第 4 部

移動計算機の支援

第 1 章

はじめに

“mobile computing” という言葉が新聞をも賑わすようになった。これはさまざまな通信システムの実用化および小型軽量の携帯型計算機の登場によるものである。たとえば未来の通信基盤としては PHP (Personal Handy Phone)、LEO (Low Earth Orbit: 低軌道周回衛星)、FPLMTS (Future Public Land Mobile Telecommunication System: 将来の公衆陸上移動通信システム)、などが提案されており、このうち PHP は実験運用が始まっている。携帯端末としては Personal Communicator (AT&T)、Newton (Apple)、Simon (Bell South)、Zoomer (Tandy)、ZAURUS (シャープ)、Think Pad (IBM)、XTEND (東芝) などがすでに発売されている。しかしこれらの機器は主にスケジューラのような電子手帳的な機能に焦点を当てており、通信機能はなおざりにされているのが実状である。

我々は計算機がネットワークを移動してもユーザには計算機の移動を意識させないための性質、ホスト移動透過性を実現するプロトコルの研究を続けている。我々の研究は以下のように進められてきた。まず 1991 年に移動ホストをサポートするための基本概念である仮想ネットワークの概念とこれを効率よく実現するための拡散キャッシュ法が提案された [26, 27]。VIP (Virtual Internet Protocol) はこれらを IP に適用したものである。続いて 1992 年に VIP の設計、実装および性能評価の実測が行なわれ [28, 29]、ホストが移動する際に VIP に付随して実行されるプロトコルについても設計、実装が行なわれた [30]。また IP との互換性の観点から、2 種類の VIP 実装方法に関する考察も行なわれた [31, 32]。さらに昨年度は VIP の改良を行ない、IP との完全互換性および無効キャッシュの効率の良い消去を実現した [33, 34]。以上のような研究の流れを受け、本年度は以下に示す研究を重点的に行なった。

第 2章では動的ホスト設定プロトコルの実装と評価について述べる。VIP において移動ホストは移動先のネットワークにおいて一時的な IP アドレスや経路情報などを得る必要がある。文献 [30] で述べられているように、この目的のため VIP では独自のプロトコルを設計し使用していた。しかし独自のプロトコルを使用していては汎用性が低くなってしまう。そこで、インターネット標準として提案された DHCP (Dynamic Host Configuration Protocol) の実装を行ない、これを用いて一時的な IP アドレスの取得などを行なうようにした。我々の実装は 1993 年 10 月に米国シアトルで行なわれた相互接続試験において良好な結果を示した。また現在 DHCP はソースコードはもとより実行形式の配布も行なわれていないが、我々は 1994 年 3 月にアルファリリースを開始した。これには DHCP

のソースコードはも含まれている。1994 年 5 月にはベータリリースが計画されている。

第 3章では MS-DOS 上に構築した移動計算機環境について述べる。今まで VIP は SONY NEWS などの UNIX ワークステーションに実装されてきた。しかし現在世界中でもっとも多く使われているオペレーティングシステムは MS-DOS である。また携帯型計算機ではメモリやディスク容量のようなハードウェアの制約から、より小さくて軽いオペレーティングシステムが望ましい。このような理由により、我々は比較的小さなハードウェアでも動作する MS-DOS に移動計算機環境を構築することにした。今回は KA9Q という TCP/IP パッケージを使用し、これに VIP を移植した。VIP の移植により、実行形式はわずか 4Kbyte 増加しただけである。この実装は DHCP の実装と合わせて 1993 年 11 月に米国ヒューストンで開かれた IETF (Internet Engineering Task Force) と 12 月に国内で開かれた IP ミーティング'93 においてデモを行ない、良好な結果を得ている。VIP を組み込んだ KA9Q のソースコードは 1994 年 5 月に配布を開始する予定である。

第 4章では IETF (Internet Engineering Task Force) の Mobile-IP ワーキンググループで提案されている移動ホストプロトコルの評価について述べる。IETF とはインターネットでの標準プロトコルを作成する組織であり、多数のワーキンググループに別れて活動している。Mobile-IP ワーキンググループは 1992 年 7 月から活動を始めた。我々は VIP を標準案の 1 つとして Mobile-IP ワーキンググループに提案したが、その後 10 種類近いプロトコルが標準案として提案された。結局 Mobile-IP ワーキンググループとしてはこれらの中から 1 つを選択するのではなく、これらを統合して現在のインターネットになるべく影響を及ぼさないプロトコルを作成することになった。我々は Mobile-IP プロトコルの実装を行ない、プロトコル自体の問題点および実装にかかわる問題点を明らかにした。

最後に第 5章では OSI (Open Systems Interconnection) のコネクションレス型ネットワーク層プロトコルである CLNP における移動ホストのためのプロトコル設計について述べる。このプロトコルは、ホスト移動をエリア[1]間およびエリア内に分け、階層的に設計されている。現在の IP (IPv4) は、経路制御やアドレス付けの点ですでに限界に近づいており、IETF では次世代 IP (IPng) の作成が行なわれている。CLNP は IPng の候補となっており、我々のプロトコルは将来のインターネットにおいて重要な役割をはたすであろう。今後はこのプロトコルを実装し、評価を行なう予定である。

[1]OSI の用語は第 5.1節を参照。

第 2 章

動的ホスト設定プロトコル

2.1 はじめに

インターネット上の計算機には識別のために全世界を通じて一意の値を持つ IP アドレスが割り当てられる。また他の計算機と通信のため他にも共有資源の割り当てを受け、計算機に設定する必要がある。逆にインターネットから計算機が撤去される場合にはこれらの共有資源を回収する必要がある。しかしながら現在にいたるまで、これらの作業の大半が人手に委ねられてきた。さらにインターネットの拡大にともなって共有資源の管理に要する労力が増加している。結果として資源の管理はおろそかになり、共有資源の利用効率が低下する。この端的な例としてあげられるのが IP アドレスの枯渇問題 [35] である。これを回避するためには IP アドレスの利用効率を向上させることが望ましい。

一方で近年の携帯型計算機および小型ネットワーク機器の登場により、計算機はインターネットへの接続/切断を頻繁に繰り返すようになった。これらの移動する計算機は比較的短い期間のみ共有資源の割り当てを要求する。このような新しい種類の要求には既存の人手による方法では対応できない。これら移動計算機に特有の要求に応えるための機構として Dynamic Host Configuration Protocol (DHCP)[36, 37] が提案されている。

DHCP の持つ特徴として以下のようなものがあげられる。

- 動的な IP アドレスの割り当てが可能
- 使用されていない IP アドレスの回収が可能
- 管理者やユーザの介入なしに動作する

このような特徴をもつ DHCP はインターネットにおいて重要な役割を担うと思われるが、現在までのところライセンスフリーで利用可能な実装は配布されていない。そこで本研究では DHCP を機種独立性の高い形で実現する際に生ずる問題点の考察と解決方法の提案を行ない、それに基づいた実装と評価を行なう。

また移動計算機をサポートするためのプロトコルとして Virtual Internet Protocol (VIP)[33, 34] が提案されている。VIP を利用することで計算機の移動透過性が実現されるが、VIP ホストは他のネットワークに移動する時には一時的な IP アドレスの取得を必要とする。本研究ではこの一時 IP アドレスの取得に DHCP を適用する方法に関しても考察し、実装と評価を行なう。

2.2 Dynamic Host Configuration Protocol

2.2.1 DHCP の問題領域

計算機への共有資源割当の問題を分析すると次のような要求が存在することがわかる。

- 計算機の自動設定 ‥ 計算機の接続と誤りがちな設定の自動化
- 計算機の集中管理 ‥ 共有資源の集中的な管理
- アドレスの一時的な割り当て ‥ 短期的な接続のための一時的な IP アドレスの割当

DHCP はこれらの要求を解決するものである。加えて近年の懸念である IP アドレスの枯渇に対処するために、次のような要求も存在する。

- 効率的利用 ‥ 使用されていない IP アドレスの回収
- 共有機構 ‥ 複数の計算機での IP アドレスの共有

DHCP はこれら 2 つの要求に関しても取り扱っている。ただし残念ながら完全な解決にまでは至っていない。

2.2.2 DHCP の概念モデル

DHCP はサーバクライアントモデルの上に構築されている。設定情報を得るために DHCP を利用するホストがクライアントであり、その要求に応えるのが DHCP サーバである。これに加えて DHCP にはリレーエージェントというものが存在する。

DHCP は BOOTP を拡張したプロトコルである。BOOTP は MAC アドレスから IP アドレスへのマッピングと数種類の設定情報の通知を行なうものである。これらはともに UDP 上に構築されていて、All 1 (宛先 IP アドレスが 255.255.255.255) のブロードキャストを用いる。All 1 のブロードキャストはルータを越えて別のサブネットに到達することはできないため、DHCP のサーバとクライアントが別々のサブネットに存在する場合に中継を行なうのがリレーエージェントである。リレーエージェントの機構はルータの転送機構とは独立しているため、リレーエージェントとルータが同一のホストである必要はない。DHCP と BOOTP のリレーエージェントは等価である。

DHCP サーバは 2 つのデータベースを管理する。1 つは「アドレスプール (Address Pool)」と呼ばれ、ネットワーク上の共有資源を含んだ静的なデータベースである。もう 1 つは「バインディングデータベース」と呼ばれ、常に更新されるデータベースである。各種設定情報 (特に IP アドレス) の割り当てはサーバがアドレスプールの中からエントリを選びだし、それをクライアントに通知する形で行なわれる。この際に「どのクライアントにどのエントリを割り当てたか」という対応を DHCP ではバインディング (binding) と呼ぶ。現在の DHCP では、アドレスプールの補充を自動的に行なうことはできないし、複数のサーバ間でアドレスプールの内容が重複することは想定されていない。

クライアントは設定情報の割り当て要求を UDP データグラムに入れて送信し、サーバからの応答を受けとることで動作する。サーバはメッセージ駆動型なので、すべての動作をクライアントが能動的に行なわなければならない。DHCP の仕様書 [36, 37] の中では有限状態機械によるクライアントのモデル化が行なわれている。

リレーエージェントはクライアントの送信した DHCP メッセージを OSI 7 層モデルにおけるアプリケーション層で受けとる。リレーエージェントは DHCP メッセージのいくつかのフィールドを書き換え、再度 UDP データグラムにのせ、サーバに向けてユニキャストで送信する。この点が一般のルータと大きく異なる。

2.2.3　割り当てモデル

DHCP では設定情報の割り当てを「動的割り当て」、「自動割り当て」、「手動割り当て」の 3 種類に分類している。

1. 動的割り当て ···· DHCP の最も特徴的な割り当て機能である。サーバはアドレスプールの中から IP アドレスを選択し、クライアントに期限つきで割り当てる。これが「アドレスの一時的な割り当て」という要求を満たす。
2. 自動割り当て ···· 同様にサーバがアドレスプールの中から IP アドレスの選択を行なう。ただし割り当ての有効期限は無期限である。これが「アドレスを自動的に割り当てたい」という要求を満たす。現在の DHCP では使用されなくなった無期限のアドレスを回収する機構は定義されていない。
3. 手動割り当て ···· 前述の 2 つとは異なり、あらかじめ管理者が割り当てておいたアドレスを通知するためだけに DHCP が利用される。いわゆる「静的割り当て」であり、既存の BOOTP の機構とほぼ等価である。

2.2.4　プロトコルの動作概要

DHCP は表 2.1 に示すような 7 種類のメッセージを用いていて設定情報の割り当てを行なう。まず新たにアドレスを割り当てる場合のプロトコルの流れを図 2.1に示す[1]。

1. DHCP クライアントは自身の接続しているサブネットに DHCPDISCOVER メッセージをブロードキャストする。クライアントは希望する IP アドレスや希望する割り当て期限を示す値を含めても良い。このときリレーエージェントが別のサブネットのサーバへメッセージを転送するかもしれない。
2. 受信したサーバ (複数存在するかもしれない) は DHCPOFFER メッセージを返送してもよいし、しなくてもよい。返送する場合は割り当てても良いアドレスを含めてブロードキャストする。ここで注意すべきことは、この時点ではサーバとクライアントの間でアドレス割り当ての合意は形成されていないということである。DHCPOFFER はあくまでも「提案」を行なうだけである。

[1] 図中の数字と説明文の数字は対応している。

表 2.1: DHCP のメッセージ

メッセージ	内容
DHCPDISCOVER	サーバ群を特定するためにクライアントがブロードキャストする
DHCPOFFER	サーバが DHCPDISCOVER に応えて送る各種情報の提案
DHCPREQUEST	クライアントが選択したサーバにパラメータを要求するブロードキャスト (他のサーバに対しては提案の辞退を示す)
DHCPACK	サーバがクライアントにアドレスを割り当てる場合に送る
DHCPNAK	サーバがクライアントの要求を断る場合に送る
DHCPDECLINE	クライアントが割り当てられた情報やアドレスに何か問題を発見した際に送る
DHCPRELEASE	クライアントが期限が切れる前にアドレスを放棄する場合に送る

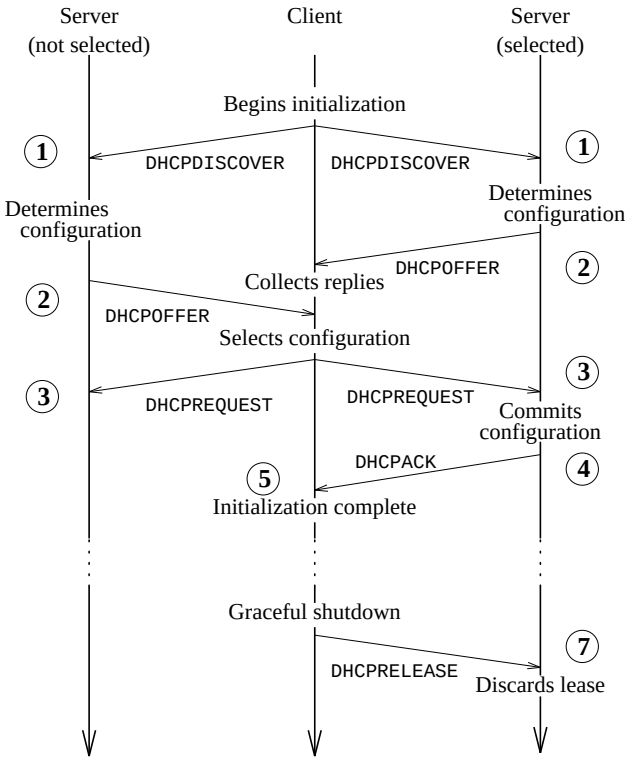

図 2.1: 新規アドレス割り当て

3. クライアントは 1 つないし複数の DHCPOFFER を受けとり、何らかの方針に基づいてサーバを 1 つ選び出す。クライアントは選んだサーバの IP アドレスを含む DHCPREQUEST をブロードキャストする。
4. DHCPREQUEST を受けとったサーバはメッセージをチェックし、自分宛てでない場合にはクライアントが提供を断ったものとみなす。自分宛てだった場合、そのアドレスを割り当てても良ければ DHCPACK を返す。何らかの理由で割り当てることができない場合[2]には DHCPNAK を返す。
5. DHCPACKを受けとったクライアントはそのアドレスの使用を開始する。もし DHCPNAK を受けとった場合には初期化を最初からやり直す。
6. クライアントが DHCPACK で受けとったアドレスや情報に問題を発見した場合[3]は、DHCPDECLINE をサーバに送信して初期化を最初からやり直す。
7. クライアントが停止する場合や別のサブネットへ移ろうとするときには DHCPRELEASE を送信して割り当てを放棄する旨を通知しても構わない。ただしこれを行なわなくても DHCP は正常に動作するようになっている。

