PPP 接続を用いる場合のほうが多いことが予想される。現状

だと自動車内の LAN からのアクセスは恐らくデジタル携帯電話経由であろうし、家庭内 LAN の接続先を切り替える場合でも、一般的には通常のモデムや ISDN を使用することが多い。今回の実装ではイーサネットに限定されていたため、DHCP によるアドレス取得しか考えられていないが、PPP 接続でのアドレス取得方法は DHCP ではなく IPCP を用いるので、運用方法の相違点を調査する必要がある。

またセキュリティに関する問題として、移動ネットワークの MR は、接続先において実際にアドレスを割り当てる必要があるため、その時点でなんらかの方法で認証をおこなうことは可能である。しかし移動ネットワーク内部のホストについては、MR さえ接続してしまえば、何の認証もなく MR 経由で通信をおこなうことが可能になってしまう。

2.6 今後の課題

- 今回の実装に間に合わなかった AMT の *Radish* への対応をおこなう。
- 移動ホストが移動ネットワークに接続した場合に、物理ネットワークのアドレスを自動的に取得できるようにするために DHCP を改造する。
- 通信媒体をイーサネットでの接続に限定していたが、電話回線経由の PPP 接続にも対応しなければならない。原理的には今回の実装で動作するはずであるが実験する必要がある。
- 今回検証しきれなかった、他の組織の移動計算機が、移動ネットワーク内部に接続する場合の通信についてもさらに実験する必要がある。
- AMT を作成する際に認証をおこなう VIP バージョン 2 と統合する。

第 3 章

PFS: 携帯型計算機のためのファイルシステム

3.1 はじめに

共有ファイルシステムは多くのアプリケーションが依存しており、これを携帯型計算機に適応させることにより、それらのアプリケーションの多くが利用可能になると考えられる。我々は、これまでに移動型計算機に適応した共有ファイルシステムとして、PFS (Personal File System) [49] を開発してきた。

従来の共有ファイルシステムでは、ファイルサーバとの間の通信路が高速であり、安定して利用できると仮定している。そのため、移動によって通信速度や通信の安定性が動的に変化し、時には通信路が使用不能になる携帯型計算機では、極端な応答性の悪化や、一時的にファイルが使用不能になる等の問題が生じる。

PFS では、携帯型計算機側にキャッシュディスクを用意し、キャッシュディスクの更新をファイルアクセスと非同期に行うことにより、通信状態に依存しない高速なファイルアクセスを実現した。しかし、実際の環境での利用により、いくつかの問題点が明らかになった。

特に、常に同じキャッシュ更新アルゴリズムを利用することにより、携帯型計算機の変化に富む通信環境に適応しきれない問題点がある。例えば、通信路が極端に低速な際には PFS の通信が通信路を占有してしまい、通信路を共有している他のアプリケーションの通信に影響を及ぼしてしまう。しかし、逆に高速な際にはその通信路の高速性を有効利用できない。

そこで本研究では、従来の PFS に新たに動作モードの概念を導入し、状況変化に細かく対応できるように PFS を再度設計、実装した。

計算機の接続している通信路の特性と利用状況に応じて、動作モードを切り換え、異なるキャッシュ更新アルゴリズムを利用することにより、動的に状況に適応することが可能になった。

3.2 現在の共有ファイルシステム

3.2.1 NFS

NFS [50] は Sun Microsystems Inc. が開発した共有ファイルシステムであり、異なる計算機間でのファイル共有を主な目的として設計されている。

NFS はシンプルなクライアント/サーバモデルに基づいて設計されている。サーバはクライアントからのファイルアクセス要求に対して処理を行ない、処理が終了すると返答を行なう。サーバからクライアントへの返答はサーバ上でその処理が完全に終了したことを表わしている。サーバは原則として状態を持たないので、何らかの障害で返答が送られない場合はクライアント側で再試行することで動作を続けることができる。

NFS が利用する NFS プロトコルは、機種やオペレーティングシステム、ネットワークアーキテクチャ、トランスポートプロトコルなどに依存しないように設計されている。さらにプロトコルの仕様が公開されているため、非常に様々な計算機に実装され、最も普及している共有ファイルシステムの一つとなっている。

携帯型計算機で NFS を利用した場合、基本的にクライアント側で応答を確認するまでは次の要求を出せない為、サーバとクライアント間の通信に遅延の大きな通信路を利用した場合には応答性が極端に悪化するとう問題点がある。また、サーバとの通信路が途絶えた場合は、クライアントは処理を続行することができなくなる。

3.2.2 AFS

Andrew File System (AFS) [51] は、カーネギメロン大学で開発された分散ファイルシステムであり、広域ネットワークにおけるファイル共有を考慮して設計されている。

AFS も NFS 同様にクライアント/サーバモデルに基づいて設計されている。AFS の特徴としてコールバックと呼ばれる機構がある。これは、あるファイルがあるクライアントのキャッシュに読み込まれだ状態で、他のクライアントがそのファイルを変更した際、サーバがキャッシュしているクライアントに変更を通知する機構である。これにより、クライアントは一度読み込んだファイルを長期間キャッシュしておくことが可能になる。

しかし、サーバとクライアントの間の通信が途絶えると、クライアントはコールバックを受け取れないために、キャッシュの内容の正当性を保証できなくなり、結果としてファイルアクセスができなくなる。

3.2.3 CODA

CODA [52] はカーネギメロン大学で開発されたファイルシステムであり、ファイルサーバとの通信路が利用不能な間もファイルアクセスを続けることが可能となっている。

CODA はクライアント/サーバモデルに基づいて設計されている。CODA クライアント

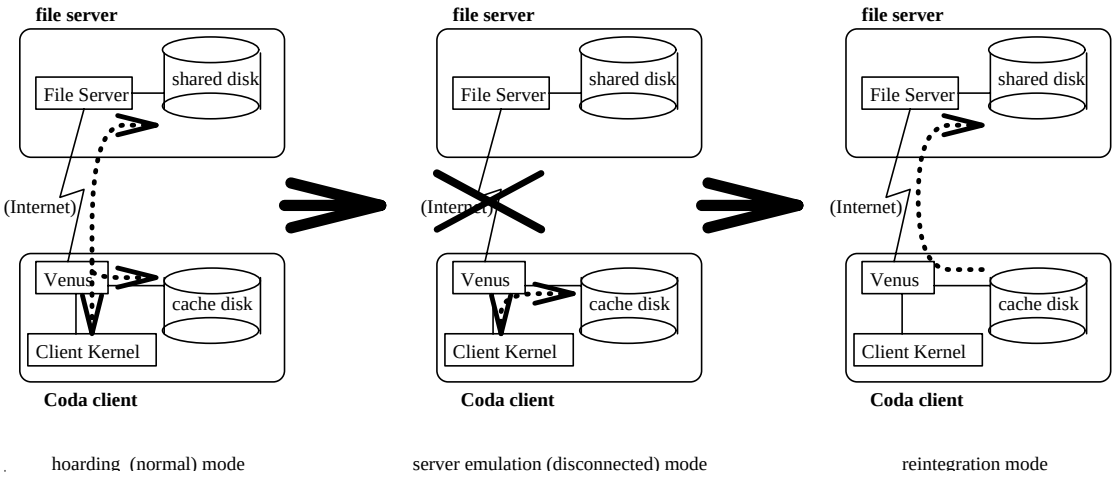

図 3.1: CODA の動作

では、Venus と呼ばれるキャッシュマネージャプロセスが動作し、キャッシュの管理とサーバとの通信をとりおこなっている。

サーバとの通信路が利用できる間にアクセスされたファイルは、キャッシュされ、通信路が切断されるとキャッシュのみの動作を開始する。通信路が再び利用可能になると、通信途絶中のファイル変更をサーバに反映させ、通常の動作に復帰する (図 3.1)。

また、通信断中の作業に必要なファイルを HDB (Hoard DataBase) と呼ばれるデータベースに登録しておくと、その内容を予めキャッシュに取り込んでおく機能を持っている。これにより、通信路が途絶える直前にアクセスしていなくても、必要なファイルにアクセス可能となっている。

しかし、CODA はサーバ/クライアント共に、オペレーティングシステムのカーネルに変更が必要であり、他の共有ファイルシステムと共存が難しいため、導入コストが高いという問題がある。また状態が、通信路が利用可能な状態と、切断中の状態の 2 つしかないため、通信路の状況によっては対応できない。

3.2.4　ブリーフケース

マイクロソフト社の Microsoft Windows 95 に付属の「ブリーフケース」は、シンプルで粗なファイルの共有を提供するアプリケーションである。

ブリーフケースは書類鞄のメタファを利用している。鞄に書類棚から取りだした書類を入れて外出し、外出先で参照/変更を行ない、外出先から帰ると書類棚に戻すという動作をファイルに対して行う。すなわち、外出前に携帯型計算機にファイルサーバからファイルをコピーし、外出先で参照/変更を行ない、外出先から帰るとファイルサーバのファイルを更新する。

この機構は直感的に理解しやすく、機構が単純なため導入も容易である。

しかし、全ての操作をユーザが行うため、ユーザは常に利用しているファイルを把握している必要がある。また、ファイルの更新も明示的に行なわなくてはならないため、更新を忘れてファイルの一貫性が崩れる等の問題が起こることが予測される。

3.2.5 LITTLE WORK Project

LITTLE WORK Project [53] はミシガン大学でおこなわれている、移動型計算機環境を構築する研究である。

LITTLE WORK ではすぐに利用可能な技術を組みあわせた移動型計算機環境の構築を目指している。計算機本体にはポータブルタイプの IBM-PC 互換機を利用し、オペレーティングシステムには Mach 2.5 [54] を、ファイルシステムには AFS (Andrew File System) [51] を改良したものを使用している。

LITTLE WORK の使用しているファイルシステムは、従来の AFS のクライアント側のみを、サーバとのネットワーク接続が断たれても動作するように、改良したものである [55]。

