修 士 論 文

移動計算機環境におけるネットワークアーキテクチャに関する研究

指導教官　中島 達夫 助教授

北陸先端科学技術大学院大学
情報科学研究科

小林 勝

1998 年 2 月 13 日

目 次

図 目 次

表 目 次

第 1 章

序論

無線を中心とするネットワーク技術の進歩によりネットワークへ接続可能な領域が大幅に広がった。また小型化により計算機を携帯することが可能になり、研究室などの屋内だけでなく野外を含む移動先でも計算機を利用する事が可能になった。この計算機は動作中に抜き差しが可能な PC カードを利用する事が可能であり、PC カードで実現された通信メディアを用いる事で計算機の動作中に通信メディアを切り替える事が出来る。このような携帯可能な計算機とネットワーク技術を用いる事により、いつどこでもネットワーク上のサービスや他の計算機を含む計算機資源を利用する事が可能になっている。

例えば、屋内の研究室などではイーサネットや FDDI、ATM などの高速な有線のネットワークを使用して、会議などで別の部屋へ移動する時には無線 LAN を使用する。さらに屋外では PHS や携帯電話を使ってネットワークに接続する、などの使い方が考えられる。このように近年の移動計算機環境は各種の通信メディアを駆使する事によりいつどこでもネットワークに接続する事ができる。

このような移動計算機環境ではシステムは計算機の場所やユーザの要求によって計算機の動作中に通信メディアを切り替えなければならない場合や、複数の通信メディアから最適なものを選択しなければならない場合がある。また、移動計算機向けに開発された無線 LAN や PIAFS はこれまでの通信メディアとは異なる特性を持っている。移動計算機環境のネットワークではこれまでのネットワークシステムでは考慮されていない問題が存在する。

現在のネットワークシステムでは IP アドレスは通信メディアに与えられる。そのため通信メディアの変更は計算機の動作中に IP アドレスが変化する事を意味する。また、移動計算機の利点である計算機の移動も IP アドレスの変化を引き起こす。ネットワーク上のサービスやアプリケーションで使用される TCP/UDP は通信合いての識別に IP アドレスを用いている。そのため IP ア

ドレスが変化すると通信を続けることが出来ない。

このような問題に対処するためにスタンフォード大学の MosquitoNet[1] などでは移動計算機環境のためのネットワークプロトコルやシステムソフトウェアの研究が進められている。現在ネットワーク上での移動を可能とするプロトコルが開発されており、そのひとつに IETF Mobile IP[2] がある。IETF Mobile IP ではネットワーク上にホームエージェントやフォーリンエージェントと呼ばれるサーバを配置する事で計算機の移動を可能としている。移動する計算機の IP アドレスはネットワークへの接続点ではなく計算機の識別子として扱っている。これによりネットワークへの接続場所に関わりなく計算機の識別を可能にしている。しかし、現在のシステムソフトウェアでは計算機を識別するための識別子とネットワークへの接続点を識別するための識別子が分離されていない。また、IETF Mobile IP の仕様にもいくつかの問題点が残されている。

本研究ではこの問題点に対処するために動的に通信メディアを選択する機構を構築する。通信メディア選択機構はプロトコルスタックと通信メディアの間に配置される。このシステムの特徴は計算機の識別子とネットワーク接続点の識別子をシステムソフトウェア上で明確に分離した事である。これまで別々に扱われていた通信メディアを統一的に管理し、プロトコルスタックに対して単一の仮想的なネットワークインターフェースを提供する。仮想インターフェースの下位で通信メディアを扱うことでこれまでのシステムでは扱いにくかった無線 LAN や PIAFS を扱うことを可能にする。また IETF Mobile IP や環境サーバ [9] とともに動作し、IETF Mobile IP の仕様上の問題点を解決する。

通信メディア選択機構は以下の機能を持つ。

通信メディアの仮想化

イーサネットや PPP といった特性の異なる通信メディアの仮想化を行い、メディア間の差異を吸収する。これにより通信メディアの動的な切り替えをプロトコルスタックとは独立して行い、通信メディアの変化を上位層から隠蔽する。

計算機の識別子とネットワーク接続点の識別子の分離

仮想ネットワークインターフェースに IETF Mobile IP のホームアドレスを、管理下の通信メディアに接続先の IP アドレスを与える事で、計算機識別子とネットワーク接続点の識別子を分離する。

IETF Mobile IP フォーリンエージェントのエミュレーション

IETF Mobile IP のフォーリンエージェントの機能を取り込む事で、IP で使用可能なすべての通信メディアで IETF Mobile IP を使用可能にする。

本研究ではこのシステムを RT-Mach(Real-Time Mach)[6] 上に構築した。通信メディア選択機構は以下のコンポーネントから構成される。

プロトコルインターフェース

- プロトコルスタックにたいして仮想ネットワークインターフェースを提供する。

デバイスインターフェース

- イーサネットや PPP などの通信メディアを仮想化する。
- IP アドレスの確保や回線の接続を行う。
- ネットワークとの接続状態を監視する。

スイッチ部

- プロトコルインターフェースとデバイスインターフェースを結び付ける。
- ユーザの要求やネットワークへの接続状態に応じて最適なデバイスインターフェースを選択する。

通信メディア選択機構では、従来のシステムでは直接結び付いていたプロトコルスタックと通信メディアはプロトコルインターフェースとデバイスインターフェースに分離されている。計算機の識別子としての IP アドレスはプロトコルインターフェースに割り当てられ、ネットワークへの接続点としての IP アドレスはデバイスインターフェースに割り当てられる。スイッチ部は状況に応じて最適なデバイスインターフェースを選択しプロトコルインターフェースと結び付ける。状況が変化して最適な通信メディアが変化した場合は、スイッチ部が動的にデバイスインターフェースを切り替える事で通信メディアを切り替える。この動作はプロトコルスタックから隠蔽されている。

以上の機能により通信メディアの変化や計算機の移動に対して透過なネットワーク環境を実現する。

本稿では、まず移動計算機環境でのネットワークの特徴とその問題点について検討する。次に問題点の解決方法を検討し、本研究で構築する通信メディア選択機構につい述べる。さらに、通信メディア選択機構をもとにシステムの設計や実装、多様な通信メディアを扱う問題点、移動計算機環境のネットワークについて検討する。

第 2 章

移動計算機環境のネットワークとその問題点

2.1 多様な通信メディア

移動計算機環境で利用可能な通信メディアは多様である。イーサネットや FDDI のような有線のメディアは高速だが屋内の特定の場所でしか利用できない。無線 LAN は有線のメディアに比べれば低速であるが、移動が可能である。しかし接続可能な範囲は基地局から数 100m の距離内に限られる。PHS を利用する PIAFS は都市部で、携帯電話はより広範囲で利用可能だが、速度は大幅に低下する。このようにいつどこでも利用可能な通信メディアは存在するが、一つのメディアですべての要求に対処できるわけではない。したがってネットワークに接続したい場所によって通信メディアを選択する必要がある。

現在の移動計算機は動的に抜き差しが可能な PC カードを利用する事ができる。移動計算機向けの通信メディアはイーサネットなどの有線のメディアでも無線 LAN や PIAFS などの無線のメディアでもこの PC カードにより実現されている。これにより計算機の動作中でも必要に応じて通信メディアを選択することが可能になっている。たとえば、研究室でイーサネットのような有線のメディアを使用している時に会議などで別の部屋に移動しなければならなくなった場合は、イーサネットの PC カードを抜いて無線 LAN の PC カードを挿入すれば通信メディアをイーサネットから無線 LAN に切り替える事ができる。さらに無線 LAN の利用可能な範囲から出てしまった場合は、無線 LAN の PC カードを PIAFS の PC カードや携帯電話用の PC カードに取り替えればネットワークへの接続を維持する事ができる。

2.2 移動計算機環境の問題点

移動計算機環境では PC カードと各種通信メディアを組み合わせる事でハードウェア的にはいつどこでもネットワークに接続する事が可能になっている。しかし、同時にこれまでになかった問題が発生している。

2.2.1 IP アドレスの変化

移動計算機環境では移動計算機の IP アドレスが計算機の動作中に頻繁に変化する。これには二つの要因がある。

計算機の移動

TCP/IP の IP アドレスはネットワークを識別するネットワーク識別子と計算機を識別するホスト識別子の組みからなる。計算機が移動して接続するネットワークが変わると移動前の IP アドレスは使用できなくなり、接続先のネットワークで新しい IP アドレスを割り当てなければならない。しかし、ネットワーク上のサービスやアプリケーションが通信に使用する TCP/UDP は通信相手の識別に IP アドレスを使用するので、計算機が移動して IP アドレスが変化するとアプリケーションは通信を続けることができなくなる。

通信メディアの切替

TCP/IP ではネットワークアドレスは計算機にではなく通信メディアに対して与えられる。このため通信メディアの変更は IP アドレスの変更を意味する。通信メディア変更後も同じネットワークに接続する場合でも計算機側ではどのネットワークに接続するかは確定できないので、ユーザが IP アドレスを指定するか DHCP(Dynamic Host Configuration Protocol)[5] で外部から割り当てる必要がある。

これまでのシステムでは IP アドレスは計算機を識別するための識別子というよりはネットワークから見た計算機との接続点となっている。したがって通信メディアの変更はネットワークへの接続点の変更であり、計算機の移動と同じ問題が発生する。

2.2.2 通信メディアの管理

移動計算機環境では計算機の動作中に使用可能な通信メディアが動的に変化する。特に PC カードを交換する場合には通信メディア自体が存在しなくなったり新しく出現することになる。その

ため計算機の動作中に通信メディアの初期化などの処理が必要になる。

また移動計算機環境で用いられる通信メディアにはイーサネットなどとはことなる特性を持つものがある。PHS などを利用した PPP 接続の場合は回線の接続を明示的に行なわなければならない。無線 LAN の一種である Netwave は基地局に異なるチャネルを割り当てることで基地局ごとにことなるサブネットを割り当てることができる。そのため Netwave でネットワークに接続するためには移動計算機側で基地局の選択を行なわなければならない。

これらは計算機が動作中に発生する。従来の環境は比較的静的で動作中に通信メディアの構成が変化することはなかった。そのため、通信メディアの制御はシステム起動時の初期化程度ですんでいた。しかし移動計算機環境は動的であり、リンク層レベルの制御を積極的に行なわなければならない。

一方、ネットワークに接続するためにはリンク層での制御の他にネットワーク層でのアドレス付や経路制御も必要である。アドレスや経路は使用する通信メディアに依存するので通信メディアの変化の影響を受ける。しかしこれまでは通信メディアの制御とネットワーク層の制御は独立して行なわれていた。例えばネットワークアドレスの確保や経路の切替えは DHCP を用いて動的に行なうことができるが、通信メディアの制御とは独立にユーザによって行なわれる場合が多かった。そのためユーザが移動や通信メディアの変化に対処する必要があり、自由に移動できるとは言えない。