次にアドレスの「再割り当て/期限延長/確認」の場合について説明する。クライアントは動的割り当てを受けたアドレスの期限が切れる前に延長を申請することができる。期限が切れた後であっても以前に使用していたアドレスの再割り当てを申請することが可能になっている。さらに DHCP は割り当てられた設定情報が変化した可能性のある場合には、任意の時点で「確認」を行なうことができる。確認と再割り当ての唯一の違いは、前者の場合はアドレスの有効期限が切れていないということである。これらの場合には、既に説明したプロトコルフローのうちのいくつかのステップを省略することができる。

1. クライアントは以前のアドレスを DHCPREQUEST メッセージに含めてブロードキャストする。期限の延長の場合には明示的に希望する期限を示す必要がある。
2. そのアドレスについての情報を持つサーバが DHCPACK か DHCPNAK で応える。
3. 以後は図 2.1の場合と同様。

アドレスの割り当てを受けたクライアントは、2 つの時刻 T1 および T2 を管理する。これらの値はサーバから通知されるか、さもなければ標準の値が利用される。クライアントは時刻 T1 がくると期限の延長申請を開始する。最初にその時点で利用しているアドレスを割り当てたサーバに対してユニキャストで延長申請を開始する。

時刻 T2 まで再転送を繰り返しても DHCPACK も DHCPNAK も返ってこない場合にはブロードキャストで延長申請をおこなう。有効期限が切れるまでに DHCPACK を受けとることができた場合には再び二つの時刻を管理する。それ以外の場合には初期状態に戻ってアドレスの取得を最初から行なう。

[2]例えば、既に別のクライアントにそのアドレスを割り当ててしまった場合。
[3]例として、ARP 要求を使ったチェックを行なうことが推奨されている。

2.2.5 関連研究

すでに触れたように DHCP は BOOTP の拡張である。しかしながら他にも DHCP の取り扱う問題の一部を解決しようと試みたプロトコルがある。本節ではそれらのプロトコルについて概要を説明する。

- Reverse Address Resolution Protocol (RARP)[38] ···· MAC アドレスから IP アドレスへのマッピングのみを行なう。
- Bootstrap Protocol (BOOTP) ···· MAC アドレスと IP アドレスのマッピングおよび数種のホスト設定情報の通知機能を持つ。しかし BOOTP は DHCP における「手動割り当て」の機構のみを提供しており、一時的なアドレス割り当てや使われなくなったアドレスの回収といった要求には対応できない。
- Network Information Protocol (NIP)[39] ···· MIT 独自のもので、MAC アドレスに対して IP アドレスの動的な割り当てを行なうことができる。NIP は “Polling/Defense mechanism” と呼ばれる機構を用いるが、これはアドレスが絶対に重複しないと保証することができない。さらに起動するたびに以前と異なるアドレスが割り当てられる可能性があるが、これは好ましいことではない。しかし取得できる設定情報は IP アドレスに限定されているし、動的な割り当ての方法にも問題がある。
- Dynamic RARP[40] ···· RARP を拡張したもので、動的なアドレスの割り当てを可能にする。しかし IP アドレスの割り当ての有効期限が 1 時間に固定されていることや、 取得できる情報の種類に限りがあるなどの問題がある。
- Point-to-Point Protocol (PPP)[41] ···· PPP 自体はポイント-ポイント間のリンク上で複数のプロトコルデータグラムを転送するためのものである。PPP は 3 つの要素から構成されているが、その中に Link Control Protocol というものが含まれている。これを IP に適用したものとして Internet Protocol control Protocol (IPCP)[42] が存在する。IPCP では PPP のコネクションを張る際にコネクションの両端の IP アドレスを設定する方法が規定されている。しかしながら実際にアドレスをどのように割り当てるかということに関しては規定されていない。

表 2.2に各方式の比較を示す。RARP, BOOTP, IPCP (PPP) は静的な割り当てしかできない。一方 NIP, DRARP は動的な割り当てしかできず、また動的な割り当ての機構も不完全である。提供できる情報の種類は

$$RARP, NIP, DRARP, PPP << BOOTP < DHCP$$

の順に豊富である。 DHCP では現在約 60 種類の情報を提供することができる。ただし常にすべての情報を必要とするわけではないため、クライアントが望む設定情報のリストをサーバに通知することができる。

表 2.2: 各方式の比較

	静的割当	動的割当	情報の種類	スケール	プロトコル完成度
RARP	○	×	×	×	○
BOOTP	○	×	△	△	○
NIP	×	△	×	×	△
DRARP	×	△	×	×	×
IPCP (PPP)	○	×	×	×	○
DHCP	◎	◎	◎	◎	?

2.3 DHCP の実装

この節ではまず DHCP を実装するにあたって設定した目標を示す。次にサーバ、クライアント、リレーエージェントのそれぞれについてそれらの目標を達成する上で生じる問題を明らかにし、その解決方法を考案する。その後各々についてさらに具体的な設計と実装の解説および評価を行なう。また相互運用性テストへの参加結果についてもあわせて述べる。

2.3.1 実装目標

今回の実装で全体的な目標としたことを以下に示す。括弧内はサーバ、クライアント、リレーエージェントのうちのいずれに対する目標なのかを頭文字で表している。

1. 一般的に使用されているオペレーティングシステム上で動くようにする (S,C,R)
2. オペレーティングシステムに対する独立性を高くする (S,C,R)
3. 実際の大規模な環境で使えるようにする (S,R)
4. ユーザの使いやすさを考慮する (S,C,R)
5. ソースコードを全て公表し、自由に変更できるようにする (S,C,R)

現在までのところ DHCP の実装は公開、配布されていないため、ライセンスフリーの実装を行なうことは重要である。DHCP を組み込んだ計算機が販売されるようになったとしても機種依存性が高くなると予想される上に、プログラムのソースコードは公開されない可能性が高い。したがって一般的なオペレーティングシステムの上で動作する機種独立性の高い実装を行なうことを最大の目標とした。

2.3.2 サーバの設計

今回は “一般的なオペレーティングシステム” として UNIX(BSD 系および SunOS4.1.3) を選択した。DHCP 自体はデータリンク層が何であるかを特に仮定していないが、本実装では Ethernet に限定する。“オペレーティングシステムに対する独立性” を高くする

ために DHCP サーバをユーザ空間におけるアプリケーションプログラムとして実装を行なう。

ユーザ空間で実装する際に生ずる最大の問題点として、通常の *socket* インタフェースでは All 1 のブロードキャストが不可能だということがあげられる[4]。さらにユーザ空間のプログラムがブロードキャストを受信したインタフェースを識別することは通常不可能である。これらの問題点への対処方法に関しては、第 2.3.3節で詳しく解説する。

“実際の大規模な環境” で使用できるために、アドレスプールのエントリ数が増えてもメモリの使用量を O(エントリ数) 以下にする必要がある。さらに多数のクライアントがほぼ同時に要求を送出したとしても、それらを処理できるようなパフォーマンスを達成しなければならない。そのためには Ethernet 上を流れる雑多なデータグラムの中から効率的に DHCP のメッセージを抽出することが重要である。また “長時間の運用” を行なっても安定して動作しなければならない。これは DHCP の実装目標として掲げられている “サーバのリブートの前後を通じてホストの設定を保つ” という要求にも深く関わってくる。

“ユーザの使いやすさを考慮する” ために、複雑なサーバ設定ファイルを記述しなくても済むようにする (既存のデーモンの設定ファイルなどと同様にする) ことと、アドレスプールデータベースなどの記述を簡潔にすることの 2 点を具体的な目標として設定した。

DHCP は BOOTP の上位互換であるため、DHCP サーバが BOOTP クライアントをサポートすることが可能なことは既に説明した。BOOTP クライアントのサポートは必ずしも行なわなくても良いのだが、今回の実装ではこれも目標に含めた。

2.3.3 サーバの実装

本実装は C 言語にして約 6400 行からなる。目標にも掲げたようにオペレーティングシステムのユーザ空間で実装されている。前節で指摘した All 1 のブロードキャストに関しては、Berkeley Packet Filter (BPF)[43] または Network Interface Tap (NIT)[44] を使用することで解決した。BPF は BSD 系の UNIX に含まれるもので、ネットワークインタフェースのドライバに直接組み込まれている。そのためデータリンク層のパケットを直接送受信できる。したがってどのネットワークインタフェースからパケットを受信したか識別することも容易であるし、 All 1 のブロードキャストを行なうことも可能である。その代わりにトランスポート層、ネットワーク層、データリンク層の処理を自力で処理する必要がある。状況によっては DHCP サーバは通常のユニキャストによる送受信を行なうが、その場合にはオペレーティングシステムの *socket* インタフェースを使用している。なお、SunOS の場合には BPF の代わりに Network Interface Tap (NIT) を利用した。図 2.2 にクライアントとリレーエージェントも含めたサーバの実装構造を示す。

BPF は強力なフィルター機能を持ち、疑似アセンブラのような独自のコマンド体系を用いて任意のフィルターを作成できる。もちろん BPF のレベルでなくサーバプログラム

[4] 厳密に言えば Net2 以降の BSD などでは不可能ではないが、問題が多い。

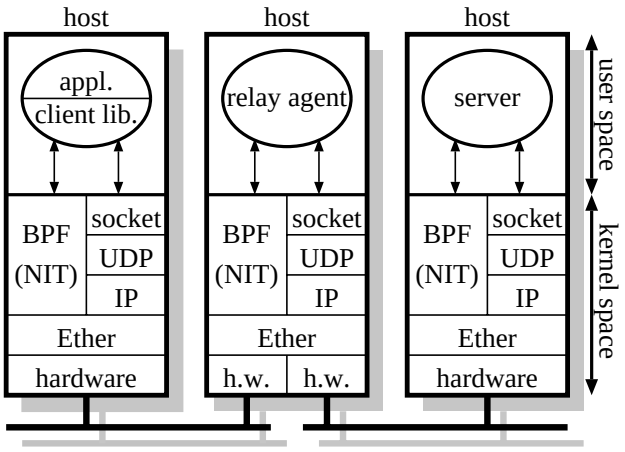

図 2.2: 実装構造

の中でフィルタリングを行なうこともできるが、速度などの観点から言って非効率的である。混雑しているネットワークでも効率的に動作するように、BPF のレベルでフィルタリングを行なった。

サーバは要求を受けとると自分自身の複製を作って処理するのではなく、自分ですべての処理を行なう。したがってオペレーティングシステムなどの性能も含めた処理能力を上回る頻度でパケットが到着すると、処理しきれなくなる。

データベースの記述を容易にするため、アドレスプールデータベースの記述はいわゆる *termcap*[45] と同形式を採用した。ただし文法に若干の変更を加えて機能を拡張してある。この部分に関しては、CMU の BOOTP サーバに含まれるソースに変更を加えることによって作成した。一方バインディングデータベースは単純にデータの羅列を 1 行に 1 エントリ記述するようになっている。リレーエージェントから DHCP メッセージが中継されてきた場合、アドレスを割り当てるためにはそこのサブネット番号を知る必要もある。今回はリレーエージェントとサブネット番号の対応を記述したデータベースを用意した。この中に記述されていないリレーエージェントからの転送は処理しないように実装したため、不当なリレーエージェントからの転送を拒否するためのアクセスコントロールリストとしての役割も持っている。

メモリの使用量を抑えるために動的にメモリを確保するのはもちろん、ポインタを可能な限り共有することでメモリの節約を目指した。その結果、各ホストエントリに対して最初に確保される領域は 208 オクテットである。

2.3.4 サーバの評価

サーバ実装の “一般的なオペレーティングシステムの上で動作する” という目標は達成できた。現在のところ本実装は BSD/386, NEWS-OS (CISC[5]版と RISC[6]版) という 2 つの BSD 系 UNIX の上で動作することを確認している。また SunOS4.1.3 の上でも動作確認を行なった。これらは Intel 80486/80386, MC68030, R3000, Sparc などの全く異なるアーキテクチャの CPU 上で動作している。これは 本実装の機種独立性が高いことを立証している。また BPF や NIT を利用することでネットワークインタフェースの種類や数に関係なく動作するようになっている。このように “オペレーティングシステムに対する独立性を高くする” という目標も達成できた。またテキストファイルを使った設定方法によってユーザの使いやすさも考慮した。*termcap* 形式のデータベースを導入したことによって設定の柔軟性も確保することに成功した。

サーバ実装は “大規模環境でも動作する” 以外の目標は達成できた。“大規模環境でも動作する” に関しては、実験を行なっていないために確認できていない。なお今回のサーバは BOOTP クライアントのサポートも目的としていた。これに関しては、KA9Q という MS-DOS 上で動作するライセンスフリーの TCP/IP パッケージに含まれる BOOTP クライアントとの相互運用テストでも良好な結果が得られた (第 3.4節参照)。

2.3.5 クライアントの設計

クライアントの場合にも、サーバ同様に UNIX 上に実装する。やはりブロードキャストの利用に関して問題が生じるが、BPF や NIT を使用して解決できる。

DHCP では現在約 60 種類の設定情報を取り扱うことが可能であるが、通常のクライアントがこれらの情報をすべて必要とするとは考えられない。そこで本実装ではライブラリルーチンの形でクライアントを提供することにする。各ユーザは個人的な要求に基づいてライブラリルーチンを使ったプログラミングを行なう必要がある。ユーザの使いやすさを考慮して、ライブラリには抽象度の高いルーチンを用意しているので、ユーザ自身で記述すべきソースコードの量は少ない。

文献 [36, 37] にはクライアントの状態遷移図が含まれるが、これをそのまま実装するといくつかの問題が生ずる。そこで実装に先だって状態遷移図の見直しを行なった (図 2.3)。

まず SELECTING 状態から REQUESTING 状態へ遷移するための条件が不明瞭であったため、新たに WAIT_OFFER 状態を付け加えた。これにより REQUESTING 状態へ遷移する条件が明確になった。

DHCP では以前使用していたアドレスを引き続き使用できるかどうか以前のサーバに確認できる。またそのサーバが何らかの理由によって作動していなかった場合には、以

[5] Complex Instruction Set Computer の略称。複雑で高機能な命令セットを持つ CPU のこと。

[6] Reduced Instruction Set Computer の略称。命令セットを基本的なものに限定することで、プログラムの高速実行を目指している CPU のこと。