通常の AFS では、サーバとの通信断等によりキャッシュの内容の正当性が保証できなくなると、キャッシュの内容を無効にしてサーバとの整合性を保つ。しかし、この機構では、ネットワークから切り離された状態で利用される移動型計算機では、クライアントがネットワークから切り離された途端にキャッシュを利用できなくなる。

そこで、LITTLE WORK では AFS のキャッシュマネージャーを改造し、ネットワークから切り離されてキャッシュの正当性を保証できなくなってもキャッシュを使い続けるようにしている。

計算機をネットワークから切り離し、ユーザの disconnect コマンドによりキャッシュマネージャーがディスコネクテッド・モードになると、以後のファイル操作が内部に記録される。再度ネットワークに接続すると、この記録をもとにサーバへ変更の反映がおこなわれる。

このような改造をおこなうことにより、ファイルの一貫性を完全に保証することはできなくなるが、ネットワークから切り離されても動作することが可能になる。一貫性はクライアントがネットワークから切り離されている間にサーバとクライアントの両方で同一ファイルに対する書き換えが起きた時にくずれるが、これは非常にまれであるとされている [55]。

LITTLE WORK を利用することにより、AFS を利用した環境下ではクライアントの改造のみで携帯型計算機とのファイル共有が可能になる。しかし、クライアント側ではオペレーティングシステムのカーネル内部の改造が必要になるため、クライアントのオペレーティングシステムを限定してしまう上、AFS を利用していない環境ではサーバを AFS に切り換えなくてはならず、導入コストが高くなる。

3.2.6　まとめ

本節で述べたように、サーバとの通信路が極端に変化する携帯型計算機環境では、通常の計算機のために設計された共有ファイルシステムはうまく動作しない。携帯型計算機での利用を考慮して設計されたファイルシステムも存在するが、導入が容易でなかったり、ユーザの負担が大きかったりする。

したがって現在、導入が容易であり、ユーザにファイルシステムの介在を意識させにくい共有ファイルシステムが必要とされている。

3.3　従来の PFS

我々はこれまでに、キャッシュを利用することにより通信途絶に耐え、かつ高速な共有ファイルシステムである PFS を設計、実装した。本章では PFS の概要を述べ、その評価によって明らかになった問題点を示す。

3.3.1　目標

PFS は携帯型計算機での利用を前提とし、容易に導入できることを目標とした、共有ファイルシステムである。

これまでに述べたように、携帯型計算機においては、ファイルサーバに対する通信路が常に確保できるとは限らず、またその通信路も動的に変化する。そこで、PFS が実現すべき目標として次のことを設定した。

- 接続するネットワークの状態に影響されずにファイルアクセスを行う
  携帯型計算機では動的にネットワークの状態が変化するため、それによってファイルアクセスが阻害されない機構が必要である。

- ユーザがストレスを感じない程度の速度を実現する
  ファイルシステムの速度はそれを利用する全てのアプリケーションの実行速度に影響を与えるため、できるだけ高速に動作する必要がある。

- 作成したファイルは必ずどこかに残る
  何らかの問題により、同名の異なる内容のファイルができてしまった場合でも、ファイル名を変えるなどして全てのファイル内容の保存をおこなう必要がある。

- 既存のファイルシステムと共存する
  既に他の共有ファイルシステムでファイル共有を行なっている環境に、携帯型計算機を付加するような場合、容易に導入を行なうために、既存のファイルシステムと共存できることが必要となる。

- 透過的なファイルアクセスを実現する
  アプリケーションの変更を不必要とし、ユーザの負担を最小限にするために、アプリケーションからは通常のファイルシステムとして透過的にアクセスできなければならない。

- できるだけ導入を容易にする
  汎用性を高め、できるだけ利用する OS などを特定しないようにする。また、既存のシステムの変更を最小限にする。

3.3.2 前提条件

PFS では、この機構を実現するために、利用環境として次の前提条件を設定した。

- 携帯型計算機は個人が専有して利用する

- 書き込むのは専有ユーザが管理するファイル (ホームディレクトリ以下のファイル) のみ
  これはサーバとクライアント間のファイルの一貫性保持を容易にするためである。

- 移動中に必要となるファイルはある程度予測可能である
  これはユーザが移動中に明らかに必要となるファイルを明示的に指定することによって、キャッシュを有効に利用するためである。前提条件として個人による専有があるため、これは現実的な条件である。

- ファイルサーバとの内容の一貫性は完全には保証しない
  これは通信が途絶えている間も書き込みを許し、移動中も作業を続行できるようにするためである。

3.3.3 設計

PFS はクライアント/サーバモデルで構成される。ファイルサーバとしては通常の計算機を利用し、携帯型計算機がファイルの供給を受けるクライアントとなる。クライアント側ではキャッシュディスクを用意し、ユーザが必要であると指定したファイルをあからじめキャッシュに読み込んでおく。ユーザのファイルアクセスは全てキャッシュに対して行い、それとは非同期にキャッシュの内容をサーバ側と一致させるように動作する。

キャッシュに読み込まれていないファイルに対してのアクセスもキャッシュ内のそれと同様に扱われ、必要に応じてサーバからの転送を行う。サーバからの転送中はアクセスはブロックされる。

キャッシュの更新をアクセスとは非同期に行なうことにより、キャッシュに入っているファイルに対するアクセスはサーバとの通信の終了を待つ必要がない。そのため、低速な回線、

または通信路が断絶した状態でも問題なくアクセスを続けることができる。また、キャッシュに入っていないファイルに関しても通信路が利用可能である限りは最初のアクセスに時間がかかるのみで、その後は高速なアクセスをおこなうことができる。

ファイルアクセスには局所性があるため、単なるアクセス頻度によるキャッシュでも効果が期待できる。PFS ではさらに、使用するユーザがそれを指定してあらかじめキャッシュに入れておく機構を用意する。これにより、非常に高いキャッシュヒット率を得ることができ、少なくともユーザの指定したファイルはキャッシュに存在させることができる。この結果、通信路が利用不能な場合でも少なくともユーザの指定したファイルに対するアクセスは継続できる。

サーバとクライアントの間の通信路としては、VIP のような機構により移動透過性が実現された環境を期待し、ネットワーク上の位置の変化に関しては PFS では原則として関知しない。ただし、導入を容易にするため、移動透過性が確保されていなくても、移動後に再接続を行うことにより動作する。

3.3.4　実装

PFS の実装はプログラムの可搬性を高めるために、全てユーザプロセスとして実装している。クライアント側 (CS) はオペレーティングシステムとのインターフェースとして、最も普及している共有ファイルシステムの一つである NFS プロトコル [50] を利用し、PFS は擬似的な NFS サーバとしてファイルの提供を行う。サーバ側 (MS) はどのようなファイルシステムも共有できるように、通常のファイルアクセスのシステムコールを用いてファイルアクセスをおこない、クライアントに対して CUP (Cache Update Protocol) を用いてファイルの供給をおこなう (図 3.2)。

この実装により、クライアント側の OS には NFS クライアントの機能が実装されていれば PFS が動作し、サーバ側では通常のファイルアクセスとクライアントとの通信が可能であればファイルの共有をおこなうことが可能になっている。また、サーバで参照しているファイルシステムは、他の共有ファイルシステムによって他のホストと共有されていても問題ないため、従来のファイル共有環境に付加する形での利用が可能になっている。

3.3.5　評価

スループット

回線速度のスループットに与える影響を、非常に広く用いられている共有ファイルシステムの NFS と、PFS とのあいだで、比較した結果を表 3.1 に示す。実験は 2 台の BSD/386 1.1 が動作している IBM-PC 互換機を、イーサーネットと、RS-232C による SLIP (Serial Line IP) で接続し、iozone ベンチマークテストによりスループットを測定した。iozone ベンチマークテストは最初に巨大なファイルを書き込み、その後そのファイルを読み込む。結

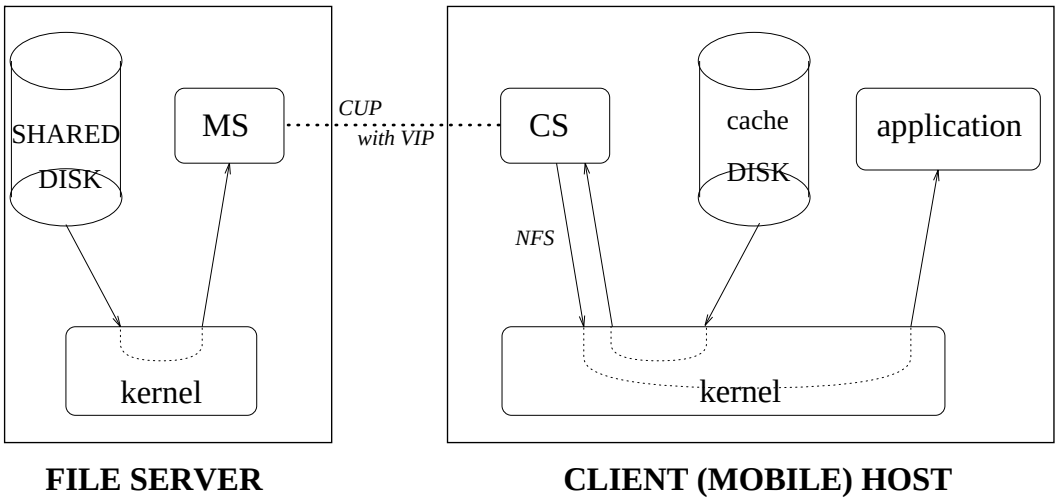

図 3.2: PFS の構造

表 3.1: 回線速度のスループットに与える影響

ファイルシステム	書き込み速度 (byte/sec)	読み込み速度 (byte/sec)
NFS (10Mbps)	75812	270032
NFS (9600bps)	599	617
PFS (10Mbps)	194994	182904
PFS (9600bps)	176301	182884

果は作成したファイルのサイズをそれぞれのファイル操作にかかった時間で除したものである。作成するファイルの大きさは操作の所要時間が 10 秒を越えるように設定される。

