MS-DOS 上での移動計算機環境の実現

慶應義塾大学環境情報学部村井研究室

渡辺 恭人

riho-m@sfc.wide.ad.jp

指導教官

村井 純

徳田 英幸

Abstract

計算機の小型高性能化とネットワークの普及により計算機の利用形態、利用環境は大きく変貌した。計算機を持ち歩いて利用する形態とコンピュータネットワークに接続する形態である。近い将来はこれら二つの形態が融合した形態、計算機をネットワークに接続しながら持ち歩くという利用形態が必要である。

しかしながら、既存のネットワークアーキテクチャは接続している計算機が移動することを考慮していないため、ネットワーク上での移動透過性を実現するためには新たなプロトコルが必要である。このようなプロトコルの一つとしてVirtual Internet Protocol(VIP)がある。

本研究は、以下のような構成となっている。第一に移動計算機環境に関して考察し、必要な概念として移動透過性を導入した。また現在のインターネットアーキテクチャに注目し、移動計算機環境を実現するために解決すべき問題点を明らかにした。その結果、インターネットアーキテクチャでのホストアドレスが位置情報を含むために、移動した場合にアドレスを付け変えられるとトランスポート層からは認識できなくなることが明らかになった。この問題点を解決したプロトコルとしてVirtual Internet Protocolに関して調査した。

第二にパーソナルコンピュータで広く使用されているオペレーティングシステムであるMS-DOS上で移動計算機環境を実現するために、VIPをMS-DOSに実装した。現在VIPは主にBSD系のUNIX実装されている。UNIX以外でもMS-DOSを使用するパーソナルコンピュータをはじめとして、コンピュータネットワークの利用の可能性が拡大していることなどを考慮すると、UNIX以外のオペレーティングシステムにVIPを実装し、整合性を検証すことは重要である。しかしながら、MS-DOSはUNIXと相違する点が数多く存在する。MS-DOSはシングルタスクシステムであることと、ネットワークの機能を自身では装備していないことである。これらを解決するために、KA9Qと呼ばれるTCP/IPのNOS(ネットワークオペレーティングシステム)を採用し、このパッケージ内にVIPを実装した。KA9Qはそれ自身でマルチタスクシステムを実現しており、UNIXのマルチタスクシステム依存部分のあるVIPを比較的に実装することが可能である。また、より容易な移動計算機環境を可能とするために移動時の手続きの簡略化を行なった。さらに、実装されたKA9Q版VIPを評価し、MS-DOS上の移動計算機環境に関する考察を行なった。

Abstract

The appearance of smaller computers with better performance and the spread of the computer network have influenced the computing environment. Today there are two types of computing environments: those that are connected to the computer network and those that can be carried around. In the near future, these two must be combined to provide a mobile computing environment with network connectivity.

However, since the current network architecture does not take into consideration free access to computer networks from portable computers, a new protocol that supports host migration transparency on networksbecomes necessary. Virtual Internet Protocol(VIP) is one of such protocols.

This research consists of two parts.

First, the mobile computing environment was described and "Host migration transparency" was introduced as a necessary concept for the mobile computing environment. Also, the problems needed to be solved were made clear. One problem is that the transport layer cannot recognize other hosts when it moves to another network and its address is changed. The Virtual Internet Protocol was investigated as a protocol that solves this problem.

Next, VIP was implemented on MS-DOS system. So far, VIP has been implemented on a series of BSD UNIX operating systems. The implementation of VIP on MS-DOS is important because recently, computer networks can be accessed with personal computers using MS-DOS.

Since VIP was originally designed on the UNIX operating system, it was important to investigate the differences between MS-DOS and UNIX before starting implementation. They are two major differences. One is that MS-DOS is a single-task operating system. The other is that MS-DOS itself has no function to access to networks. In order to solve these problems, a TCP/IP package called KA9Q was selected and VIP was implemented on it. KA9Q is a multi-task operating system so it is easy to implement VIP, which depends the on multi-task system of UNIX. Also,to make mobile computing environment easier to use, procedures for migration were simplified. Furthermore, the implementation of VIP on KA9Q was evaluated, and mobile computing environment on MS-DOS system was considered.

目 次

図目次

表目次

第 1 章

序論

1.1 はじめに

従来、計算機というものは固定された端末としての利用を前提としたものであり、それが当然であるかのように大型であった。しかし、技術革新により計算機は高性能ではるかに小型で軽量なケースに集積した、ノートブック型コンピュータと呼ばれる計算機が登場して、新種の計算機としての地位を確立した。これにより、人々は高性能でかつ小型な計算機を持ち運んで利用するようになった。他方で、計算機同士を相互に接続して情報交換する、いわゆるコンピュータネットワークが普及し、現在では広域化した。当初はUNIXのワークステーションなど一部の計算機でのみ利用可能だったネットワークの機能も現在ではパーソナルコンピュータからスーパーコンピュータまであらゆる計算機で利用可能である。このような技術を基盤として将来の計算機環境は、これら二つの利用形態が融合した形態、つまり、ネットワークに接続しながら持ち運ぶという移動利用形態が必要となるのである。しかし、現在のネットワークアーキテクチャではこのような利用形態は考慮されていないため、持ち運んでいる計算機からネットワークに接続することはできない。ネットワークに接続しながら持ち運ぶという計算機の移動利用形態を実現するためには、ネットワークアーキテクチャに修正を加える必要がある。

現在、計算機の移動利用形態をサポートするための研究は盛んに行なわれているが、それらは全て現在広く使用されているInternet Protocol(IP)を主に対象とし、それに何らかの機構を追加することで移動を実現している。「仮想ネットワーク」[1][2]という概念を、IP用に適用したプロトコルとして、Virtual Internet Protocol(VIP)[3][4][5][6][7]がある。

MS-DOS及びその互換性のあるDOSは全世界の数千万台パーソナルコンピュータ(PC)上で使用されており、最も普及しているオペレーティングシステムである。したがって、MS-DOSはパーソナルコンピュータの事実上の標準オペレーティングシステムである。その機能は、ファイルシステム、最小限度のメモリ管理である。世界最多のユーザ数を誇るこのオペレーティングシステムは、アプリケーションの数も多く、より優れたオペレーティングシステムが登場している昨今に至っても、MS-DOS環境に固執する人々は多い。そのシステムは基本的に、一度に単独のプログラムのみ実行可能というシングルタスクであり、また一人だけでの使用に限定される。

しかし、コンピュータネットワークが普及するにつれて、ネットワークの機能をPC

でも利用可能になり、対応するアプリケーションやハードウェアも安価に供給されている。オペレーティングシステムの構造上、UNIX と全て同様の機能が使用できるわけではないが、十分にネットワークを利用できるようになっている。また、PC 本体自体が小型化高性能化され持ち運んで作業を行なうことが可能になった。このため、移動した先々でネットワークに接続して端末として使用するといった利用も多く行なわれている。

また、必要最小限の機能を装備しているため、UNIX 等のマルチタスク処理を行なうオペレーティングシステムと比較して、メモリーやディスクのような記憶装置が少なくて済み、負荷の軽い実行環境が得られるのである。このため、固定型に比較してメモリやディスクの制約が大きいノートブック型といった携帯型の PC では、MS-DOS 上でネットワーク環境の移動透過性を実現することが必要である。

このように、MS-DOS 上でのネットワークの利用と移動が一般化しつつある現状では、UNIX 上で実装されている Virtual Internet Protocol(VIP) を MS-DOS 上に実装することは大いに意義があり、VIP 自体の UNIX 以外のオペレーティングシステム環境との整合性も検証可能である。

VIP は現在主に BSD 系 UNIX 上で稼働しているが、本研究ではこれをパーソナルコンピュータのオペレーティングシステムである MS-DOS 上に実装し、MS-DOS 使用の移動計算機環境について考察する。ネットワークがパーソナルコンピュータでも利用できるようになっていることを考慮すると、それらで使用される MS-DOS というオペレーティングシステム上においても計算機の移動利用環境を実現することは重要である。VIP を DOS 環境に適用させ、移動利用形態が実現できることを実証する。

1.2 本論文の構成

本論文では、第 2 章で移動計算機環境について述べ、第 3 章では、ホスト移動を実現する VIP の概要について説明する。第 4 章では、オペレーティングシステムの機能とプロトコルスタックについて述べ、第 5 章では、MS-DOS 上の TCP/IP パッケージである KA9Q NOS について述べる。第 6 章では、MS-DOS への VIP の実装について詳細を説明し、実装したものを第 7 章で評価、考察する。最後に、結論と今後の課題について述べる。

第 2 章

移動計算機環境

この章では、ネットワークに接続しながら計算機を持ち運ぶという移動利用形態について考察し、これからのネットワークアーキテクチャに必要な概念として、移動透過性 [1][2] という概念について説明する。

2.1 移動透過性

現在の計算機の利用形態は、固定と移動という二つに分類される。固定された計算機を利用する形態は従来から存在していた形態であり、ユーザはスタンドアロンで使用する場合もあれば、モデムやイーサネットなどでネットワークを介して計算機を接続して利用する場合がある。もう一つは、ノートブック型といった小型の計算機を持ち運ぶ場合で、スタンドアロンで使用したり電話回線とモデムを使用して他の計算機と接続して利用するものである。後者も実質的には前者の固定された計算機と差はない。しかし近年、計算機の技術革新は目覚しく小型化高性能化が進んだため、十分携帯可能なものが多数存在する。また、コンピュータネットワークの普及と移動をサポートする機器等の環境も整いつつあり、広域に渡る大規模な分散環境が構築されつつある。このような環境において、ユーザはネットワークに接続しながら移動する計算機の利用形態が必要である。そのような形態では、広域ネットワークのどの地点に計算機を接続しても、常に同じ計算機環境をユーザに提供するという性質、「移動透過性」[1][2] が重要となる。

2.1.1 計算機移動の定義

この節では、広域ネットワーク環境における計算機の移動を定義する。

現在普及しつつある広域ネットワークとは、多数のサブネットワークまたはローカルエリアネットワーク (それ自身もサブネットワークの相互接続によって構成される) の相互接続によって構成され、ホストは一時的にはいずれかのサブネットワークに接続されている。このような環境における計算機の移動とは、計算機があるサブネットから別のサブネットに移ることであると定義できる。すなわち、ネットワークアドレスが変化する場合を計算機の移動と定義する。従って、計算機があるサブネットワークから切断され、同じサブネットワークの別の地点へ接続した場合はネットワークアドレスは変化しないので、[1]計算機の移動とは考えない。