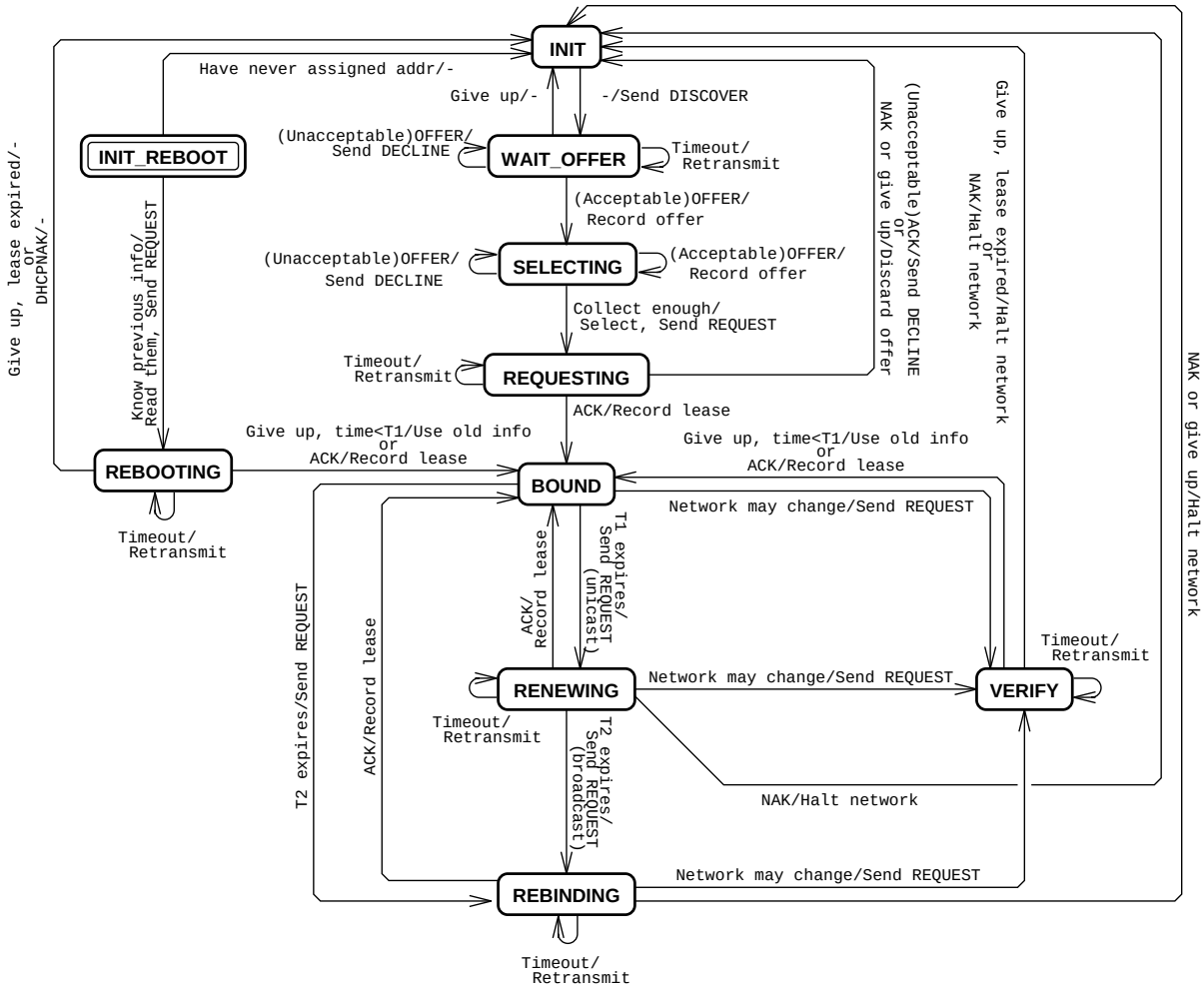

図 2.3: 改良した状態遷移図

前のアドレスを有効期限が切れるまで使い続けることができる。これらは REBOOTING 状態から BOUND 状態へ遷移することに相当する。しかし元の図には DHCPACK を受信した場合、つまり確認に成功した場合についてしか記述されていなかった。

また実際には有効期限の残り時間によってはただちに RENEWING 状態や REBINDING 状態へと遷移する必要がある。しかし DHCP の仕様書の中では BOUND 状態に遷移する場合しか記述されていない。したがって BOUND 状態から REBINDING 状態への遷移を付け加えた。

さらに “ネットワークが変化したかも知れない” 場合に行なう確認 (Verify) に相当する VERIFY 状態を追加した。BOUND 状態、 RENEWING 状態、REBINDING 状態のどの状態でも VERIFY 状態に遷移することができる。これにより “ネットワークが変わったかも知れない” という非同期の事象にいつでも対応できるようになった。

このように変更された状態遷移図に基づいてクライアントライブラリの設計を行なった。状態遷移図に忠実な実装をするために、単一の関数で全てを処理するようにする。また “ネットワークが変わったかも知れない” 場合、つまり確認 (verify) をしたい場合に非同期にイベントを通知するための手段としてシグナルを用いる。DHCPRELEASE を送って割り当てを放棄するというのも非同期に発生するイベントだが、この通知にもシグナルを用いるものとする。

2.3.6 クライアントの実装

設計に従い、*dhcp_client()* という単一の関数を用意した。引数は次の 6個である。

1. ネットワークインタフェースの名前などを含む構造体へのポインタ
2. “Parameter Request List” オプションに含める値、DHCPOFFER を収集する秒数、“Client Identifier” などの情報を含んだ構造体へのポインタ
3. 割り当てられた情報を利用するための関数へのポインタ。この関数はヘッダファイルに定義された構造体 dhcp_param を引数として受けとる
4. 複数の DHCPOFFER(dhcp_param のリストとして渡される) の中から、一つを選択する関数へのポインタ
5. DHCPOFFER を受けとった際に、提案された設定情報の内容に問題がないか調べる関数へのポインタ
6. DHCPACK を受けとった際に、提案された設定情報の内容に問題がないか調べる関数へのポインタ

さらに VERIFY 状態への遷移を引き起こすためには SIGUSR1 を、割り当てを放棄することをライブラリに通知するためには SIGUSR2 を利用することにした。また内部的に使用している *config_if()*(ネットワークインタフェースにアドレスを設定する) や *set_route()*(ルーティングテーブルを設定する)、*flushroutes()*(ルーティングテーブルをすべて消去する) などの関数は内部に隠蔽せず、ユーザが実際にクライアントを実装する際に利用でき

るようにしている。ライブラリは各状態に対応する関数を設けることによってモジュール化を進め、今後の仕様の変更や拡張に容易に対応できるようにした。

2.3.7　クライアントの評価

サーバが動作した OS のすべてにおいてクライアントも動作した。サーバと同様に基本的な部分に関しての移植は容易である。しかしネットワークインタフェースにアドレスを設定する方法や、ルーティングテーブルを設定するための方法などが同じ UNIX でも系統によって異なる。

2.3.8　リレーエージェントの設計・実装・評価

リレーエージェントの実装方針もサーバやクライアントと同様である。したがって All 1 のブロードキャストを用いたり、どのネットワークインタフェースから DHCP メッセージを受信したかを認識したりする必要がある。リレーエージェントも簡潔なテキストファイルを記述することによって設定できるようにする。

リレーエージェントは C 言語で約 950 行である。クライアントやサーバと同様に BPF や NIT を使用している。ただしクライアント・リレーエージェント間は All 1 のブロードキャストを使用するが、サーバ・リレーエージェント間は通常のユニキャストで通信すれば良いので、BPF(NIT) と socket の双方を使用している (前掲の図 2.2参照)。

複数のサーバが存在する場合に信頼性を向上させるため、リレーエージェントはクライアントから受けとった単一の DHCP メッセージを複数のサーバに送ることができる。これによりどれか一つのサーバに障害が発生していても、残ったサーバが処理をすることができる。したがって DHCP 機構の信頼性が高くなる。

設定ファイルには、DHCP サーバの IP アドレスを列挙する。また信頼性を向上させるために、いくつのサーバへ転送を行なうかを指定する。もちろん複数のサーバへ転送を行ないたくない場合には 1 を指定すればよい。複数サーバへ転送する場合にはクライアントから受信した DHCP メッセージを元にハッシュを行ない、転送先を選出する。このように転送先を分散させることによって特定のサーバへ負荷が集中しないように負荷分散を試みている。

“ユーザの使いやすさ” に関してはテキストファイルによる設定という方法で考慮したものの、設定の柔軟さの点でやや課題が残る。リレーエージェントもサーバ同様に大規模環境での動作テストは行なっていない。

2.3.9　相互運用性テスト

DHCP に関しての議論は Internet Engineering Task Force (IETF) の Internet Area における DHC Working Group で行なわれている。メーリングリストによる通常の議論の

ほかに、有志による相互運用性テストが 1993 年 8 月、1993 年 10 月 29 日、1994 年 1 月 19 日の 3 回開催されている。その目的は独立に行なわれた各人の実装を持ち寄り、実際にテストを行なうというものである。本実装もシアトルで開催された第 2 回相互運用性テストにて実験を行なった。

当日は 7 つのサーバと 12 のクライアントが参加した。参加組織を以下に示す。オペレーティングの種類が特に記述されていないものは UNIX と思われる[7]。

1. WIDE ‥ サーバ、クライアント、リレーエージェント
2. Microsoft ‥ WINDOWS NT サーバおよびクライアント、DOS クライアント
3. Sun Select ‥ サーバ、クライアント
4. HP ‥ クライアント
5. Boeing ‥ サーバ、クライアント
6. DEC ‥ クライアント
7. SGI ‥ サーバ、クライアント
8. Competitive Automation ‥ サーバ、クライアント
9. FTP Software ‥ WINDOWS サーバ、OS/2 サーバ、WINDOWS クライアント、DOS クライアント

テストの結果、本実装は非常に良好な結果が得られた。また仕様への忠実度に関しては極めて厳密な部類に属していた。そのため他の実装との相性は良かったといえる。

2.4 移動計算機への応用

本節では移動計算機サポートプロトコルの一つである Virtual Internet Protocol (VIP) への DHCP の適用について述べる。最初に DHCP を VIP へ組み込む方法を提案し、クライアントライブラリを利用した実装の解説と評価を行なう。また実際にこれらを運用した実験の結果についても解説する。

2.4.1 DHCP の VIP への応用

ここでは、一時アドレスを自動的に取得してくる DHCP クライアントを外部から制御するためのプログラムである vipd を設計し、その実装と評価を行なう。今回、我々の設計したライブラリはシグナルによって非同期にアドレス確認 (verify) を行なうことができるため、vipd に非常に適している。また抽象化のレベルが大きいためにクライアントの内部構造をほとんど意識する必要がなく、DHCP クライアントと vipd の制御構造との切り分けが明確に行なえる。

[7] 具体的なオペレーティングシステムは確認できなかった。

図 2.4は vipd の中心部分である。全体として vipd はポーリングを行なう親プロセスと、DHCP のクライアントである子プロセスの二つが協調して動作する。親プロセスは子プロセスを生成すると無限ループに入る。ポーリングを行なって何のパケットも受信できないと子プロセスにシグナルを送る。シグナルを受けとった子プロセス (DHCP クライアント) は状態遷移を起こし、確認をおこなうための VERIFY 状態に移る。もちろん VERIFY 状態に移るべきでない場合 (例えばシグナルが届いた時に SELECTING 状態だった場合など) は遷移しない。逆にシグナルを利用することで有効期限の延長を行なっている最中であっても確認を行なえるようになっている。

```
if ((childpid = fork()) < 0)
  exit(1);
if (childpid == 0) {
  if (execl(dhcpc_path, dhcpc_name,
      "-d", argv[0], argv[1], (char *) 0) < 0) {
    exit(1);
  }
}
while(1) {
  sleep(INTERVAL);
  if (wait3(NULL, WNOHANG, NULL) == childpid) {
    syslog(LOG_WARNING, "DHCP client had died");
    exit(1);
  }
  if (polling(argv[0]) != 0) {
    kill(childpid, SIGUSR1);
  }
}
```

図 2.4: vipd の中心部分

2.4.2　vipd の評価

今回実装した vipd はおおむね良好な結果を示した。シグナルを使ったライブラリを利用することで、vipd は DHCP の有効期限を遵守することができる。また DHCP クライアントと vipd を別プログラムにすることによって制御構造が非常に簡潔にまとまっている。

現在は BPF や NIT によるポーリングによってしか、ネットワークの変化を検出できないうえに、実際に (真に) ネットワークが変化したかどうかは不確定である。しかしながら本クライアントライブラリは VERIFY 状態への遷移が非同期なシグナルにより行なえる。したがって将来的にネットワークデバイスのドライバを拡張できたような場合には、*polling()* 関数を変更するだけで容易に対応可能である。

2.4.3 実験結果

我々が実装した vipd や DHCP を用いて、第 28 回 IETF ミーティングおよび IP ミーティング'93 にて実験デモを行なった。実験には DHCP サーバ、DHCP リレーエージェントの他に、VIP を実装した BSD/386 マシン (vipd を使った DHCP クライアント) と DOS マシン (BOOTP クライアント) を用いた。そのときのネットワーク構成を図 2.5に示す。この実験デモにより DHCP サーバと DHCP リレーエージェントが安定して動作すること、KA9Q 上の BOOTP クライアントと DHCP サーバの組合せでも動作することなどが確認できた。デモでは図中の SONY NEWS から移動計算機 (BSD/386) に ico[8]のウィンドウを開き、ネットワークを移動しても ico が動き続けることを視覚的に示した。これは VIP によって移動透過性が保たれることと、DHCP によって (vipd によって) アドレスを自動的に取得できることによって初めて可能となる。KA9Q マシンでも同様にリモートログインの論理的通信路を維持したまま移動可能なことを示した (第 3.4.3節参照)。

