第 5 部

トンネリング技術

第 1 章

はじめに

インターネットは歴史的には研究活動の支援を主目的として構築され、科学技術の発展に大いに貢献してきた。近年では商用インターネットの発達により産業や経済の発展にも貢献している。インターネットがいかに有用であるかは、接続ホスト数が毎月約 10%ずつ増加していることからも推測できる。もはやインターネットなしでの研究活動は考えられない研究者は多い。産業や経済においても、いずれはインターネットなしでは語れない日がくるであろう。

初期のインターネットの主な利用目的は、電子メイル、ファイル転送、遠隔ログインであった。今日では WWW を活用した情報提供・検索・収集の場としての利用が増えている。さらに最近では、インターネットの新しい利用形態として移動ホストの提案や、サービス提供の新しい形態としてサービスにアドレスを振る Service Switch が提唱されるなど、従来は異なる利用形態やサービス提供形態が始まろうとしている。

これらの新しい利用形態やサービス提供形態をインターネットで実現する際にはインターネットのアドレス方式が問題となる。インターネットは、技術的には基盤となる個々の物理ネットワークをサブネットとして抽象化し、サブネット間を接続して全体を 1 個の大きなネットワークとみたてるものである。このとき、個々のサブネットにサブネット番号を割り振り、そこに接続する個々のホストにホスト番号を割り振る。インターネット・アドレスとしてはこの両者を組み合わせたものを用いる。そのため、インターネット・アドレスはホストの位置情報を示すと同時にホストの識別子にもなっている。ところが移動ホストや Service Switch を実現するためには、ホストやサービスの識別子の意味しか持たない特別なインターネット・アドレスが必要となるのである。

このような場面で有効な解決策を与えるのが仮想ネットワークである。仮想ネットワークは論理的なネットワークであり、通常の物理ネットワークと違って物理媒体とは直接対応していない。しかし仮想ネットワークは、論理的にネットワークアドレスを割り振ることにより通常のネットワークと変わりなく存在しているとみなすことができたり、他の物理ネットワークと同じネットワークアドレスを割り振って物理ネットワークを論理的に延長してみせたりすることができる。このとき、仮想ネットワークに割り振られたアドレスは、識別子としての意味しか持たない特別なインターネット・アドレスであるため、これを用いれば移動ホストや Service Switch のサーバが実現可能となる。仮想ネットワークは

この他にも経路制御におけるトポロジーの補助線としても利用されている。

この仮想ネットワークを構築するための技術としてトンネリングが注目されている。トンネリングはネットワーク層のプロトコルを他のネットワーク層以上のプロトコルで運ぶものである。歴史的には、組織間を IP で接続する際に既存の X.25 等のネットワークをデータリンクとみなして IP の送受信に用いるという形で利用されてきた。IP による広域ネットワークが発達した現在では、IPX、Appletalk、OSI などのネットワーク・プロトコルを IP ネットワークを用いて送受信する場面でも利用されている。さらに最近では、IP 自身の転送に IP を使う形、つまり “IP over IP” の利用が注目されている。上記の仮想ネットワークの構築で利用するのは “IP over IP” の形のトンネリングである。

今回の報告書では、トンネリングに関してこれまでの DDT WG の研究成果のまとめを行なう。

第 2 章

研究の背景

2.1 トンネリングとは

トンネリングとは、広義にはネットワーク層のプロトコル x (例えば IP) を伝達するために、他のネットワーク層以上のプロトコル y (例えば X.25) をデータリンク層のプロトコルとみなして利用する技術である。このときプロトコル x のパケットはプロトコル y のデータとして扱われる。この状態を "x *over* y" と呼ぶ。狭義には、特に x と y とが等しい場合を指す。

以下では、プロトコル x をプロトコル y の上に載せる操作を「カプセル化」または "encapsulation"、それを行なうルータまたはホストを「トンネルの入口」、そのとき付加されるヘッダを「カプセル化ヘッダ」または "encapsulation header" と言う。プロトコル x をプロトコル y から取り出す操作を「脱カプセル化」または "decapsulation"、それを行なうルータまたはホストを「トンネルの出口」と言う。