以上のように移動計算機環境では複数の通信メディアをネットワークアドレスや経路といったネットワーク層も含めて制御することが必要とされている。

2.3 まとめ

移動計算機環境の特徴である通信メディアを使い分けて移動しながらネットワークに接続する事は、使用する通信メディアやネットワークとの接続位置が移動先やその時の環境によって常に変化する事を意味する。これは通信メディアやネットワークの接続位置が変化しないというこれまでの計算機環境とは大きく異なる。そのため、従来のネットワークプロトコルやオペレーティングシステムのネットワークサポートでは移動計算機環境に対応し切れない部分が生じている。

第 3 章

関連研究

3.1 IETF Mobile IP

計算機の移動に伴う問題点に対処するためのプロトコルとして IETF Mobile IP や VIP[3] がある。ここでは IETF Mobile IP の概要について述べる。

3.1.1 IETF Mobile IP の概要

IETF Mobile IP では移動する計算機は移動ホストと呼ばれ、ホームネットワークとホームネットワーク上の IP アドレスであるホームアドレスを持つ。ホームネットワークにはホームエージェントがあり、移動ホストのホームアドレスを管理している。移動ホストはホームネットワークから他のネットワークへ移動する事ができる。移動ホストが移動したネットワークにはフォーリンエージェントが配置されている。(図 3.1)

IETF Mobile IP の基本的な動作は他のネットワークに移動した移動ホスト宛の IP パケットをホームエージェントが移動ホストへ転送することである。移動ホストはホームネットワークから他のネットワークに移動すると、移動先のネットワーク上で気付アドレス (care of address) を取得する。気付アドレスの取得方法は移動先のネットワーク上のフォーリンエージェントから割り当ててもらう場合と、移動ホスト自身が DHCP などを用いて移動先のネットワーク上の IP アドレスを確保しそれを気付アドレスとする場合がある。

気付アドレスを確保した移動ホストは気付アドレスをホームエージェントに登録する。ホームエージェントはホームネットワークに送られてきたホームアドレス宛の IP パケットを移動ホストの替わりに受け取り、IP in IP のカプセル化をおこなって移動ホストが登録した気付アドレスに転送する。気付アドレスがフォーリンエージェントの場合はフォーリンエージェントがカプセル化さ

れた移動ホスト宛の IP パケットを取り出し、移動ホストに転送する。気付アドレスが移動ホストが確保したアドレスの場合は移動ホストが自分でカプセル化された IP パケットを取り出す。このような手順でホームアドレス宛の IP パケットを転送する事により移動ホストは移動先でも自分宛の IP パケットを受け取る事ができる。移動ホストが送信する IP パケットのソースアドレスは移動先でもホームアドレスを用いる。これにより移動ホストの通信相手は常にホームアドレス宛に IP パケットを送るので、移動ホストはどのネットワークに移動しても通信を続ける事ができる。

このように IETF Mobile IP ではホームアドレスはネットワークインターフェースに割り付けられたアドレスではなく移動ホストの識別子として働く。ネットワークへの接続点としての IP アドレスは気付アドレスとして移動したネットワーク上でその都度確保される。

3.1.2 IETF Mobile IP の問題点

IETF Mobile IP ではホームエージェントが移動ホストへ IP パケットを転送する事で計算機の移動に伴うネットワークアドレスの変化に対応している。しかし、その仕様には以下に示すいくつかの問題が残されている。

フォーリンエージェントから移動ホストへの転送

移動ホストの移動先のネットワークで、フォーリンエージェントから移動ホストへ転送される IP パケットの宛て先アドレスはホームアドレスとなっている。この場合ホームアドレスは別のネットワーク上のアドレスなので、通常の IP パケットのルーティングを用いる事ができない。このためフォーリンエージェントはリンク層を直接制御して移動ホストにパケットを転送しなければならない。リンク層がイーサネットの場合、フォーリンエージェントは移動ホストの MAC アドレスを指定してパケットを転送する。

移動ホストが PPP で接続する場合、移動ホストは PPP サーバを介してフォーリンエージェントの配置されたネットワークに接続される。そのためフォーリンエージェントがホームアドレス宛のパケットを転送する先は PPP サーバになる。したがって PPP サーバも IETF Mobile IP を認識しなければならない。このようにフォーリンエージェントから移動ホストへの IP パケットの転送はリンク層の制御が必要になり、移動ホストが使用する通信メディアごとに IETF Mobile IP に対応する必要がある。

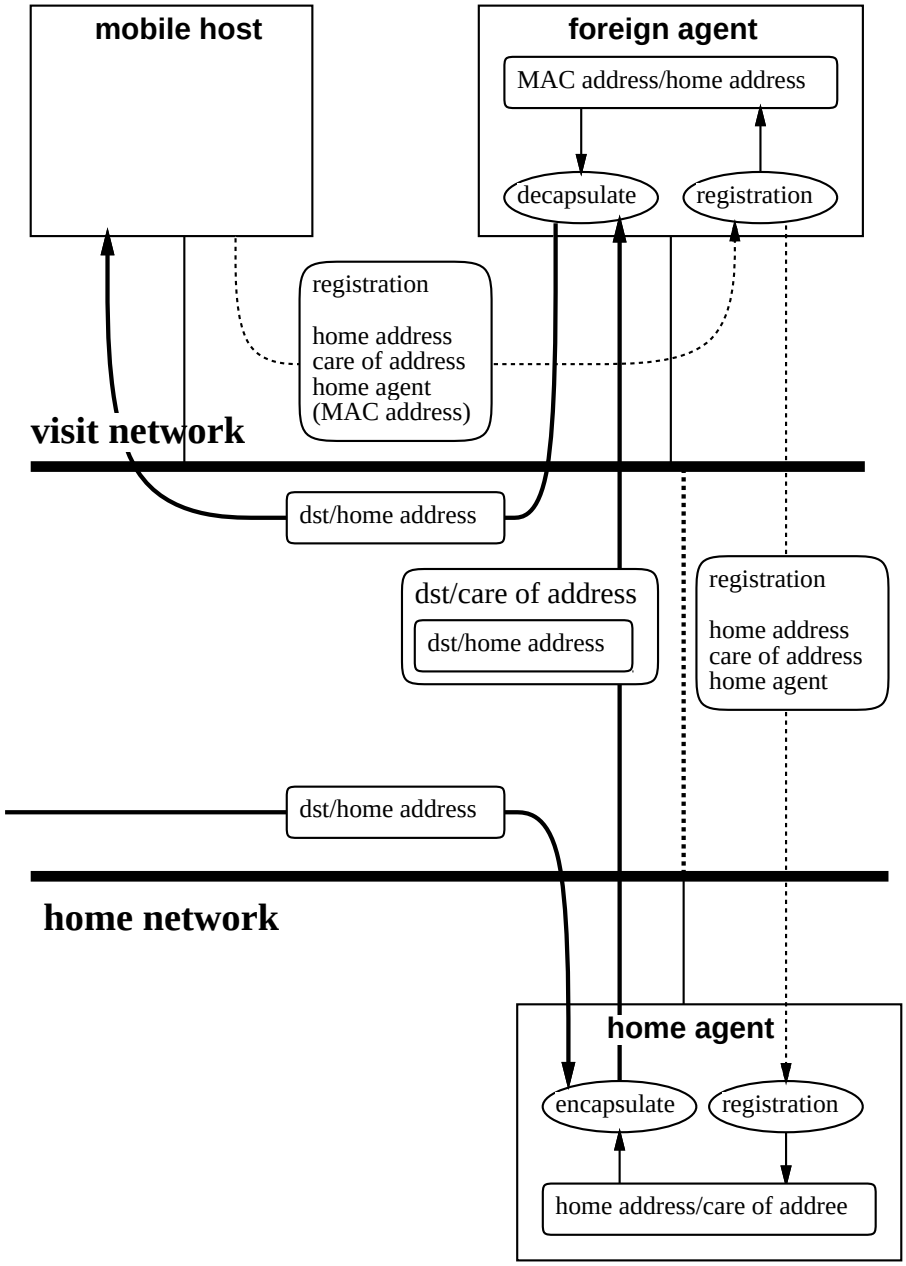

図 3.1: IETF Mobile IP の動作

ホームアドレスと気付アドレス

気付アドレスを移動ホスト自身が確保する場合、ホームエージェントから転送されてくるカプセル化 IP パケットを受け取るために、移動ホストの通信メディアは気付アドレスで動作しなければならない。一方移動計算機から通信相手のホストへ送信される IP パケットのソースアドレスには気付アドレスではなくホームアドレスを使用しなければならない。そのため、ネットワークに接続された通信メディアはネットワークからの ARP などに対しては気付アドレスを持ったホストとしてで動作し、送信される IP パケットにはホームアドレスを持ったホストとして動作するという変則的な動作が必要になる。

3.2 MosquitoNet

スタンフォード大学の MosquitoNet では状況に応じて複数の通信メディアを使い分ける事が出来る Mobile IP を開発している。

3.2.1 MosquitoNet での MobileIP

MosquitoNet ではプロトコルとしては IETF Mobile IP をベースとしているが、フォーリンエージェントを用いていない。必要とされるサーバはホームエージェントのみで、気付アドレスは移動ホストのネットワークへの接続点となる。

MosquitoNet は拡張されたルート検索関数 (ip_rt_route() 関数)、ルーティングのための移動ポリシーテーブル (Mobile Policy Table)、IP in IP のカプセル化を行なうための仮想ネットワークインターフェース (VIF) から構成される。(図 3.2) 上位層からパケットを渡された IP 層はパケットを送るためのインターフェースを決定するために ip_rt_route() を呼び出す。拡張された ip_rt_route() は移動ポリシーテーブルを調べ、ソースアドレスをホームアドレスとする必要はあるか、カプセル化を行なう必要があるかといった Mobile IP の処理が必要かどうかを決定する。必要がなければ IP パケットは通常の手順でリンク層に出力される。ソースアドレスをホームアドレスにする場合はソースアドレスにホームアドレスを着けてリンク層に出力する。カプセル化が必要な場合は仮想インターフェース VIF に出力する。MosquitoNet ではカプセル化のためのプロトコル IPIP がトランスポート層に追加されており、VIF は渡されたパケットを IPIP に送る。IPIP はパケットをカプセル化し IP 層に渡す。IP 層は再び同じ様な処理を行なうが、このカプセル化されたパケットはこれ以上 Mobile IP としての処理は必要ないのでそのままリンク層に出力される。