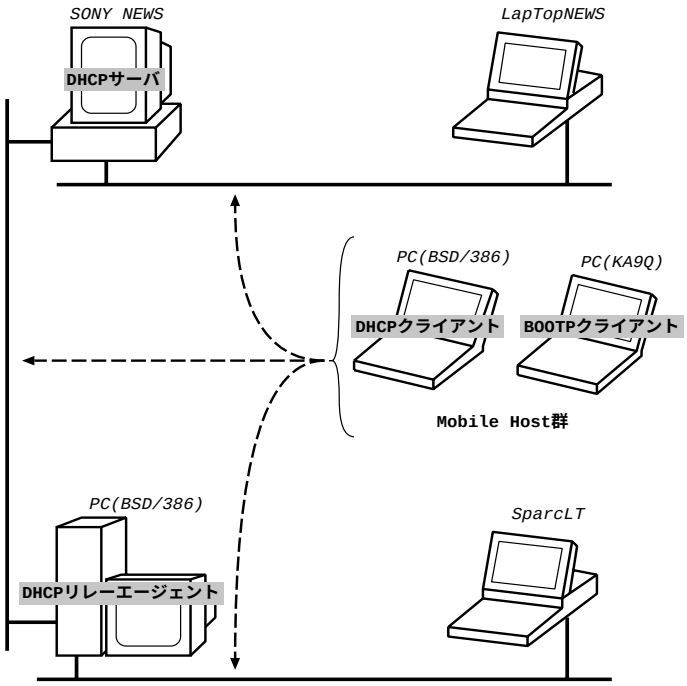

図 2.5: デモ用ネットワーク

このように DHCP と VIP を組み合わせることにより、ユーザがネットワークアドレス等の詳細を意識せずに、しかも作業を中断する事なく移動できる[9]。また任意の第三者が移動計算機に向かって新たに論理通信路を開く場合にも VIP アドレスさえ知ることができれば良く、相手の位置 (IP アドレス) を意識する必要がない。VIP は既存の IP ホスト

[8]X ウィンドウ上で動作するプログラム。多面体がウィンドウ内を動く。
[9]ただしアプリケーションによっては時間切れとして処理を終了してしまうものもある。

との互換性を保っている。したがって世界中のどこに移動しても一時的に使用する IP アドレスが取得できれば通信が可能となる。DHCP はインターネットにおいて広く提案されているため、今後世界中で利用可能となることが予想される。したがって VIP の持つ利点をそのまま活かすことができるだろう。

2.5　DHCP に関する考察

本節では現在の DHCP の持つ問題点と仕様上の限界を示す。それらの問題点や限界をクリアするためには複数の DHCP サーバ同士が協調して動作することが必要である。そのための仕組みとして DHCP サーバ間プロトコルについて考察する。

2.5.1　複数サーバ問題

DHCP ではサーバやリレーエージェントを一つのサブネットに複数設置することによってアドレス割り当て機構の信頼性を向上させることが可能である。しかし RFC1541 に規定されている仕様にしたがうと、サーバ同士が通信することができないために以下に説明するような問題が発生する。

1. DHCPOFFER と DHCPNAK の競合
2. DHCPACK と DHCPNAK の競合

前者は、クライアントが送信した DHCPDISCOVER に対し別々のサーバが DHCPNAK と DHCPOFFER を返送する場合である。クライアントは DHCPNAK を受けとるとその後に届く DHCPOFFER は無視してしまう。この場合、利用できるアドレスが存在するにもかかわらず DHCP クライアントはアドレスの割り当てを受けることを断念してしまうかも知れない。最新のドラフト [37] では「DHCP サーバは割り当て可能なアドレスがない場合でも DHCPNAK を返さない」と規定することによってこの問題を回避している。しかしながらその場合は逆に、両方とも割り当て可能なアドレスを持たない場合に DHCPNAK を受けとることはない。したがってタイムアウトと再送を繰り返し、最終的に DHCP の利用を断念するまでユーザは待たされてしまう。

後者は、クライアントが送信した DHCPREQUEST に対して別々のサーバが DHCPACK と DHCPNAK を返送する場合である。このような競合は稀にしか発生しないが、2 次記憶を使わない (情報を保存しておかない) クライアント[10]が存在するとこのような状況が頻繁に発生する可能性がある。

2.5.2　アドレスプール枯渇問題

現在の DHCP では、DHCP サーバはネットワークアドレスの集合を保持していて、クライアントから届くアドレス割り当ての要求を満たすことができると仮定する。したがっ

[10] 例えば X 端末などが考えられる。

てアドレスプールに割り当て可能なアドレスがなくなった場合のことについては触れられていない。このように現在の DHCP サーバはアドレスプールが枯渇しても補充することができない。また複数サーバ間でアドレスプールが重複する場合は考慮されていないため、アドレスプールは互いに独立した集合である。

近年のインターネットにおいて、IP アドレスは近い将来に確実に不足することが指摘されている。DHCP では使われなくなった IP アドレスをサーバが回収し、別のホストに割り当てることで IP アドレスの効率的な利用ができることになっている。しかし実際にはアドレスプールは管理者の手によって設定される固定した集合である。したがって既存の人手による管理に比べて格別に使用率が改善されることは考えられない。

2.5.3 サーバ間プロトコルに関する考察

複数サーバによる競合問題に関しては簡単な peer-to-peer のサーバ間プロトコルで対処可能と思われる。しかしアドレスプール枯渇問題や、ひいては IP アドレス自体の枯渇問題に対処するためには、階層的なサーバ群の構築が必須である。

階層的なサーバ間プロトコルを設計する際に障害となることがいくつか考えられる。第一に、通常のホストの取り扱いをどのように行なうかという問題があげられる。IP アドレスの効率的な利用を行なうためには IP アドレスの付け替えが発生すると思われる。インターネット上の全ホストが DHCP クライアントであり、しかも比較的短い有効期限を持つ動的割り当てである場合には、アドレスの付け変えもある程度自動的に行なうことができるだろう。しかしながら実際には通常のホストも存在するし、有効期限の長いホストも存在する。こういったホストの IP アドレスを付け変えるためには人間の介入が必要とされる。また IP アドレスの付け変えに伴って生ずる様々な影響 (例えば複雑な経路制御の記述や、Domain Name System への登録内容をどのように変更するかなどの問題が発生する) に、どのように対処するかという問題もある。

第二に、サーバ間プロトコルにおいて、どのようにアドレスプールのエントリを表現するかということが問題となる。現在の DHCP は一つのメッセージにつき一つのエントリに関する情報しか含まない。しかしながら多数のエントリを一度に交換する必要のあるサーバ間プロトコルでは、より効率的で強力な記述力を持つ表現方法を考案する必要がある。

また階層的にアドレスプールを管理する場合に、下位のサーバにアドレスプールを配分することになる。この際に上位サーバが、下位サーバに対して割り当てなどの方針 (Policy) を伝達する必要があるかも知れない。そのような方針をどのように表現して伝達するのかが問題となる。これらの諸問題に対する解決方法を考案することが今後の大きな課題である。

2.6　おわりに

インターネットにおいては絶えず計算機が接続と切断を繰り返す。インターネットに接続される計算機は IP アドレスなどのネットワーク上の共有資源を割り当てを受けて、設定に組み込まなければならない。逆にインターネットから撤去される計算機の使用していた共有資源は回収する必要がある。資源の割り当て作業や計算機の設定作業は煩わしいものである上に間違いを犯しやすい。したがってこれらの作業を集中的・自動的に行なう機構が求められていた。そのために提案されているのが DHCP である。

本報告では第一に、初めてライセンスフリーの形で DHCP の全てのモジュールを実装した。本実装は非常に機種独立性が高く、オペレーティングシステムのユーザ空間で動作する。これにより汎用性の高さを確保することが可能となり、複数の UNIX 上で動作する実装を行なった点が特徴的である。サーバやリレーエージェントについては大規模性や設定の柔軟性を確保することを目標とした。クライアントはユーザの個別の要求に柔軟に対応できるようにライブラリ形式の実装を行なった。本実装の完成度の高さは、アメリカで開催された第 2 回相互運用性テストにて実証済みである。

第二に、DHCP の応用例として VIP への適用を行なった。 DHCP と VIP は非常に相性が良く、機能を補完しあう。DHCP 単独ではごく初歩的な移動しかサポートできないが、DHCP を VIP と組み合わせることによって携帯型計算機の透過的かつ円滑な移動が可能となることを示した。その可能性を実証するために、動的に IP アドレスの取得を行なって設定を行なう vipd の実装と評価を行なった。世界中の任意の場所で IP アドレスが取得可能となる DHCP を応用することによって、VIP の有効性が向上した。

最後に現在の DHCP のもつ矛盾を示す事例を 2 つ指摘した。この問題は同一サブネットに複数のサーバが存在する場合に各々のサーバからの応答が互いに矛盾したものであるために発生する。これはサーバ間での情報の授受が行なわれない限り本質的に発生するものであり、問題を解決するためにサーバ間プロトコルが必要であるということを示した。

さらに IP アドレスの枯渇問題などのより複雑な問題に対処するためには、サーバ間プロトコルを階層的なものにする必要があることを示した。サーバ間プロトコルを設計する際には、通常のホストをどのように取り扱うのか、あるいは DNS との関係はどうなるのかという問題がある。またアドレスプールのエントリをどのように効率的に記述するかという問題も発生する。また最大の懸念である “サーバ間プロトコル” の設計を完了し、それをインターネットに提案することが望ましい。そのためには “サーバ間プロトコル” の実装を行なって評価する必要がある。本報告では、これらの問題点についても指摘し、今後の展望とした。

第 3 章

MS-DOS 上での移動計算機環境の実現

3.1 はじめに

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。MS-DOS は PC の事実上の標準オペレーティングシステムとなっている。コンピュータネットワークが普及するにつれて PC 上でもネットワークの機能が利用可能になり、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。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。

MS-DOS は機能を必要最小限に限定しているため、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。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。

3.2 KA9Q(KA9Q NOS パッケージ) の概要

3.2.1 MS-DOS でのネットワーク機能実装上の問題点

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く、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、タイマー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考えられる。

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。

後者の例として KA9Q というパッケージが存在する。後述するように KA9Q は簡易オペレーティングシステムとして機能し、疑似マルチタスクを実現している。このためネットワークの処理系を作るのに適している。本研究では KA9Q パッケージを利用することにした。

3.2.2 KA9Q の特徴

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 氏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。さら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。

3.2.3 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。またアドレス解決プロトコルである 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 3.1に示す。

表 3.1: 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

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

3.2.4 KA9Q の構造

KA9Q パッケージでは図 3.1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。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。

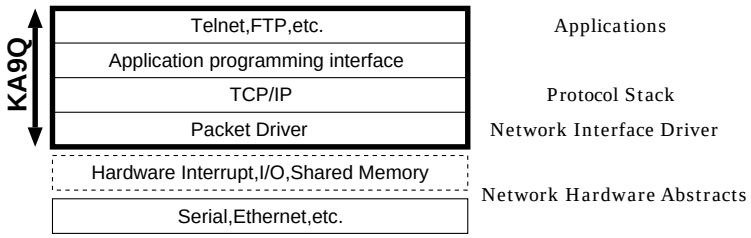

図 3.1: KA9Q のネットワーク構造

3.2.5　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 3.2参照)

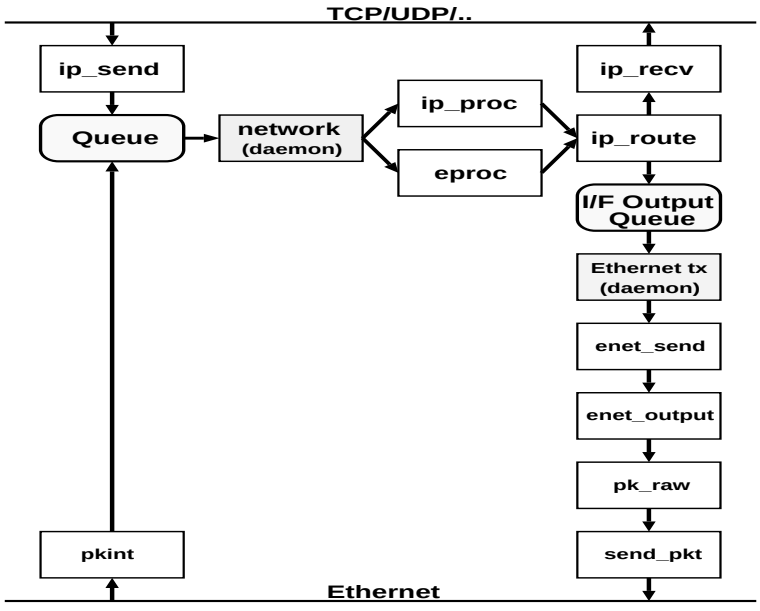

図 3.2: KA9Q の IP 関係ブロックダイアグラム (Ethernet の場合)

KA9Q では 2 つのデーモンプロセス (`network` および `ethernet_tx`) と 2 つのキュー (`queue` および `i/f output queue`) によって処理が成立している。受信・中継・送信を簡単にまとめると以下のようになる。

- 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())

- 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())

- 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())

3.2.6 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。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 3.3のような表示とプロンプトが出て入力待ちとなる。

ネットワークの設定に関しては UNIX で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いる。図 3.4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 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

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

net> _
```

図 3.3: KA9Q 起動時

```
 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 3.4: ネットワークの設定例

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している。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。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。