この表から、PFS のスループットは回線速度に殆ど影響されず、かつユーザプロセスで実装されていても、通常の NFS とほぼ同程度の速度が実現できることが確認できた。

それに対して、NFS では 9600bps のような低速な回線には適応できず、極端にスループットが悪化している。

特に書き込みにおいて、高速な回線においても PFS は NFS よりも好成績をあげているが、これは NFS では NFS リクエストの単位毎に、サーバでのディスクへの完全な書込み保証とクライアントでの応答確認を義務づけているのに対し、PFS ではクライアントのローカルファイルシステムへの書き込みが終了した時点で NFS サーバとしてのファイル操作が終了する為であると考えられる。

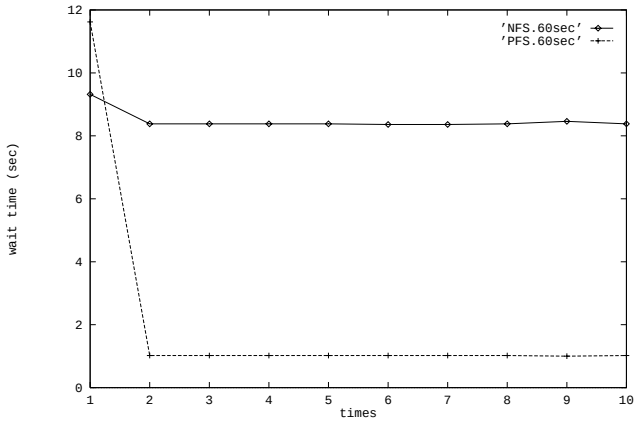

図 3.3: NFS と PFS の応答速度の差

表 3.2: PFS と他のファイルシステムとの比較

	導入の容易さ	WAN での利用	通信途絶中の動作	一貫性保持
NFS	容易	速度等に難あり	不可能	完全
AFS	容易ではない	良好	不可能	完全
LWP †	AFS 環境では容易	良好	可能	通信途絶中は保証なし
CODA	容易ではない	良好	可能	通信途絶中は保証なし
PFS	容易	良好	可能	完全には保証なし

†LITTLE WORK Project による改良型 AFS

応答性

9600bps の回線を用いた NFS と PFS において、20 個のエントリが存在するディレクトリを ls コマンドを用いて 60 秒の周期で読み出すのに要する時間を、測定した結果を図 3.3 に示す。横軸が試行回数、縦軸が所要時間である。どちらもキャッシュが消去された状態から測定を開始した。

このグラフより、低速な回線において NFS よりも良好な応答性を実現していることが読み取れる。

定性的性能

他のファイルシステムとの定性的な比較を、表 3.2 に示す。

3.3.6 問題点

PFS の実装により、携帯型計算機におけるキャッシュを利用したファイルアクセスの有効性は確認できたが、実際に利用した結果、次のような問題点が明らかになった。

- 通信回線が利用可能な場合、それがどんなに低速でもサーバとの通信を行うため、回線を共有している他のアプリケーションの通信を阻害する事がある。
- 高速な通信回線を利用できる場合にも、ファイルの一貫性の保持が不完全となる。
- ファイルサーバとの間でファイルの内容の一貫性が取れなくなった時にいずれかのファイルが突然もう片方の内容に変わってしまい、元のファイルにアクセスするには別のファイル名でアクセスする必要がある。
- ファイルの内容をそのまま転送している為、低速な回線の時に転送効率が悪く、広域ネットワークを利用した際などには盗聴の恐れがある。また、転送を中断するなどの細かな制御が不能である。
- ネットワーク的に近い場所に、同等の資源があるにもかかわらず、遠いサーバの資源を利用してしまう。
- キャッシュディスクの容量が限られている場合、作業に十分なファイルが保持できない事がある。

次章以降でこれらの問題点を解決する改良を提案する。

3.4 改良型 PFS の設計

PFS に次のような改良を加えることにより、問題点を解決する。

- 動作モードの導入
- ファイル実体の切り替え機構
- サーバ切替え機構
- ファイル転送方式の改良
- 中間サーバの導入
- キャッシュ内容の圧縮

本研究では特に動作モードに着目し設計を行なった。それぞれの改良について、詳細を述べる。

3.4.1　動作モードの導入

従来の PFS は、通信路の状況やユーザの意思に関わらず、NFS アクセスとは非同期に常にキャッシュを最新の状態にするように動作していた。

しかし、ファイルサーバとの間で高速な通信路が利用できる時には、NFS アクセスとは非同期のキャッシュ更新ゆえの、ファイル内容の伝播遅延が、ファイルの一貫性保持のうえで、大きな問題となる。

逆に、通信路の容量が少ない環境で利用している時には、キャッシュ更新のための通信量が通信路を圧迫し、他のアプリケーションの利用を妨げる結果となる。

この問題は、PFS が常に同じアルゴリズムでキャッシュの更新を行なっているために生じている。したがって、計算機の置かれている状況にあわせたアルゴリズムの切り替えを行えば解決可能である。

そこで、PFS に動作モードを導入し、モードによってキャッシュ更新アルゴリズムを切り替える。モードの切り替えはクライアントが PFS 自身の通信状況をモニタし、自動的に行なう。その際にユーザがモード遷移の方針を指示することにより、PFS が利用する通信量を、制御することを可能とする。

モードの基本集合としては、今回は以下に述べる 4 種類を用意しした。

NFS compat mode

NFS compat モードは NFS と同様にファイルアクセスと同期してキャッシュの更新を行なうモードである。NFS が利用できるような高速で安定した通信路が利用可能な場合を想定している。

利点としては、サーバとのファイルの一貫性の保証が行われることである。欠点は、低速な通信路では極端に応答性が悪化し、通信路が途絶えた状態では全くファイルアクセスができなくなることがある。また、必要とする通信量も多い。

Async Update mode

Async Update モードは従来の PFS と同様にファイルアクセスと非同期にキャッシュの更新を行なうモードである。通信路の状態が変化するような状況を想定している。

利点としては、あらゆる通信路で良好な応答性を得られることがあげられる。また、NFS モードに対して必要な通信量も削減される。欠点は、サーバとのファイルの一貫性保持が不完全であることである。

Fetch Only mode

Fetch Only モードはファイルアクセスは通常通り行なえるが、キャッシュへの更新に関してはサーバからの読み込みのみを行ない、キャッシュ内容のサーバへの書き戻しは抑制されるモードである。抑制された書き戻しは、Async Update mode 等の書き戻しが許され

表 3.3: ユーザの意思とモード

項目	要求/状態				
一貫性と応答性	一貫性優先		応答性優先		
キャッシュ更新	any		書き戻す	書き戻さない	更新しない
回線速度状態	高速	低速	any	any	any
モード	NFS	Async Update		Fetch Only	Disconnected

るモードに遷移した後に実行される。利用できる通信路の容量が小さいという状況を想定している。

このモードでは、キャッシュがミスヒットしたファイルはサーバから読み込むが、変更したファイルのサーバへの書き戻しは抑制されるため、クライアント上で可能な操作を変えずに書き戻しにかかる通信量を削減することが可能となっている。

欠点は、書き戻しが許されるモードに遷移するまでは、クライアント上でのファイル書き込みがサーバに反映されないことである。

Disconnected mode

Disconnected モードはキャッシュ内のファイルに対するアクセスは行なえるが、キャッシュの更新は抑制されるモードである。キャッシュに存在しないファイルに対するアクセスはエラーとなる。書き込まれたファイルのサーバへの反映は、Fetch Only mode と同様に、それが許されるモードに遷移した後に実行される。

利点は PFS のキャッシュ更新の為の通信が一切行なわれず、他のアプリケーションの通信を阻害しないことである。

欠点はアクセス可能なファイルがキャッシュ内のものに限定されることと、他のモードに遷移するまではクライアント上でのファイル書き込みがサーバに反映されないことである。

ユーザの意思と、回線状況に対する、適切なモードを表 3.3 に示す。この表におけるユーザの意思は一貫性と応答性のどちらを優先するか、どの程度のキャッシュ更新を行なうかである。

モードにより PFS が通信路をどのように利用するかを管理することが可能となり、通信路の状況に応じた、通信路の利用と、一貫性保持の程度の制御が、可能となる。ユーザの指定をモードの遷移に反映させることにより、ユーザが PFS の使用する通信量を制御することができる。

本研究では、本節で提案している改良点のうち、特にこのモードの導入に焦点をあてる。

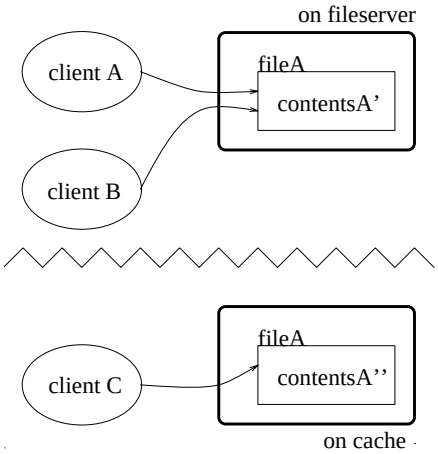

図 3.4: 通信途絶中の一貫性保持の失敗

3.4.2　ファイル実体の切り替え機構

通信路が利用不能な状態でもファイルの書き込みを許すため、サーバとクライアントの両方で別々にファイルの更新が行われ、ファイル内容の一貫性がとれなくなる可能性がある (図 3.4)。その際、従来の PFS ではどちらかのファイルを優先し、もう片方を別のディレクトリに保存するという方法で内容を一致させ、ユーザに通知している。これにより、変更のあったファイルを必ず保存するという目標を実現し、かつファイル内容を一致させている (図 3.5)。

しかし、別のディレクトリに移動された側のファイルを参照していた利用者にとっては、それまで参照していたファイルが突然別のファイルに切り替わることになり、混乱のもととなった。

そこで、一貫性がとれなくなった場合、クライアントのファイルをファイルサーバ上で新たな別のファイルとして扱い、双方でアクセスを継続できるようにする。

これによりユーザは、サーバとの一貫性がとれなくなっても、実行中の作業を継続することができ、さらに後にそれを統合するための全てのファイルを得ることができるようになる。