MosuquitoNet の特徴は使用する通信メディアの選択やカプセル化を行なうかどうかといった判

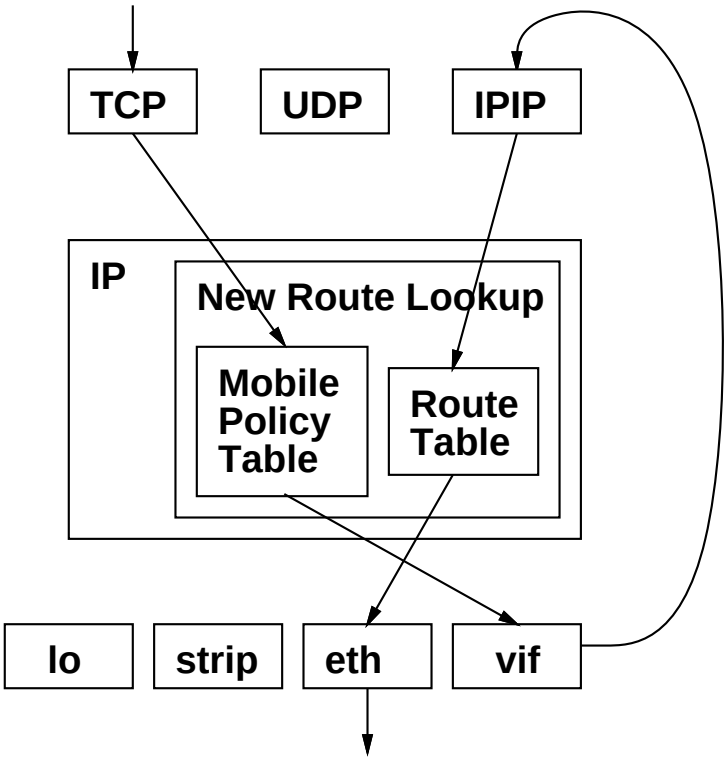

図 3.2: MosqutoNet の構成

断をルーティングの問題として扱っている事である。

3.2.2 MosquitoNet の問題点

MosquitoNet では IP 層のルーティングに Mobile IP のためのポリシーを追加する事で既存のシステムに Mobile IP を追加している。トンネリングの機能はカプセル化のためのトランスポートプロトコルとリンク層に仮想インターフェースを配置する事で実現しており、フォーリンエージェントの機能を取り込んでいる。しかし通信メディアの管理は既存のシステムの範囲内で行なっており、通信メディアの特性はあまり考慮されていない。移動計算機環境の通信メディアには PHS のように回線の接続/切断という動作が必要で、通常は切断状態の物がある。MosquitoNet でこのような通信メディアを使用するためには、外部から回線を接続してルーティングやポリシーを更新してやらなくてはならない。

MosquitoNet が扱う範囲はネットワークプロトコルに限定されており、回線の接続や使用するメディアの選択などは上位の処理としているようである。しかし通信メディアの制御や選択は Mobile IP の動作と密接に関係しているので、協調して動作する必要があると思われる。また通信メディアの扱いは従来のシステムと同じであるが、無線 LAN や PHS などの新しいメディアは従来のネッ

トワークインターフェースの抽象化では扱い切れない部分があると思われる。

3.3 PMI

動的に通信メディアを管理するシステムとしてOGI(Oregon Graduate Insutitute) のPMI(Physical Media Independence)[4] がある。PMIでは通信メディアの使用可能性を定義するモデル、通信メディアの変化によるネットワークシステムの再構成を行なうための手続きが定義されている。

通信メディアは以下の特性が与えられている。

- **物理的な接続**
- **リンクレベルの接続**
- **ネットワークアドレス**
- **電源**
- **コスト**
- **ユーザの選択**

これらの条件が満たされると通信メディアは使用可能となる。

上記の特性を与えられた通信メディアはステートマシンとして記述される。ステートマシンは通信メディア毎のに定義することができる。PCカードの抜き差しやリンクレベルの接続性などが外部イベントとして定義されている。外部イベントに対して内部イベントが定義されている。

通信メディアの特性の変化を検出するために特性毎にガードと呼ばれるモジュールが用意されている。ガードは通信メディアの特性を検出するモジュールに配置され、特性が変化するとステートマシンに状態の遷移を起こさせる。ガードは物理的な接続ではオペレーティングシステムのPCMCIAモジュール、リンクレベルの接続性は通信メディアのデバイスドライバというように各特性を検出できるシステムレベルのモジュールに配置される。

このシステムはネットワークシステムを適応させるためレイヤー間の通信モデルを持っている。ネットワークシステムの各レイヤーには適応のためのモジュールが与えられている。上位レイヤーのモジュールからは適合のためのポリシーに関する情報 (メタデータ) があたえられ、下位レイヤーからはイベントの情報が通知される。通信メディアの使用可能性の変化はリンク層に結びつけられており、この情報が上位レイヤーに伝えられる。各レイヤーは伝えられた情報を元に各レイヤーの適応を行なうことでネットワークシステムの適応が行なわれる。

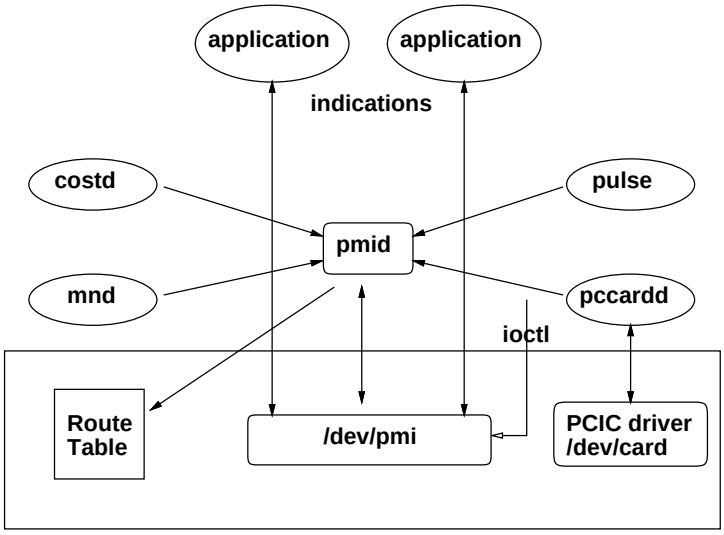

図 3.3: PMI の構成

システムは図 3.3のように構成されている。ガードやレイヤーの適応モジュールは既存のモジュールの変更を最小限にするためすべてが対象のレイヤーに配置されてはおらず、デーモンとして実現されているものもある。主要なモジュールには以下のものがある。

- **pmid**
  各通信メディアの使用可能性ステートマシンの実装、ガードからのイベントやメタデータの配送をおこなう
- **pulse**
  通信メディアのリンクレベルの接続性を監視するデーモン
- **pccardd**
  PC カードサポートのためのデーモン
- **costd**
  コストモデルをもとに各通信メディアのコストを監視する
- **nmd**
  Mobile IP のデーモン

PMI では pccardd や pulse などのデーモンに配置されたガードからの情報を pmid で配送することで複数の通信メディアの管理や使い分け、それにともなうネットワークシステムの適応を可能としている。ネットワーク上での移動はネットワークシステムの適応の一部として Mobile IP

により実現されている。全体として既存のシステムの変更を最小限にするために各種デーモンの集合体という構成になっており、ネットワークシステムの適応も通信メディアの変化に応じてネットワークアドレスや経路の再構成というような範囲に留まっている。

3.4 まとめ

移動計算機環境の問題点に対処するために移動を可能にするプロトコルの研究やシステムソフトウェアの拡張が行われてきた。しかしこれらは問題点の一部に焦点をあてており、全体としては移動透過とはなっていない。移動計算機環境がシームレスな移動を可能とするにはネットワークプロトコルやシステムソフトウェアを統合したシステムが必要である。

第 4 章

通信メディア選択機構の設計と実装

4.1 問題点の検討

移動計算機環境のネットワークでは計算機がネットワーク上を移動してもその計算機を識別するために計算機の識別子とネットワークへの接続点としての識別子の分離が必要になる。ネットワークプロトコルは IETF Mobile IP などのホストの移動に対応した拡張が行われ、計算機の識別子とネットワークへの接続点としての識別子の分離が行われている。

一方、現在のオペレーティングシステムのネットワークコードは計算機は移動せず、通信メディアなどのハードウェア構成が動的に変化する事は無いという前提で設計されている。ネットワークへの接続点である通信メディアは動作中に取り替えられる事はないので、IP アドレスは通信メディアに直接割り当てられる。IP アドレスはネットワークへの接続点であり、ホストの識別子とはなっていない。IETF Mobile IP の残された問題点もネットワークコードがホストの識別子とネットワークへの接続点の識別子を分離していない事に起因すると考えられる。

4.2 プロトコルスタックと通信メディアの分離

現在のネットワークコードでホストの識別子とネットワークへの接続点としての識別子の分離することは、一つの通信メディアに複数の IP アドレスを割り当てる事になる。これはプロキシ ARP によってネットワークに対しては通信メディアに割り当てられた IP アドレスのほかに別の IP アドレスとしても見せたり、トンネルデバイスを用いて出力される IP パケットのソースアドレスを書き換えるなどの方法で対処する事ができる。これらの方法は現在の枠内での解決方法ではあるが、ネットワークコードが想定している使い方ではない。

問題点はプロトコルと通信メディアが密に結び付けられているため IP アドレスが通信メディア

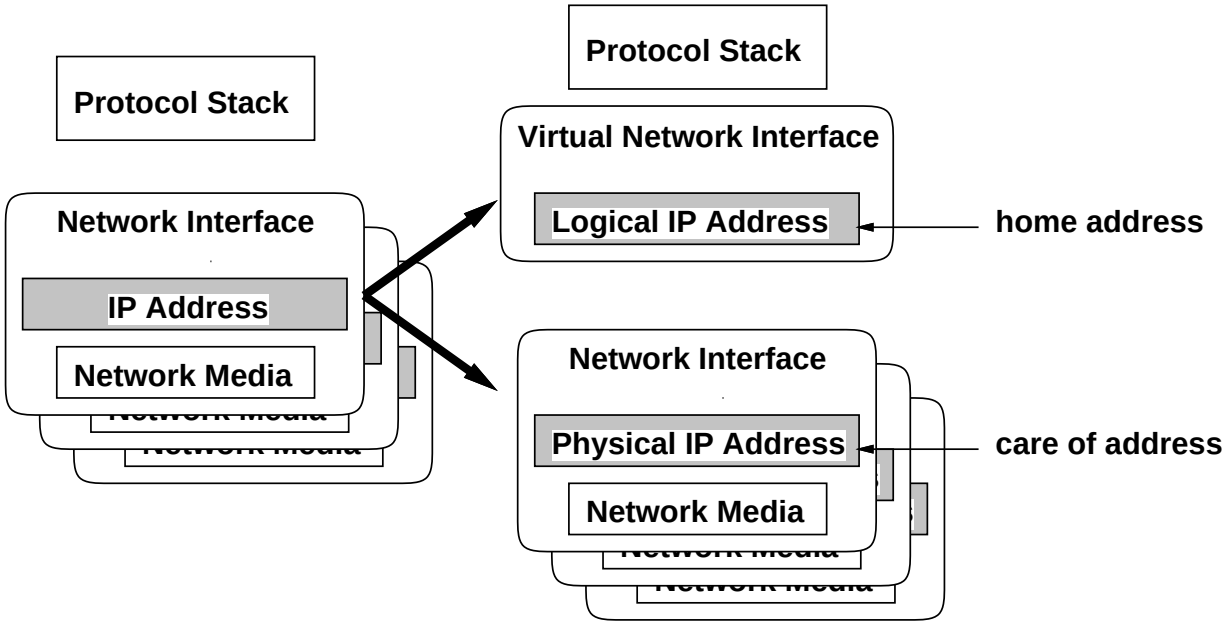

図 4.1: ホストの識別子とネットワーク接続点の識別子の分離

に直接割り当てられてしまい、ホストの識別子を持つレベルが無い事である。そこでプロトコルと通信メディアを明確に分離し、プロトコルの下位に通信メディアを管理する層を配置する。プロトコル側はホストの識別子としての IP アドレス、通信メディア側にはネットワークへの接続点としての IP アドレスを与える。具体的にはプロトコルに IETF Mobile IP を用いる事を前提として、プロトコル側の IP アドレスは IETF Mobile IP のホームアドレスを与え、通信メディア側の IP アドレスは接続先で DHCP などでその都度割り当てる。これによりホストの識別子とネットワーク接続点の識別子を分離する。(図 4.1)