計算機の移動には、二つの形態がある。「on-line 移動」と「off-line 移動」である。on-line 移動とは、移動中もネットワークに接続できる形態である。これに対して、off-line 移動とは、移動中はネットワークから切断されており、スタンドアロンでの使用に限定される形態である。計算機とネットワークとの接続を確立するメディアとして、有

---

[1] OSI ではサブネット間で移動してもアドレスが変化しないこともある

線と無線がある。現在のコンピュータネットワーク環境では、有線の回線が一般的で、ローカルエリアネットワークから広域にネットワークに至るまで最も多く使用されている。これに対して、無線のネットワークは相対的にはローカルエリアネットワークなど小規模で移動範囲も狭く、利用形態も限定されている。このため現在はoff-line移動のみである。近年の無線技術の革新によって広域な無線ネットワークも登場しつつあり、これを利用するとon-line移動も可能になる。

2.1.2 移動透過性

近年計算機システムは分散環境によって提供されていることが多い。例えば、計算機上の作業がNetwork File System(NFS)などのように他の計算機上の特定のファイルやサーバプロセスに依存することがある。このような分散環境の場合、移動により自分の計算機が相手の計算機から正しく認識されなくなった場合、他の計算機のファイルやサーバプロセスにアクセスできなくなる。従って、ユーザにとっての移動透過性とは「広域ネットワークのどこに接続しても、常に同じ計算機環境が得られるという性質」である。ユーザにサービスを提供するのは、アプリケーション・エンティティ同士の通信である。移動によって計算機のアドレスが変化した場合、他の計算機からは移動した計算機を識別できず、アプリケーション・エンティティも識別できなくなる。アプリケーション・エンティティ同士が移動を意識することなく相手を識別し通信を行なうためには、アプリケーション・エンティティから移動を隠蔽する必要がある。システムにとっての移動透過性は、「移動してもアプリケーション・エンティティは移動を意識することなく、また何の変更も加えずに相手エンティティを識別して通信できること。」であると規定できる。このようにユーザにとっての移動透過性はシステムにとっての移動透過性に依存しており、下位レベルであるシステムにおいて移動透過性を実現することで上位レベルのユーザにとっての移動透過性を実現できるのである。

2.1.3 計算機移動のモデル

計算機移動には、on-line移動とoff-line移動という二つの形態が存在することを述べたが、ここでは、実際の動作をモデルを用いてこの二つの形態について説明し、それらの特性と相互の関係について考察する。

まずoff-line移動を図2.1に示す。これは移動中はネットワークにアクセスすること

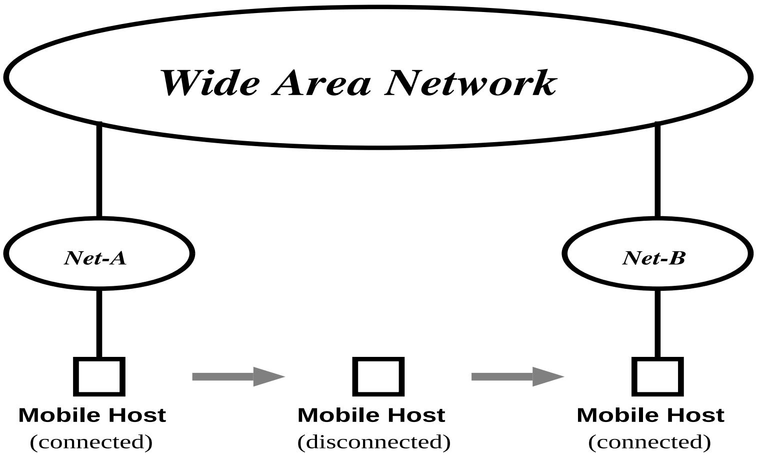

図 2.1: off-line 移動のモデル

ができない移動形態である。現状では無線のネットワークが未成熟であり普及度が低いために、この移動形態のみが主に実現可能なものである。次に on-line 移動を図 2.2

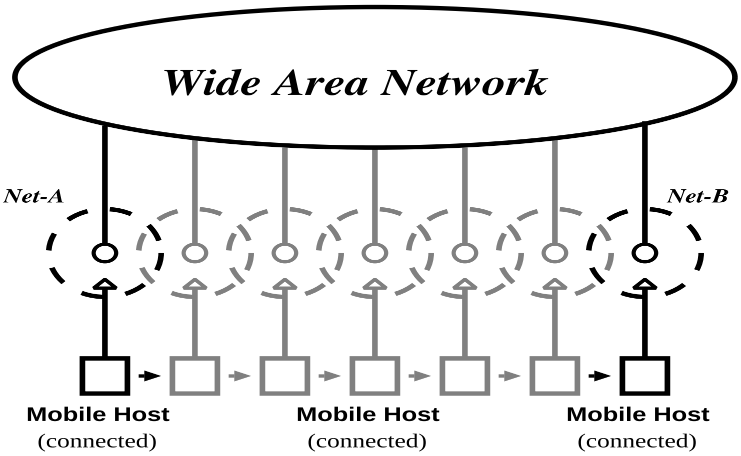

図 2.2: on-line 移動のモデル

に示す。これは移動中もネットワークにアクセス可能な移動形態である。図のように計算機は移動するたびにサブネットワークを連続的に移動していく。この移動形態は、無線ネットワークの普及によって可能になる。しかし、無線は有線と比較して安定度に問題があるため、突発的な事故や障害によって回線から切断されることが起こり得る。また、無線ネットワークでは電波の到達可能範囲が限定される。このため、無線

ネットワークがあらゆる位置に設置されても完全にカバーするのは不可能である。建築物などの障害物によっても電波の到達不能範囲は生じる。

この場合、off-line 移動がその短所を補完するべきである。回線の接続状態を検知して、アプリケーション ·エンティティに通知する機能や、アプリケーション ·エンティティでは、回線の切断、接続に対応してそれぞれのアプリケーションに依存した処理を行なうことなどが考えられる。これらにより、システムにとっての移動透過性だけでなく、ユーザにとっての移動透過性が実現される。

2.2 インターネットアーキテクチャの問題点

ネットワーク層プロトコルである IP では、ホストをインターネットアドレスまたは IP アドレスと呼ばれる 32 ビットのアドレスで特定する。IP アドレスは、ネットワークを識別する netid とネットワーク上のホストを識別する hostid の対で構成される。 netid はネットワーク全体を通じて一意であり、 hostid は netid が示すサブネットワークの中で一意である。netid はホストの識別には無関係であり、 インターネット全体では IP アドレスはホストの識別子ではなく位置情報を示すものであると考えられる。しかし、トランスポート層以上のプロトコルとの通信では IP アドレスは、相手を特定するための識別子として使用されている。

このようなアドレス形式を規定するインターネットアーキテクチャでは、相手ホストが他のサブネットワークに移動すると相手ホストの IP アドレスが変化する。したがって、相手ホストの識別することができないため、通信できない。または、アプリケーション ·エンティティでアドレスの変化を認識する必要が生じる。そのため、移動透過性は実現されていない。

ホストの位置を意識することなくホストの識別を行なうためには、ホストアドレスが位置情報を含まない識別子のみで構成される必要がある。

2.3 解決方法

前節のような問題を解決するための研究は、最近盛んになっている。現在、ネットワーク分割方式 [9]、パケット転送方式 [10] 等が提案されている。これらの方式は、Internet Protocol(IP) を基盤としているものであるが、ネットワークアーキテクチャに

関して十分な考慮をしておらず、単に移動するホストとの通信方法の提案に留まっている。一方、IBM 社ではより一般的な移動ホスト用ネットワークアーキテクチャモデルを提案している [8]。

数多く登場しているこれらの研究に対して、IETF[2]の MobileIP-WG ではプロトコルの一本化と勧告案の作成を行なっている。そのドラフトでのプロトコル案は、現在のインターネットへの影響を極力抑えつつ、移動ホストと通信できることに重点が置かれているため、さまざまな欠点がある。[11][12]

本研究では、仮想ネットワーク [1][2] の概念を IP に適用することで移動透過性を実現している Virtual Internet Protocol(VIP)[3][4][5] [6][7] を採用し、このプロトコルの実装を行なう。

---

[2] Internet Engineering Task Force の略。インターネット上での標準プロトコルを作成する組織で、多数のワーキンググループ (WG) に分かれて活動している

第 3 章

Virtual Internet Protocol(VIP) の概要

3.1 仮想ネットワーク (Virtual Network)

ネットワーク層において移動透過性を実現するために「仮想ネットワーク」[1][2] という概念を導入する。仮想ネットワークとは既存のネットワーク (「物理ネットワーク」) より上位に位置付けられる論理的なネットワークである。トランスポート層からは仮想ネットワークのみが見える。各ホストは、それぞれ一つの物理ネットワークに接続されると同時に一つの仮想ネットワークにも接続される。物理ネットワーク上でホストが移動しても、仮想ネットワークではホストは移動しない。すなわち、各ホストは常に一つの仮想ネットワークに接続されていると規定するのである。このようなネットワークをそのホストの「固有ネットワーク」と規定する。

この場合、各ホストは、物理ネットワークでのアドレス (物理ネットワークアドレス) と仮想ネットワークでのアドレス (仮想ネットワークアドレス) を持つ。仮想ネットワークアドレスは移動しても変化しないアドレスで、位置情報を持たない完全なホストの識別子である。物理ネットワークアドレスはネットワーク上のホストの位置を表し、パケットの経路制御に使用される。トランスポート層からは相手がどこに移動しても仮想ネットワークアドレスで相手を指定できる。基本的にはトランスポート層から物理ネットワークアドレスは見えない。

プロトコル階層は、図 3.1に示すように、ネットワーク層を二つに分割して上半分を仮想ネットワーク副層、下半分を既存のネットワーク副層 (物理ネットワーク層) とする。実際の動作では、仮想ネットワーク層は、トランスポート層から相手の識別子である仮想ネットワークアドレスを受けとり、それを実際のホストの位置を表す物理ネットワークアドレスに変換して既存のネットワーク層に渡すことが必要になる。