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 3.5)

```
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 3.5: session コマンドによるセッションの状態表示

セッションが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。) またコマンド close [session #] で FIN を送るとセッションは終了する (または、control-c)。

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。さらに 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の検査を行なうことができる。

3.3 VIP の MS-DOS への実装

3.3.1 設計

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 3.2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。

VIP では移動ホストがホームネットワーク宛てに一定間隔で *VipCAMT* を送信する。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。たとえばタイマーによって起動される vipslowsend() というデーモンは以下のように書くことができる。

- 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 3.6のように書く。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。

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

図 3.6: timer driven デーモンプロセスとするモジュールの記述例

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とき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。これを自動的に行なうために Bootstrap Protocol (BOOTP)[46] や Dynamic Host Configuretion Protocol(動的ホスト設定プロトコル)[47, 36, 48] 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され、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が取得される。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。

3.3.2 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 3.7に示す。太線で表した部分が今回追加した部分である。送信時には TCP/UDP 層から ip_send が呼ばれ、その

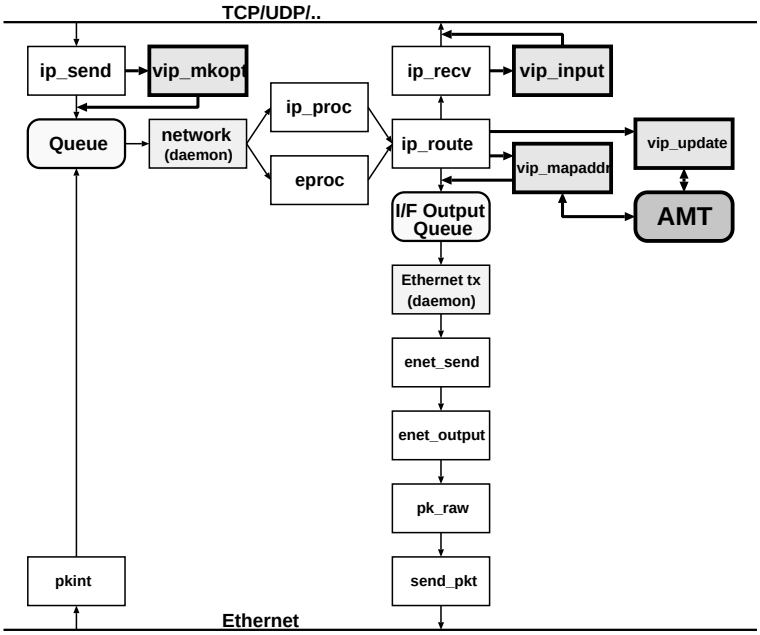

図 3.7: VIP 実装後の KA9Q のブロックダイアグラム

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に対し、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) でヘッダが付けられ、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。

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については、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。

3.4　評価と考察

3.4.1　KA9Q 版 VIP の現在の実装

当初 KA9Q の 911229 バージョンを使用して VIP を実装した。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、こちらにも VIP の実装は終了しており (KA9Q+VIP (940115))、現在試験中である。

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、バージョン 940115 (80386 コードコンパイル) では約 240 キロバイトとなっており、VIP を含まない場合と比較して約 4 キロバイト増加しただけである。ただ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。

3.4.2　VIP の実装状態

現在は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。

また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している。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である。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。しかし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。指定されたキーを押すことにより BOOTP クライアントを起動する。さらに回線との接続・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ている。AMT の確保の方法としてはファイルを利用することも考えられるが、検索と更新に高速性を必要とするため、適当ではない。また現在のところ、最大エントリ数は 256 としている。エントリ数は多いほどよいのであるが、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を図っている。

VIP は経路情報に関する処理は行なわない。したがって移動後の BOOTP によるパラメタ取得時、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経路情報が残留する。このため経路情報も消去するようにした。

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

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 3.8に示す(第 2.4.3節参照)。

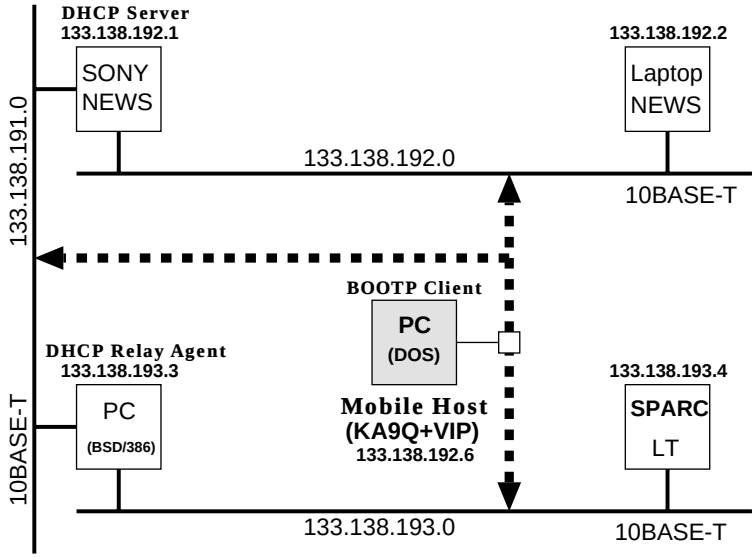

図 3.8: IETF'93 及び IPmeeting'93 での実験環境

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。DHCP に関係するホストが 2 つ存在している。1 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 1 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。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 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ときに VIP アドレスと IP アドレスとの対で構成される AMT エントリが作成され、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 アドレスから IP アドレスが解決され、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。

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。

3.4.4　考察

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。

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。

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。したがって切断・接続を自動的に検出し、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。

3.5 おわりに

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。MS-DOS はその構造上メモリやディスクの制約が少なく、負荷も軽い実行環境が得られる。したがって 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 を利用した。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。

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、取得できる情報が限られている。より多くの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と考えている。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。

第 4 章

IETF Mobile-IP プロトコル

4.1 はじめに

現在移動ホスト用のプロトコルの研究は企業の研究所や大学などで広くおこなわれているが、そのほとんどは Internet Protocol (IP) を対象とし、IP に何らかの機構を付加することによって移動を実現している。これらは Internet Engineering Task Force (IETF) の IP Routing for Wireless/Mobile Hosts WG (Mobile-IP WG) に提案され、多くの議論がなされてきた。しかしここにきてこれらの方式のうちから 1 つをインターネットの標準案として提示するのではなく、WG としての標準案を一本化しようという動きがあり、実際にドラフト [49] が作成された[1]。この方式は既存のネットワークに変更を加えずに移動ホストが通信できるようにしているため、プロトコル自体に数々の問題点を含んでいる。またネットワークプロトコルにおいては仕様上に整合性があっても実際に通信をおこなわなければ顕在化しない問題がある場合が多い。

本研究ではこの標準案を実際に実装し、プロトコルと実装の両方の問題点を明らかにした。さらに他の方式と比較して、インターネット標準として全世界で利用するのに最適であるかどうかを検討した。

4.2 Mobile-IP ワーキンググループ標準案

4.2.1 手続き

本プロトコルでは新たな手続きを定めることによって移動ホストが接続先のネットワークで一時的な IP アドレスを取得することなしに、通常のインターネット上のホストと正しく通信できるようにしている。また既存のホストやルータのソフトウェアに一切修正を加えないということも前提としている。この手続きは、ビーコン/ビーコン要求、登録、パケットのカプセル化と転送、経路の最適化、という 4 種類の機能に分類される。これらの手続きのうち、パケットのカプセル化と転送以外には User Datagram Protocol(UDP) が使用される。

[1] 1994 年 4 月に新しいドラフトが出されたが、パケットフォーマットなど細かい部分の変更のみであり、基本的な差異はない。

4.2.2 必要な構成要素

これらの手続きは 4 種類の構成要素が担当する。移動ホスト (Mobile Host)、ホームエージェント (Home Agent)、訪問先エージェント (Foreign Agent)、移動検知ホスト (Mobile Aware Host) と呼ばれるものである。以下にそれぞれの機能を示す。

移動ホスト 名前の通り移動してネットワーク上の接続点を変える。また移動ホストは恒久的に割り当てられた IP アドレスを識別子として持ち、これはネットワーク上の接続点が変化しても変わらない。この IP アドレスはホームアドレスと呼ばれ、通常の経路制御で正しくパケットが送られてくる場合、移動ホストはホームネットワークにいるという。

訪問先エージェント これも移動しないという前提のホストまたはルータであり、以下のような機能を持つ。

1. 定期的にビーコンを出し、移動ホストが接続してくるのを待つ
2. 移動ホストの接続要求を受け、ホームエージェントへ通知する
3. ホームエージェントからカプセル化して送られてきた移動ホスト宛のパケットを脱カプセル化して転送する

ホームエージェント 移動しないという前提のホストまたはルータであり、以下のような機能を持つ。

1. 移動ホストへの経路情報をアナウンスする
2. 移動ホストの IP アドレスと、その移動ホストが現在サポートされている訪問先エージェントの IP アドレスの組合せの書かれたテーブルを管理する
3. 移動ホストがホームネットワークにいないとき、その移動ホスト宛のパケットを訪問先エージェントへカプセル化して転送する

移動検知ホスト 移動ホストと現在それがサポートされている訪問先エージェントの組合せのテーブルを持ち、移動ホスト宛のパケットは訪問先エージェントへ直接カプセル化して送る。またこの構成要素も移動はしないことが前提となる。

4.2.3 ビーコン/ビーコン要求

移動ホストがネットワークに接続して通信できるようになるためには、訪問先エージェント、あるいはホームエージェントに登録要求を出さなければならない。そのためには移動ホストはエージェントの IP アドレスを知らなければならない。エージェントはサブネット内に定期的にビーコンを流して移動ホストが接続可能であることをアナウンスしている。

また、移動ホストは接続先のサブネットに訪問先エージェント、またはホームエージェントが存在するかどうかを確認する手段をもつ。これは例えばビーコンの間隔が長いとき、仮に 30 秒である場合、移動ホストはエージェントの IP アドレスを知ってから登録

作業にはいるまで最悪 30 秒待たされることになる。このような事態を避けるため、移動ホストはサブネット内に存在するエージェントに対してビーコンを要求できる。エージェントはビーコン要求メッセージが来た場合、ただちにビーコンを出さなければならない。

4.2.4　登録

登録作業とは、移動ホストに関する情報をホームエージェント、訪問先エージェント、移動ホストの間で交換することである。具体的には移動ホストのホームアドレスと現在それをサポートしている訪問先エージェントの組合せの書かれたテーブルを正確な状態に保つことである。これにより通常のホストが移動ホストのホームアドレス宛にパケットを送っても通信が適切におこなわれるようになる。

ここで最も一般的な事例として、移動ホストがホームネットワークを離れて他のネットワークに接続するという状況を考えてみる。登録作業は図 4.1にあるように 4 種類のメッセージを交換しておこなう。

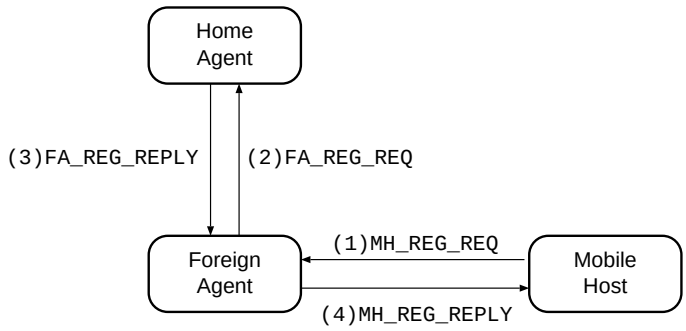

図 4.1: 登録時のメッセージ交換

1. 移動ホストは MH_REG_REQ メッセージを訪問先エージェントに送り、登録作業を開始する。
2. 訪問先エージェントは FA_REG_REQ メッセージをホームエージェントに送り、移動ホストのサポートの開始について許可を求める。
3. ホームエージェントは訪問先エージェントに対して FA_REG_REPLY メッセージを送り、移動ホストのサポートの開始を許可するか拒否するかを通知する。
4. 訪問先エージェントは移動ホストに対して登録作業の結果を MH_REG_REPLY メッセージとして通知する。

この登録作業が成功した場合、ホームエージェントと訪問先エージェントの両方に移動ホストと現在それをサポートしている訪問先エージェントの組合せのテーブルが作成される。ホームエージェントのテーブルを “転送先リスト”、訪問先エージェントのテーブ