3.4.3　ファイル転送方式の改良

従来の PFS ではファイルの転送を行う際、ftp などと同様に新たに TCP のコネクションを設定し、その回線をそのまま使って転送を行う方法をとっている。

この方法は、TCP によるフロー制御等をそのまま使えるため単純だが、一方で、転送を中断するなどの細かな制御ができない。また、ファイルの内容によっては圧縮可能なデータをそのまま転送することになる。

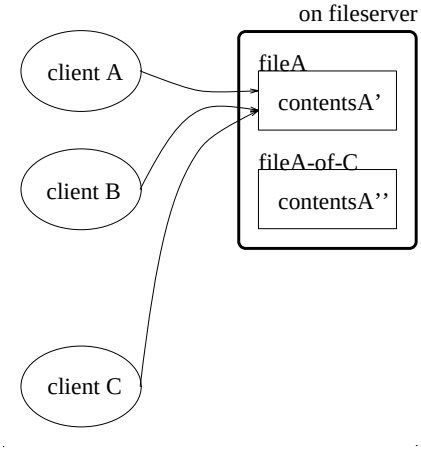

図 3.5: 従来の PFS における一貫性の回復

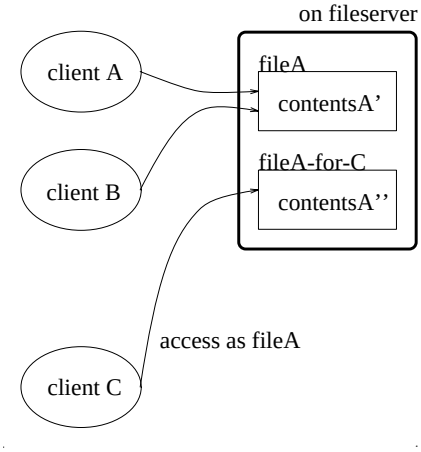

図 3.6: 改良した PFS の一貫性の回復

転送時データが圧縮可能なものであり、かつ圧縮/伸張の時間よりも転送時間の方が長いような場合には、圧縮してから転送した方が効率がよい。さらに移動先の信用できないネットワークに接続することを考慮すると、盗聴を防ぐため、転送時に暗号化を行うことが考えられる。

そこで、ファイル転送方式を可変長のパケットに分割して送信する方法に変更し、転送データに型を導入する。型には従来通りの生データの他、圧縮型、暗号化型等を用意し、必要に応じてデータ形式転送を変更できる機構にする。

これにより、ファイル転送を中断するなどの細かな制御が可能となり、かつ、安全で効率のよい転送がおこなえる。

3.4.4　サーバ切替え機構

従来の PFS はファイルサーバとサーバ上のマウントすべきディレクトリ名の組をボリュームとして管理し、複数のボリュームを扱うが、それぞれは関連づけられずに、別のファイルツリーとして処理される。一方で特定のオペレーティングシステムの特定のシステムディレクトリ以下のファイルなどは、どのファイルサーバでも同じ内容である。例えば SunOS 4.1.3 の /usr/bin ディレクトリ以下のファイル群はどのサーバでも同じ内容であると考えることができる。

クライアントの計算機が移動した際に、ファイルサーバとのネットワーク的な距離が遠くなり、通信状態が悪化することがある。そのような場合に移動先の近傍に同等のファイルサーバが存在すれば、それを利用することにより効率のよいアクセスが可能となるが、それまで利用していたサーバとの関連づけがなされないため、キャッシュしていた内容が全て無駄になる。

そこで、単一のボリュームに複数のファイルサーバとトップディレクトリを登録できるようにし、その中から最も効率のよいアクセスが可能となるサーバを選択することを可能にする。

これにより、移動をおこなっても効率のよいアクセスが可能なサーバに切り換えることが可能となる。

3.4.5　中間サーバの導入

ファイルサーバとの通信路の状況によっては、キャッシュミスの際に起こるファイル転送の待ち時間が非常に長くなる。また、キャッシュの書き戻しも同様に長時間を要し、キャッシュディスクの容量が少ない場合には、これもユーザの待ち時間となる。

この待ち時間を減少させるため、補助的なキャッシュとして働く中間サーバを導入する。ネットワーク的近傍に配置した中間サーバを経由してファイル転送を行うことにより、ユーザの待ち時間を減少させる (図 3.7)。

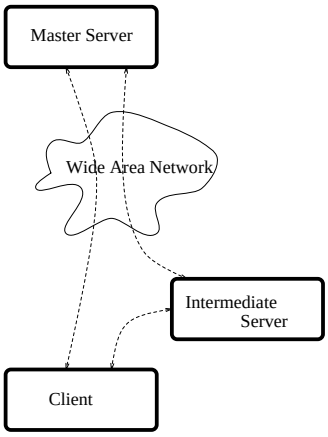

図 3.7: 中間サーバの概念図

これにより、中間サーバまでの通信が高速に終了すればクライアントは次の作業に移ることができるようになり、ユーザの待ち時間が減少することが期待できる。

3.4.6 キャッシュ内容の圧縮

従来の PFS のキャッシュはファイルサーバ上のファイルと 1 対 1 に対応するファイルを通常のファイルシステム上に用意し、その中に内容を格納している。

この方式の利点は、ファイルの管理が容易となり、クライアントのオペレーティングシステムで最適化されたファイルアクセス機構をそのまま利用できる点である。さらに、キャッシュディスクとして利用するファイルシステムの種別は問わないため、半導体ディスクや圧縮ディスク等を利用することも可能となる。

圧縮ディスクはメディアにファイル内容を格納する際に、すべてまたはアクセス頻度の低いファイルを圧縮して格納し、アクセス時には伸張して読み出す機構である。これにより、メディアの容量以上のファイルを格納するができる。これは、ファイルアクセスの時間に対して、圧縮/伸張にかかる時間が無視しうる程高速な場合に有効である。

しかしながら、多くの携帯型計算機は通常の非圧縮ファイルシステムを利用しており、ファイルを圧縮して保存することによりディスクの有効利用をはかれることが多い。また、計算能力が低く、全てのファイルを無条件に圧縮/伸張するとアクセス速度が問題になることもある。

そこで、PFS の管理するファイルアクセス頻度情報を利用し、アクセス頻度の低いファイルに関しては圧縮して保持する機構を導入する。全てのファイルを圧縮しないのは、アクセス頻度の高いファイルは伸張のオーバーヘッドなしに参照可能にするためである。

これにより、少ない携帯型計算機の資源の一つであるキャッシュディスクを最大限に利

用することが可能となる。

3.5　実装

本研究では、改良点のうち特に PFS にとって特徴的な動作モードに着目して実装をおこなった。

改良型 PFS の実装は、BSD/OS 2.0.1 上で C 言語で行なった。Sun RPC [56] の呼び出しインターフェースと XDR [57] データ変換ルーチンは XDR 言語でプロトコルを記述し、OS に附属の rpcgen コマンドによって C 言語のソースコードを生成した。NFS 及び mount プロトコルの記述には、Sun Microsystems Inc. が配布している rpcsrc.4.0 中の nfs_prot.x を利用した。

基本的構造は改良前の PFS と同様だが、複数のキャッシュ更新アルゴリズムを適用できるように、キャッシュ更新部の他の部分からの独立性を高め、各ブロックのスケジューリングを変更したため、全面的に再実装した。同時にサーバ、クライアント共に複数の通信を同時に行えるようにした他、各ブロックに前回の実装の経験をもとにした新しい工夫をおこなった。

3.5.1　サーバ

サーバは、MS (Master Server) と呼ぶ、単一のユーザプログラムとして実装した。

MS は同時に複数のクライアントにファイルを供給可能であり、管理者権限で動作するが、一般ユーザ権限で動作させてもよい。その場合、他ユーザのファイルへの書き込みは失敗する。

MS は単一プロセスとして動作し、クライアントからの接続を受け、要求に応じて動作する。複数の通信を同時に行うために内部にスケジューラを持ち、UNIX の non-blocking I/O を利用して低速な通信が他の通信に影響を与えないように実装した。

CUP

MS の通信プロトコルは CUP (Cache Update Protocol) と呼ばれ、データの標準化に XDR を利用した可変長のメッセージを、TCP/IP の全二重回線を通じて送ることによって通信を行う。

MS に対する要求メッセージには要求種別とシリアル番号が付けられ、それに対する応答には同じ番号が含まれる。これにより、MS は即時処理ができない要求を保留して、次の要求を処理することが可能になっている。さらに、サーバからの能動的なメッセージを応答の合間に送ることも可能になっている。

応答の種別には、正常終了とエラーの他、保留、再試行要求があり、更に細かな原因が伝達される。これにより、キャッシュ更新時にサーバのディスク容量が不足するなどの一時

的なトラブルに対処することが可能になっている。

ファイルの転送には通常の要求メッセージとは別の TCP 接続を行ない、可変長のメッセージを用いて転送をおこなう。したがって、ファイルは他の要求の送受信と並行して転送されることになり、ファイル転送中も他の要求を続けておこなうことが可能となっている。転送されるデータが圧縮済みかどうかなどの、データの種別は CUP のメッセージで伝達される。

3.5.2 クライアント

クライアントは CS (Cache Server) と呼ぶ、単一のユーザプログラムとして実装した。

CS は仮想的な NFS サーバとなり、クライアント計算機のオペレーティングシステムからの NFS プロトコルによるファイルアクセス要求を受けつけ、キャッシュ内容を元にそれを処理する。

キャッシュ内のファイルは「ボリューム」と呼ばれるグループに分けられ、管理されている。各ボリュームには、ボリューム名が付けられ、そのファイル群を供給可能なサーバのホスト名と、サーバ上のディレクトリ名の組のリストなどが、ボリューム設定ファイルとして用意される。