アドレス変換を行なうには、ホストの識別子 (仮想ネットワークアドレス) を鍵として位置情報 (物理ネットワークアドレス) を得るための表が必要となる。このため、アドレス変換テーブル (Address Mapping Table) を導入する。また、この表の管理のために拡散キャッシュ法を導入する。これらについては後の節で説明する。

3.2 Virtual Internet Protocol の概要

前節で説明した仮想ネットワークの概念を IP に適用した、Virtual Internet Protocol(VIP) [3][4][5][6] [7] の概要について述べる。

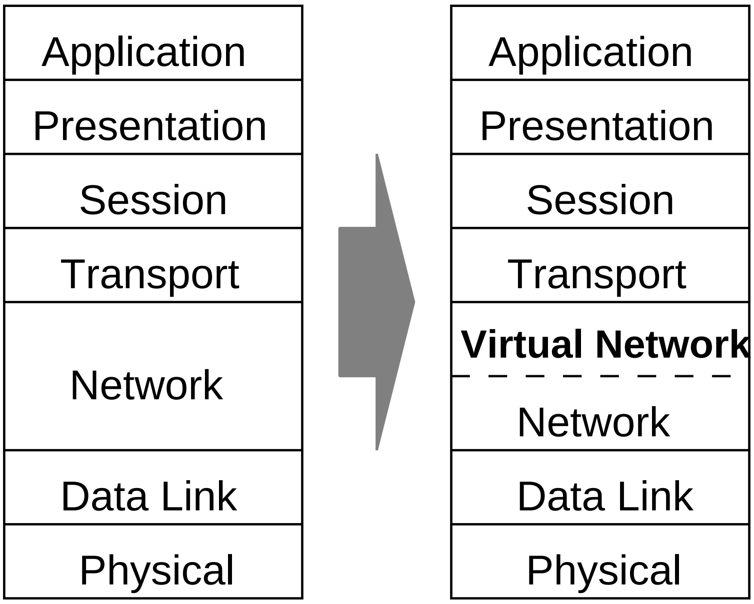

図 3.1: OSI7 層モデルでのネットワーク層の分割

3.2.1 仮想 Internet アドレス (VIP アドレス)

仮想ネットワークの概念を導入するために、ノードの識別子として新たに仮想ネットワークアドレス (VIP アドレス) を導入する。Internet Protocol に導入する際 IP アドレスとの変換を容易にするため、IP アドレスと同様に 32 ビット値とする。これは、IP の上位層である TCP や UDP は下層が IP であることを前提としているため、下層に対して IP アドレスを渡すからである。

3.2.2 Address Mapping Table

仮想 Internet Protocol 層では、ホストの識別子である仮想 Internet と位置情報である IP アドレス変換するに必要がある。そのために、各ホスト、ルータでは、VIP アドレスを鍵として IP アドレスを得るための表、アドレス変換テーブル (Address Mapping Table(AMT)) を保持する必要がある。

Internet に接続される全てのホストは膨大な数であり、これら全てのアドレス変換表を持つことは不可能である。このため VIP では各ホストがアドレス変換表の一部を保持することにし、インターネット全体としては、全てのアドレス変換表は分散して

存在する。

3.2.3 拡散キャッシュ法

膨大なアドレス変換表 (AMT) を分散して保持するために、拡散キャッシュ法を使用する。この方法では、各ホスト、ルータはアドレス変換のためのキャッシュを保持する。これは、ネットワーク上での通信の際のパケットによってマップを拡散させる方法である。ルータでは、パケットの転送の際にそのパケットの始点情報を使って自分のキャッシュを更新する。つまり、そのパケットを送信したホストに対するエントリを更新、追加するのである。

3.2.4 Virtual Internet Protocol のプロトコル仕様

実装すべき VIP のプロトコル仕様について説明する。ここでは、改良された VIP[5][7] を取り上げ、そのプロトコル仕様について説明する。

これまでの VIP では、IP との互換性、孤立した AMT エントリの削除、固有ネットワークのルータ上の AMT エントリなどに関して問題があった。これらを解決し VIP をより実用的なものにしている。

1. 仮想 Internet アドレス (VIP アドレス)

   VIP アドレスは、相手を特定するという性質上ホスト毎に付けるアドレスである。ホームネットワーク (home network) では VIP アドレスは IP アドレスに等しい。

2. ヘッダフォーマット

   VIP のパケットヘッダのフォーマットを図 3.2に示す。データパケットとコントロールパケットの 2 種類がある。

   (a) VIP 層の実現

   VIP 層の実現には、VIP 層を独立した層に設ける方法ではなく、IP のオプションとして挿入する方法を選択している。

   (b) それぞれのフィールドについて

   ver:VIP のバージョン。現在は 0。

   type:VIP パケットのタイプ。タイプには 3 種類ある。

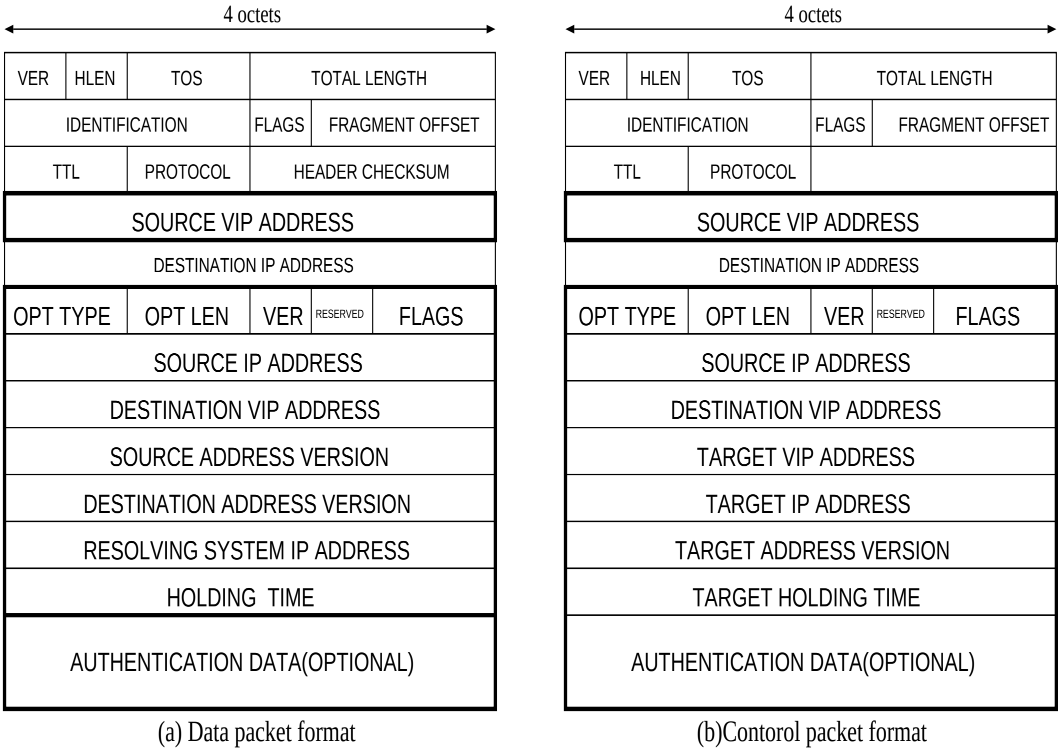

図 3.2: VIP パケットのヘッダフォーマット

- *VipData* .. .. 通常のデータ
- *VipCAMT* .. .. ターゲットに関する AMT エントリを作成、更新する。(コントロールパケットのタイプ)
- *VipInvAMT* .. .. ターゲットに関する AMT エントリに無効フラグを立てる。(コントロールパケットのタイプ)

flags .. .. パケットの取り扱いを制御するフラグで現在次の 4 つが定義されている。

- *Data/Ctrl* .. .. パケットがデータパケットかコントロールパケットかを示す。
- *DtCacheIR* .. .. 途中のルータでの AMT エントリの更新、作成の禁止を示す。
- *DtRslvIR* .. .. 途中のルータでのアドレスの対応づけの禁止を示す。
- *BcastInvAmt* .. .. 途中のルータが VipDAmt パケットを受けとった時

のブロードキャストの抑制を示す。

source VIP address .. .. パケット送信者のVIPアドレス

destination VIP address パケット受信者のVIPアドレス

source address version .. .. 送信者のVIPアドレスとIPアドレスの組のバージョン番号

destination address version .. .. 受信者のIPアドレスとVIPアドレスの組のバージョン番号

resolving system IP address .. .. アドレス対応付けを行なったシステムのIPアドレス。

holding time .. .. パケットによって作成または更新されるAMTエントリを保持する時間(秒)

target VIP address .. .. コントロールパケットが対象とするホストのVIPアドレス。(コントロールパケットのみ)

target IP address .. .. コントロールパケットが対象とするホストのIPアドレス。(コントロールパケットのみ)

target address version .. .. コントロールパケットが対象とするホストのアドレス対が作成された歴時間。(コントロールパケットのみ)

target holding time .. .. コントロールパケットが対象とするホストを鍵とするAMTエントリを保持する時間。(コントロールパケットのみ)

(c) AMTエントリ

VIP address .. .. VIPアドレス。検索の鍵として使用される。

IP address .. .. IPアドレス。仮想Internetアドレスを鍵として検索された結果、対応して得られる。

version .. .. VIPアドレスとIPアドレスの組が作成された歴時間。

timer .. .. エントリのタイムアウトを管理するためのタイマ。このタイマによりタイムアウトのイベントが発生した時、エントリが削除される。無効フラグが立っていない時のみ有効。

holding time .. .. 無効フラグが立ってから動作しはじめるタイマ。タイムアウトするとAMTエントリは削除される。

flags .. .. テーブルを管理するフラグ。

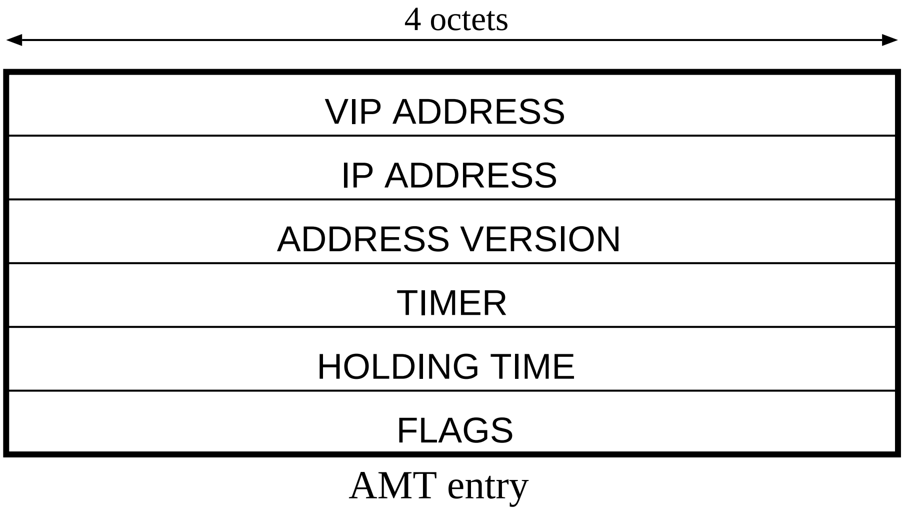

図 3.3: AMT エントリ

- $AmtPerm$ .. .. エントリがタイムアウトしないことを示す。
- $AmtInv$ .. .. 無効フラグ。エントリがすでに古くなっていることを示す。
- $AmtInUse$ .. .. エントリが使用中である。