2.2 トンネリングの応用分野

インターネット技術におけるトンネリングの利用例及び提案を、目的別に以下にまとめる。

1. 新しいプロトコルや異なるプロトコルを運ぶ
   トンネリングの提供する仮想的なネットワークは、他のネットワーク層のパケットを運ぶ目的で利用することができる。具体的には以下の例が挙げられる。

   - マルチキャスト
     マルチキャストは一対多通信のためのメカニズムである。マルチキャスト・プロトコルは比較的新しく、現段階ではインターネット標準とはなっていない。そのため、マルチキャスト・パケットを正しく扱えないルータは多い。そのようなルータをスキップしてパケットを転送するために、現在はトンネリングが利用されている。マルチキャスト・パケットはカプセル化されているため、マルチキャスト配送上の次のルータに届くまでどのルータにも処理されることはない。現在、この方法を使って MBONE と呼ばれる世界規模のマルチキャスト

の実験環境が構築されている。トンネリングのプロトコルとしては “IP in IP” (ipproto=4) が使われている。

- IPng への移行
  現行の IP (IPv4) に伴う問題点を解決するために次世代 IP (IPv6) が議論されている。IPv4 から IPv6 への移行を円滑に行なうために、移行初期には “IPv6 over IPv4” の形で IPv6 を IPv4 の上にトンネリングで載せて運ぶことが検討されている [29]。
- 異なるプロトコルを運ぶ
  IP ネットワークを利用して他のプロトコルを運ぶために “IPX over IP”[30], “OSI CLNP over IP”[31] などが定義されている。マルチプロトコル環境で相互にトンネリングするための汎用的なプロトコルとして GRE (Generic Routing Encapsulation)[32][33] がある。

2. ネットワーク・トポロジーの補助線
   トンネリングの提供する仮想的なネットワークは、IP を運ぶ目的で利用することができる。このときトンネリングはネットワーク・トポロジーの補助線として機能させることができるので、経路制御を簡単にするなどの目的で利用することができる。具体的には以下の例が挙げられる。

   - 経路制御技術の制限の回避
     ルータの forwarding 機能における技術的なの制限 [34] もしくは経路制御プロトコルの限界 [35] を避ける手段としてトンネリングの利用が提案されている。
   - AS を通過するデータグラムの転送
     AS を通過するデータグラムを border gateway 間で運ぶ方法のひとつとして IDRP ではトンネリングの利用を検討していた。これは、AS 内の経路制御と AS 間の経路制御を分けて考えることができるようにするためである。
   - 事業所間通信
     企業が事業所間の通信をインターネット経由で行なうとき、利用している IP アドレスがインターネットで正式に割り当てられたものでない場合にはそのままでは通信できない。しかしその場合でも、事業所間にトンネリングによる仮想的なリンクを作り、事業所間トラヒックをすべてそのリンクを通すようにすれば問題を回避することができる [36]。
   - 提案されているプロトコル例
     “IP over IP” のプロトコルとして “Internet Encapsulation Protocol”[37] が提案されている。“PPP over TCP” も利用可能である [38]。

3. 特殊ネットワークの構築
   現バージョンの IP では、IP アドレスはホストの位置情報とホストの識別子のふたつ

の意味を持っている。ところが、仮想ネットワークに付与した IP アドレスは位置情報とは独立であり、識別子の意味しか持っていないため、移動ホストに応用できる。さらにホスト以外のもの (例えばグループやサービス) の識別子として IP アドレスを利用できるようにもできる。このような識別子の意味しか持たない IP アドレスへのパケット配送機能を実現する方法として、すべてのルータにその機能を持たせる方法と、一部のルータやホストのみがその機能をサポートしてそれらの間をトンネリングによる仮想ネットワークを利用して配送する方法とがある。ここでは後者に注目する。具体的には以下の例が挙げられる。