ルを “訪問者リスト” と呼び、2 つは同じ内容である。この時点でようやく他のホストは移動ホストと通信が可能になる。また訪問先エージェントは今までサポートしていた移動ホストが別のネットワークへ移動した場合、新しい訪問先エージェントと移動ホストの組合せを “位置情報リスト” と呼ばれるもので管理する。

訪問先エージェントが自分宛のパケットを受けとった場合、この “訪問者リスト” と “位置情報リスト” に従って以下の 3 種類のふるまいをする。

- 脱カプセル化して中のパケットを “訪問者リスト” にある移動ホストに配送する。
- “位置情報リスト” にある以前ここにいた移動ホストへ転送するために、現在その移動ホストを管理している別のネットワークの訪問先エージェントへ再びカプセル化して転送する。
- 取り出したパケットを何もせずにそのまま通常のルーティングテーブルに従って転送する。

ホームエージェントはルータとして働かなくてはならない。その理由は移動ホストへの経路情報を流し、もし移動ホストがホームネットワーク以外のところで接続していれば移動ホスト宛のパケットを横取りしてカプセル化して訪問先エージェントに転送しなければならないからである。ホームエージェントの接続形態は図 4.2のようなものが考えられる。またホームネットワークは物理的なサブネットワークでも仮想的なサブネットワークでもよい。しかし、(a) の形態でのメッセージの交換については現在のところドラフトにおいては何も述べられていない。

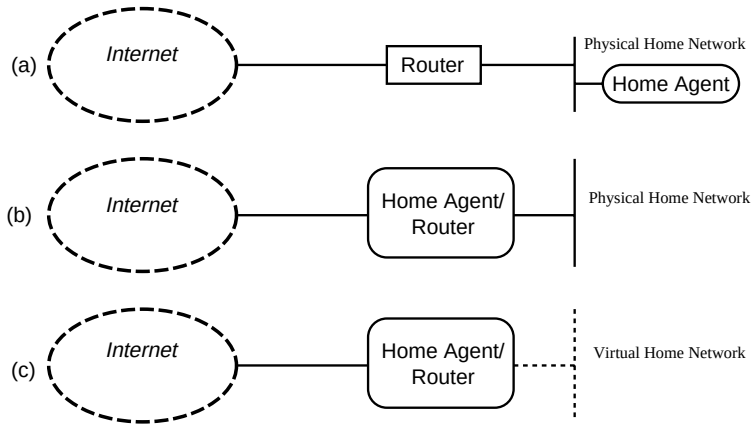

図 4.2: ホームエージェントの接続形態

- (a) のようにインターネットとは通常のルータで接続され、ホームエージェントは物理的なサブネットに接続されている 1 つのホストである場合
- (b) のようにホームネットワークのルータがホームエージェントも兼ねている場合
- (c) のようにホームネットワークが仮想的なサブネットであり、移動ホストが実際に接続できるホームネットワークがない場合

4.2.5　パケットのカプセル化と転送

通常のホストが移動ホストと通信する場合を例として考える。移動ホストがホームネットワーク以外のところに接続され、なおかつ登録作業が完了している場合、具体的には以下のような手順 (図 4.3参照) で通信が行なわれる。

1. 通常のホストは移動ホストのホームアドレスへ向けてパケットを送り出す。
2. ホームエージェントが移動ホストへの経路情報を流しているため、通常の経路制御プロトコルにしたがってパケットはホームエージェントに到達する。
3. ホームエージェントはパケットを受けとった後、“転送先リスト” に従ってカプセル化して訪問先エージェントへ向けてパケットをトンネリングする。

   (もし移動ホストがホームネットワークに接続されていた場合はそのまま配送する。しかし、移動ホストがホームネットワークに存在せず、“ 転送先リスト” にも登録されていなかった場合は、パケットを配送することができない)
4. 通常の経路制御プロトコルによってカプセル化されたパケットは訪問先エージェントに配送される。
5. 訪問先エージェントは脱カプセル化して “訪問者リスト” に基づいて、同じネットワークに接続されている移動ホストへ転送する。

ここで経路最適化機能が実装されていれば、直前まで移動ホストをサポートしていた訪問先エージェントが、新しく登録した別の訪問先エージェントへパケットを転送することも可能である。

4.2.6　通常のホストと移動ホストの通信例

図 4.3に通常のホストと移動ホストが通信する場合のパケットの流れを示す。ホスト H

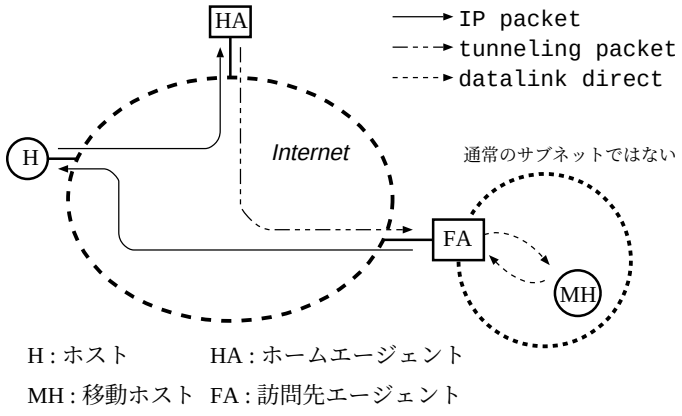

図 4.3: 通常のホストと移動ホストの通信

が移動ホスト MH へパケットを送信する場合、H は宛先アドレスとして MH のホームアドレス指定する。MH の ホームエージェントである HA が MH のホームアドレスへの経路情報をアナウンスしているので、このパケットは HA に配送される。HA はこのパケットが現在 FA という訪問先エージェントのサポートを受けている MH 宛のものだと知り、これを別の IP ヘッダでカプセル化して FA へ送信する。このような IP パケットを別の IP パケットのデータ部としてカプセル化し別のホストへ転送することを一般的にトンネリングという。FA はこのパケットを受信すると脱カプセル化して元のパケットを MH に転送する。また、MH から H へのパケットは、MH から見ると FA がデフォルトのルータとして認識されているので、FA へむけて送信する。FA は通常の経路情報に従ってこのパケットを中継する。以上のように移動ホストとの通信ではパケットは H、HA、FA の三角形の経路をたどる。

4.2.7 経路の最適化

経路最適化の 1 つの方法として、移動検知ホストに対してホームエージェントが移動ホストと現在それを管理している訪問先エージェントの組合せを通知することが可能である。これによって移動検知ホストは移動ホスト宛のパケットを直接カプセル化して訪問

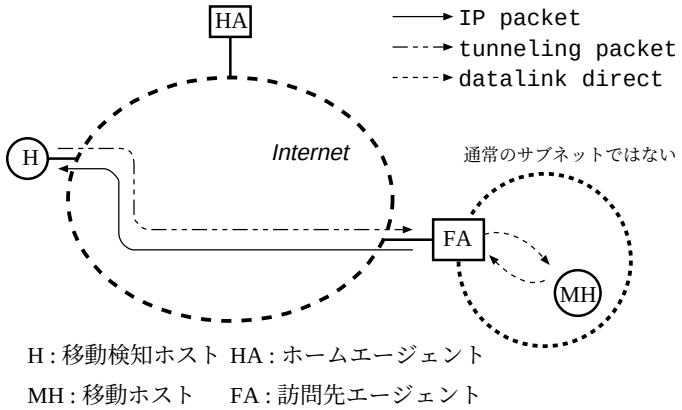

図 4.4: 移動検知ホストと移動ホストの通信

先エージェントに送ることができ、ホームエージェントを経由しなくて済む。よって図4.4に示すようにパケットは三角形の経路をたどらずに、最適な経路をたどることができる。また移動ホストが新たに違うネットワークに接続して登録作業をはじめるとき、新しい訪問先エージェントは直前にサポートしていた訪問先エージェントに自分が新しい訪問先エージェントであることを通知する。これにより以前の訪問先エージェントは、移動ホストと通信していた移動検知ホストが新たなネットワークへ移動したことを知らずに直接パケットをカプセル化してきた場合、新しい訪問先エージェントに再びカプセル化してパケットを転送することができる。

4.2.8　認証の枠組

ドラフトには認証の具体的な手続きについては一切述べられておらず、認証の手続きのため以下のような枠組だけが用意されている。今後これらの枠組に基づいて認証の方法を決定しなければならない。

- ビーコン/ビーコン要求をおこなう場合に対しては認証は全く考慮しない。
- 登録時にはすべてのメッセージにおいて自分で認証を行なう。つまり認証に必要な情報がメッセージに含まれているため、相手に確認をとるという形の認証はおこなわない。
- 経路最適化の過程においては、相手と情報を相互に交換することによって認証をおこなう。

4.3　実装

本研究では 4.3 BSD UNIX オペレーティングシステムを基盤とした BSD/386 というオペレーティングシステム上に IETF Mobile-IP プロトコルを実装した。本節ではこの実装について具体的に説明する。

4.3.1　設計方針

ホストの移動を実現するためのこのシステムの 4 種類の構成要素のうち今回実装した 3 種類、すなわち移動ホスト、ホームエージェント、訪問先エージェントに共通する設計方針を以下に挙げる。

1. 移植性を高くする
    - ユーザプロセスとして実装する
    - カーネルの変更を最小限にする
2. 既存の通信の仕組みを利用する

今回の実装の目的は、このプロトコルがきちんと動作するかを検証し、プロトコル自体の問題点を指摘することである。またこのプロトコルの仕様の背景には既存のネットワークへの変更を最小限にするという目標がある。したがって今回はオペレーティングシステムになるべく変更を加えないで実装するという方針をとった。

また将来的には各サブネットごとに訪問先エージェントが存在しなければならない。そのためには特定のオペレーティングシステムに依存する実装は、普及に時間がかかるという欠点になるため避けなければならない。いいかえれば移植性が高い実装をおこなわなければならないということである。よって可能なかぎりユーザプロセスとして実装し、ユーザプロセスでは不可能なトンネリング機構を実現する部分のみカーネルに変更を加えた。

4.3.2 移動ホストの実装

移動ホストとエージェント間でのビーコン/ビーコン要求にはパケットフィルター(BPF)という機構を利用した。BPF によりイーサネットのパケットを直接読み書きすることが可能なので、デーモンはビーコンメッセージから訪問先エージェントの IP アドレスとイーサネットアドレスの両方を得ることができる。デーモンはそれらを得た後以下に述べる動作をする。この結果移動ホストから外部に送信されるすべてのパケットは訪問先エージェントに配送されることになる。

1. ARP テーブルとルーティングテーブルのエントリをすべて消去する。
2. ARP テーブルに訪問先エージェントのエントリをセットする。
3. ルーティングテーブルのデフォルトの経路を訪問先エージェントにする。
4. インターフェイスに NOARP フラグを立て ARP 要求/応答をおこなわないようにする。

以上の設定をおこなえば、移動ホストは訪問先エージェントとは通常の方法で通信をおこなうことが可能になるため、登録作業は通常のソケットを使用する。現状では認証の方法が規定されていないため、認証は全くおこなっていない。またこのプロトコルではホームエージェントなどのアドレスが可変長であっても対応できるように設計されているが、インターネットの現状を見るかぎり IP アドレス以外のアドレス体系を考慮することはあまり現実的ではない。そこで実装に際しては処理効率をあげるためアドレスの長さは 4 オクテットの固定長としている。

登録に成功するとホームエージェントと訪問先エージェントのそれぞれが移動ホストをサポートすることが可能な期間を得ることができる。そこでその期間を過ぎた場合再び登録作業を開始しなければならない。デーモンはこの期間の管理もおこなっている。

4.3.3 訪問先エージェントの実装

訪問先エージェントはデーモンとして実装した。ビーコン/ビーコン要求は以下のように行なう。訪問先エージェントは移動ホストのサポートが可能であることネットワーク内にアナウンスするためビーコンを定期的に送信している。これは移動ホストのビーコン要求メッセージの実装と同様に宛先アドレスが “255.255.255.255” のブロードキャストをしている。前述した理由により BPF を使用する。また移動ホストからのビーコン要求メッセージに対してすぐにビーコンメッセージを送信する。さらにビーコンメッセージ内の`INCARN_NUM` フィールドの値をファイルに記録しておき、再起動するたびにその値を増やして送信する。

登録作業は以下のように行なう。まず移動ホストからの`MH_REG_REQ` メッセージを受信して、`FA_REG_REQ` メッセージをホームエージェントに送信する。ホームエージェントのアドレスは移動ホストの`MH_REG_REQ` メッセージ内に記されている。ホームエージェントでの認証の結果や移動ホストをサポートする期間が`FA_REG_REPLY` メッセージとして

返ってくるのでその結果とホームエージェント、訪問先エージェントの移動ホストをサポートする期間などをMH_REG_REPLY メッセージとして移動ホストに返答する。ここではMH_REG_REPLY メッセージを移動ホストに正しく返すために、MH_REG_REQ メッセージはBPF を使用して移動ホストのイーサネットアドレスを得なければならない。MH_REG_REQメッセージを受信してからFA_REG_REQ メッセージを送信するまでの訪問先エージェントの動作を以下に示す。

1. MH_REG_REQ メッセージを BPF で受信する
2. ARP テーブルに移動ホスト用のエントリを加える
3. ルーティングテーブルに移動ホストの IP アドレスを到達コスト 0 で設定する
4. FA_REG_REQ メッセージを送信する

移動ホストのホームネットワークのエントリがルーティングテーブルにある場合でも、同時に移動ホストのエントリがあればそちらが優先されるので通信が可能になる。

訪問先エージェントとホームエージェントのトンネリング機構に関しては、現在 WIDE プロジェクトで開発中の DDT[50] というトンネリング用の仮想的なネットワークインターフェイスを使用した。これは IP パケットを IP パケットでカプセル化し、外側の IP パケットのプロトコルフィールドには 94 番 (IP within IP)[51] を用いている。

4.3.4　ホームエージェントの実装

ホームエージェントも訪問先エージェントと同様にデーモンとして実装した。移動ホストへの経路をアナウンスする機能は、Routing Information Protocol (RIP) または Open Shortest Path First (OSPF) という経路制御情報を扱うプロトコルがそのまま利用できる。実装としては routed や gated などが広く一般的に使用されており、何も特別なことはおこなわずにそのまま利用した。

訪問先エージェントと同様に、移動ホストがホームネットワークに戻ってきて再び接続する場合に備えてビーコンメッセージを定期的に送信する。ビーコンメッセージを送信する方法は訪問先エージェントと同様に BPF を使用している。

登録もほぼ訪問先エージェントの場合と同じ方法でおこなっている。ただMH_REG_REQ メッセージを受信したあと、FA_REG_REQ メッセージを送信する必要がないという点のみが異なっている。またMH_REG_REPLY メッセージを返す必要があるので、MH_REG_REQ メッセージは BPF で受信して移動ホストのイーサネットアドレスを得なければならない。MH_REG_REQ メッセージを受信した後MH_REG_REPLY メッセージを返答するためにホームエージェントは ARP テーブルにその移動ホストのエントリを加える必要がある。ここで訪問先エージェントの登録作業と異なるのはルーティングテーブルにはエントリを加える必要がない点である。その理由はホームネットワークに戻ってきた移動ホストとホームネットワークへの経路が異なったインターフェイスになるという可能性は考えられないからである。

また訪問先エージェントから送信されたFA_REG_REQ メッセージに対して、移動ホストの認証をおこなってからFA_REG_REPLY メッセージを返すが、現在の実装では認証の方法が未定義であるために認証は全くおこなっておらず、DDT インターフェイスの設定をおこない移動ホストのサポートが可能であるかのみを判定している。この部分は認証の方法が確定され次第実装をおこなう予定である。

さらにルーティングテーブル内の訪問先エージェントのエントリには DDT インターフェイスの一つを指定する。DDT インターフェイスの設定は、ホームエージェントと訪問先エージェントの間で整合性がなければならないので登録時におこなっている。

現在ホームネットワークにいない移動ホスト宛のパケットのカプセル化は DDT インターフェイスがおこなう。これはルーティングテーブルさえ登録時にきちんと設定されていれば移動ホスト宛のパケットは問題なくカプセル化されて訪問先エージェントへ転送される。

4.4 考察

本節ではプロトコル自体が持つ問題点、またそれを実装する場合の問題点を詳細に検討する。

4.4.1 訪問先エージェントの存在

プロトコルの仕様上当然ではあるが、移動ホストが接続したネットワークには必ず訪問先エージェントが存在しなければならない。しかし将来的には接続先のネットワークに訪問先エージェントが存在しない場合にも DHCP[36] を利用して動的にアドレスを割り当ててもらい、移動ホスト自体が専用訪問先エージェントとなる可能性も考えられている。

4.4.2 サブネットモデル違反

移動ホストとそれを受け持つ訪問先エージェントが接続されているネットワークはサブネットモデルに違反している。つまり単一の物理ネットワークに異なる net-ID を持つホストが同時に複数存在することになり、通常の経路制御が利用できなくなる。しかしサブネットモデルは現在非常に一般的に利用されているが、Request For Comments (RFC) などの公式な文書などでサブネットの定義がなされているわけではない。ここでは違反と呼んでいるが、実際に規約があってそれに反しているというわけではなく、ただ現在の一般的な慣習に反しているだけであるという点に注意したい。

4.4.3　冗長な経路

通常のホストと移動ホストが通信をする場合、移動ホストから通常のホストへのパケットは普通の経路をとるが、通常のホストから移動ホストへのパケットはホームエージェントを経由する。そのため相互に通信する場合にはパケットの経路が三角形となり、ネットワークの回線容量が小さい場合や転送時間の長いの通信においてはかなりのオーバーヘッドになることが予想される。一概にはいえないが Round Trip Time (RTT) というパケットが自局から送信されて戻ってくるまでの時間は平均して通常の 1.5 倍程度にはなるはずである。単純計算ではあるがホスト間の通信にかかる時間を 1 とすると、通常の通信では往復時間は 2 かかることになるが、本方式においてはホストとホームエージェント、ホームエージェントと移動ホストの間の通信に 2 かかるため、往復時間は合計 3 かかるという理由からである。