第 4 章

オペレーティングシステムの機能とプロトコルスタック

本章では、ネットワーク機能を実現する場合にオペレーティングシステムに必要な機能と実装構造について考察する。

4.1 ネットワークの実装階層

ネットワークの実装階層は、図4.1のように大きく3つの層に分かれる。

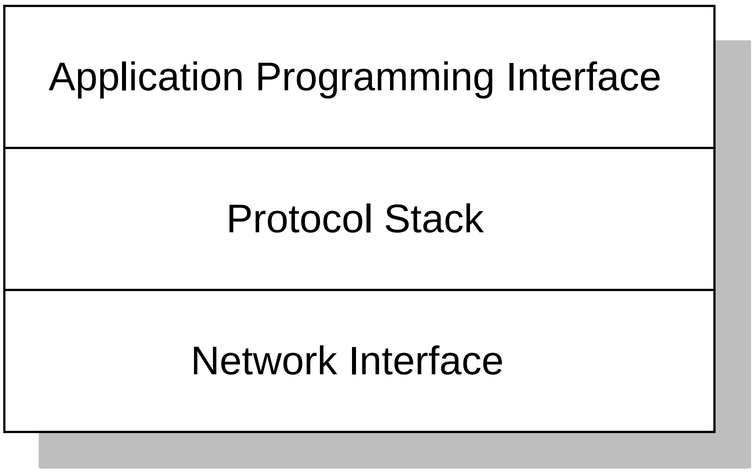

図 4.1: ネットワークの実装階層

ネットワークインタフェース層では、データ送受信のためのネットワークハードウェアへの一貫したインタフェースを提供する。プロトコルスタック層では、TCP/IPなどのプロトコルスタックが構築される。アプリケーションプログラミングインタフェース層では、アプリケーションプログラム作成に必要な構造的なインタフェースが提供される。また、ネットワーク通信に必要なメモリの大きさは激しく変動するため、専用のメモリ管理を使用して効率的なメモリの割当と解放が行なわれるべきである。

4.1.1 UNIX(4.3BSD)

UNIXでのネットワーク機能はネットワークインタフェースとプロトコルスタックをカーネル内部に、ユーザレベルからはソケットを使用してプロトコルにデータを受け渡して通信を行なう。

メモリ管理には、mbufと呼ばれるデータの構造体を使用する。これにより、大きさの決まった構造体や大きさの変化する構造体の要求するメモリに対応することができる。

また、UNIXでは非同期のネットワーク処理(ネットワークインタフェース層から

ネットワークプロトコルでの処理) のスケジューリングにソフトウェア割り込みを使用し、プロトコルを独立したプロセスとしては実行させない。

4.1.2 MS-DOS

MS-DOS には本来、ネットワークの機能が含まれておらず、シングルタスクの小型オペレーティングシステムである。このような場合、ネットワークインタフェースのドライバが用意される。例えば、パケットドライバ[1]、ODI[2]、NDIS[3]などである。これらは、デバイスドライバまたは常駐型プログラムとして組み込まれる。

ネットワークインタフェースドライバの上位のプロトコルスタックとアプリケーションプログラミングインタフェースはそれぞれ必要な部分だけがプログラムに組み込まれ、それぞれ個々の独立した実行ファイルとして存在することが多い。これは、MS-DOS がシングルタスクシステムであることと、メモリに制約があるためである。この場合、個々のアプリケーション内部でスケジューリングを行なうが、一度にそのアプリケーションのみの実行が保証される。

プロトコルの実装の場合にはプロトコルスタックをプリミティブな形で用意する必要がある。

4.2 VIP の機能と実装構造

第 3 章で述べたように、VIP はネットワーク層を分割して既存のネットワーク層とトランスポート層の間に VIP 副層を設けている。しかし、新たな層としてではなく IP オプションとして実装される。以下に VIP の機能と実装構造をまとめた。

4.2.1 VIP ヘッダ

- IP アドレスと VIP アドレスという二重のアドレスを持つ。
- VIP のヘッダは IP のオプションとして挿入される。
- 入力時.. .. VIP ヘッダのアドレス検査 ·入れ換え

---

[1] FTP Software 社、Clarkson 大学

[2] Oprn Datalink Interface、Novell 標準

[3] Network Driver Interface Spaecification、3COM 社、Microsoft 社

- 出力時.. .. VIP ヘッダ挿入

4.2.2 AMT

- 各移動ホストで VIP アドレスと IP アドレスの表 (AMT) を保持する。
- 入力・出力されるパケットの VIP アドレスを AMT を使ってマップし、AMT を更新する。

4.2.3 コントロールパケット

各移動ホストは、一定間隔でホームネットワークにコントロールパケットを送信して VIP アドレスと IP アドレスの組を通知する。

4.3 MS-DOS への実装

VIP は、IP のプリミティブな部分を修正する必要があるためアプリケーションのプログラミングインタフェースではなく、TCP/IP プロトコルスタックのプリミティブな実装を必要とする。また、定期的なコントロールパケットの送信を行なうためには、複数のプロセスが並行して実行されるような環境が必要となる。

KA9Q は、ネットワークインタフェースドライバにパケットドライバを利用した TCP/IP パッケージである。ネットワークインタフェース層以上では TCP/IP プロトコルスタックがプリミティブに実装されており、カーネルを修正して再構築することが可能である。疑似的ながらマルチタスクシステムを実装しており、複数のプロセスを並行して実行可能である。メモリ管理にシンプルな mbuf 構造体を使用している。このため、メモリの割当と解放が効率的に行なえる。但し KA9Q は、簡易オペレーティングシステムの機能、TCP/IP プロトコルスタック、ソケットによるアプリケーションのプログラミングインタフェースを一つの実行ファイルで実現しているため、メモリを大量に消費するという欠点がある。

第 5 章

KA9Q(KA9Q NOSパッケージ)の概要

この章ではKA9Qの概要について説明する。

5.1 MS-DOS でのネットワーク機能実装上の問題点と解決法

5.1.1 MS-DOS の構造上の問題点

MS-DOS はシングルタスクシステムである。また、MS-DOS にはネットワーク関係の処理を行なう機構は含まれておらず、物理層のデバイスドライバ、データリンク層のネットワークインタフェースソフトウェア (例えば FTP software 社のパケットドライバ) が必要である。アプリケーションはその上で稼働する。

VIP も含めてネットワーク関係のプログラムは、UNIX で設計実装されたものが多い。故にマルチタスクシステムに依存した部分が存在する。また UNIX で利用されているデーモン等をそのまま実装することはできず、割り込みなどにより解決することになる。しかし、この解決法は、プログラムへの負担部分が大きく、汎用性にも欠ける。

MS-DOS 上でネットワークのシステムを構築する場合、このような制約を考慮すると以下のような解決法がある。

1. サーバ専用、クライアント専用のホストに分けてネットワークシステムを構築する。
2. (疑似) マルチタスク ·システムになるようなプログラムを走らせ、その上ですべてを実行する。

5.1.2 KA9Q による解決法

前節で説明した二つの解決法のうち、後者を選択する場合を考える。このような要求は、KA9Q というパッケージを使用することで解決される。KA9Q は MS-DOS を使用する IBM-PC とその互換機上での MS-DOS、Macintosh、SystemV 系 UNIX、HP-UX、 ATARI などで動作する TCP/IP のパッケージであり、それらのバイナリとソースを含んでいる。これは Phil Karn が中心となって骨格部分が作成され、アマチュア無線でのパケット通信ネットワークで盛んに使用されているものである。そして、さまざまな人々によって核となるプロトコルは精力的に試験され、実際のネットワークを構築するための頑丈なプラットフォームとなっている。また、最新のアプリケーションやプロトコルが次々にパッケージに追加されている。更に、ソフトウェアのポータ

ビリティが増加し、より多くのハードウェアインタフェースをサポートされ、多数のネットワーク技術(非同期、RS-232C、Ethernet、各種無線パケットなど)が利用できるようになっている。

また、プログラミングのインタフェースがより簡潔な形で実装されており、新たなプロトコルやアプリケーションの実装や実験を行なうことができる。

KA9Qは、それ自身で簡易的なオペレーティングシステムとして機能する。そして、最大の特徴はカーネル内部でシングルプロセスマルチスレッド型のプロセス制御を行なっていることである。これにより、複数のプログラムをメモリが許す限り並行して実行することができ、MS-DOSではできないデーモンを利用したプログラミングが容易になる。また、複数のネットワーク・セッションを並行して実行可能なので、ユーザは効率的な利用環境を得ることができる。

5.2 KA9Qでのサービスの提供

KA9Qは基本的なTCP/IPファミリのプロトコルをサポートしている。それらは、IP、ICMP、TCP、UDP、FTP、TELNET、SNMPである。また、アドレス決定プロトコルであるARPをEthernetとAX.25に利用することができる。

また、KA9QのユーザによってもKA9Qのサービスや、EthernetやSLIP以外のアマチュア無線のパケット通信用ハードウェアへの対応ドライバも増加し、より汎用性が増加している。現在の対応を表5.1に示す。