CS では NFS プロトコルによるファイル操作の処理の他に、NFS のマウントプロトコルの処理と、キャッシュ更新の為に各サーバとの通信を行う。各処理に必要な情報の共有を密にするため、これらの処理は全て単一のプロセスとして実装し、内部で以下のようなブロックに分けられている (図 3.8)。

- Cache Block
  キャッシュの入出力とファイルシステムの構造の管理を行う
- NFS Block
  キャッシュの内容を利用して、擬似的な NFS サーバとして動作する。
- Mount Block
  NFS のマウントプロトコルを処理し、NFS Block へのアクセスに必要なファイルハンドルを NFS クライアントに供給する
- Cache Update Block
  MS との通信によってキャッシュの更新を行う
- Volume Block
  ボリュームに関する情報を管理する
- Scheduler Block
  各ブロックのスケジューリングを行う

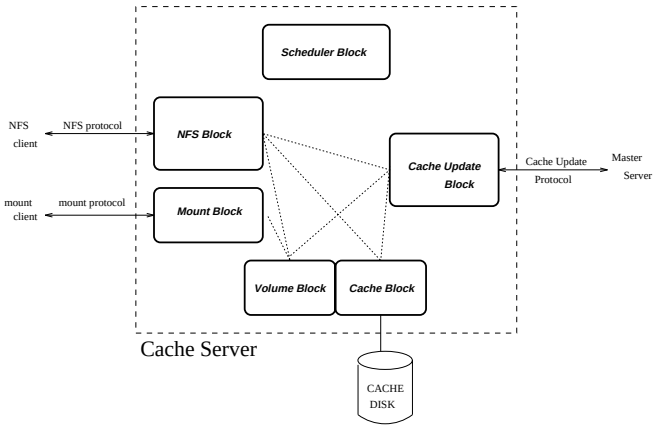

図 3.8: CS の内部ブロック

3.5.3　モードによる動作の変化

動作モードは大局変数に格納され、各ブロックからの参照が可能になっている。モードによる各ブロックの動作は以下のようになる。

NFS compat mode

NFS compat モードでは、NFS hook subblock においてファイルアクセスの前または書き込みの場合は後に、キャッシュ内の該当ファイルがサーバと同一のものかどうかの確認をおこなう。その際、確認が終了するまではファイルアクセスはブロックされる。

NFS 要求のタイムアウトによる再送が行なわれた場合の性能悪化を防ぐため、同一の NFS 要求がごく短時間に連続した場合には確認を省略する。また、一定時間 (初期値 10 秒) の間サーバからの応答がなかった場合には、NFS モードを利用するには不適当な通信路であると判断し、Async Update モードに遷移する。

Async Update mode

Async Update モードでは、NFS アクセス時に該当ファイルがキャッシュ内に存在するかどうかの確認をおこなう。キャッシュ内に存在しなければ、サーバからキャッシュへの読み込みをおこなう。キャッシュにファイルが存在すれば、NFS 要求に対してキャッシュ内容を元に応答し、アクセスされたファイルの一覧であるアクセスリストに追加する。この際、サーバとの通信が停止してれば開始する。

また、ファイルアクセスと並行してサーバに対する要求の待ち行列を処理する。処理すべき待ち要求がなくなると、アクセスリストを元に新たにキャッシュ更新のリクエストを生成し、キャッシュの更新を行なう。

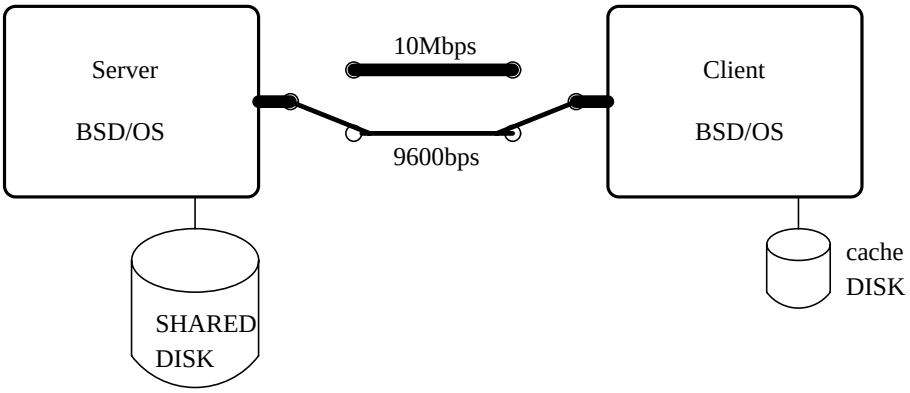

図 3.9: 実験環境

Fetch Only mode

Fetch only モードは、基本的に Async Update モードと同じ動作をおこなうが、書き戻しのリクエストを生成しない。さらに待ち行列に存在する書き戻しのリクエストは待ち行列の送信時に先送りにされる。

Disconnected mode

Disconnected モードに遷移するとサーバとの通信を終了する。以後 NFS アクセスに対応してアクセスリストへの登録は行なうが、サーバへのリクエストは生成しなくなる。登録されたアクセスリストは他のモードに遷移した後に参照され、書き込みの処理等に利用される。

3.6 評価

評価は、BSD/OS を動作させている 2 台の IBM-PC 互換機を使用した。通信路には、低速な回線として 9600bps の RS-232C ケーブルによる SLIP による接続と、高速な回線として 10Mbps の ethernet による接続を用意し、切り換えて使用した (図 3.9)。

3.6.1 回線使用率

モードによる通信回線の使用率の違いを確認するために、9600bps の通信路において、Async Update モードと Fetch Only モードで、約 3Kbyte のファイルのコピーに PFS が

使用するトラヒックを測定した。コピー元ファイルはファイルサーバ上に存在し、キャッシュにはディレクトリ情報のみが存在する状態で測定を開始した。

測定には tcpdump コマンドを利用し、PFS の使用する全 TCP コネクションのデータ長を加算して算出した。

結果を図 3.10 に示す。このグラフから Async Update モードの際には、サーバからのコピー元ファイルの読み込みと、コピー先ファイルの書き戻しの 2 つの大きなトラヒックの山があるが、Fetch Only モードの際には読み込みのみでサーバへの書き戻しは抑制されていることが読みとれる。

この PFS のトラヒックが他の通信に与える影響を測定した結果が図 3.11 である。このグラフは、上記の PFS の通信と並行して 56 バイトの ICMP echo パケットをクライアントホストからサーバホストに 1 秒毎に送信し、その応答が到着するまでの時間 (RTT: Round Trip Time) を測定したものである。

グラフから、通常の応答時間は 185msec 程度で対話的用途にも十分利用できる回線が、PFS が回線を利用すると同時に急激に応答性を悪化させ、最悪時には応答時間が 2229msec と極端に長くなっていることが読み取れる。Async Update モードの際にはこの悪化がトラヒックと同様に 2 回起るが、Fetch Only モードの際には書き戻しの抑制により 1 回に減っている。さらにグラフには含まれていないが、Disconnected モードでは PFS の通信そのものが抑制されるので、PFS による通信の阻害は起らなかった。

3.7　一貫性保持

PFS の Async Update モードでは、サーバでファイルを書き換えたという情報がクライアントに伝達されるまでに時間がかかるが、NFS Compat モードではファイルの参照前にサーバに問い合わせるので、高速なネットワーク接続が行われている限りは即時に反映され一貫性が保証された。

3.8　自動モード切り変え

PFS が NFS Compat モードで動作している時に、サーバの応答に要する時間を測定しており、何らかの原因でサーバの応答が極端に低速になると、自動的に Async Update モードに遷移することを確認した。

3.9　結論

携帯型計算機で利用可能な共有ファイルシステムである PFS に、キャッシュ更新モードを導入し、状況に応じて最適なキャッシュ更新アルゴリズムを利用することができるよう