4.4.4　耐故障性

通常の 1 対 1 の通信においては、自分とその通信相手の間の経路が切れていた場合は通信できなくなる。しかしこのプロトコルにおいて移動ホストと通常のホストが通信する場合には、通常のホストとホームエージェント、ホームエージェントと訪問先エージェント、訪問先エージェントと通常のホストの 3 箇所の経路のうち 1 つでも切れていたら通信が不可能になる。つまり通常の通信に比べて耐故障性が 3 分の 1 になってしまう。これに対処するために移動検知ホストが定義されているが、移動検知ホストと移動ホストが通信する場合でも最初のパケットの交換は三角形になり、耐故障性が充分高いとはいいがたい。

4.4.5　セキュリティ

ドラフトでは認証の枠組だけがあげられているのみで、具体的な認証方法については全く定められていない。移動ホストでスーパーユーザになれる場合、IP アドレスを変更することは非常に容易である。これはネットワーク上での一意の識別子を変更することが可能であり、つまりは他のホストになりすますことができるということである。登録時には相手と情報をやりとりすることなくパケット内に含まれるデータのみで認証をおこなうとあるが、本方式でなりすましを防ぐことが可能なのかという点については疑問が残る。現在認証の方法についてはメーリングリストにおいて活発な議論が交わされている。

4.4.6　ファイアーウォール

現在企業などではインターネットとの接続を 1 点とし、そのホストをファイアーウォールとして運用しているところが多い。これは機密漏洩を防ぐために、インターネットの外側のホストから企業内部のホストへのアクセスを防止し、企業内部のホストからも直

接インターネットへアクセスできないようにするために、ゲートウェイのホストまたはルータのパケット中継機能 (IP Forwarding 機能) を無効にするというものである。

本来は情報の流通を自由にして、重要な情報は暗号化して通信するのがあるべき姿であると考えるが、現在の IP では暗号化をはじめとするセキュリティ機能がほとんど備わっておらず、また暗号化のためのオーバーヘッドがかなりあるため効率が悪い。このため次善の策として企業ではファイアーウォールのような方法をとらざるをえないのが現状である。

ここで、ある移動ホストが企業内のネットワークに接続することを想定する。たまたまそのネットワークに訪問先エージェントが存在しても企業内のネットワークから外部には出られないためホームエージェントと直接通信ができず、登録に失敗することになる。また逆にある企業内の移動ホストが、外部のネットワークに接続して、そこの訪問先エージェントがホームエージェントに登録しようとしても、ファイアーウォールがパケットを通過させないため、同様に登録に失敗する。

結局、通常のネットワークをホームネットワークとする移動ホストは、ファイヤーウォール内のネットワークに接続することはできず、逆にファイアーウォール内にホームネットワークを持つ移動ホストは、その組織内しか移動できないことになるので、ホストの移動という点に関してはファイアーウォールという運用形態はあまり好ましくない。

以下は実装上の問題点である。

4.4.7 DDT インターフェイスの問題点

DDT インターフェイスは現在開発中でありそれ自身多くの問題点を抱えている。しかしここではこのプロトコルの実装に使用した場合の問題点について述べる。まず DDT インターフェイスの数はカーネルの構築時に決定されるため、動作中に動的にインターフェイスの数を増やすということが不可能である。例えばホームエージェントの DDT インターフェイスの数が 10 個である場合、移動ホストがサポートを受けることのできる訪問先エージェントも 10 箇所までとなる。よって多くの移動ホストがあちこちに接続した場合対応しきれない可能性がある。このような場合あらかじめ移動ホストの数だけ DDT インターフェイスを作成しておけばこのような事態は避けられるが、インターフェイスの数が多いとその分ルーティングテーブルもおおきくなりパケット転送時のオーバーヘッドとなる。訪問先エージェントでも事態は同様である。あるネットワーク内でサポートできる移動ホストの net ID の数は、そのネットワークに存在する訪問先エージェントの DDT インターフェイスの数に依存する。

また登録時にホームエージェントと訪問先エージェントの両方の DDT インターフェイスが正しく設定されなければトンネリングができないので、片方のエージェントの DDT インターフェイスが足りない場合はトンネリングが不可能となり移動ホストの登録はできない。

これらの問題はオペレーティングシステムに依存する。今回実装をおこなった BSD/386

では動的にインターフェイスを増やしたりすることは不可能であるが、他のオペレーティングシステムにはこのような動作が可能なものもあり、その場合上記の問題は生じない。

4.4.8　エージェントと移動ホストの通信

今回の実装では移動ホストと訪問先エージェントの通信のためにルーティングテーブルと ARP テーブルを書き換えるという方法をとり、従来の通信の仕組みをそのまま利用した。この実装方法だとカーネルの変更が必要ないためデバッグの時間を減らすことができた。また移植も比較的容易におこなえるはずである。また移動ホストの数にかかわらずインターフェイスの数は変わらないため実行時のオーバーヘッドがないという利点がある。しかし既存の通信方法をそのまま利用したためにプロトコルのモデルとの対応が悪いという欠点があげられる。

また一方で、イーサネットを 1 対 1 通信を疑似的におこなうインターフェイスを作成するという方法も考えられた。この方法だとエージェントと移動ホストとの間のデータリンク層が抽象化されておりプロトコルのモデルとの対応がよいことがあげられるが、イーサネットインターフェイスを改造して 1 対 1 通信を疑似的に実現するインターフェイスを新たに開発しなければならない。またそのインターフェイスは通常のイーサネットインターフェイスとハードウェアを共有しなければならないため非常に複雑になることが予想される。また上記のような実装を行なった場合、インターフェイスの数がカーネル構築時に規定されるため、前述した DDT を使用した場合の問題と全く同じ問題が発生することになる。

4.4.9　ホームネットワークでの通信

このプロトコルにおいては移動先のネットワークに接続して通信する場合よりも、特にホームネットワーク内での通信に問題が多い。

まず通常のホストが移動ホストへパケットを送る場合を考える。ARP テーブルに移動ホストのイーサネットアドレスがない場合 ARP 要求パケットをブロードキャストする。しかし移動ホストは登録後はインターフェイスに NOARP フラグが立っているため、ARP 要求に対して返答が不可能になる。そこで代理 ARP という手法を用いてホームエージェントが代わりに返答してやる。移動ホスト宛のパケットはホームエージェントに送られ、ホームエージェントが移動ホストへパケットを転送してやるという手順をふむ。つまり同一ネットワーク内であるにも関わらず、移動ホストと通常のホストは直接通信することは不可能である。よって通常の 2 倍のトラフィックが生じることになる。

また移動ホストが ARP 要求に答えるようにインターフェイスを設定した場合には、通常のホストに移動ホストのエントリがキャッシュとして残るが、エントリが残ったまま移動ホストが移動した場合、ARP テーブルのエントリが消去されるまで移動ホストへのパケットが送信できなくなる。通常 ARP テーブルのエントリは 20 分で消去される。また移動ホストが他のネットワークに移動中の場合も ARP 要求に答えるホストがないため

通信ができないことになる。

よって通常のホストから移動ホストへの ARP 要求はホームエージェントが答えなければならないが、現在のところ代理 ARP をおこなうデーモンの実装は Network Interface Tap(NIT) という BPF とは別の種類のパケットフィルターを使用するものしか存在していない。このデーモンを BPF 上で動作するように現在移植作業をおこなっている。

4.4.10 他の方式との比較

ここでは 3 種類のプロトコル、分割ネットワーク方式 (コロンビア方式)[51]、経路指定方式 (IBM 方式)[52]、仮想ネットワーク方式 (VIP 方式)[33, 34] と Mobile-IP ワーキンググループの方式 (mobile-ip 方式) を比較する。

各方式での移動ホストの IP アドレスを表 4.1にまとめた。この中で VIP 方式のみが

表 4.1: 移動ホストの IP アドレス

	コロンビア	IBM	VIP	mobile IP
IP アドレス	不変	不変	動的取得	不変

IP アドレスを接続先のネットワークで動的に取得することを要求している。IP アドレスの動的な取得は、従来 BOOTP や RARP などの方法があり、主にディスクレスの計算機のためのものであった。これらは最低限の設定のみ可能であり、また移動ホストのための IP アドレスの一時的な使用という概念はなかった。しかし、DHCP という新しいプロトコルが提唱され、その実装の配布も間近であるため、IP アドレスの動的取得の手段がないという点での VIP 方式の不利な点は解消された。

また、残りの 3 種類の方式はともに移動ホストの IP アドレスが不変であるが、コロンビア方式のみ同一の net-ID を持たなければならないという制限がある。この場合クラス A のアドレスを移動ホスト用に取得すれば約 1600 万台までの移動ホストをサポートが可能であるが、インターネット全体で使用するためには世界中の移動ホストに IP アドレスを割り当て、管理する機構が必要となり現実的ではない。もともとこの方式はキャンパス内の移動のサポートを目的として設計されたため、インターネット全域で使用するということはあまり考えられていない。

各方式での移動ホストへのパケットの転送方式を表 4.2 にまとめた。まず、IP オプショ

表 4.2: トンネリングと IP オプション

	コロンビア	IBM	VIP	mobile IP
転送方式	トンネリング	IP オプション	IP オプション	トンネリング

ンを使用している IBM 方式と VIP 方式であるが、IBM 方式が正式な IP オプションを

使用しているのに対し、VIP 方式では新しいオプションを定義しているという違いがある。しかし IP オプションは、正式なものであってもパケットのヘッダが可変長となるため、処理の効率が悪いという観点から実装されない場合が多い。また IP オプション自体の処理を実装していない機器も数多く、そのような機器はオプションが付属する IP パケットを受信すると動作不能になる場合が多いという点が特に問題になる。さらに IBM 方式で使用されている LSRR オプションは通常の経路情報を無視するため、セキュリティの問題上そのパケットを破棄する機器も多い。また VIP 方式ではホームルータと移動ホストのみが VIP に対応していれば通信できるが、それでは拡散キャッシュ法のメリット、つまり移動ホストへのパケットの経路の最適化が自動的におこなわれるという特徴を十分に享受することはできない。

一方トンネリングの場合は、既存のネットワーク機器にまったく修正をおこなう必要がない。しかし、実装の手段が多過ぎてどのレベルで抽象化するべきであるのかが定まらない。本実装はインターフェイスという形で抽象化をおこなっているが、IP パケットを処理する部分でトンネリング用の処理をおこなってもまったく構わない。さらにパケットを見ただけでは送信者を特定することが不可能である点、エラーが発生した時に実際の送信者でなくカプセル化をおこなったホストへ報告されてしまう点、カプセル化をするためのオーバーヘッドがある点、無限にカプセル化されてしまう危険性がある点、経路制御が複雑になる点、階層化モデルに違反している点など多くの欠点があげられる。

結局、現在の IP と互換性を保ちつつ移動ホストをサポートするには、どちらの手段を使用しても一長一短であるといえる。

4.5　まとめ

本研究では IETF において標準化が進められている移動ホストをサポートするためのネットワークプロトコルを実装し評価をおこなった。具体的にはインターネットドラフト [49] において仕様が規定されている 4 種類の構成要素のうちの 3 種類、移動ホスト、ホームエージェント、訪問先エージェントを実装した。また現時点ではオプションの経路最適化以外の機能を実現した。

第 4.4節で述べたように、このプロトコルには現状ではまだ問題が多い。さらに他の方式に比べて格段に優れているというわけでもない。しかしこれは現状のインターネットを構成する機器に変更を加えずに移動ホストをサポートでき、WG での統一案として提唱されているため、今後インターネットの標準となる可能性が高い。

さらに第 4.3節で述べた設計方針はプロトコル自体の評価という目的からみれば、カーネルにほとんど変更を加えずに実現し、移植性も高く、実験をおこなうためには十分な機能を有しているため今回の設計方針は正しいといえる。しかし今後プロトコルの仕様が確定し標準となった場合には、カーネル内部に移動ホストサポートのための機構を実現し、大規模なネットワークでの実用に耐える速度を確保する必要がある。そのためにはトンネリング機構は DDT のようなインターフェイスとして実現することはオーバー

ヘッドが大きく、またインターフェイスであるがゆえの制限もあるのでカーネル内部のネットワーク処理をおこなう部分で実現する必要がある

また実際に運用しなければわからないパラメータも数多いので、今回の実装を基にして認証機構の開発、相互運用性の確認とともに、ネットワークの規模に応じたビーコンの間隔やエージェントがサービスする時間などの、実際の運用に必要なパラメータの最適値を求める実験をおこなうための最低限の枠組を用意できた。また実装することによってあきらかになった仕様の不備な点を指摘した。

今後これらの実験をおこなうことにより得られた結果を、プロトコルの仕様にフィードバックして標準のプロトコルの決定に貢献し、携帯可能な計算機の普及と無線 LAN などの新たなネットワーク機器の出現に備えて、移動するホストをインターネット全体でサポートする仕組みを早急に完成させるべきであると考える。本研究ではこれらの環境の基本的な枠組を提供したといえる。

第 5 章

CLNP における移動ホストプロトコル

5.1 はじめに

OSI ではコネクションレス型ネットワーク層サービスを提供するプロトコルとして CLNP (Connectionless-mode Network Protocol)[53] が標準化されている。CLNP は IP とほぼ同等の機能を持っているが、エラー報告機能も含んでいること、IP が 4 オクテット固定長のアドレスを扱うのに対し CLNP は最大 20 オクテットの可変長アドレスを扱うことができること、などの違いがある。

OSI のネットワーク体系では、ある組織の管理下にあるルーティングドメインをエリアと呼ばれるサブドメインに分割し、エリア内とエリア間のルーティングを階層的に行なうことにより効率の良いルーティングを行なえるようにしている。そのため、ホストのネットワークアドレスがエリアに割り当てられるエリアアドレスとエリア内で一意なシステム ID から構成されている。このようなネットワークアドレスをもつホストがエリアを跨る移動を行う場合は、IP ネットワークでの移動と同様にネットワークアドレスが変化するため、移動透過性を保証する枠組が求められる。

エリア内での移動の場合、つまりネットワークアドレスが変化しない場合は上記のような問題は生ぜず、基本的には既存のルーティングプロトコル (ES-IS プロトコル [54, 55]、IS-IS プロトコル [56]) の枠組でも移動をサポートすることが可能である。しかしリンク状態プロトコルである IS-IS プロトコルはホストが移動するたびに所在地の変更を示すパケットをエリア全体に送信するためトラフィックが増大するという問題がある。このように CLNP ネットワークにおけるホストの移動はエリア間移動とエリア内移動に分けてとらえることができ、この章では CLNP ネットワーク環境においてこれらの階層別にホストの移動を効率良くサポートする手法について論ずる。