5.3 KA9Qの構造

KA9Qパッケージでは図5.1のような構造でネットワークが実装されている。

MS-DOSのカーネルにはネットワーク関係のコードが含まれていない。そのため、各々のハードウェアに対応したデバイスドライバとその上層にネットワークのプログラムを実装するためのプログラムインタフェースを提供するドライバが必要となる。KA9Qパッケージには、FTP Software社の”PC/TCP Version1.08パケットドライバの仕様”と互換性のあるパケットドライバが提供されている、これは、データリンクレベルでネットワークインタフェースが使用できるプログラム用のインタフェースである。

Contents	Contents
SM0RGV mailbox server	ARCnet via PACKET driver
NNTP Netnews client	KISS TNC code
FTP server/client	High speed (56kbps) modem driver
Finger server/client	Hamilton Area Packet Network driver code
SMTP server/client	Eagle card driver
RIP routing protocol	PI card driver
IP path tracing command (hopcheck)	FTP Software's Packet Driver interface
SLIP redial code	PAC-COM PC-100 driver code
NET/ROM async interface	Appletalk interface (Macintosh)
NET/ROM network support	DRSI PCPA slow-speed driver
LZW-compressed sockets	PE1CHL generic scc driver
SLIP(Serial line IP on built-in ports)	Asynch driver code
PPP(Point-to-Point Protocol code)	SLFP packet driver class supported
BOOTP(Boot Strap Protocol) server/client	CDMA mobile DM interface
Van Jacobson TCP compression for SLIP	CDMA QTSO data interface

表 5.1: KA9Q Version 930622 で提供されるソフトウェアオプションとドライバ群

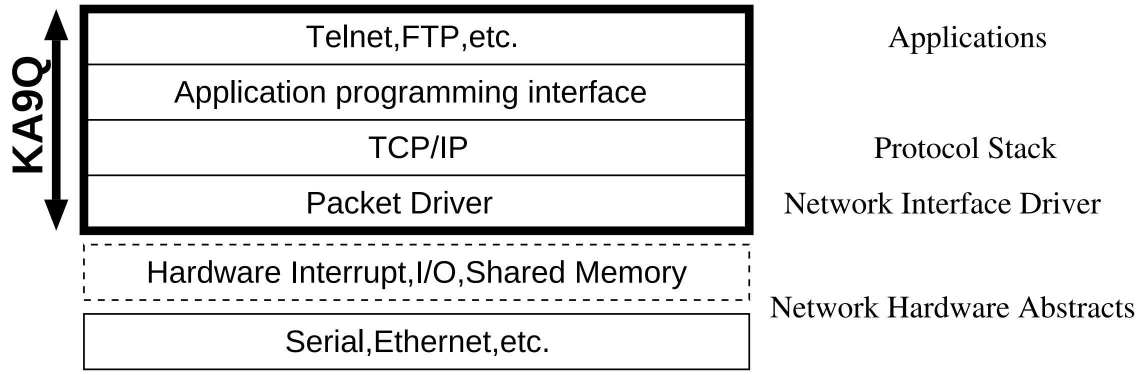

図 5.1: KA9Q のネットワーク構造

FTP Software 社は、各社のネットワーク用のインタフェースドライバの統一を呼びかけ、仕様を定め公開した。この仕様公開により多くのパケットドライバが作られた。また、Clarkson 大学の Russ Nelson 氏 (現 Crynwr Software) が中心となって、フリーのドライバが多く作られた。

このパケットドライバの仕様の利点は PC でのネットワーク用インタフェースの製造業者 (Ethernet ボードなどの) が各種ネットワークのアプリケーションで使用するために導入できる一つのドライバを書けば良いことである。パケットドライバを使用しない独特なハードウェアはそれぞれ固有のドライバが必要となる。パケットドライバは、データリンクレベルでネットワークインタフェースを複数のアプリケーションによって共有できるプログラム用のインタフェースを提供している。この仕様に基づくプロトコルの実装は、完全にどの PC 上においても共存できる。また、パケットドライバを使用することによって、TCP/IP やその他のプロトコルを実装できる。

5.4　KA9Q でのネットワーク実装

この節では、VIP を実装するに当たって KA9Q でのネットワークの実装、特に IP 関係の実装、処理の流れについて説明する。(図 5.2参照)

KA9Q では、二つのデーモンプロセスと二つのキューによって処理が成立している。まず、ネットワークインタフェースがパケットを受信すると割り込みによって `pkint` が呼び出され、受信パケットはキューに挿入される。また、上層から送信されるべきパケットは、 `ip_send` にて IP ヘッダを付加されて IP パケットとなり、同じくキューに挿入される。ここで `network` という名のデーモンプロセスはキューを監視しており、パケットを取り出す。パケットがキューに入力されたことがわかると、 `network` はそれがどのようなパケットか調べ、IP パケットであれば、 `ip_proc` を通じて `ip_route` に渡される。 Ethernet のパケットであれば、Ethernet ヘッダを取り除き IP パケット化して `ip_route` へ渡す。 `ip_route` では、渡された IP パケットの終点アドレスをチェックし、自分宛ならば IP ヘッダを取り除いて上層へ渡し、そうでなければ、出力インタフェース用のキューにパケットを渡す。そこでもう一つのデーモンプロセス、`ethernet_tx` がキューを監視して、パケットを取り出す。この場合は Ethernet に送信するために、 `enet_send` で Ethernet アドレスの対応づけを行なう。`enet_output` で Ethernet ヘッダを付加し、それを `pk_raw` でパケットドライバに渡す。最後にパケット

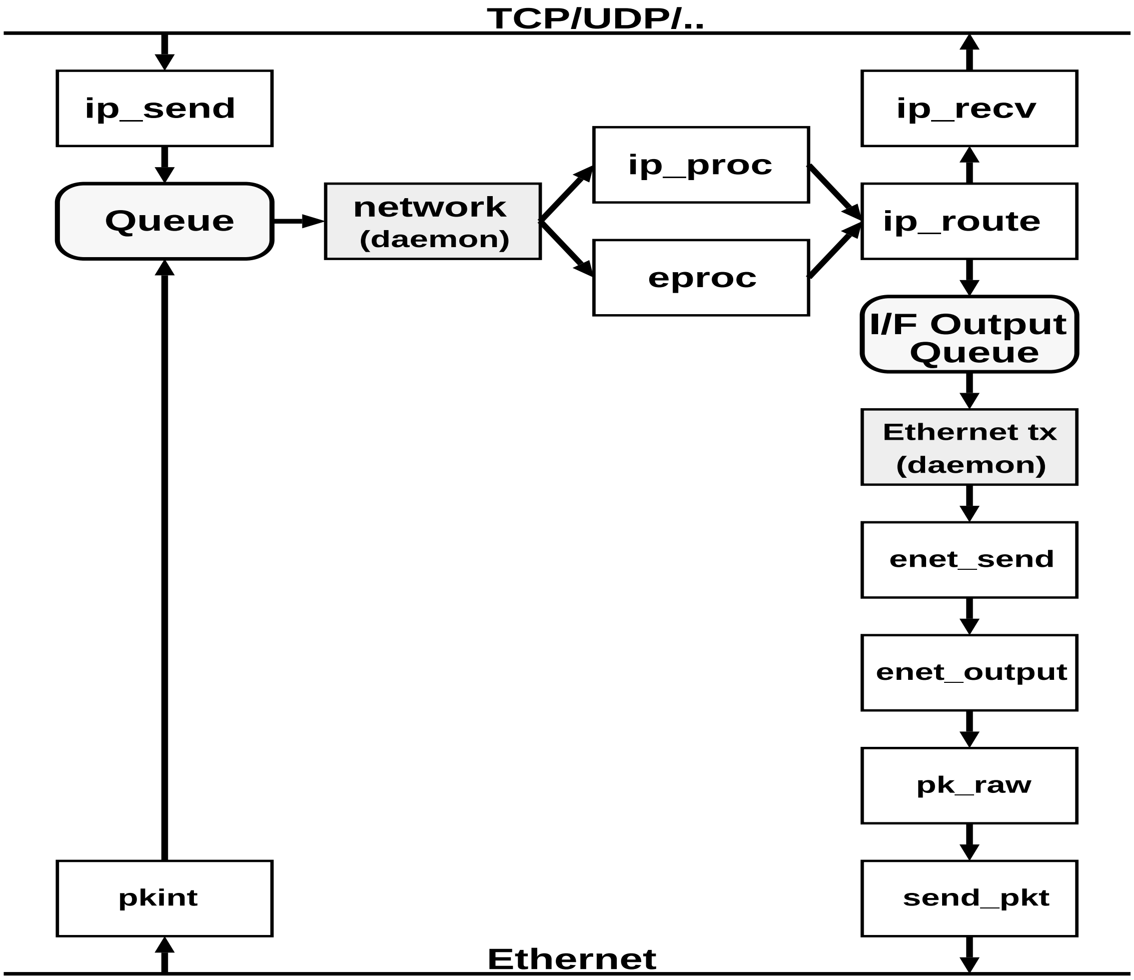

図 5.2: KA9Q の IP 関係ブロックダイアグラム (Ethernet の場合)

ドライバの送信ルーチン `pk_send` より送信される。受信・中継・送信を簡単にまとめると以下のようになる。

5.4.1 受信時

1. 割り込みにより受信したパケットはキューに保持される。(pkint())
2. キューからパケットを取り出し、パケットを見る。(network())
3. Ethernet のパケットなので、Ethernet のヘッダを取り外して、IP のパケットにする。(eproc())

4. 終点アドレスをチェックする。(ip_route())

5. 自分宛なので上層へ送る。(ip_recv())

5.4.2 送信時

1. 上層から IP パケットを渡され、キューに送る。(ip_send())

2. キューからパケットを取り出し、パケットを見る。(network())

3. 自分から出された IP パケットである。(ip_proc() から ip_route() へ)

4. 自分宛ではないので出力用キューに挿入する。(ip_route())

5. ethernet_tx デーモンがキューからパケットを取りだし、下層へ送る。(enet_send() → enet_output() → pk_raw() → send_pkt())

5.4.3 中継時

1. 受信したパケットはそのままキューに保持される。(pkint())

2. キューからパケットを取り出し、パケットを見る。(network())

3. Ethernet のパケットなので、Ethernet のヘッダを取り外して、IP のパケットにする。(eproc())

4. 自分宛かどうか見る。(ip_route())

5. ethernet_tx デーモンがキューからパケットを取りだし、下層へ送る。(enet_send() → enet_output() → pk_raw() → send_pkt())