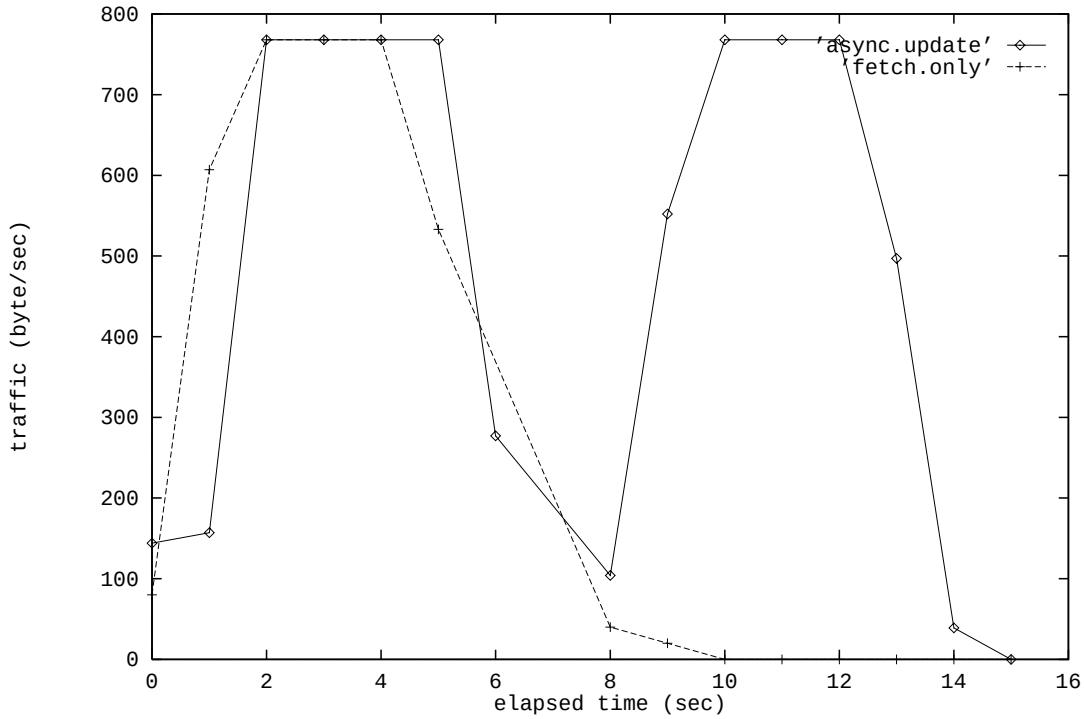

図 3.10: ファイルコピーのトラヒック

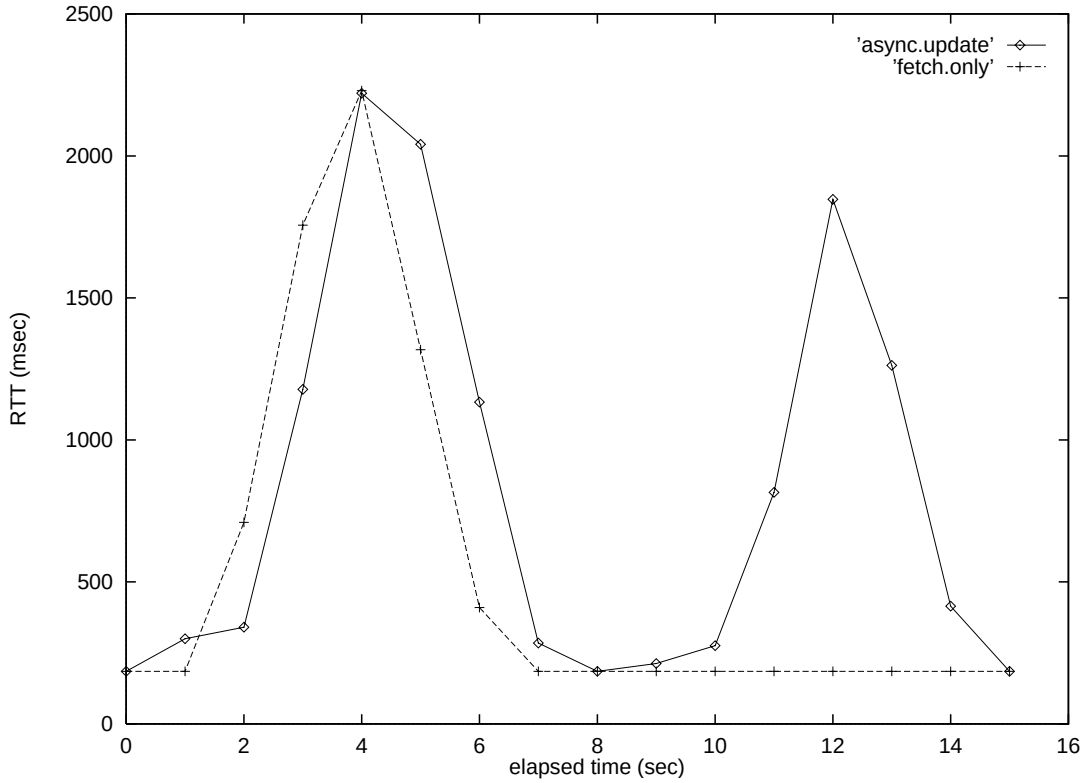

図 3.11: ファイルコピーの他の通信への影響

になった。これにより、通信回線の状況に応じたきめの細かな制御が可能となり、PFS の実用性が向上した。

3.10 今後の課題

3.10.1 動作モード以外の改良点

今回の実装では改良点のうち、モードの導入に関して特に着目したため、他の中間サーバやファイル実体切替え機構等は、一部の実装のみを行なった。モード以外の実装状況を以下に示す。

ファイル実体切り替え機構 切り替えファイル名の検出まで

サーバ切り替え機構 順次接続を試み、最初に接続したサーバを利用

ファイル転送方式の改良 圧縮、暗号化転送は実装されていない

中間サーバの導入 中間サーバに関する情報は持つが、接続は行なわない

キャッシュ内容の圧縮 実装はしていない

今後、これらの実装もおこなう必要がある。

3.10.2 モード選択の自動化

現在の実装では、ユーザの意思はモードの選択という方法で指定しているが、今後は、動作モードの選択をより自動化し、ユーザが直感的に意思を指定できる機構の構築が必要である。

3.10.3 アプリケーションによる解決の限界

PFS の実装の経験から、携帯型計算機の利用環境を整備するには、アプリケーションのみでは解決できないことが明らかになった。

PFS では、携帯型計算機環境においても TCP/IP による通信の連続性を確保し、移動透過性を保証するために VIP を利用している。VIP によりクライアントのネットワーク上の位置は隠蔽され、サーバはクライアントの VIP アドレスによってクライアントを識別することができ、移動前のファイル転送を移動先で継続することも可能になっている。

しかし同時に、位置情報が隠蔽されるために、近傍のサーバを自動選択するなどの、位置を利用した動作ができないという問題が生じる。

また、携帯型計算機にとってバッテリーの電力は動作に必要な非常に貴重な資源であるが、ハードディスクのモーターはバッテリーを大量に消耗する。

これに対処するため、多くの携帯型計算機ではディスクアクセスがない間はモーターの回転を止め、アクセスが必要になってから再度回転させる事により、バッテリーの消耗を押えている。

このような環境では、ディスクへの読み書きはモータが回転している間に集中して行う事が望ましいが、現在の計算機システムではこのような情報を適切に取得/利用する機構がほとんど存在せず、オペレーティングシステムからアプリケーションまでこのような情報を利用できないため、不定期にアクセスを行う。そのため、アクセス時に停止しているモータが、回転を始めて安定するまでの待ち時間がそのままユーザの待ち時間になるほか、モータの回転と停止を繰り返して結局電力を消耗する結果となることがある。

これらのことから、携帯型計算機の利用環境を整備するには、アプリケーション層を始めとする各層個別の対応だけでは限界があり、各層で情報を隠蔽するだけでなく、必要に応じて他の層に情報を提供する機構が必要となってくると考えられる。

第 4 章

地理的位置情報システム

本章では，Geographical Location Information (GLI) System を提案する．GLI System では現実世界におけるエンティティとインターネット上の識別子の対応付けを提供する．GLI System は次の 3 つから構成されている．1) それぞれのエンティティの地理的位置情報をデータベースで管理するサーバ，2) サーバにエンティティの地理的位置情報を登録するエージェント，3) エンティティまたは位置を問い合わせるクライアント．サーバは，エンティティの識別子と地理的位置情報からなるエントリを管理する．クライアントは指定した位置を鍵として識別子を検索し，逆に指定した識別子から位置を検索することができる．また，プロトタイプの GLI System の設計及び実装について述べる．このプロトタイプでは，インターネット上の識別子として IP アドレスを使用する．

我々のプロトタイプを基にした実験によって GLI System の有効性が示された．また，GLI System はマシンのアーキテクチャに依存しておらず，BSD/OS，SunOS，NEWS-OS といったオペレーティングシステムでテストされている．このシステムを使用した実験は現在インターネット上で継続中である．

4.1 はじめに

現在インターネットはその劇的な普及とともにかつて以上に日常生活とのより密接な関連が生じている．この分野での研究は，移動する計算機やネットワークに接続された非計算機エンティティを利用した実用的な応用を目標としている．その結果，インターネットに接続されたエンティティの数は絶えず増加しており，またその位置が広域に分散し，トポロジは常に変化している．これらのエンティティに実用的な手法でアクセスするためにはインターネットに接続されたエンティティの地理的な位置情報を獲得することが必要となる．

TCP/IP のネットワークアーキテクチャではすでに IP アドレスが接続されたホストの論理上の位置を識別する手段を持っている．しかしながら，インターネットを通じてアクセス可能なエンティティの地理的位置情報を提供するアーキテクチャやシステムは，それが強く必要とされているにも関わらず存在しない．そこで現実世界のエンティティとその地理的な位置との関係を定義する必要がある．本章では エンティティの地理的な位置とイン

ターネット上の識別子を対応付ける Geographical Location Information (GLI) System を提案する．

プロトタイプではインターネット上の識別子として IP アドレスを使用している．そのシステムによって，物理的な位置に基づいてエンティティの識別子を検索し，エンティティの識別子に基づいて位置を検索することができる．したがってユーザは，現実世界のさまざまなエンティティをインターネットを通じて検索し，リアルタイムに指定したエンティティの最新の位置情報を獲得することが可能になる．

4.2　プロトタイプ設計

ここでは，前節で述べた問題を考慮しながら，Geographical Location Information (GLI) System のプロトタイプの設計について述べる．

4.2.1　設計概要

GLI プロトタイプシステムでは，ホストの位置情報を管理する．本システムを通じてアクセス可能なエンティティはホストであり，IP アドレスがエンティティの識別しとして使用される．従って，本システムではホストの GLI と IP アドレスとの対応付けを提供する．そのような対応付けの結果として，ユーザは次のような問い合わせ行なうことができる．

- 指定したエンティティの地理的位置の要求
- 指定した地理的位置でのエンティティの識別子 (IP addresses) の要求

以上の問い合わせに対応する必要な機能は次のようになる．

- WIT:Who Is There?
  地理的位置に基づくエンティティ識別子の検索
- WAY:Where Are You?
  エンティティ識別子に基づく地理的位置の検索

GLI パラメタ

表 4.1に，本プロトタイプで使用される GLI パラメタを示す．基本的な GLI パラメタは location, velocity, device type, datum, data type, time からなる．location は緯度経度高度の座標で表される．Velocity は北方向-東方向-上方向の速度からなる．device は位置を取得する装置のタイプを表す．datum は，測地系で GLI をローカルエリアでの GLI に変換するために使用される．data type は精度を表し，time は GLI が取得された時刻を表す．

表 4.1: GLI Parameters

location	Latitude-Longitude-Altitude
velocity	North-East-Up
device	GPS etc.
datum	WGS-84,OSGB-36,NAD-27, etc.
data type	C/A,P,RDGPS, Kinematic, etc.
time	time of fix

位置情報管理

サーバではエンティティの GLI を管理する．本プロトタイプではシングルサーバアーキテクチャを使用する．分散サーバは今後のバージョンで考慮される．

位置情報注射

一般的にエンティティは接続された装置から GLI を取得する．そのような装置を持たないエンティティのために，位置情報を別のエンティティに注射する注射機能を導入する．本プロトタイプでは注射するエンティティが注射をされるエンティティのどのくらい近くにいるかは考慮してない．