通信メディアを管理する層は複数の通信メディアを管理し、状況やユーザの要求によって最適な通信メディアを選択してネットワークに接続する。管理する通信メディアにはイーサネットや PPP などがある。ネットワークへの接続に必要な IP アドレスの確保や ARP などの処理はこの層で行なう。各通信メディアの違いはこの通信メディア選択機構で吸収し、プロトコルからは単一のネットワークインターフェースとして仮想化する。プロトコルはこの通信メディア選択機構にホスト識別子としてホームアドレスを与えたネットワークインターフェースとして使用する。

通信メディア選択機構はプロトコルに対してフォーリンエージェントとしても動作する。気付アドレスに通信メディア選択機構が確保した IP アドレスを用いる事でホームエージェントから移動ホストへの IP パケットの転送を IP を用いて行なう事ができる。ホームエージェントから転送されたカプセル化されたパケットは、通信メディア選択機構がカプセルから取り出して上位のプ

ロトコルに渡す。これにより、プロトコルは変則的な動作を行なう必要がなくなる。

プロトコルスタックから見えるネットワークインターフェースは通信メディア選択機構が提供する仮想インターフェースのみなので、プロトコルスタックの経路は常にこの仮想インターフェースを使うように設定するだけで良い。実際に使用するゲートウェイは通信メディアを管理するレベルである物理インターフェースをネットワークに接続する過程で DHCP などを用いて決定する。これにより接続するネットワークや使用する通信メディアが変わっても再構成が必要なのは通信メディア選択機構内で使用する通信メディアの切替のみでよい。

通信メディア選択機構は必要な構成要素が一つの層に集中しており、MosquitoNet や PMI にくらべ非常にシンプルになっている。接続するネットワークや使用する通信メディアが変化した時に再構成が必要な部分が一箇所に集中しているので、通信メディアの切替も簡単に行なうことができる。そして再構成に外部のモジュールを必要とせず、単独で環境の変化に適応できる。

4.3 実装上の検討

通信メディア選択機構はプロトコルスタックとネットワークデバイスドライバの間に位置する。実装する場所としてはプロトコルスタックやデバイスドライバに組み込む方法とサーバとして実装する方法がある。UNIX の場合、プロトコルスタックはデバイスドライバとともにカーネル内に実装されている。前者の方法の場合は通信メディア選択機構をカーネル内に実装することになる。カーネル内のプロトコルスタックとデバイスドライバは比較的密に結び付いており、この部分に通信メディア選択機構を組み込むには大きな変更が必要になる。

一方、今回実装の対象とするマイクロカーネル方式の RT-Mach の場合、プロトコルスタックはユーザレベルのサーバである Lites[7] 内に、デバイスドライバはマイクロカーネル内に実装されている。したがって通信メディア選択機構はユーザレベルのサーバである Lites(図 4.2) とマイクロカーネルのどちらかに組み込むことになる。プロトコルスタックとデバイスドライバがサーバとカーネルに分離しているので UNIX ほど密な関係ではないが、その間のインターフェースの変更は必要になるので Lites やカーネルの大きな変更が必要と考えられる。

サーバ方式の場合はユーザレベルのプロセスとして実装する。プロトコルスタックに対してはデバイス側がユーザープロセスから読み書き可能なネットワークインターフェースとして接続し、デバイスドライバに対してはカーネルに直接接続する。これにより既存のプロトコルスタックなどを変更せずに機能を組み込む事が出来る。また、サーバ方式は既存の部分から完全に独立しているので変更が容易である。デバイス側を読み書き可能なネットワークインターフェースは UNIX、Lites ともにトンネルデバイスとして実装されている。デバイスドライバとのインターフェースに

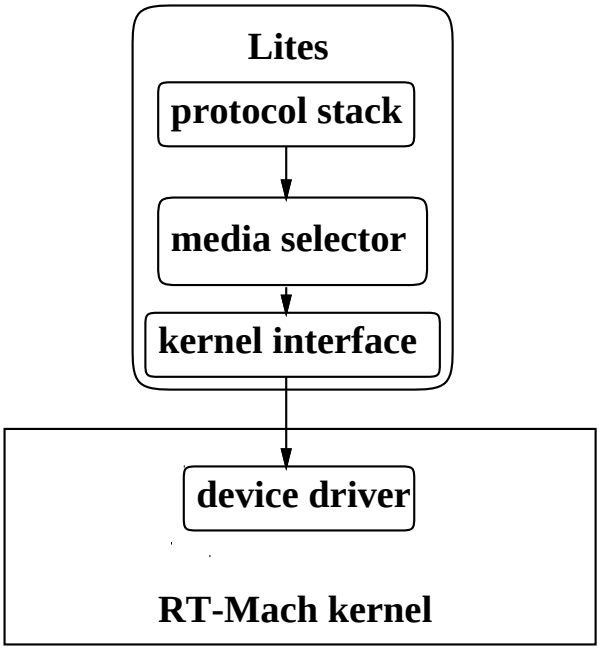

図 4.2: 通信メディア選択機構の Lites への実装

はパケットフィルタと呼ばれるインターフェースがカーネル（デバイスドライバ）内に実装されている。

そこで、今回の実装では既存の部分を変更する必要がないサーバ方式 (図 4.3) で実装を行なう事にした。

4.4 システムの構成

通信メディア選択機構は図 4.4のようにプロトコルインターフェース、デバイスインターフェース、スイッチ部から構成される。プロトコルインターフェースはプロトコルスタックとの入出力を行ない、デバイスインターフェースは通信メディアのデバイスドライバとの入出力を行なう。スイッチ部はデバイスインターフェースとプロトコルインターフェースを結び付ける。デバイスインターフェースはデバイスドライバの種類 (通信メディアの種類) ごとに定義され、スイッチ部がプロトコルインターフェースに接続するデバイスインターフェースを切り替える事で複数の通信メディアを使い分ける事が出来る。プロトコルインターフェース、デバイスインターフェースとスイッチ部の間でインターフェースが定義されており、スイッチ部はどのプロトコルインターフェース、デバイスインターフェースとも同じ方法で接続する事が出来る。

プロトコルインターフェースの主な機能はプロトコルスタックとの入出力である。現在はトン

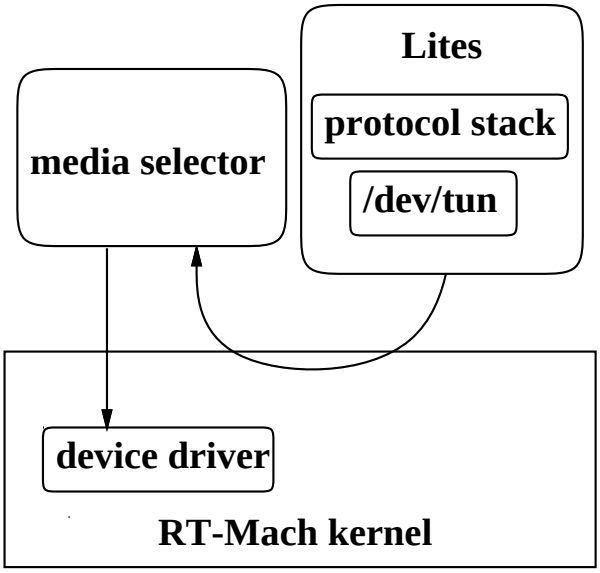

図 4.3: サーバ方式での通信メディア選択機構

ネルデバイスを想定しており、トンネルデバイスとの入出力のほかに初期化や終了処理を想定したインターフェースを定義している。各機能はスイッチ部から呼び出される関数として実現されている。

デバイスインターフェースはデバイスドライバとの入出力のほかにネットワークとの接続も管理する。デバイスドライバとの入出力には RT-Mach マイクロカーネルのパケットフィルタを使ってデバイスドライバと直接接続して行なう。入出力やパケットフィルタの設定などの機能はスイッチ部から呼ばれる関数として実現されている。一方、ネットワークの接続の管理はスレッドとして実現されている。スレッドは ARP、DHCP の機能のほかにネットワークとの接続状態の監視機能も持っている。ネットワークに接続されてから IP アドレスを確保するまでの処理はこのスレッドが行なっている。デバイスインターフェースとスイッチ部の間では IP パケットをやり取りしており、リンク層のための処理はすべてデバイスインターフェース側で行なっている。このためスイッチ部はリンク層を意識する必要はなく、イーサネットと PPP を同等に扱う事が出来る。

スイッチ部はプロトコルインターフェースとデバイスインターフェースを結び付け IP パケットの入出力を行なう。スイッチ部はアップコール用とダウンコール用の二つのスレッドを持っており、IP パケットの入出力を平行して行なっている。アップコールスレッドはデバイスインターフェースを用いて IP パケットを取り込みプロトコルインターフェースを用いてプロトコルスタックに渡す。ダウンコールスレッドは反対の動作を行なう。スイッチ部はそのほかに IETF Mobile IP のフォーリンエージェントとしての機能を持っている。移動ホストがネットワークへの接続の監視

やフォーリンエージェントの探索に用いる拡張されたICMPルータディスカバリへの応答、ホームエージェントとの間のレジストレーションメッセージの中継を行なう。スイッチ部はデバイスインターフェースを切り替えるためにデバイスインターフェースの状態を監視している。切替の時にはアップコールとダウンコールのスレッドで同時に切り替えなければならないので、ミューテックスとコンディション変数を用いた同期機構が用意されている。