5.5 KA9Q の使用例

前節までに説明した KA9Q(Version 930622) の実際の使用例を紹介する。

5.5.1 KA9Q の実行

KA9Q は一つの実行ファイルに全ての機能が含まれているため、実行するときは、MS-DOS プロンプトから、`net.exe` と入力するだけで良く、図 5.3のような表示とプロンプトが出て入力待ちとなる。

```
C: net
KA9Q NOS version 930622(KA9Q)
Copyright 1986-1993 by Phil Karn, KA9Q


net> _
```

図 5.3: KA9Q 起動時

5.5.2 ネットワークの設定

UNIX でのネットワークの設定用コマンドと全く同じか類似のコマンドが用意されており、簡単である。図 5.4にその一例を示す。

1 行目では、ネットワークインタフェースのドライバの設定を行なっている。ここでは使用するドライバがパケットドライバでそのインタフェース名は pe0 とする。また、使用するバッファの大きさが 8192 バイト、MTU[1]は 1024 バイトとする。

2 行目では、ホスト名の設定を行なっている。

3、4 行目では、ネットワークインタフェースの設定を行ない、インタフェースの IP アドレス、サブネットワークマスク、ブロードキャストアドレスを設定する。

5～10 行目は、`ifconfig` コマンドでインタフェース pe0 の設定内容を表示している。MTU は 1024 バイト、パケットのトレースを行なっている場合のフラグ、サブネットワークマスク、ブロードキャストアドレスの値をそれぞれ表示する。

11 行目は、経路のエントリに記述がなかった IP アドレスを終点に持つパケットが制御されるべき経路を記述する。デフォルトルート。

---

[1] 与えられた物理ネットワーク (イーサネットなど) を通じて転送できるデータの最大のかたまり

```
 1:net> attach packet 60 pe0 8192 1024
 2:net> hostname foobar
 3:net> ifconfig pe0 ip 133.138.192.6 netmask ffffff00
 4: broadcast 133.138.192.255
 5:net> ifconfig pe0
 6:pe0 IP addr 133.138.192.6 MTU 1024 Link encap Ethernet
 7: Link addr 00:80:c7:0d:08:73
 8: trace 0x0 netmask 0xffffff00 broadcast 133.138.192.255
 9: sent: ip 5 tot 5 idle 0:00:04:30 qlen 0
10: recv: ip 0 tot 0 idle 0:00:04:53
11:net> route add pe0 133.138.192.1
12:net> start ftp
13:net> start echo
14:net> start rip
15:net> start finger
```

(左端は便宜上の行番号)

図 5.4: ネットワークの設定例

12～15行目では、各サーバを起動する。これらはバックグラウンドプロセスとして動作する。

このような設定のうち、KA9Q起動と同時に動作させたい設定は、`autoexec.net`ファイルに記述することができる。これにより、起動時に同じコマンドを繰返し入力する必要はなくなる。

5.5.3 セッションの状態と切り変え

1. 起動された(ネットワーク)セッションの状態を見る
   起動されたセッションの状態は、`session`コマンドで知ることができる。起動したセッションは、コマンドを入力するコンソールとは別のスクリーンに表示されているので、F10キーを押してコマンドモードにする。(図5.5)

```
net> telnet 133.138.191.3 7
net> session
 #  S#  Snd-Q State     Remote        socket    Command
*1  8289    0 Estab     133.138.191.3:7 telnet 133.138.191.3
```

*は、現在のセッションを示す。

図 5.5: `session` コマンドによるセッションの状態表示

2. 起動されたセッションの切替え
   複数起動されている場合、コマンド `session [ session # ]` で切替えることができる。(キーでは、F8 キーと F9 キーで切替える。)

3. 起動されたセッションの強制終了
   コマンド `close [ session # ]` で、FIN を送ってネットワークの接続を終了する。(または、control-c)

5.5.4 パケットのモニタリング

KA9Q では特定のインタフェースに流れるパケットをディスプレイ画面上に表示することができる。例えば、`net> trace pe0 011` とすると、パケットのヘッダのみで入力側出力側ともにモニタすることになる。

5.5.5 IP パケットの経路検査

UNIX で使われている `traceroute` と同様なもので `hop check [IP address]` というコマンドにより与えられたアドレスまでの経路を検査する。

5.6 まとめ

以上のように、KA9Q は MS-DOS 上の TCP/IP パッケージとしては実装基盤として十分なソフトウェアであると言える。フリーソフトウェアでありながら UNIX のネットワーク機能と同等のものを実装されており、また多くの人の手によってテストされ

ることにより、強化され、安定化されているからである。早くから無線で使用されてきていることもあって、無線上の経路制御プロトコル (例えば、RSPF–Radio Shortest Path First) などのソースも存在する。

したがって、現在のところ MS-DOS 上にプロトコルを実装するには、KA9Q が最も適当であると言える。

第 6 章

Virtual Internet Protocol(VIP) の MS-DOS への実装

本章では、IP ホストの移動をサポートする通信プロトコルである Virtual Internet Protocol(VIP) の MS-DOS(KA9Q) への実装について説明する。

6.1 MS-DOS(KA9Q) への実装上の問題点及び設計目標

6.1.1 問題点と設計目標

1. **IP パケットから VIP パケットへの変換**

   MS-DOS での計算機環境を考える場合、MS-DOS 上のホストは、ネットワーク上でルータのような重要なホストとしては必ずしも使用されない。よって、移動計算機環境においても MS-DOS はルータの機能を持つ必要はないといえる。VIP におけるルータの機能は、『中継すべきパケットが普通の IP パケットでそのパケットを受信すべきホストに対する AMT エントリを持っていた場合、そのパケットを VIP パケットに変換してから中継する。』ことである。現段階では KA9Q には必要ない。また、図 5.2のように、KA9Q では受信されるパケットも送信されるパケットも一度キューに保持され、次に `ip_route` に渡され、そこで下層か上層かの方向付けが決定される。このため、IP パケットの中継に対応するモジュールは必要ない。

2. **timer driven** なモジュール VIP では、移動ホストがホームネットワークに対して一定間隔で *VipCAMT* を送信するためのタイマによって動作するモジュール `vip_slowtimo` がある。 これを KA9Q に実装する場合デーモンプロセスにする必要があるが、KA9Q を利用することでデーモンプロセスが容易に使用可能である。

3. 移動後のアドレスなど設定情報の割り当て

   VIP では、ホストの識別子である VIP アドレスは、不変である。一方、IP アドレスは、ホストが移動して別のサブネットワークに接続した場合、そのネットワークから新たにアドレスを割り当ててもらう必要がある。その際、ユーザが `ifconfig` のようなネットワークインタフェースを設定するコマンドを使用して自分で書き換えることもできるが、移動する度に手作業で書き換えるのは非効率である。よって、移動後の手続きはより単純で、自動化するべきである。

6.1.2 解決法

1. timer によって一定間隔で呼び出されるデーモンプロセスとなるモジュールの作成

- 起動時に実行するデーモンプロセスのテーブルに登録する。(名前, スタックサイズ, 関数へのポインタ)
- デーモンプロセスにするモジュールを図6.1のように書く。

```
vipslowtimo()
{
while(1){
pause(1000L);/* 1000msec interval */
.................................
        }
}
```

図 6.1: timer driven デーモンプロセスとするモジュールの記述例

`pause()` は、カーネル内の `timerproc` によって管理され、1000msec のタイマが 0 になるまではラウンドロビンで別のデーモンプロセスを実行する。

2. 移動時のホスト設定情報の取得

   これを自動的に行なうために Bootstrap Protocol (BOOTP)[14] や Dynamic Host Configuretion Protocol(動的ホスト設定プロトコル)[15][16][17] といった計算機の自動設定用プロトコルを使用する必要がある。 KA9Q には、BOOTP が実装されているため、この BOOTP のクライアントのみを利用する。新たな接続先で、ユーザが指定されたファンクションキーを押すと、 BOOTP クライアントが起動し、IP アドレス、ネットマスク、ブロードキャストアドレス、デフォルトゲートウェイアドレスを取得する。当然ながら、そのネットワークに BOOTP あるいは DHCP のサーバが存在する必要がある。