位置情報更新

サーバは，エンティティの情報を収集する．エンティティはネットワークを通じてサーバへ情報を送信する．更新情報の送信時期は mobility factor と，前回取得された GLI と現在の GLI，精度との比較によって決定される．

4.2.2 基本構造

GLI System の基本構造を図 4.1に示す．GLI を管理するために相互に協調する 3 つのモジュールを導入する．

Agent

エージェントは各々のエンティティで動作し，GLI を集め，サーバに登録する．さらにエージェントは他のエンティティからの GLI 注射要求を受け取って，注射を必要とするエンティティに対して GLI を返す．エージェントはまたクライアントからの最新 GLI 要求を受け取り，その情報を返す．

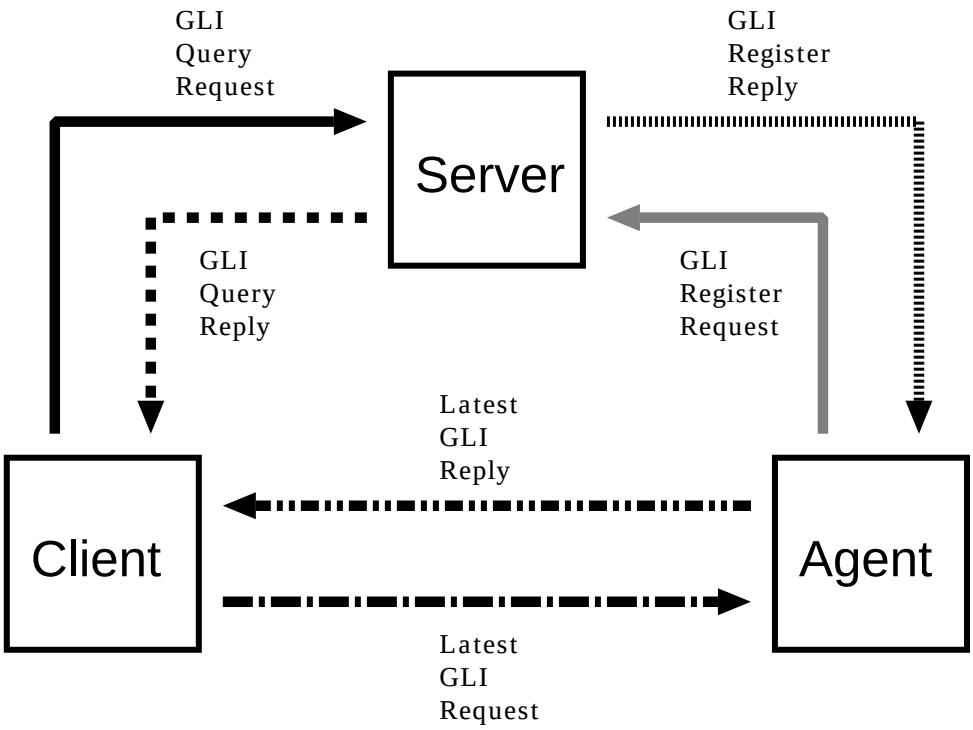

図 4.1: プロトタイプ・アーキテクチャ

Server

サーバはエージェントから送信された GLI をデータベースに管理し，クライアントからの問い合わせ要求を受け取り，その結果をそのクライアントに送信する．エージェントから送信されたデータはサーバのデータベース上に管理される．サーバ上のデータベース管理方法はエージェントからの連続的な要求，特に移動エンティティの最新情報を提供するための登録要求は短い間隔で送信されるので，高速に処理することが要求される．

サーバで管理されるそれぞれのエントリには 4 つのフィールドがある．第 1 のフィールドは IP アドレスで, 本プロトタイプではエンティティの識別してして使用される．第 2，3 のフィールドは GLI である．一つは最新の GLI，もう一つは前取得の GLI である．過去 2 段階の GLI の値を保持することで，エンティティの移動状況を把握することができる．二つの値の差分から移動の間の速度や進行方向を計算して求めることができる．エージェントが何らかの事由によって長い間隔で更新情報を送らなかった場合に，過去 2 段階の GLI から現在地を推測可能である．最後のフィールドは付加情報である．これは，エンティティの名前やタイプなどの情報を文字列で記述できる．

Client

クライアントはユーザと GLI System との間にインタフェースを提供する．クライアントはユーザからの問い合わせ要求 (WAY, WIT) をサーバに送信し，その返事をユーザへ返す．また，クライアントは特定のエンティティの最新の GLI をそのエンティティに直接アクセスすることによって求めることができる，

4.3 実装

ここでは GLI System のプロトタイプ実装について述べる．

4.3.1 ハードウェアアーキテクチャ

本稿でのプロトタイプ実装では，地理的位置情報を取得する装置として，a Global Positioning System(GPS) を使用している．使用した GPS はトリンブル社製の 6 チャネル GPS センサである．GPS に依存した部分は gli_update という名前のモジュールに分離して実装した．これは本プロトタイプでは装置に依存した唯一のモジュールである．このシステムは，IBM PC/AT 互換機用の BSD/OS 2.1，NEWS-OS4.2.1，SunOS4.1.4. に実装されている．

4.3.2　ソフトウェアアーキテクチャ

GLI System は TCP/IP を使用したアプリケーションプログラムとして実装されている．サーバとクライアント，サーバとエージェント，クライアントとエージェント，それぞれの間の通信は TCP (Transmission Control Protocol) を使用している．データフォーマットには XDR(eXternal Data Representation)[57] を使用している．

したがって，異なるマシンアーキテクチャが異なるバイトオーダを使用する場合にも問題は生じない．図 4.2に，プロトタイプ実装のソフトウェア構造を示す．

4.2節で述べた GLI System のエージェント，サーバ，クライアントは実装次のように実装される．エージェントは glid と gli_update という 2 つのモジュールからなる．サーバとクライアントはそれぞれ glis と glic というモジュールから実装される．

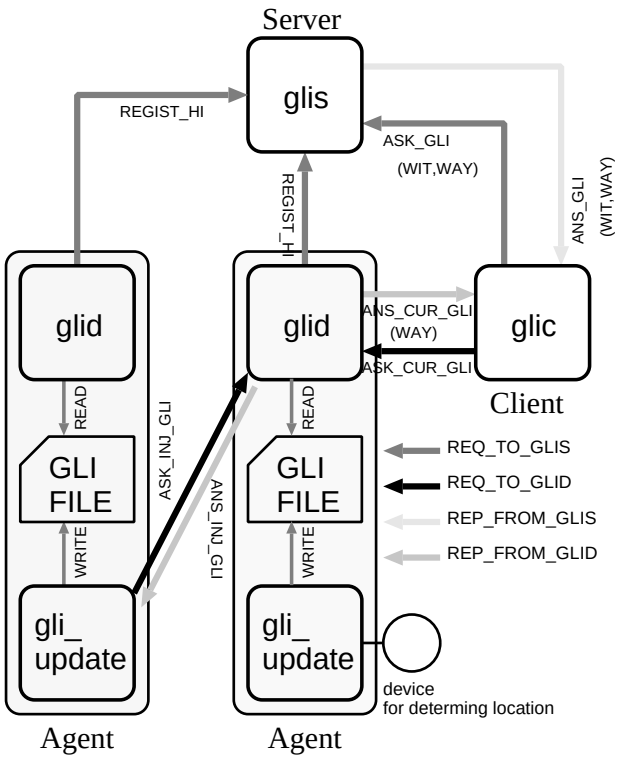

図 4.2: GLI ソフトウェアアーキテクチャ

図 4.2においてモジュールはそれぞれの名前 (gli_update, glid, glis, glic) がついた正方形で表されている．glid，gli_update，GLIFILE を囲む長方形はこれら 3 つが同じエンティティに存在することを示す．gli_update は GLI を GPS のような装置かまたは他のエンティティから取得する．gli_update は装置から情報を受け取り，それを GLI へ加工する．また，gli_update は注射エンティティ上の glid と通信する．gli_update は得られた GLI を glid によって参照されるファイル GLIFILE に書き込む．GLI を得る装置を持つエンティティで

は，gli_update は GLI を取得しそれを GLIFILE へ書き込む．glid は GLIFILE を読んで，GLI をサーバに登録する．glic は問い合わせ要求を glis と glid に送信する．

4.3.3 パケット処理

パケットヘッダ

図 4.3に GLI System でやりとりされる各パケット共通のヘッダを示す．

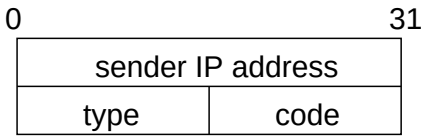

図 4.3: GLI パケット共通ヘッダ

それぞれのフィールドについて次に述べる．

- sender... このパケットを送信するエンティティの IP アドレス
- type... 表 4.2に示されるパケットのタイプ
- code... それぞれの type ごとに定義されたパケットのコード

表 4.2: パケットタイプ

Packet type	Contents
REQ_TO_GLID	Request to glid
REQ_TO_GLIS	Request to glis
REP_FROM_GLID	Reply from glid
REP_FROM_GLIS	Reply from glis

パケットコード

要求パケット (REQ_TO_GLIS) には二つの異なるコードを持つパケットがあり，glis へ送信される．これらには，glic からの WAY と WIT のような問い合わせ要求 (ASK_GLI) と glid からの登録更新要求 (REGIST_HI) がある．

glis からの返答パケットには一つのコード (REP_FROM_GLIS) がある．これは glic 向けの返答である (ANS_GLI).

要求パケット (REQ_TO_GLID) は glic かまたは別のエージェントの glid から glid へ送信される．これらは，glic から最新情報の検索要求 (ASK_CUR_GLI) と GPS を持つエージェントから GLI 注射させる要求 (ASK_INJ_GLI).