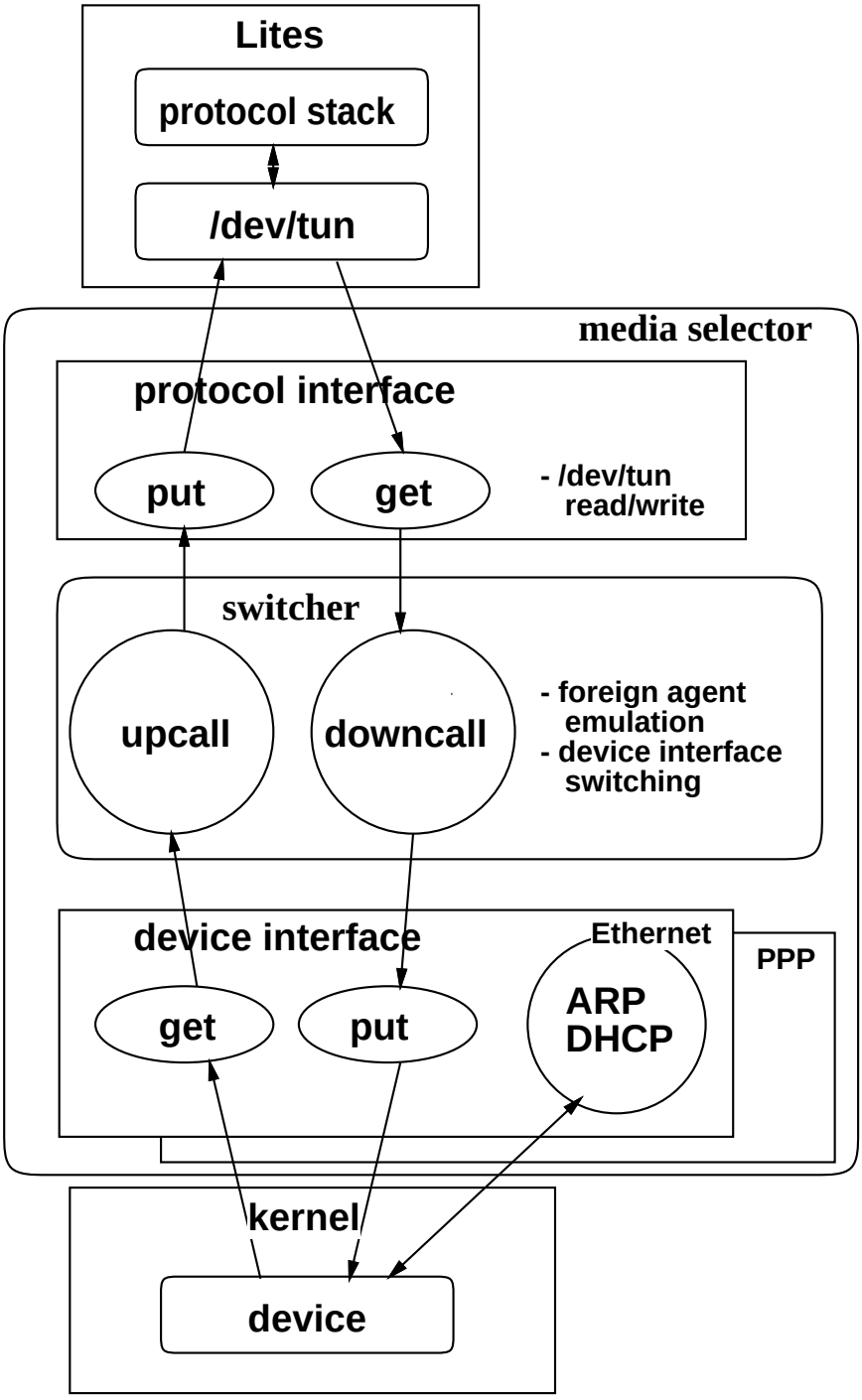
Lites
protocol stack
/dev/tun
media selector
protocol interface
put
get
- /dev/tun
read/write
switcher
upcall
downcall
- foreign agent
emulation
- device interface
switching
device interface
Ethernet
PPP
get
put
ARP
DHCP
kernel
device

図 4.4: システムの構成

第 5 章

評価

RT-Mach 上に実装された通信メディア選択機構を用いて計算機の移動の実験を行なった。実験では移動ホストに PORTEGE 610CT(Pentiunm 90MHz) と Thinkpad530、ホームエージェントに GATEWAY2000(Pentium 90MHz) を用いた。ホームエージェントは FreeBSD2.2.2 上で動作させている。ネットワークはホームネットワークはイーサネット、移動先のネットワークはイーサネットと無線 LAN の Netwave、28.8kbps のモデムカードで接続した PPP を用いた。

移動はイーサネットのホームネットワークと無線 LAN の移動先ネットワーク、イーサネットの移動先ネットワークと PPP の移動先ネットワークという組合せで行なった。(図 5.1) 移動の際は使用する通信メディアをネットワークに接続しておき、使用中の通信メディアを切り離すことで通信メディアの切替を起こさせた。切替え前に移動ホストから telnet で他の計算機にリモートログインしておき、通信メディア切替後も telnet セッションが維持されることを確認した。

切替に擁する時間

切替えに要する時間は大部分が IETF Mobile IP のレジストレーションの切替の時間で、モーバイルエージェントの動作の周期で決まる。通信メディアの切替が発生したタイミングにもよるがモーバイルエージェントの動作に対して 2 周期以内で切替えは完了している。

通信メディアの切替えに要する時間は、Pentiunm プロセッサーが持つタイムスタンプカウンターレジスタを用いて計測した。切替時間は切替えの動作が開始されてから完了するまでとし、無線 LAN からイーサネットへの切替を 10 回計測し、その平均をとった。

計測結果を表 5.1に示す。通信メディアの初期化と後処理はカーネルのパケットフィルタの設定と削除の処理である。スレッドの同期は使用する通信メディアなどの内部データを更新するために必要なアップコールとダウンコールの両スレッドとの同期をとるための時間である。

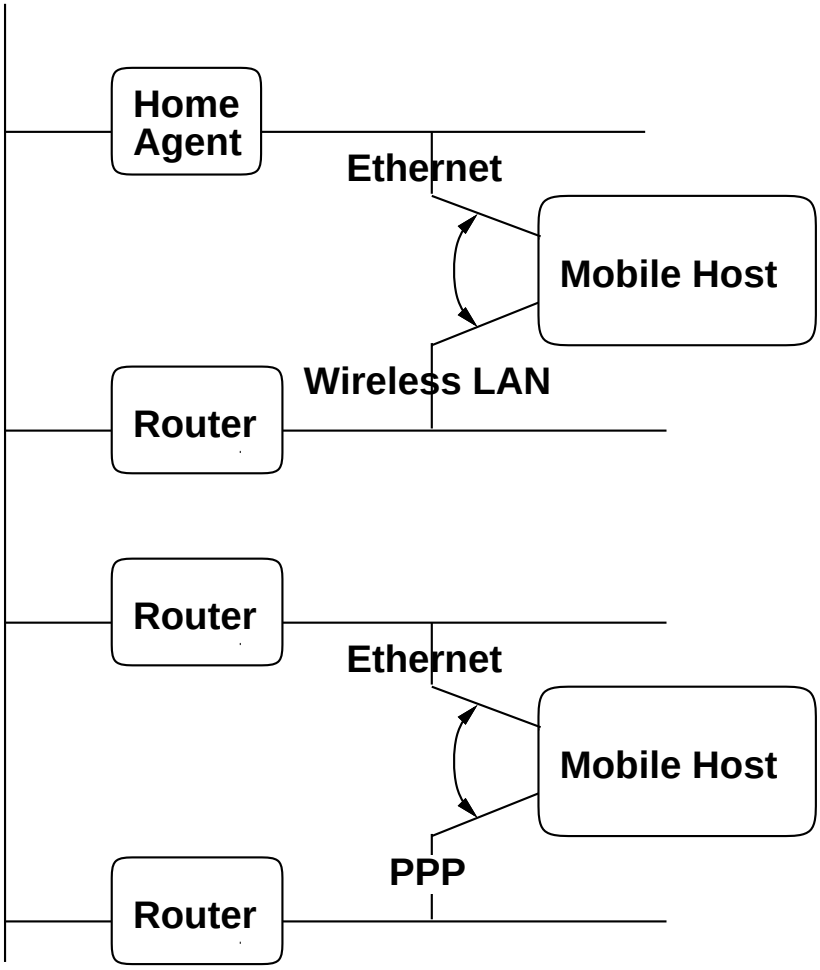

図 5.1: 実験ネットワーク

表 5.1: 通信メディアの切替え時間

	時間 (μ s)
新しい通信メディアの初期化	1499
スレッドの同期	6469
内部データの更新	69
切り離した通信メディアの後処理	97
合計	8134

表 5.2: 通信メディア選択機構によるラウンドトリップタイムの変化

	未使用	使用
ラウンドトリップタイム	1.393	11.411

この結果から切替えに要する時間の大部分はスレッドとの同期に費やされている。

オーバーヘッド

通信メディア選択機構のオーバーヘッドを調べるために、通信メディア選択機構を使用した場合としない場合のパケットのラウンドトリップタイムを比較した。計測ではホームネットワークに接続した状態で同じサブネット内のホストから ping を行ない、その往復時間を計測した。(表 5.2)

通信メディア選択機構の処理時間はアップコール、ダウンコールの各スレッドでパケットを取り出してから出力するまでの時間をタイムスタンプカウンターを使って計測した。(表 5.3)

ラウンドトリップタイムは 10ms 増加しているが、スレッドの走行時間は 0.5ms 程度しかない。このことから、オーバヘッドの大部分はパケットの入出力で発生する通信メディア選択機構とカーネルの間の切替えの時間と考えられる。

表 5.3: スレッドのパケット処理時間

	時間 (μ s)	
アップコール	400	
ダウンコール	88	

第 6 章

考察

6.1 システムの構成と実装

6.1.1 システムの構成

本研究で設計、実装した通信メディア選択機構はネットワークプロトコルスタックとネットワークデバイスドライバの間に挿入される。実現法方としては既存のモジュールに追加することで直接挿入する方法と、本体はサーバとして作成し既存のモジュールにはプロトコルスタックとデバイスドライバの間に本体への迂回路のみを組み込む方法が考えられる。

既存のモジュールへ追加する方法

プロトコルからデバイスドライバまでは処理を高速化するために密接に作られている。実際、ネットワークデバイスドライバのカーネルインターフェースは RT-Mach でも UNIX でもネットワーク専用になっている。とくに UNIX ではプロトコルスタックもカーネル内に実装されていてデバイスドライバとプロトコルスタックは直接結び付いている。通信メディア選択機構を直接この部分に挿入する事は、カーネルの大幅な変更が必要になると考えられる。