6.2 KA9Q 版 VIP における処理系の構成

今回実装した VIP の処理系のブロックダイアグラムを図 6.2に示す。太矢印線で表した部分が今回追加した部分である。

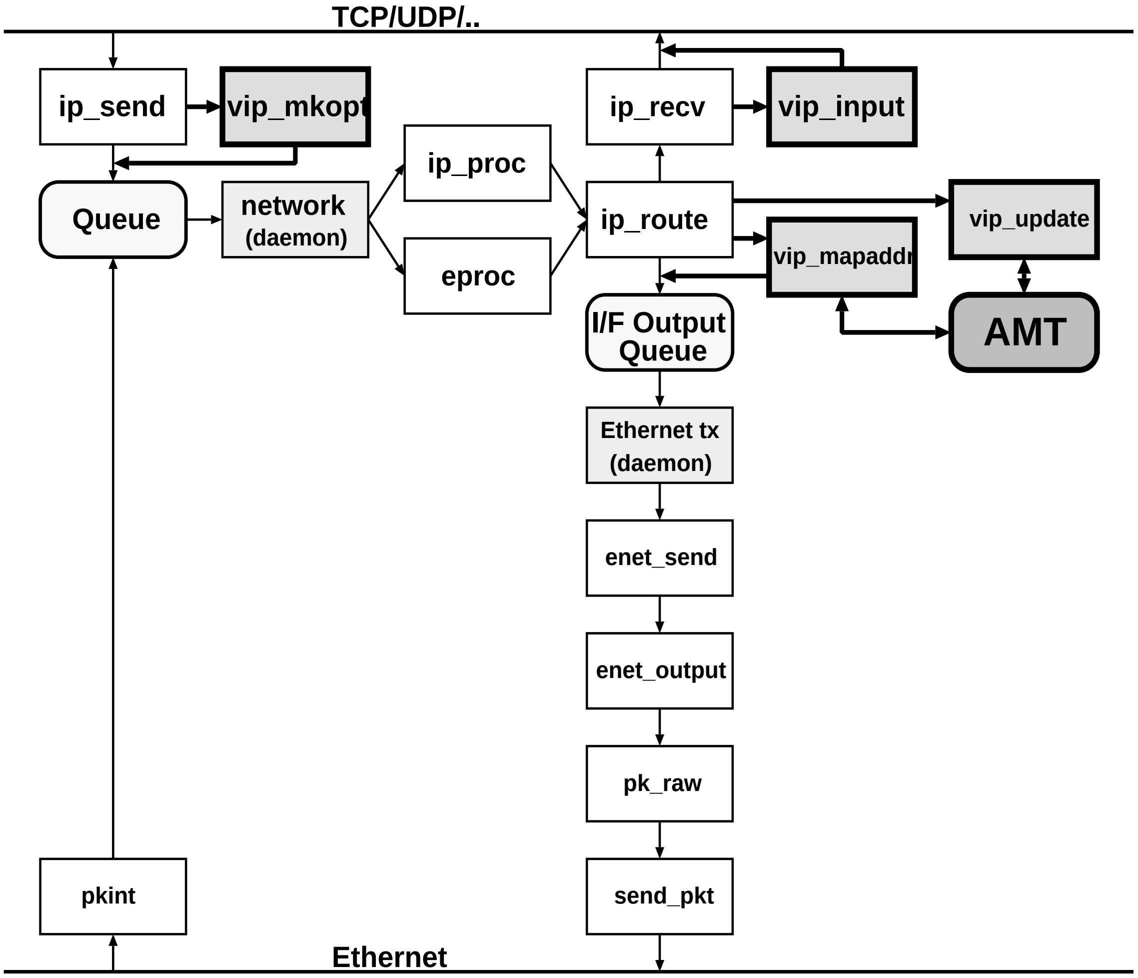

図 6.2: VIP 実装後の KA9Q のブロックダイアグラム

6.2.1 送信時

送信時には、TCP/UDP 層から ip_send が呼ばれ、その中で vip_mkopt を呼び VIP オプションを作成して、IP ヘッダに付加する。そして、IP パケットはキューに入れられる。その後、キューから出されたパケットは、ip_route で、 vip_mapaddr が呼ばれ受信ホストのアドレスの対応付けが行なわれる。アドレスの対応付け後は、パケットを送出するインタフェースが決定され、 VIP オプションの送信側 IP アドレスを設定する。その後、対応するインタフェースの send に渡され (この場合は enet_send)、対応するインタフェースの output(この場合は enet_output) でヘッダを付けられ、pk_raw に送られる。最後にパケットドライバの送出ルーチン send_pkt から送出される。

6.2.2 受信時

インタフェースがパケットを受信すると割り込みが入り、`pkint` が呼び出され、パケットはキューに入れられる。キューから出されたパケットは、`ip_route` で、`vip_update` が呼ばれ送信ホストの VIP アドレスを鍵とする AMT エントリを更新し、IP アドレスの確認を行ない、自分宛のパケットならば `vip_input` を呼び出す。`vip_input` では、VIP アドレスの確認と、そのパケットがコントロールパケットかデータパケットかの確認を行ない、自分宛のデータパケットなら受信側 IP アドレスに VIP アドレスを代入し、上位層のプロトコルに渡す。コントロールパケットは `vip_input` で処理を行なった後、破棄し上位プロトコルには渡さない。

第 7 章

評価と考察

この章では、実装したKA9Q版VIPを評価・考察する。

7.1 KA9Q版VIPの現在の実装

7.1.1 実装環境

当初KA9Qは、911229バージョンを使用して実装していた。その実装バージョンは、KA9Q+VIP(931014)である。また、最新のKA9Qは930622バージョンであり、こちらも実装は終了している(KA9Q+VIP(940115))が、試験は完了していない。

使用したDOSは、IBM DOS Version J5.02/VとIBM PC DOS J6.10/Vであるが、英語MS-DOS 5.0等でも未確認だが動作可能であろう。

使用したコンパイラは、Borland C++ 3.1Jである。これは、当初からKA9QがBorland C++のソースで書かれていたためである。MS-Cなど他のコンパイラは試していない。

新たに作成したソースファイルは、`vip.(c,h)`, `vipcmd.c`, `amt.(c,h)`, `amtcmd.c`であり、約800行程度である。また、既存のファイルに追加した部分が約100行である。これにより、コンパイルされた実行ファイルは、Version931014(8086コードコンパイル)では約280キロバイト、Version940115(80386コードコンパイル)では約240キロバイトとなっている。但し、ソフトウェアオプションとしてはRIP、パケットのトレース、IPパケット経路検査、BOOTPのコードを含み、ハードウェアオプションとしてはパケットドライバを含むものとした。

7.1.2 VIPの実装状態

- ルータになることを想定していない。これはMS-DOSでの移動計算機環境では、そのような役割は現状では果たす必要がないと考えられたからである。しかし、近い将来、無線ネットワーク用の経路制御プロトコルによっては、全ての移動ホストがルータになることも予想される。

- 移動の際の手続き簡略化のためにBOOTPのクライアントを使用する。DHCPはBOOTPの上位互換のため、DHCPのサーバからも設定情報を取得可能。これにより、移動時の手続きは簡略化されている。しかし、BOOTPでは取得できる情報が限定されるので、DHCPの実装が望ましい。

- 移動の際の回線の切断を自動認識できないため、ユーザが明示的にBOOTPのク

ライアントを起動することで、移動先でのIPアドレスその他の設定情報を取得するようにしている。KA9Q+VIP(931014)ではF9キーを押すことによりBOOTPクライアントを起動する。またKA9Q+VIP(940115)ではF8、F9、F10がセッション切替用に予約済みのため変更予定。更に回線との接続・切断の検知を含む、自動化を検討中である。

- AMTはKA9Qのカーネル内部に静的変数として確保した。AMTの確保の方法としては他にファイルがあるが、検索と更新に高速性を必要とするため、これは使用しない。また現在のところ、その最大エントリ数は256としている。このエントリ数は大きいだけよいのであるが、メモリの制約からこの値にした。エントリの検索にはハッシュを使用して高速化している。また、エントリがいっぱいになった場合は新たなエントリは追加しない。
- VIPではルートエントリに関する処理は行なわない。したがって、移動時の、BOOTPによるパラメタ取得、あるいは`ifconfig,route`コマンドによる設定変更時に、古いアドレスでのルートエントリが残留する。このため、これを削除、その他のルートエントリもフラッシュするようにした。これは、移動前のルートエントリが残っていると移動後に送信されるコントロールパケットが正しく経路制御できなくなるのを防止するためである。

7.2 IETF'93,Fall 及び IPmeeting'93 での実験環境とその使用例

7.2.1 実験環境

VIPは、インターネットでの移動サポートプロトコルの標準案候補として提案されている。昨年は、IETF(Internet Engineering Task Force)ミーティングという技術者会議で実験デモを行ない、VIPを紹介した。また、国内でのネットワーク管理者の会議であるIPmeeting'93においても同様の実験デモを行なった。実験環境の概略図を図7.1に示す。

実験環境は、3つののサブネットワークから構成されている有線(10BASE-T)のネットワークである。その3つのサブネットワークをホストが移動することになる。

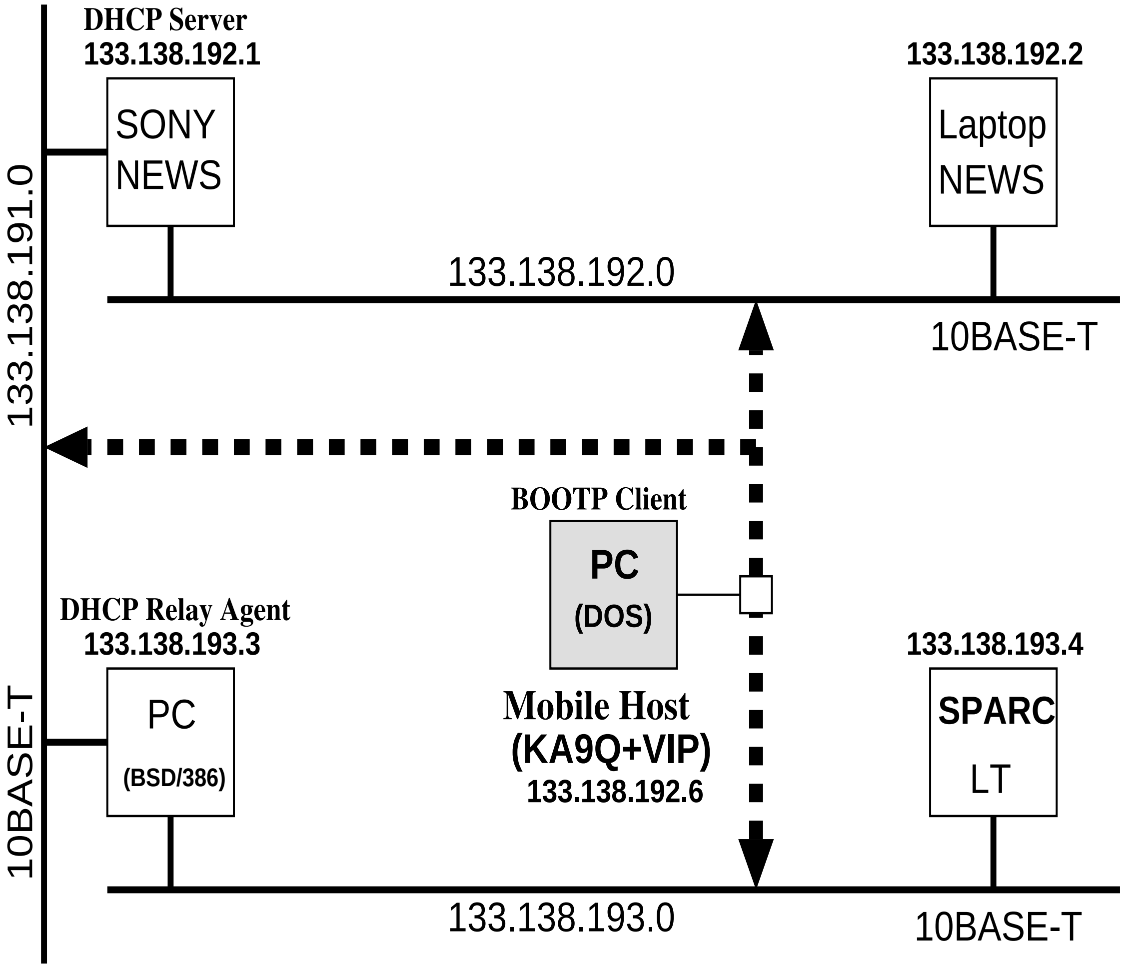

図 7.1: IETF'93 及び IPmeeting'93 での実験環境

DHCP に関係するホストが二つ存在している。一つはサーバであり、実際にホストに設定情報を割り当てる。もう一つはリレーエージェントと呼ばれるもので、サーバと異なるサブネットワークでクライアントのホストからリクエストがあった場合に、そのクライアントが接続しているサブネットワークに存在してそのリクエストをサーバに転送する役割を果たす。これは DHCP のサーバが必ずしもサブネット毎に存在する必要がなく、複数のサブネットにまたがってホストの設定情報の一元管理を行なうことができるためである。この DHCP により、KA9Q 版 VIP の移動ホストはどのサブネットワークに移動し接続しても BOOTP クライアントによって DHCP にリクエストを送ることによって IP アドレス、ネットマスク、ブロードキャストアドレスを取得