- コロンビア大学方式移動ホスト支援 [39]
  移動ホスト (Mobile Host: MH) はキャンパス全域に広がる仮想ネットワークに無線で接続することでインターネットと通信が可能になる。その仮想ネットワークはキャンパスの各所にある MSR (Mobile Support Router) の間をトンネリングで結ぶことで実現している。トンネリングのためのプロトコルとして “IP within IP” (ipproto=94) を独自に定義し、利用している。

- Service Switch[40]
  Service Switch は、サービスの要求者や時刻などによって、実際に提供するサービスを変化させるものである。これをネットワーク層で実現するものが Servicecast というメカニズムであるが、Servicecast の実現にはトンネリングの利用が検討されている。

4. IP option の拡張
   現在の IP の IP option 長は最大 40 オクテットしかない。それ以上の長さの情報を IP データグラムに対して付加する必要がある場合、トンネリングの形を借りて、トンネリング・プロトコルにその情報を運ばせることができる。

   - swIPe[41]
     swIPe は暗号化機能をネットワーク層で実現するものである。swIPe では個々のパケットをカプセル化して送る。このとき、復号化のための情報はトンネリング・プロトコルのヘッダ部に格納している。これにより、40 オクテットを越える長さの復号化情報を利用することが可能となる。

このようにトンネリングは様々な応用例が存在する。しかし、その実装方法は個々の応用ごとにばらばらになされており、汎用的な方法は存在しない。トンネリングの汎用的な利用モデルが存在すれば、トンネリングはより利用しやすくなり、その応用は拡がるであろう。

2.3 トンネリングの問題点

実験や文献調査を通して、インターネット技術におけるトンネリングの問題点を洗い出し、分類を行なった。以下にそれを示す。

1. 経路上の問題もしくは経路制御に関する問題

TP-1 モデルとの相性

トンネリングは次に示すように現在の階層型プロトコル・モデルでは記述できない現象である。

- 広義のトンネリングは複数のプロトコル・ファミリにまたがった現象である。ところが、現在の階層型プロトコル・モデルはひとつのプロトコル・ファミリで閉じた場合しか考えていない。
- 狭義のトンネリングは階層モデル違反 (Layer Violation) を含んだ現象である。ネットワーク層のプロトコルの上にネットワーク層のプロトコルが載っているため、単純には現在の階層型プロトコル・モデルを当てはめることはできない。

TP-2 迷子パケット

パケットは、目的地に到着できない場合はすみやかに消滅させられなければならない。ところがトンネリングにおける TTL の機構が確立されていないため、目的地に到着できず迷子になったパケットが必要以上に長い寿命を持ったり、消滅しなかったりする可能性がある。そのようなパケットが、トンネルに入った後そのトンネルから抜け出ることなく再び同一トンネルに入ってしまうような無限ループに突入してしまったとする。以後このようなループを「クラインの壷型ループ」と呼ぶ。

このとき、パケットはループを 1 周するたびに新たなカプセル化ヘッダを追加されて、パケットの大きさが徐々に大きくなっていく。パケットの大きさが経路上の最小 MTU を越えてると、パケットは MTU 以下の大きさのパケットに分割される。その結果、ループが無限に続くと最小 MTU 以下の大きさを持つパケットが多数発生してしまい、ネットワーク・メルトダウンを引き起こすことになる [42]。

TP-3 QoS

多くのプロトコルでは、パケットには通過する経路に対する要求を表す QoS (Quality of Service) が書かれている。ところがカプセル化する時に元々のパケットの QoS が無視されたり、カプセル化おいて十分に反映されなかったりする場合が多いため、パケットが不適切な経路を通ってしまう可能性がある。

TP-4 パケットの識別

カプセル化を行なうと元々のパケットの発信者、受信者、プロトコルの種類などの識別が難しくなる。また元々のパケットの識別情報を詐称することも簡単にできる。その結果 packet filter がパケットを誤って識別する可能性が発生し、セキュリティ上大きな問題となる。またネットワークの利用統計を正しく取ることも難しくなるため、ネットワーク利用度を評価する上で支障となる。