マイクロカーネルである RT-Mach の場合はプロトコルとデバイスドライバはユーザレベルのサーバである Lites とマイクロカーネルに分けられている。通信メディア選択機構をプロトコルとデバイスドライバの間に挿入する事を考えた場合、Lites のプロトコルスタックの下位に組み込む、サーバとして Lites とマイクロカーネルの間に挿入する、マイクロカーネル内のデバイスドライバの上位に組み込む、の 3 種類が考えられる。

Lites またはカーネルに配置する場合、Lites とカーネル間のインターフェースとプロトコルスタック、またはデバイスドライバの間に挿入することになる (図 4.2)。この時 Lites とカーネルの

間のインターフェースを変更しないようにすると、UNIX と同じ様に密接に作られた部分に追加する場合と変わらない可能性がある。Lites はユーザレベルのサーバなので UNIX のようにカーネルを変更する場合よりは軽い作業のように思われるが、Lites は 4.4BSD のサービスを提供するかなり大きなサーバなので実際の作業は UNIX の場合とあまり変わらない可能性が高い。Lites とカーネルの間のインターフェースを変更する場合は Lites またはカーネルの変更は小さくなるかもしれないが、Lites とカーネルの両方を変更しなければならない事も考えられる。したがって、既存のモジュールに追加する方法は RT-Mach でもかなり重い作業になると予想される。

サーバによる方法

サーバ方式では、プロトコルスタックとデバイスドライバの間でパケットを横取りし必要な処理を行なった後プロトコルスタックまたはデバイスドライバに出力する。プロトコルスタックにはデバイス側をユーザプロセスから読み書き可能なトンネルデバイスという仮想ネットワークインターフェースが実装されている。一方デバイスドライバ側はパケットフィルタと呼ばれるデバイスドライバと直接接続できるインターフェースが既に存在する。これらのインターフェースは UNIX、RT-Mach ともに持っている。UNIX の場合はこれらのインターフェースを使って通信メディア選択機構のサーバを実装する事が出来る (図 6.1)。

この方法ではプロトコルスタックやデバイスドライバに必要なインターフェースが既に実装されているので、既存のモジュールは変更する必要がない。さらにサーバはユーザプロセスとして実現されるのでカーネル内に組み込む場合に比べてはるかに軽い作業ですむと予想される。

RT-Mach の場合はすでに Lites とマイクロカーネルという形でプロトコルスタックとデバイスドライバが分離されているので、UNIX のようなトンネルデバイスとパケットフィルタを用いる方法のほかに Lites とマイクロカーネルの間に配置する方法も考えられる。しかし、Lites とマイクロカーネルの間は Lites が個々のデバイスドライバと接続されるので、ここにネットワークデバイスを統一的に扱おうとする通信メディア選択機構を配置する事は非常に難しい。少なくとも Lites 側のインターフェースも変更が必要になる。

一方トンネルデバイスとパケットフィルターを用いる方法は UNIX と同じ様に既存のモジュールの変更は必要ない。既存のインターフェースを用いてサーバを実装するのならば UNIX でもマイクロカーネルでも一つのユーザプロセスなので、どちらの OS でも大きな違いはないと思われる。

以上の検討から、密接に結び付いたモジュール間に挿入する通信メディア選択機構は、直接組み込むよりはサーバ方式の方が変更量や作業は軽くなると考えられる。また、RT-Mach の場合、Lites とカーネルの間に配置する方法もあるが、コストは小さくない。マイクロカーネルは各機能

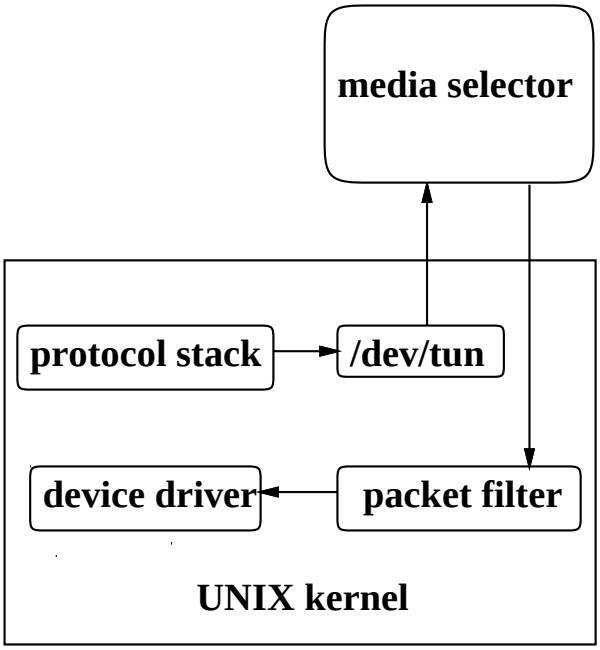

図 6.1: UNIX でのサーバ方式による通信メディア選択機構

がサーバとして実現され機能の変更や拡張を行ないやすいといわれていたが、サーバ間にまたがったりインターフェースをそのまま使えないような場合のコストは決して小さくないと考えられる。

サーバ方式の問題点としてユーザモード、カーネルモードの切替が増えオーバーヘッドが増加する事がある。UNIX の場合、カーネル内のプロトコルの処理の間に一度ユーザ空間に出て再びカーネル空間に戻ってくることになり、空間の切替が 2 回発生する。RT-Mach の場合も、Lites からサーバへの切替に 2 回、サーバからマイクロカーネルに 1 回の切替が必要であり、従来の Lites からマイクロカーネルへの 1 回から 3 回になっている。どちらもオーバーヘッドは増加するが、増加の度合は RT-Mach の方が小さい。

6.1.2 実装

通信メディア選択機構はプロトコルスタックからの IP パケット、ネットワークデバイスドライバからのフレーム、ネットワークとの接続ための ARP への応答などの処理を同時に行なわなければならない。このような複数の入出力を同時に扱う方法は、UNIX で提供されている select システムコールのような入出力が可能になったかどうかを同時にチェックできる機能を使う方法と、RT-Mach で提供されているスレッドを使う方法がある。

select システムコールを用いる場合は入出力をイベントと見立てて、select システムコールでイベントを検出してイベント毎の処理を行なうイベント駆動型の設計を行なうことになる。この時、

一つのデータの流れの中の入出力を別々のイベントとして扱わなければならない場合がある。この場合一つのデータの流れにもかかわらず入出力は別々のイベントに対する処理として設計しなければならない。さらに、複数のイベントを同時に扱う必要上、イベント処理の中で待ち合わせを行なうことができず、待ち合わせ自体もイベントや内部状態として持たなければならない。このため、イベントや内部状態とそれに対する処理はかなり細かく定義しなければならない。

スレッドを用いる場合は入出力データの流れ毎にスレッドを割り当てて並列に処理させる。これにより個々のデータの流れに対してシーケンシャルに処理する事が出来るので、各データの流れが独立している場合はイベント駆動型に比べ設計は容易である。しかし、データの流れが密接に関係している場合は複数のスレッド間で同期が必要になる。処理の流れの中で同期が必要な場所が多くなると各処理毎の同期の関係が複雑になり、デッドロックの可能性が生じる。これを避けるためには処理の流れを細かく分解し、注意深く同期をとらなければならない。このため、複雑な同期を行なわなければならないような密接な関係を持つデータの流れが存在する場合はスレッドの利点がなくなってしまう。

select システムコールとスレッドの比較を元に、通信メディア選択機構の実装方法を検討する。検討項目には実装の容易性と拡張性があげられる。

実装の容易性

両方式を比較すると、複雑に関係した複数の流れがある場合はイベント駆動型が有利で、関係があまりない場合はスレッドの方が有利と考えられる。通信メディア選択機構の場合、データの流れはプロトコルスタックとデバイスドライバの流れ、各デバイス毎の ARP や DHCP などのデバイス毎の処理がある。これらの流れはほぼ独立しており、別々に扱っても支障が無い。プロトコルスタックとデバイスドライバの流れにはアップコールとダウンコールの二つの流れがある。デバイス切替の時は同時に切り替えなければならないので同期が必要になるが、必要な同期はこれだけなのでたいした問題にはならない。したがって、通信メディア選択機構の場合はスレッドを用いた設計の方が select システムコールを用いたイベント駆動型の設計より有利と考えられる。

拡張性

通信メディア選択機構は複数の特性の異なる通信メディアを扱わなければならない。例えばイーサネットと PIAFS などを使った PPP ではデバイスドライバとのインターフェースがまったく異なり、また入出力の手順も異なる。また、イーサネットの場合は ARP と DHCP の処理が必用であるが、PPP の場合 ARP は必要なく IP アドレスの割り当ては回線接続の過程で行われる。この

ようにイーサネットと PPP では処理手順がまったく異なり、イベントとしてみた時にはまったく別々に扱わなければならない。したがって、イーサネットのみで設計された通信メディア選択機構に PPP を追加する場合、イベント駆動の設計ではイベントを検出する中心部分を変更しなければならないだけでなく、イベント全体の再設計が必要になる可能性もある。しかし、データの流れとしては独立しているのでスレッドが使える場合は別々のスレッドに割り当てれば良く、新しいデバイスを追加する場合もデバイスインターフェースの追加だけすむと予想される。

6.1.3 まとめ

以上、RT-Mach 上の通信メディア選択機構の設計と実装について UNIX との比較も含めて検討を行った。RT-Mach のようなマイクロカーネルは機能をサーバとして持つので拡張性が高いといわれているが、通信メディア選択機構のようなサーバやカーネルとのインターフェースの変更が必要な場合やカーネルやサーバの変更が必要な場合は拡張に必要な労力は UNIX のようなモノリシックな OS とあまり変わらないと考えられる。本研究で採用したサーバ方式では既存のインターフェースを利用した事もあり、RT-Mach と UNIX では余り差がなさそうである。サーバ方式はパフォーマンス上は不利であるが、管理するデバイスの追加など今後の拡張を考えるとサーバ方式の方が好ましい。

一方複数の入出力を扱うことを考えると、通信メディア選択機構が扱うデータの流れは独立性が高い事もありスレッドを前提とした設計の方がはるかに有利と考えられる。最近の UNIX はユーザレベルスレッドを持っているが、デバイスドライバのようなカーネル内での待ちが発生する場合はユーザレベルスレッドでは全スレッドが停止してしまうので、通信メディア選択機構には不十分である。この点ではカーネルレベルスレッドの使える RT-Mach の方が有利と思われる。また、通信メディア選択機構が扱うデバイスの追加や機能の追加を考える時、スレッドは各モジュールの独立性を高めるのに役立っている。

全体として、システム構成上は RT-Mach と UNIX では余り差が無いが、実装上は RT-Mach が有利と思われる。通信メディア選択機構は今後対応する通信メディアの拡大などの拡張が予想される。この事も含めると通信メディア選択機構の設計と実装は妥当な物と思われる。