5.2 CLNP ネットワーク

OSI 環境でコネクションレス型サービスを提供するネットワーク層プロトコルは CLNP (Connection-Less-mode Network Protocol) [53] である。CLNP ではデータ、エラー、エコー要求、エコー応答の 4 種類の PDU (Protocol Data Unit) が規定されている。ネットワーク層でやりとりされる PDU を NPDU (Network PDU) と呼ぶ。各 NPDU はあて先

アドレス、送信元アドレス、オプションフィールドなどのフィールドを含む。オプションフィールドには QOS、ソースルーティング、ルート記録などの付加情報が格納できる。

エンドシステム (ES) とはトランスポート層以上の層を有するシステムのことで、おもに NPDU を送受信する主体となるシステムのことをいう。中間システム (IS) とは NPDU を中継するシステムであるが、NPDU の送受信を行なうこともある。リンクとは ES と IS もしくは IS と IS をつなぐ物理的媒体のことを指す。リンク上では適切なデータリンクプロトコルが使用できるものと仮定する。

エリアとは ES および IS の集合である。エリアにはユニークなアドレスが付与されており、これをエリアアドレスと呼ぶ。エリア内で IS は互いにリンクを介して連結されている。ES は 1 つ以上の IS にリンクを介して連結されている。エリアが集まってドメインを構成する。

ES および IS にはエリア内で一意な識別子が与えられており、これをシステム識別子と呼ぶ。ES および IS のアドレスはエリアアドレスとシステム識別子の組で表される。レベル 1 IS とは、エリア内の情報として IS の接続形態と ES の存在および居場所を交換し合い、エリア内宛ての NPDU のルーティングを行なう機能を持つ IS のことである。これに対しレベル 2 IS とはエリア間の情報として IS の接続形態と各々のエリアアドレスを交換し合い、異なるエリア宛ての NPDU の目的エリアまでのルーティングを行なう機能を持つ IS のことである。レベル 2 IS 間のリンクがつくるグラフは強連結であるものとする。

ES-IS プロトコル [54, 55] は ES と IS が保持タイムを付与したハローパケットを定期的に出すことにより互いの存在を確認するプロトコルである。ES および IS は互いの存在が確認できるとき隣接関係にあるという。ES の送信するハローパケットを ES ハロー (ESH)、IS の送信するハローパケットを IS ハロー (ISH) と呼ぶ。ES はアドレス要求 (RA) パケットを送信することにより、一時的に使用するアドレス (一時アドレス) を IS に要求することができる。IS はアドレス割り当て (AA) パケットで ES に一時アドレスを割り振ることができる。AA パケットにはアドレスの使用有効期限を示すアドレスホールディングタイムパラメータが含まれる。

IS-IS プロトコル [56] は、IS 間で IS-IS ハロー (IIH) と呼ばれるパケットを出し合って隣接関係を確認し、ES-IS プロトコルで得た隣接 ES 情報とともに他の IS と接続形態の情報を交換し合い、得られた完全マップをもとにルーティングを行なうための仕組みである。レベル 1、レベル 2 の二階層構成とすることにより効率を高めている。リンク状態 PDU (LSP) とは IS 間で定期的および情報内容の変更時に情報を交換し合うためのデータ単位のことである。レベル 1 LSP およびレベル 2 LSP がある。LSP に ES および IS のネットワークアドレス (あるいはその ID 部) を含めて送信することを隣接関係を報告するという。図 5.1はドメインの一例である。図において小さな丸は ES、一重四角はレベル 1 IS、二重四角はレベル 2 IS を表す。実線はシステム間のリンクを表す。細線はレベル 1 回線、太線はレベル 2 回線を表す。複数のシステムを囲む大きな破線による丸は一つのエリアを示す。矢印は ES の移動を表す。

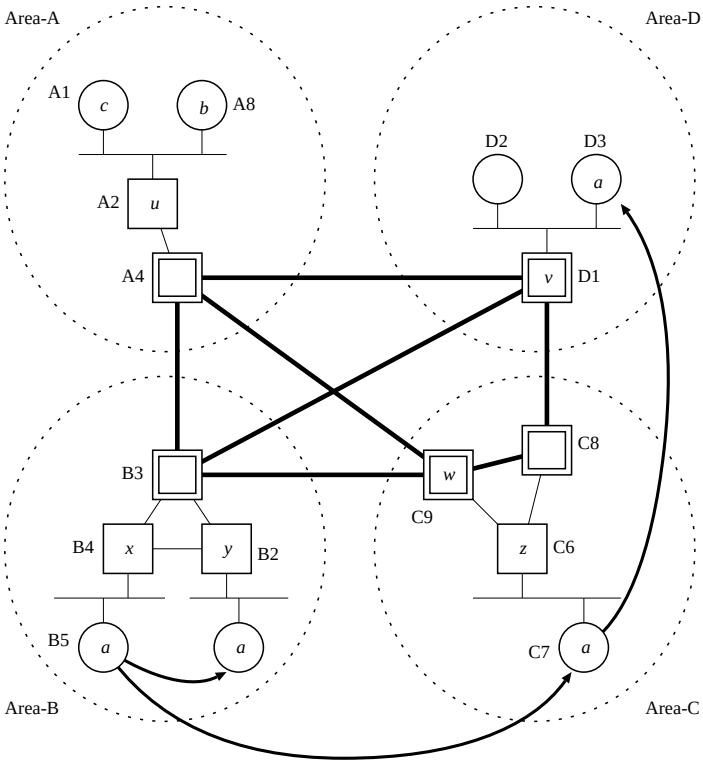
Area-A
Area-D
A1
c
b
A8
D2
D3
a
A2
u
A4
v
D1
C8
B3
w
C9
B4
x
y
B2
z
C6
B5
a
a
C7
a
Area-B
Area-C

図 5.1: ドメインの例

5.3 移動体対応プロトコルの分類

第 5.2節で述べたように、OSI におけるアドレス体系ではネットワークアドレスがエリアアドレスとシステム識別子から構成されるため、エリアを越えて移動する場合は ES のアドレスが変化する。しかしエリア内を移動する場合はアドレスが変化しない。そこでアドレスが変化する移動をエリア間移動、変化しない移動をエリア内移動と呼び区別する。

エリア間移動ではネットワークアドレスが変化するため、移動透過な識別子を上位層に提供する必要がある。その識別子と現在の所在地との対応付けをどのように解決するかが問題となる。エリア内移動は基本的には既存の ES-IS、IS-IS プロトコルでも対応が可能であるが、移動のたびに LSP がブロードキャストされるため効率が悪いという問題がある。

以下ではそれぞれの問題を別の問題としてとらえ、各々別のプロトコルを用いて解決を試みる。つまりエリア間移動とエリア内移動に対して独立なプロトコルを提供し、これらの階層的な使用による問題の切り分けと効率を高めることを考えている。これらのプロトコルはいずれも既存の OSI システムとの相互動作も考慮している。

5.4 エリア間移動

エリア間移動のためのプロトコル [57] はエリア間かつドメイン内の移動を想定するものである。ポリシールーティングを実現するためのドメイン間のルーティングプロトコルである IDRP[58] を採用している場合で、ポリシー上の都合で情報が送信できなかったりフォワードができなかったりする場合を除いてドメイン間の移動にも対応可能である。

エリア間の移動に対しては VIP と同様の考え方を適用して対応することが可能である。すなわち移動体はデフォルトアドレス (VIP アドレスに相当) とカレントアドレス (IP アドレスに相当) を持つ。デフォルトアドレスは移動体の識別のために用いる。カレントアドレスはルーティングに使用し、エリア間移動のたびに変更される。ただし IS-IS プロトコルと連携させるために寿命値パラメータをうまく使う必要がある点が異なる.

移動体はデフォルトアドレスが存在するエリア内の IS にデフォルトアドレスを登録しておく必要がある。この IS を AIS(Administrative IS) と呼ぶ。またエリア間移動のたびにカレントアドレスを AIS に通知する。移動体あての NPDU はいったんデフォルトアドレスあてにフォワードされた後、AIS によってカレントアドレスにフォワードされる。しかし送信元 ES や途中の IS がカレントアドレスをキャッシュ[26] している場合はキャッシュ情報にしたがってあて先が書き換えられ、効率良いルーティングが可能となる。ある時点における移動体の隣接 IS を CNIS(Current Neighbor IS)、移動体がエリア間移動を行なったときの移動前の CNIS を PNIS(Previous Neighbor IS) と呼ぶ。

移動体のデフォルトアドレスとカレントアドレスの対を他のシステムに知らせるために次のパラメタからなる移動体情報を交換する。

デフォルトアドレス (DA) — 移動体のデフォルトアドレスを表す。

カレントアドレス (CA) — 移動体のカレントアドレスを表す。
寿命値 (t) — 情報が有効である期間を表す。
シーケンス番号 (n) — 情報の新旧を判断するために使用する。

移動体情報は NPDU のアドレスフィールドおよびオプションフィールドを用いて運ばれる。移動体は移動すると移動体情報をデフォルトアドレスと移動前のカレントアドレスあてに送信する。この移動体情報はそれぞれ AIS と PNIS が受信し、保持する。単位時間ごとに寿命値を 1 ずつ減らし、0 になると廃棄する。

デフォルトアドレス宛てには寿命値が切れない間隔で移動体情報を送る必要があるが、移動前のカレントアドレスには一度だけ送ればよい。それは次のような理由による。ドメイン内のどこかの IS に古い情報が残っており、それに基づいて NPDU が PNIS にフォワードされることがある。しかしその古い情報の寿命値が切れると PNIS へフォワードされることはなくなるので、その時間だけ PNIS が CNIS へフォワードする役割を果たせばよい。このように寿命値を適切に設定することにより、古いキャッシュエントリがネットワークに残っていたとしても他の ES と通信を行なうことができる。このような手法を用いることにより、古いキャッシュエントリを消去するために制御 PDU を伝搬する必要がなくなる。

さらに IS-IS プロトコルとの連携の際にも寿命値は重要な役割を果たす。寿命値によって PNIS は移動体との隣接関係をいつまで報告すればよいかがわかる。適当な契機に隣接関係の報告をやめると、古いアドレスあてにフォワードされてきた NPDU がエリア入口であて先到達不可エラーとして廃棄されてしまう。また寿命値の使用には移動先で割り当てられた一時アドレスの使用期限を明示的にネットワークに広めることができるという利点もある。

5.5　エリア内移動

エリア間移動に対応するプロトコルと違ってエリア内移動に対応するプロトコルでは、ES が移動を行なっても自ら移動通知を送るなどの動作を行なわないという ES の拡張不要性を重視する。これは ES 側の負担を軽減するためのものであるが、特にエリア内のプロトコルを移動体の数や移動頻度などの運用上の諸条件によって、エリア毎に異なったものを採用する場合に、ES 側はプロトコルを切替える必要がないため有効である。

DFP(デフォルトフォワーディングプロトコル) [59, 60] は効率とスケーラビリティを考慮したエリア内移動体対応プロトコルである．エリア間移動プロトコルにおけるデフォルトアドレスとカレントアドレスに対応する枠組を導入する。移動体との隣接関係を報告している IS をデフォルト隣接 IS、移動体が実際に隣接している IS をカレント隣接 IS と呼ぶ。移動体が新たに隣接したことを検出した IS は、その移動体のデフォルト隣接 IS あてに移動通知を送る。デフォルト隣接 IS は、この移動通知によって移動体が現在隣接している IS を知ることができる。移動体あての PDU はいったんデフォルト隣接 IS あてにフォワードされたのち、カレント隣接 IS へフォワードされる。

データ転送の効率をあげるために IS はキャッシングを行なう。カレント隣接 IS がデフォルト隣接 IS へ送る移動通知の他に、データ PDU のオプションフィールドにもカレント隣接 IS 情報を設定でき、これを学習することによって無駄なフォワーディングを避けることができる。キャッシュは適切に更新する必要があるが、移動体情報は前述のように ES ではなく IS が送信するために、適切なシーケンス番号や寿命値を割り当てるのが困難である。そのため情報の新旧判断を適切に行なうために次のようなルールを新たに導入し、部分的に比較を行なうようにしている。

> 新旧判断条件: 次のいずれかの場合にキャッシュよりも受信した PDU に含まれる情報の方が新しいと判断でき、キャッシュを更新する。
>
> 1. PDU の送信元がデフォルト隣接 IS またはカレント隣接 IS である。
> 2. PDU の送信元が自局の保持するキャッシュのカレント隣接 IS と一致する。

このような方法でもループが生じず、しかも効率が上がることが文献 [59] で示されている。

エリア内移動のためのプロトコルとして DFP 以外のものも考えられ、エリアの特性によってエリアごとにさまざまな方式が選択できる。文献 [61] ではこれらの手法の効率の比較を行なっている。

1. IS-IS プロトコル (レベル 1)。
   移動が少ない場合に有利である。
2. ブロードキャスト問い合わせ法。
   中継時にエリア内の他のすべての IS に移動体の位置情報を問い合わせる。移動時には情報を送信しないため、高頻度の移動に強い。
3. デフォルト問い合わせ法。
   DFP と同様、移動体をデフォルト隣接 IS/カレント隣接 IS の組で管理する。IS が中継時に、デフォルト隣接 IS に対してカレント隣接 IS を問い合わせる。無駄なフォワーディングをしないため、長データ PDU を多く使う場合に DFP よりバンド巾の使用効率が良い。
4. DFP.
   移動と通信がある程度頻繁な場合に有効である。

IS-IS プロトコルならびにブロードキャスト問い合わせ法は、それぞれ移動時あるいは NPDU フォワーディング時に制御 PDU をブロードキャストする。IS 間のリンクの多重度を上げるとブロードキャストにかかるコストも上がることになる。そのためデータ通信にかかるコストとのトータルなコストでみた場合、移動が極端に少ないかまたは多いかのどちらかの場合に有効となる。これに対しデフォルト隣接 IS とカレント隣接 IS との間でのみ移動体情報を交換するデフォルト問い合わせ法と DFP は移動頻度からみて一般的な利用状況において有効である。

5.6　関連研究

OSI 環境での移動体対応プロトコルとしては Carlberg [62] が提案するものがある。この手法において、移動体は論理アドレスとルーティングアドレスを有する。ルーティングアドレスはサブネットワークが変わるごとに変わる。移動体の識別は論理アドレスで行なう。

エリア内移動 エリア内のレベル 1IS 間で隣接 ES 情報を交換する。レベル 1LSP に隣接する移動体の論理アドレスとルーティングアドレスを設定するためのフィールドを追加する。移動体あての NPDU のあて先は論理アドレスとするが、エリア内でのフォワードはルーティングアドレスにしたがって行なう。移動毎に LSP がエリア内にブロードキャストされるが、ルートの再計算やフォワーディングデータベースの変更は必要ない。

エリア間ドメイン内移動 レベル 2IS が他のレベル 2IS にエリア内に存在する移動体の論理アドレスを知らせる。エリア内に存在するすべての移動体の論理アドレスはレベル 1LSP の情報からわかるので、レベル 2IS はそれをレベル 2LSP に設定してドメイン内の他の IS に広める。個々のレベル 2IS はドメイン内のすべての移動体の論理アドレスを保持することになる。NPDU のフォワードは、NPDU のあて先が保持する論理アドレスの一つに一致する場合はその LSP を生成したレベル 2IS の存在するエリアにフォワードされ、それ以外の場合は、NPDU のあて先アドレスにしたがってフォワードされる。

ドメイン間移動 ディレクトリ [63] を使用して移動体のドメイン情報を知る。移動時にドメイン情報をディレクトリに登録する。

NPDU のフォワードの際、そのあて先が保持する論理アドレスに一致するものがなく、しかもドメイン内のアドレスではない場合は、ドメイン外へフォワードされる。境界 IS においては、ディレクトリに問い合わせて得られた結果あてに NPDU をカプセル化する。

しかしこの手法では移動体の移動毎に LSP が広められ、さらにレベル 2 IS はドメイン内のすべての移動体のエントリを保持しなければならず、大規模なネットワークには向かない。また DA 回線 (課金網などを動的に接続して使用する場合) は LSP が流れないため、ドメインを分けなければならないという運用上の制約もある。さらにディレクトリサーバの応答が高速でないとドメイン間のオンライン移動は不可能である。本稿で示した手法はこれらの問題点を効率的に解決している。

5.7　おわりに

この章では OSI ネットワークにおいて移動体をサポートするためのプロトコルについて論じた。OSI の階層的なルーティング体系の効率の良さを損なわないようにするため、

エリア間とエリア内の移動に対してそれぞれ提案されている別々のプロトコルの適用について論じた。またエリア内移動に対しては特性の異なる複数の手法を用意し、ネットワークトポロジーやエンドシステムの移動頻度などの条件によって、使い分けが可能である点についても述べた。各々のプロトコルの独立性を重視したため、柔軟で効率的な運用が可能となる。今後は各プロトコルの使い分けを動的に行う方式と、使い分けのためのより具体的な指針を示す必要がある。