TP-5 経路制御

トンネリングによる仮想ネットワークを含むネットワークでは通常のループを避ける以外に次の点に注意しなければならない。

- 定常状態の維持 — トンネリングを正しく機能させるため、トンネルの出入口への経路を確保する。
- 障害時のフェイルセーフ — クラインの壷型ループを避ける仕組みが必要となる。
- 最適な経路選択 — トンネルの cost/metric を物理ネットワークの状態に応じて正しく評価し、意味のないトンネリングは避ける。

ところが、トンネリングによる仮想ネットワークを含んだネットワークにおける一般的な経路制御方法は知られていない。そのため経路情報が誤った方法で処理されたり、誤った経路情報がアナウンスされたりして、ネットワークに混乱を招く可能性がある。

2. プロトコルに関する問題

TP-6 エラー報告

パケットがトンネル通過中にエラーが発生した場合、エラー発生箇所のルータがトンネリングを特に支援していない限り、エラーの報告はパケットの元々の発信者ではなくトンネルの入口のルータに送られてしまう。そこで、如何にしてパケットの元々の発信者に正しいエラー報告を返すかが問題となる。具体的な方法はプロトコル・ファミリごとに考えねばならない。いずれの場合にもトンネリング・プロトコルに工夫が必要となるであろう。具体的には以下のような点を考慮しなければならない。

- トンネル内のルータがパケットの元々の発信者に対して直接エラーの報告を送るようにする場合には、エラー報告パケットの送り先である元々の発信者の見つけ方と、エラー報告パケットを正しく生成するための方法が必要になる。
- トンネルの入口がパケットの元々の発信者に対して間接的にエラーの報告を送るようにする場合には、トンネルの入口が報告を受け取った時、それを無視すべきかあるいは元々の発信者に知らせるべきかをの判断基準 [37] が

必要になる。また、トンネル入口のルータが受け取ったエラー報告のパケットから、元々の発信者にとって意味のあるエラー報告パケットを正しく生成するための方法が必要になる。

TP-7 オーバーヘッド

トンネリングを利用すると、元々のパケットにカプセル化ヘッダが付加されることから、その分バンド幅を余分に消費する。またカプセル化、脱カプセル化などの操作のために CPU を余分に消費する。

3. 実装に関する問題

TP-8 MTU

トンネリングによる仮想ネットワークを仮想インタフェースを用いて提供するような実装を行なう場合、実際には存在していない MTU の値を決めなければならない。MTU の値が大きすぎると、実際に物理ネットーワークに送り出すためにはフラグメント処理が必要になる。逆に MTU の値が小さすぎると、小さなパケットを多く送出することになりネットワーク利用効率が悪くなる。

4. ネットワーク管理上の問題

TP-9 制御不能なトポロジー

トンネリングを用いるとユーザは任意の相手と自由に仮想リンクを張ることができる。その結果、ネットワーク管理者がネットワークのトポロジーを把握できなくなったり、各組織のポリシーに反するリンクが秘密裏に張られたりするなどネットワークの管理・運用に支障が出る可能性がある。

第 3 章

トンネリングの汎用的な利用モデル

トンネリングには多くの応用分野が存在するにもかかわらず、汎用的な利用方法が存在しない。そのため、現状ではトンネリングを必要とするモジュールは目的に合わせて個々に設計・実装しなくてはならない。そこで本章では、トンネリングを利用しやすくするために、汎用的なトンネリングの利用モデルである “DDT”[1] を提案する。“DDT” は既存のトンネリングの応用例のほとんどに適用可能と考えられる。さらに 3.2 節では、トンネリングの新たな応用も提案する。

3.1 DDT の提案

3.1.1 DDT とは

トンネリングとは基本的には仮想ネットワークを提供する手段と言える。トンネリングによる仮想ネットワークの汎用的な利用モデルにおいては、仮想ネットワークも Ethernet などの物理ネットワークと区別なく利用できるのが望ましい。その結果、例えば、経路制御の場面でも物理ネットワークと同様な方法で経路情報を伝えることができたり、TOS を利用して経路選択ができたりするからである。そのためには、仮想ネットワークも物理ネットワークと同様にネットワーク・インタフェースの形で提供され、それを通してアクセスする形態が望まれる。