返答パケット (REP_FROM_GLID) は glid からの返答を送る際に使用される．これらには glic に送り返される最新 GLI(ANS_CUR_GLI) と gli_update によって注射される GLI (ANS_INJ_GLI) がある．

管理されるデータ

GLI の更新のために glid から glis へ送信されるパケットのデータ (type = REQ_TO_GLID, code = REGIST_HI) は，表 4.3に示されるデータから生成される．

表 4.3: glid で管理されるデータ構造

Server Address	4bytes
Latest Location Info	36bytes
Time of Fix	4bytes
Mobility Factor	2bytes
Additional Info	64bytes
Device	2bytes
Datum	2bytes
Surveying Location	2bytes
Total	116 bytes

追加，更新，削除などのデータベースの機能は，flags に指定される．

パケットによって作られるデータのエントリのタイプを表 4.4に示す．それは glis 上のデータベースで管理される．

データエントリは単純な線形リストで実装されている．エントリごとに 148bytes，リストにエントリは 64 個ある．このリストの検索には線形探索が使用される．より高速なアルゴリズムや手法が最適化のために将来導入される．

glic はユーザと glis のインタフェースを提供する．特定のエンティティ位置の検索要求の場合，glic は対応するエンティティの IP アドレスを鍵とした要求パケットを glis に送信する．特定のエンティティの最新 GLI の問い合わせ要求の場合，glic はそのエンティティに要求パケットを送信する．特定の領域の最も近いエンティティの検索要求の場合，glic は特定の位置を鍵として要求パケットを指定した glis へ送信する．文字列の検索要求の場合，glic は文字列を鍵として要求パケットを送信する．

表 4.4: glis で管理されるデータ構造

Agent Address	4bytes
Latest Location Info	36bytes
Time of fix	4bytes
Previous fix location info	36bytes
Time of previous fix	4bytes
Additional Info	64bytes
Device	2bytes
Datum	2bytes
Surveying Location	2bytes
Total	148bytes

結果として，glic は glis または glid からの返信を受け取り，受け取った情報を表示する．

glic から glis への要求パケット (ASK_GLI, ASK_CUR_GLI) は，flags を持つ．これらにはエンティティの識別子に基づく地理的位置の検索要求 (WAY) と地理的位置に基づくエンティティの識別子の検索要求 (WIT) がある．

4.4 実験環境と評価

図 4.4に実験環境を示す．この実験はイーサネットからなるネットワークと 800Mhz 帯の携帯電話の回線を利用し，一つのサーバとクライアントとエージェントが動作するいくつかのエンティティ(その一つは移動している) で行なわれた．san-marino と shark という名前のエンティティはイーサネットに固定されている．docile という名前のエンティティは車と共に移動しており，携帯電話の回線でモデムを用いて shark に接続されている．

図 4.5に shark 上の glic がエンティティdocile の GLI を san-marino 上の glis に問い合わせ要求した結果を示す．

図 4.6に shark の glic が docile の最新 GLI を docile に問い合わせた結果を示す．

図 4.7に docile の glic が指定した位置に近いエンティティの検索を san-marino の glis に要求した結果を示す．

以上のように，我々のプロトタイプは効果的に動作しており，固定および移動ホストの GLI を検索することができている．

また，この最初の実装から，大規模な環境でのサーバ位置と GLI System 内での情報の機密性維持の重要性が認識された．

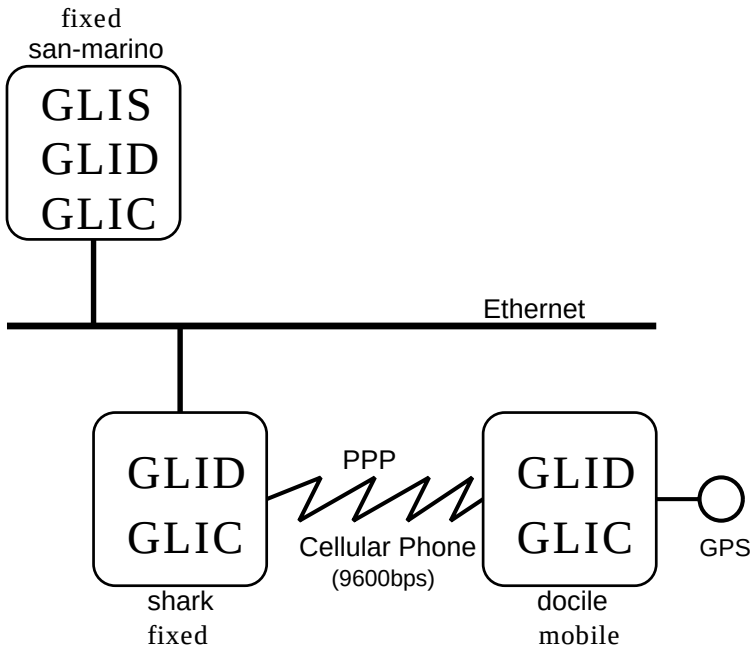

図 4.4: 実験環境

```
% glic -s san-marino -t docile
[docile]
Latest Info:
Position:  35 18.321 N  139 30.652 E     29.74 m
Vel  ENU:        -11.30       1.63      -1.67
Time of fix: Thu 03:47:03.44
Prev fixed Info:
Position:  35 18.316 N  139 30.677 E     33.51 m
Vel  ENU:        -10.38       3.09      -0.79
Time of fix: Thu 03:46:59.94
```

図 4.5: shark の glic が san-marino の glis に docile の検索要求を行なう

```
% glic -t docile
[docile]
Latest Info:
Position:  35 18.338 N  139 30.490 E      9.38 m
Vel  ENU:         -11.83       1.00       -0.95
Time of fix: Thu 03:47:22.94
```

図 4.6: shark の glic から docile への最新 GLI 問い合わせ要求を行なう

```
% glic -s san-marino -l 35.38 -o 139.42
[san-marino]
Latest Info:
Position:  35 23 19.432 N  139 25 39.166 E     39.59 m
Vel  ENU:       0.00       0.00       0.00
Time of fix: Sat 07:57:38.31
Prev fixed Info:
Position:  35 23 19.432 N  139 25 39.166 E     39.59 m
Vel  ENU:       0.00       0.00       0.00
Time of fix: Sat 07:57:38.31
```

図 4.7: docile の glic から san-marino の glis へ指定した位置に近いエンティティの検索を要求する.

4.5　関連研究

インターネットの実用的な応用の研究に関心が集まりつつある．いくつかの応用，それらはモーバイル・コンピューティング [58] やユービキタス・コンピューティング [59, 60] といった研究分野の発展の結果であるが，現実世界のエンティティとコンピュータ・ネットワーク上のユーザを疑似的に実現，あるいは対応付けることを目指している．そのようなアプリケーションには現実世界エンティティの位置情報が必要となる．

ユービキタス・コンピューティングは Mark Weiser によって提唱された概念であり，この概念ではコンピュータは至る所に存在し，かつユーザには見えずに存在する．ユーザはその存在を意識せずに使用することができる．ユービキタス・コンピューティングの環境では，コンピュータは逆にユーザの位置を認識する必要がある．この課題を取り扱っている研究は Active Badge Location System[61] である．Active Badge system は，建物内の個人の位置を身につけた Active Badge の位置から得る手段を提供する．この小さな装置は 10 秒ごとに独自の赤外線信号を転送する．オフィスはそれぞれこれらの信号を検知するネットワークに接続されたセンサが備え付けられている．よって身に付けられたバッジの位置はこれらのセンサからの情報で認識できる．

Active Badge に似たシステムを用いるアプリケーションや研究もある．[62]-[63]. その一つは Context-Aware Computing Applications[62] である．これは，Active Badge のような赤外線のインタフェースを持つ Parc Tab という小型のコンピュータを用いる．Parc Tab のユーザは位置を変えたり，人間のグループに入ったり抜けたり，他のコンピュータとやりとりしたりする．Context-Aware Computing のソフトウェアは，その位置，属しているホストや人間のグループ，そばにあるアクセス可能な装置などの状況に適応する．

位置情報を取得する最もポピュラーな装置としては Global Positioning System (GPS) がある．GPS は地球上の全てのエリア，海上，陸上，飛行機，ロケット，船上などで連続測位可能である．また，移動するエンティティの速度や進行方向や極めて正しい時刻を得ることもできる．様々な分野で多くの応用で使用されている．

しかしながら，これらの新しい概念や装置の出現があっても解決すべき問題は多くある．以下にそれを列挙する．

- Active Badge location system は主に個人向けであり，個人とその位置を対応付けている．その他のエンティティは固定されているとみなしている．よってこのシステムでは現実世界の全ての種類のエンティティを同レベルで取り扱わない．

- Active Badge や Parc Tab は建物などのような限定された領域で使用されるので，それが提供する位置情報はローカルで，グローバルな領域には対応できない．

- GPS は広域な環境で使用可能である．しかしながら，位置情報だけを提供し，通信の機能は貧弱である．分散環境においては GPS を持つネットワークに接続されたコンピュータが位置情報を処理することが必要である

この分野のその他の研究としては個人の位置情報のプライバシの保護やセキュリティの問題がある．[64, 65].

4.6 結論

我々は，エンティティの地理的位置情報とインターネット上の識別子の対応付けを提供する Geographical Location Information (GLI) System を提案した．プロトタイプではインターネット上の識別子として IP アドレスを使用している．本システムにより地理的な位置に基づいてエンティティの識別子を検索したり，逆にエンティティの識別子に基づいて地理的な位置を検索することが可能となっている．

また，GLI System のプロトタイプを設計し，TCP を使用したアプリケーション層のプログラムとして実装し，実験によってその有用性を示した．GLI System はマシンアーキテクチャに非依存であり，BSD/OS, SunOS, NEWS-OS でテストされている．この GLI System を多くのホストを使用してインターネット上で実験している．

現在我々はプロトタイプの GLI System を使用してそのパフォーマンスの改善と本章で提案した GLI の階層的表現構造の導入のために実験を継続している．また，このシステムのためのアプリケーションプログラムも開発中である．