し、自動設定することが可能になっている。

133. 138. 192. 6 が PC の VIP アドレスであり、移動しても変化しない。133. 138. 191. 0 のサブネットワークでは、133. 138. 191. 6、133. 138. 192. 0 のサブネットワークでは 133. 138. 192. 6、 133. 138. 193. 0 のサブネットワークでは 133. 138. 193. 6、の IP アドレスがそれぞれ DHCP により PC に割り当てられる。それぞれのサブネットワークで IP アドレスが割り当てられた時に、VIP アドレスと IP アドレスとの対で構成される AMT が作成され各ホストやルータに伝播する。したがって相手ホストが移動した場合には AMT により VIP アドレスと IP アドレスの変換が行なわれ、相手の位置に関係なく通信を継続することが可能になる。

7.2.2 現在の有線での移動方法

現在の実装では、有線のネットワークにおいて次のような手順でホスト移動が行なわれる。

1. ケーブルを外して、別のサブネットワークにケーブルを接続する。
2. 接続先で BOOTP クライアントを起動。
3. BOOTP により、IP アドレス、ネットマスク、ブロードキャストアドレス、デフォルトゲートウェイアドレスを取得。
4. VIP のコントロールパケットを送信。
5. 通信再開。

このように移動しても、TCP のコネクションは保存される。また、位置に関わらず、相手の VIP アドレスを指定することにより通信することができる。

7.3 考察

7.3.1 Virtual Internet Protocol

実験環境においては、異なるサブネット間を移動してもセッションが保存され、通信が継続できることが明らかになった。これは、VIP がホストの識別子として VIP アドレスを使用することで移動透過性を実現していること実証している。

さらに、VIP の機構が比較的単純であるために新たに追加する部分・変更を要する既存の部分は非常に少ない。これは、VIP 自体のポータビリティの高さを示すものである。

7.3.2 MS-DOS 上での移動計算機環境

現在の KA9Q 版 VIP の実験では、telnet を使用している。しかし、telnet では計算機環境を遠隔ホストに依存している。つまり、プロセス自身と自分の情報、(ファイルなど) 両方の実体は移動ホストには存在しないのである。移動計算機環境では、自分の情報は仮想的に移動ホスト上にあって移動ホスト側の計算機資源を利用して作業する形態、さらに固定された遠隔ホスト上の自分の情報を移動ホストのファイルシステムと共有して作業する形態、の二つの形態がが考えられる。このいずれの場合でも情報の効率的操作を支援する機能が必要がある。

7.3.3 トランスポート層の問題

VIP を使用することで他層の問題点も明らかになった。トランスポート層、特に TCP を使用するアプリケーションでは、回線を切断して移動して再接続するまでに一定時間以上経過すると、タイムアウトによりそのセッションは終了してしまう。これは TCP の仕様上の問題である。このような状況は、有線ほどの安定度を持たない無線ネットワークで生じる可能性がある。このため、この問題は無線ネットワークでは致命的になる。したがって、切断・接続を検知し、オペレーティングシステムレベル、アプリケーションレベルでこのような状況を支援する機能が必要である。

第 8 章

結論と今後の課題

8.1 結論

本研究では、まず計算機の利用形態が変化して移動計算機環境を実現する必要があることを述べた。そのためには移動透過性という性質を実現することが重要になることについて説明した。しかしながら、既存のネットワークアーキテクチャではホストを特定するアドレスが完全なホスト識別子でないために、移動透過性を保証していなかった。移動透過性を実現するためのプロトコルとして、仮想ネットワークという概念をInternet Protocolに適用したVirtual Internet Protocolについてその概要を紹介した。これまで、VIPは主としてBSD系のUNIXで実装されていた。しかし、ネットワークの普及につれてUNIXだけでなくMS-DOSを使用する計算機上でもネットワークに接続し利用できるようになってきた。また、MS-DOSはその構造上メモリやディスクの制約がそれほどなく、負荷も軽い実行環境が得られる。以上により、MS-DOS上にVIPをMS-DOS上のTCP/IPパッケージを実装し移動計算機環境を実現ことは重要である。MS-DOSはシングルタスクシステムである。よってUNIXのマルチタスクシステムに依存した処理を持つVIPを実装するために、疑似的なマルチタスクシステムを内部に持つKA9Q NOSに実装した。VIPをMS-DOS上に実装することにより、VIPをMS-DOS環境に適応させ、移動透過性が保証できることを実証した。また、VIPが持つUNIXへの依存性が明らかになり、MS-DOS環境への整合性が検証された。

8.2 今後の課題

VIPによりネットワーク層での移動に対する透過は実現できているが、オペレーティングシステムレベル、アプリケーションレベルでの移動計算機環境に対する支援は十分ではない。

例えば、移動時にネットワークとの接続が長時間絶たれている場合、つまりoff-line移動時には作業を継続的連続的に行なうことができない。NFSのクライアントが切断を検知してその間を移動ホストのローカルのディスクにあらかじめキャッシュしておいたファイルで作業させる。再び接続された場合にサーバ側の実体ファイルに書き込みを行なう、といった解決法が考えられる。

ネットワークとの切断・接続がどのレベルでどのように発生し、どのように処理すべきであるかを考える、別の面からの研究が必要であることを示している。

また、KA9Q 上の移動計算機環境の支援もまだ不十分な点が多い。現在、KA9Q 版 VIP では再接続をユーザによる明示的な操作により認識するが、移動の際の手続きもユーザから隠蔽することが必要であり、自動化するべきである。ネットワークへの計算機の自動設定には現在 BOOTP を使用しているが、取得できる情報は、IP アドレス、ネットマスク、ブロードキャストアドレス、デフォルトゲートウェイアドレスに限られる。しかしこのほかに、例えばネームサーバアドレスなどさらに多くの情報を必要に応じて取得できる必要がある。これには、約 60 種に及ぶ設定パラメタを取得できる DHCP のクライアントを実装するべきである。その他の課題としては、VIP ルータとしての機能の実装、SLIP ・PPP での利用実験、日本語化、KA9Q による統合移動計算機環境の構築などがある。これらの課題の解決と試験による安定化実現の後、本ソフトウェアの配布を予定している。

謝辞

本研究を進めるにあたり、多大なる御助言と御指導を頂きました、指導教官の村井純助教授、徳田英幸助教授に御礼申し上げます。ソニーコンピュータサイエンス研究所の寺岡文男氏、電気通信大学大学院の植原啓介氏に深く感謝致します。そして、中村修氏と楠本博之氏に深く感謝致します。また、研究を進めていくにあたり、小生を暖かく支えて頂いた村井徳田研究室の方々、有益な議論をして下さった VIP Working Group をはじめとする WIDE Project の皆様に感謝致します。

参考文献

[1] Fumio Teraoka,Yasuhiko Yokote,and Mario Tokoro: A Network Architecture Providing Host Migration Transparency. In *Proceedings of ACM SIGCOMM*,September 1991.

[2] 寺岡文男, 横手靖彦, 所 真理雄: 広域ネットワークにおけるホスト移動透過性, コンピュータソフトウェア,Vol.9,No.3(1992),pp.50-64.

[3] Fumio Teraoka,Kim Claffy,and Mario Tokoro: Design,Implementation,and Evaluation of Virtual Internet Protocol In *Proceedings of the 12th International Conference on Distributed Computing Systems*,June 1992

[4] Fumio Teraoka and Mario Tokoro: Host Migration Transparency in IP networks: The VIP Approach. *ACM Computer Communication Review*,Vol.23,No.1,January 1993

[5] Uehara, K., Teraoka, F., Sunahara, H., and Murai, J.: Enhancement of VIP and Its Evaluation, In *Proceedings of INET'93*, August 1993.

[6] Fumio Teraoka and Keisuke Uehara: The Virtual Network Protocol for Host Mobility, Internet-Draft,*draft-teraoka-mobileip-vip-00.txt*,July 1993.

[7] 植原啓介, 寺岡文男: VIP の改良及びその評価, WIDE 研究会,1993 年 5 月

[8] C.Perkins and Y.Rekhter.: Support for mobility with Connectionless Network Layer Protocols. In *Internet Draft*,January 1993.

[9] John Ioannidis and Gerald Q.Maguire Jr.: The Design and Implementationof a Mobile Internetworking Architecture In *Proceedings of 1993 Winter USENIX*,January 1993.

[10] Hiromi Wada,Takashi Yozawa,Tatsuya Ohnishi,and Yasunori Tanaka: Mobile Computing Environment Based on Internet Packet Forwarding. In *Proceedings of 1993 Winter USENIX*,January 1993.

[11] Charles Kunzinger: *Routing Support for IP Mobile Hosts.* Octorber 1993,Internet-Draft

[12] David Johnshon,Andrew Myles,annd Charles Perkins: *The Internet Mobile Host Protocol.* Octorber 1993,Internet-Draft.

[13] Phil Karn,et al: KA9Q Network Operating System(NOS) package,1989.

[14] W. Croft and J.Gilmore: *Bootstrap Protocol*,RFC951,September 1985

[15] R. Droms: *Dynamic Host Configuration Protocol*,RFC1541,Octorber 1993

[16] S. Alexander and R.Droms: *DHCP Options and BOOTP Vender Extentions*,RFC1533,Octorber 1993

[17] 冨永明宏, 寺岡文男, 村井純: 動的ホスト設定プロトコル (DHCP) の実装の評価 情報処理学会 マルチメディア通信と分散処理研究会 予稿集,1993 年 12 月