6.2 通信メディアの抽象化

多様な通信メディアを統一的に扱うためには通信メディアを何等かの形で抽象化する必要がある。

これまでのシステムでもリンク層でネットワークインターフェースの抽象化が行われてきた。これまで用いられてきた通信メディアはイーサネットのような屋内で用いられる有線のメディアを

想定している。無線 LAN や PIAFS などの新しい通信メディアはこれまでの有線のメディアと異なった特性を持っており、従来の抽象化では扱い切れない面がある。

6.2.1 新しい通信メディアの特性

移動計算機環境向けに開発された通信メディアは従来のものとは異なる特性を持っている。

ネットワークへの接続手順

PHS などを利用した PPP による接続の場合は物理的に接続しただけでなくダイアルして回線を繋がなければならない。さらに接続するための認証が必要な場合も多い。PHS や携帯電話などのダイアルに要する時間は１０数秒もあり、必要になったからといってすぐに使用をはじめる事が出来ない。すぐに使用できるように常時回線を接続しておくという選択もあるが、PHS などは公衆回線を使用するので課金がありトラフィックがない時も回線を接続しておく事はコスト的に適切ではない。一方イーサネットなどは物理的にケーブルを接続するだけで良い。

ネットワークとの接続状態

PHS や無線 LAN といった無線を利用する通信メディアは状況によってバンド幅などの特性が大幅に変化する。この特性の変化は基地局との距離や位置関係による電波状態や回線の品質といった物理的な要因による。イーサネットなどのこれまでの通信メディアも特性が変化したが、これは接続されている計算機の数やトラフィックといった利用率による物で物理的な要因ではない。

PHS や無線 LAN の特性の変化は物理的な要因によるが、この要因はハードウェア的に知る事が出来る場合がある。無線 LAN の一種である Netwave の場合、無線の状態などのハードウェアレベルの接続状態を取り出す事が出来る。これにより基地局との間のバンド幅などの状態を知る事が出来る。バンド幅の現在の状態を知る事が出来れば、バンド幅が大幅に低下した場合はそのメディアの使用をあきらめ別のメディアに切り替える事が可能になる。

複数の接続先

PHS はダイアル番号によって複数の接続先を持つ。無線 LAN の一種である Netwave はドメインと呼ばれる複数のチャネルを持ち、このドメイン番号を基地局に割り当てることにより接続する基地局を複数の基地局の中から選択することができる。これらの接続先はソフトウェア的に選択される。そのためこれらの通信メディアを使用するためには通信メディアを

物理的に切り替えるだけでなく、ソフトウェア的に接続先を選択しなければならない。イーサネットなどのこれまでの通信メディアは接続先は物理的にもソフト的にもひとつしかなく、通信メディアそのものがネットワークへの接続先となっている。

接続先は移動計算機と基地局の位置関係によって決まり、固定的なものではない。また Netwave の場合同じドメインをことなるネットワークの基地局に割り当てることができるので、接続先とネットワークアドレスは必ずしも一致しない。したがって通信メディアの IP アドレスは常時変化していると考えなければならない。

以上の特性からこれらの通信メディアはパケットを流すまでネットワークアドレスや経路が完全には確定しないといえる。にもかかわらず、接続を確立し使用可能な状態になるまでの時間がこれまでのメディアに比べ非常に長い場合がある。

6.2.2 移動計算機環境を考慮した抽象化

新しい通信メディアはこれまでの通信メディアと比べるとネットワークアドレスや経路が安定的でない。通信メディアにパケットを流すまでネットワークアドレスや経路が確定しない場合もある。このような通信メディアの取り扱いは大きな問題である。

問題点を整理するために移動計算機環境のネットワークについて考えると、移動計算機には二つのカテゴリーが考えられる。ひとつは移動するサブネットのルータ (移動ルータ) であり、もうひとつはユーザが持ち歩く携帯型計算機である。ここでは携帯型計算機について考える。

携帯型計算機の基本的な用途はクライアントである。この場合計算機はネットワークに接続できればよく、ネットワークへの接続点を複数持つような要求はない。実際 DHCP による設定や PPP によるダイアルアップ接続でのネットワークの経路は、接続先のネットワークのデフォルトルータをデフォルト経路として設定するのが一般的である。このことから移動計算機環境では接続可能なインターフェースが一つあれば十分と考えられる。

以上の条件で新しい通信メディアの抽象化を検討する。

複数の接続先

いくつかの通信メディアは複数の接続先をソフト的に切替えることができる。現在の抽象化ではネットワークの経路はネットワークインターフェース単位になっており、このような経路は扱えない。

新しいネットワークインターフェースのオプションには複数のタイプのコネクタを持つネットワークインターフェースにおいて使用するコネクタを指定するためのオプション (media) がある。

このオプションは複数の接続先を選択するという意味では似ているが、ネットワークの経路は同じなのでこのオプションでは扱い切れないと思われる。接続先ごとに仮想的なネットワークインターフェースを定義し経路を設定する方法も考えられるが、Netwave の場合は接続先とネットワークは 1 対 1 で対応するわけではないので固定的な経路を割り当てることができない。

しかし、携帯型計算機を前提とする場合、ネットワークの経路はネットワークに接続した時に接続先のネットワークに合わせて再構成する必要があるので、複数の接続先の経路の維持は意味がない。このことを前提とすれば複数の接続先があっても一つのネットワークインターフェースとして扱っても問題はなく、接続先の選択はネットワークインターフェース側で行なえば良い。通信メディア選択機構の場合仮想的なネットワークインターフェースのもとで複数の通信メディアを管理しており、プロトコルスタックはネットワークインターフェースとしてはこの仮想インターフェースのみを使用する。接続先は仮想インターフェース側で管理しプロトコルスタックから隠蔽去れるので、経路はこのインターフェースをデフォルトの経路とすることができる。

以上の検討から、無線 LAN や PHS のような複数の接続先を持つ通信メディアは移動計算機を前提とした場合は通信メディア選択機構のような仮想インターフェースとして扱うほうが有利と考えられる。

回線の接続/切断とそれに要する時間

PHS などの公衆回線を利用する通信メディアなどは通常は回線を切断しておいて必要な時だけ接続するという運用が要求される。現在の抽象化ではネットワークインターフェースは使用可能/不可能の区別しかない。回線の切断状態を使用不可能の状態とするとそのインターフェースはルーティングから切り離される場合がある。一方常時使用可能にしておき必要な時に回線を接続するという選択もある。この方法は回線の接続に要する時間が数秒以下の ISDN で用いられている。TCP は信頼性のない通信路を仮定しておりコネクションを切断するまでの時間は短い実装でも 2 分なので、回線接続の時間程度ではコネクションをリセットすることはない。

しかし PHS や従来の公衆回線は接続に１０数秒かかり、アプリケーションのタイムアウト値がこれ以下の場合はアプリケーションが異常として扱ってしまいコネクションが切れる場合がある。これはアプリケーションの対応がないと解決できない問題である。

以上の検討からネットワークプロトコルの観点からはオンデマンドで回線を接続するネットワークインターフェースとして扱えるが、アプリケーションを含めて考えるとアプリケーションとの連係できる機能が必要と思われる。

6.2.3 まとめ

移動計算機環境で使用される通信メディアは動作がオンデマンドでかつ経路が固定的でないというこれまでのネットワークシステムの抽象化では扱い切れない部分がある。通信メディア選択機構は仮想的なネットワークインターフェースを使い通信メディアの違いを隠蔽しているが、この方法はこのような特性に対しても有効と考えられる。しかし回線接続による遅延はアプリケーションによる対応が必要な場合もあり、通信メディア選択機構だけでは対応し切れていない。

6.3 通信メディアの選択方法

移動計算機環境では複数の通信メディアが同時に使用可能な場合がある。このような時に最適な通信を行なうためには使用する通信メディアを決定するためのしくみが必要である。通信メディアを選択するためのパラメータとしては通信メディアの特性、ユーザ (計算機) の場所などが考えられる。通信メディアの特性にはバンド幅、遅延、エラー率、回線を使用するためのコストがある。ユーザの場所からは接続可能な基地局や基地局との位置関係から通信メディアのバンド幅などの期待値を得る事が出来る。これらの情報とユーザの要求から使用する通信メディアを選択する。

6.3.1 ルール

どのような条件の時にどの通信メディアを選択するかというルールを記述しておき、ユーザの要求を条件として対応する通信メディアを検索する。例えばバンド幅が大きいメディアを選択する、コストの小さいメディアを選択する、というように通信メディアの特性を条件としてルールを記述しておき、指定された特性を満たす通信メディアを選択する。実際の使用環境を考えると、指定されたコストの範囲で最もバンド幅の大きい通信メディアというように複数の条件を指定できる必要がある。

バンド幅とコストのように複数の条件を指定する場合複数の条件の間で優先順位やトレードオフを記述する必要がある。このような記述は多次元のマトリクスで記述する方法がある。しかし、次元数が２や３ぐらいまでは記述できるがそれ以上になると記述が非常に複雑になり、矛盾のない記述が難しくなる。

6.3.2 評価関数

ユーザの指定する条件に対して通信メディアの特性の有用性をあらわす評価関数を設定し、評価値がもっとも大きい通信メディアを選択する。この方法では評価関数による評価値の大小関係

で通信メディアが選択されるので、ルールを記述できないというような問題は発生しない。しかし通信メディアの特性や条件を満たせるような評価関数が必要になる。通信メディアの特性はバンド幅やコストのように値の大小関係で比較できるので、評価関数としては特性値にたいして増加、又は減少しつづけるような関数を使用できる。また上限や下限を指定する場合、その値を越えると評価値が 0、あるいは大幅に低下する必要がある。

その様な特性を持つ評価関数として本研究では低域フィルタの近似のひとつであるワーグナ特性の近似式 $|H(j\omega)|^2 = 1/(1+\omega^{2n})$ を使って評価を行った。特性の値を ω の値としてワーグナ特性の値を計算する。指定された特性の上限値/下限値で各通信メディアの特性値を規格化しワーグナ特性の値を計算した結果を評価値とする。特性の優先順位は近似式の n の値にあてはめる。ワーグナ特性では n が大きくなるほど規格化した値付近での傾きが急激になるので、上限値/下限値を満たさない特性の評価値が低下する。これにより条件を満たさない通信メディアの評価値は小さくなり、選択されなくなる。

ワーグナ特性を評価関数としていくつかの条件で通信メディアの選択を行った。条件値の値と通信メディアの特性値が一定の範囲内ではおおむね期待した選択が出来るが、場合によっては評価値が逆転する場合がある。

例としてバンド幅が非常に大きいがコストも大きい通信メディアを考える。今回用いた評価関数では値が増加、あるいは減少しつづけるのでコストが上限値を下回ってもバンド幅の評価値が大きくなり、全体の評価値はコストの条件を満たす通信メディアを上回る場合がある。コストの優先度を大きくしてもバンド幅の指定しだいではコストの条件を満たさない通信メディアの評価値が高くなってしまう。