以上の考察から、トンネリングの汎用的な利用モデルでは、トンネリングによって提供される仮想ネットワークに対し、仮想インタフェースを通してアクセスする形態が良いということが言える。これは次のようにまとめられる。

> **トンネリング**を用いて構築された**仮想ネットワーク**を、**仮想インタフェース**を通して提供する。

この利用形態を DDT モデルとして提案する。

[1] DDT は “Doki Doki Tunneling” もしくは “Delightfully Dangerous Tunneling” の略であり、歴史的理由からこう呼ばれている。

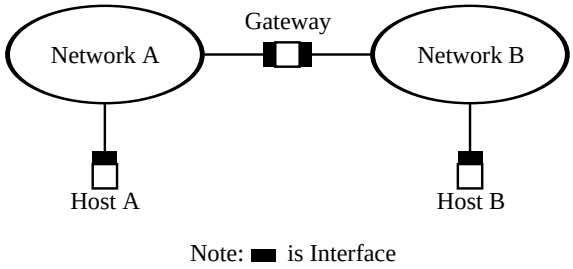


図 3.1: インターネット・アーキテクチャ

3.1.2 インターネット・アーキテクチャにおける DDT

DDT モデルのインターネット・アーキテクチャへの適合を試みる。DDT モデルでは、トンネリングによる仮想ネットワークを物理ネットワークと同様に扱う。そこで仮想ネットワークを、インターネット・アーキテクチャにおけるネットワークとして扱えるようにしなければならない。

インターネット・アーキテクチャ・モデルは次のようになっている。

1. インターネットはネットワーク、ゲートウェイ、ホストから成る。
2. すべてのゲートウェイ及びホストはネットワークとの接続のためにインタフェースを持つ。複数持っていてもかまわない。
3. すべてのゲートウェイは複数のネットワークに対してインタフェースを持ち、パケット転送機能を提供している。
4. すべてのネットワークはネットワーク識別子を持つ。
5. すべてのインタフェースはネットワーク識別子＋ホスト識別子という形式のアドレスを持つ。

トンネリングによる仮想ネットワークをこのモデルに適合させるには、この記述において「ネットワーク」になっている箇所を「物理ネットワーク及び仮想ネットワーク」に置き換えても矛盾しないようにすれば良い。したがって、次の 3 点を行なえば良いことがわかる。

1. ゲートウェイ及びホストが仮想ネットワークにアクセスするときには仮想的なインタフェースを通して行なうようにする。
2. 仮想ネットワークにネットワーク識別子を割り振る。

3. 仮想的なインタフェースはネットワーク識別子＋ホスト識別子の形のアドレスを付与する[2]。

3.1.3　DDT の “IP over IP” への適用例

図 3.2 は、“IP over IP” トンネルの場合の DDT の例を示す。図ではホスト H_a と H_b はそれぞれが物理ネットワークと仮想ネットワークの両方に接続されている。H_a, H_b の物理インタフェースのアドレスはそれぞれ P_a, P_b であり、仮想インタフェースのアドレスはそれぞれ V_a, V_b である。仮想ネットワーク N_v はトンネリングにより H_a, H_b の間に作られた 2 点間リンクであり、ネットワークアドレス N_v を持っている。このとき、仮想インタフェースのアドレス V_a, V_b はネットワーク N_v に属するアドレスである。

図 3.2 中央付近に、パケットが仮想ネットワークを通過する様子を示す。図 3.2 中では、ホスト H_a が H_b に IP データグラムを送ろうとしており、そのための経路として仮想ネットワーク N_v を選択している。H_a から仮想インタフェース (“DDT”) を通して仮想ネットワーク N_v に送り出されたデータグラムは、始点アドレス V_a、終点アドレス V_b を持ち、データ部には例えば TCP のヘッダとデータが入っている。このパケットは H_b の仮想インタフェース (“DDT”) を通して H_b に届く。