今回用いた評価関数では特定の特性の評価値が大きく影響する場合があり、優先度が逆転してしまう場合があった。評価の条件には現実の通信メディアを想定しているので、この結果はこの方法もすべての場合で使用できるわけではない事を示している。

6.3.3 まとめ

通信メディア選択のポリシーはユーザの要求と通信メディアの特性が複雑に関係しており非常に難しい。本研究ではいくつかの方法を検討したがどの方法も十分ではない。この問題は今後の研究課題である。

6.4 移動計算機環境のネットワークシステム

本研究で開発したシステムは移動計算機環境の問題のうち移動計算機側の問題を解決するものである。しかし移動計算機環境はネットワーク全体を含むシステムであり、移動計算機上の対処だけでは解決できない問題も存在する。

6.4.1 通信の最適化

移動計算機環境で用いられる通信メディアの特性は通信メディア毎に大きく異なる。例えばバンド幅を考えると現在使用可能な物は高速な物でも 2Mbps 程度で、屋外での中心的なメディアである PIAFS では 32kbps しかない。このバンド幅も公称最大値であり、計算機と基地局の位置関係などの要因で大幅に低下する場合がある。したがってユーザからみれば計算機のレスポンスが使用中に大幅に変動する事になる。他の計算機にリモートログインして作業しているなら計算機のレスポンスが急に悪くなったり良くなったりする。Web サーバにアクセスしている場合はデータの転送が急に遅くなったり止まったりする。マルチメディアアプリケーションではビデオの再生がぎこちくなくなったり音がおかしくなったりする。

このように移動計算機環境では単にネットワークがつながっているだけでは不十分である。すなわちバンド幅などの通信環境が変動してもユーザにたいするサービスの質を維持できる機構が必要である。例えば通信メディアの特性に合わせて通信手順を変えたり、バンド幅に合わせてデータ量を調整したりデータを圧縮するなどが考えられる。これらの対応はアプリケーションで行なう事も可能であり、マルチメディアアプリケーションではネットワークのバンド幅をユーザが指定する事で動作を変えるという事が行われている。しかし通信環境の変動はアプリケーションの動作中にも発生する事、変動は予測可能ではない事を考えるとシステム側でも対応が必要である。

IETF Mobile IP による移動計算機環境を考えると、フォーリンエージェントを使用する場合移動ホストはフォーリンエージェントに接続される。この場合フォーリンエージェントは移動ホストのルータとして位置付けられている。すなわち移動ホストの通信はすべてフォーリンエージェントを経由する。そこでフォーリンエージェントと移動ホストの間で通信の最適化を行なう事ができる。例えば移動ホストが処理する必要の無いパケットをフォーリンエージェントで破棄し、移動ホストとの通信量を減らす事が考えられる。更に移動ホストへ送るデータの圧縮や間引きといったより積極的にデータ量を調整する事が考えられる。

6.4.2 移動計算機環境のサービス

ネットワークとの接続性だけでなく通信というサービスを提供するというモデルが考えられる。すなわち移動計算機はネットワークに接続するのではなくネットワークが提供するサービスに接続するのである。

ネットワークにはマルチメディアデータや Web などの各種サービスを提供するサービスプロキシが配置される。移動計算機は移動した場所のサービスプロキシを経由してネットワークに接続する。サービスプロキシは移動計算機から要求されたサービスと接続に使用された通信メディアに対応してデータ量や通信手順を最適化する事で移動計算機のユーザに安定した質のサービスをサービスを提供する。

この段階ではネットワークは論理的なネットワークと物理的なネットワークに分離される。論理的なネットワークはこれまでのネットワークのな通信路に見える。その下位では物理的なネットワークが通信路に応じたデータの圧縮などの処理や経路制御を行ない、仮想的なネットワークを提供する。(図 6.2) たとえば、IETF Mobile IP は IP によるネットワークの上にトンネリングによる仮想的なネットワークを構築したものと捉えることができる。

本研究の通信メディア選択機構はシステムを論理的なレベルと物理的なレベルに分離してており、このようなネットワークに適合できる。

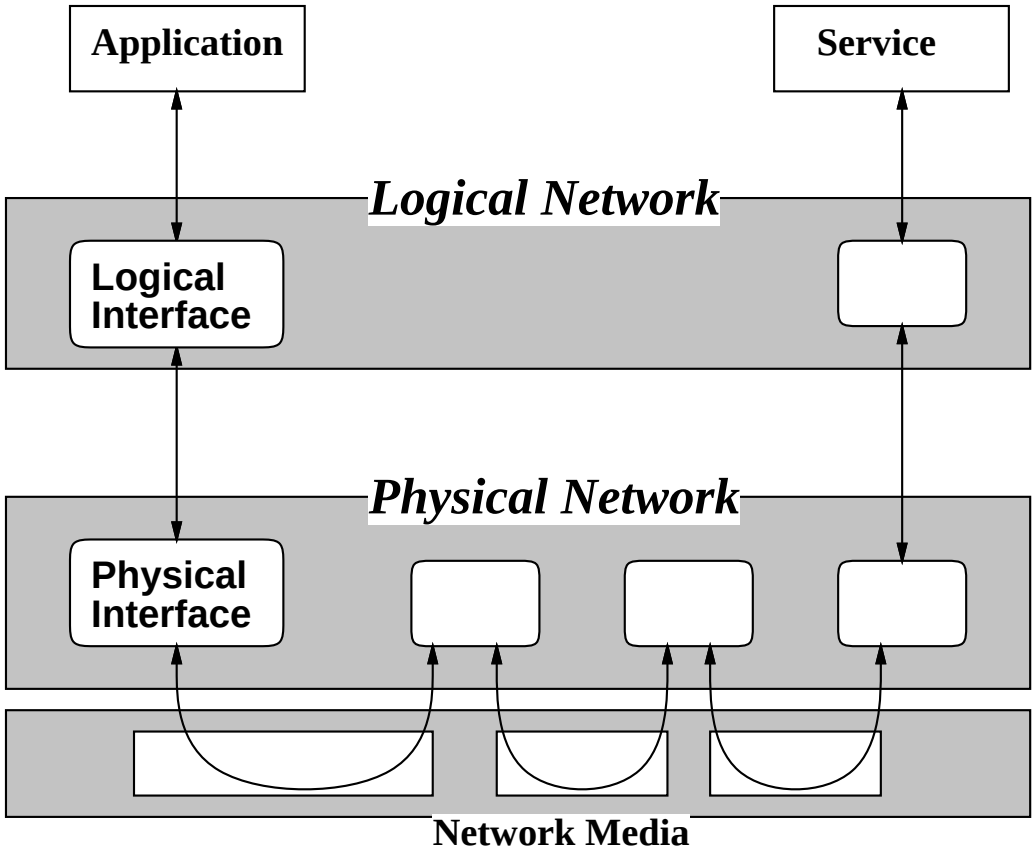

図 6.2: 論理ネットワークと物理ネットワーク

第 7 章

まとめと今後の課題

移動計算機環境では通信メディアの変化や移動によってネットワークでの通信を維持できないという大きな問題がある。本研究では動的な通信メディアの切り替えを可能とするシステムを構築し、IETF Mobile IP と組み合わせて使用する事でこの問題の解決を試みた。RT-Mach 上に実装された試作システムは通信メディアとしてイーサネット、無線 LAN、モデムカードによる PPP を使ってネットワーク上を移動しても通信を維持する事が出来た。

移動計算機環境のネットワークサポートはおもにプロトコルスタックのルーティングを拡張する事で対応されてきた。そのため、IP アドレスの管理や通信メディアの管理は独立して行われる場合が多い。しかしこれらは密接に関連しており、システムが自立的に適応するにはひとつのシステムとして動作する必要がある。

本研究では問題の鍵は計算機識別のためのアドレスとネットワークに接続するためのアドレスが明確に分離されていない事と考え、この分離を試みた。リンク層レベルで仮想的なインターフェースを定義してその下位で通信メディアを管理する事で、分離を行った。これによりプロトコルスタックとは独立して通信メディアを切り替えたり、無線 LAN や PIAFS のようなこれまでの通信メディアとは異なる特性のメディアを扱う事を可能にしており、自力で環境の変化に適応できる。本研究で実装した試作システムはサーバ方式で実装されており、試作システムという前提においては妥当な設計と考えられる。

計算機の位置や通信状況に適応するためには最適な通信メディアを選択する事が重要であるが、非常に難しい。本研究でもルールや評価関数を用いる方法を検討したが、決して十分な結果は選られなかった。この問題は今後の課題である。

本研究では通信メディア選択機構によってネットワーク上の移動を可能としたが、移動計算機環境のネットワークでは単にネットワークにつながるだけでは十分ではない。ユーザに十分な質

のサービスを提供するには状況に応じた通信の最適化が必要である。IETF Mobile IP のフォーリンエージェントは移動計算機との通信を最適化する手段として有望であるが、仕様上の問題により現状では使用に制限がある。

今後は十分な能力を持つ通信メディアの選択方法の開発やフォーリンエージェントの拡張による通信の最適化を行ない、ユーザに十分な質のサービスを提供できるネットワークシステムの構築を目指す。

参考文献

[1] Mary G.Baker, Xinhua Zhao, Stuart Cheshire, Jonathan Stone, "Supporting Mobility in MosquitoNet", Proceedings of the 1996 USENIX Technical Conference, San Diego, CA, January, 1996.

[2] C. Perkins, "IP Mobility Support", RFC2002, October, 1996.

[3] F. Teraoka, Y. Yokote and M. Tokoro "A Network Architecture Providing Host Migration Transparency", ACM SIGCOMM'91, 1991.

[4] Jon Inouye, Jim Binkley, Jonathan Walpole "Dynamic Network Reconfiguration Support for Mobile Computers", Proceedings of the Third ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom '97), Budapest, Hungary, September 26-30, 1997.

[5] R. Drome, "Dynamic Host Configuration Protocol", RFC1541, October, 1993.

[6] T. Tokuda, T. Nakajima, and P. Rao, "Real-Time Mach: Towards a Predictable Real-Time System", Proceedings of the 1st USENIX Mach Symposium, October, 1990.

[7] J. Helander, "Unix under Mach: The Lites Server", Helsinki University of Technology, Master's Thesis, 1994.

[8] Akihiro Hokimoto, Kuniaki Kurihara, Tatsuo Nakajima, "An Approach for Constructing Mobile Applications using Service Proxies", The 16th International Conference on Distributed Computing System(ICDCS'96), March, 1997.

[9] Tatuo Nakajima, Hiroyuki Aizu, Masaru Kobayashi, Kenji Shimamoto, "Environment Server: A System Support for Adaptive Distributed Application", The 2nd International Conference on Worldwide Computing and Its Applications'98 (WWCA'98), March, 1998.