図 3.2 の下部は、実際のパケットの動きを示している。H_a の経路制御モジュール (図中 “Network (IP)”) により DDT 仮想インタフェース (“DDT”) に送り出された IP データグラムは、実際には一旦ポンプモジュール (“DDT pump”) に吸い上げらる。そこでカプセル化された上で再び経路制御モジュールに投げ入れられ、経路選択が行なわれて物理インタフェース (“Ether”) を通して物理ネットワーク (“N_a”) に送り出される。このとき送り出されたデータグラムは、図 3.2 の中央やや下にあるように、始点アドレス P_a、終点アドレス P_b を持ち、データ部には仮想的に N_v を通過しているデータグラムがそのまま入っている。H_b の物理インタフェース (“Ether”) に届いたパケットは、ネットワーク・プロトコル処理モジュール (“Network (IP)”) を経由して一旦ポンプモジュールに送られて脱カプセル化された上で、DDT 仮想インタフェースを通して再びネットワーク・プロトコル処理モジュールに渡される。その後は通常と同じくプロトコルに応じて TCP 処理モジュールなどに送られる。

3.2　応用例

DDT モデルを導入し、トンネリングを仮想ネットワークと考えることで、トンネリングの新たな応用を提案しやすくなった。以下に、いくつか提案を行なう。

[2] もちろん、実装上は unnumbered link もありうる。

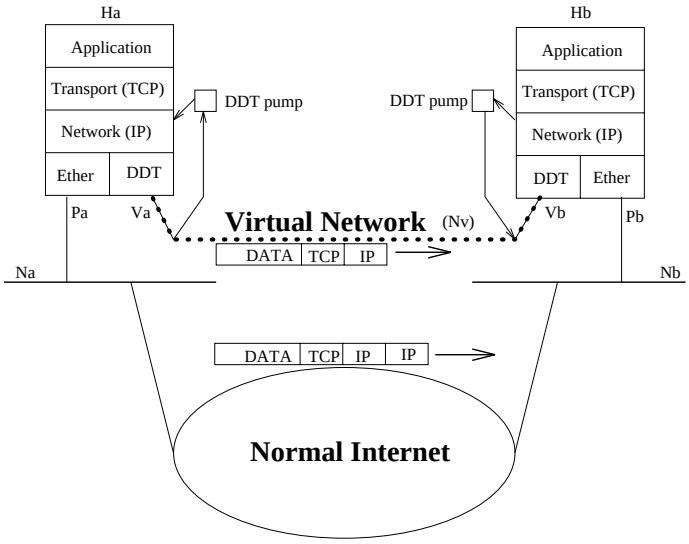


図 3.2: IP over IP の場合の DDT

3.2.1 離れ小島の接続

組織がインターネットに接続するとき、外部からの悪意の侵入を防ぐために組織の入口に firewall を設けることがある。特に企業の場合は厳重なセキュリティ管理が行なわれ、企業内の限られた小数のホストからしかインターネットに直接アクセスできなくしている場合が多い。その結果、ゲートウェイから離れた部署がインターネットに直接アクセスすることは難しくなっている。というのは、それを実現するにはそのような部署とゲートウェイと間のすべてのルータにインターネットの経路情報を教えねばならいことから、当該部署とは関係ない部署までもセキュリティ上脅威にさらさねばならなくなるからである。そのため、どうしてもインターネットに直接アクセスしなければならない場合は、当該部署とゲートウェイとの間に新たに直接回線を引かねばならなかった。

この問題は、当該部署とゲートウェイとの間に DDT による仮想ネットワークを作ることで解決できる。その結果、ちょうど直接回線を引いたのと同じく、途中のルータにインターネットの経路情報を知らせる必要は全くなくなる。当然、当該部署の仮想リンクの端点周辺においては経路制御情報を慎重に扱わなければならない。また、厳重なセキュリティ管理も必要となる。しかし、これもちょうど実際に回線を引いた場合も同様に発生する問題である。つまり、DDT を用いると実際に直接回線を引いた場合とほぼ同じ状況を新たな出費なしに構築できるわけである。

同様に、障害によりネットワークが分断されて間には異なる番号のネットワークしか存在しないような場合に、分断されたネットワークを仮想ネットワークを用いて緊急避難的に接続する目的で DDT を利用することができる。

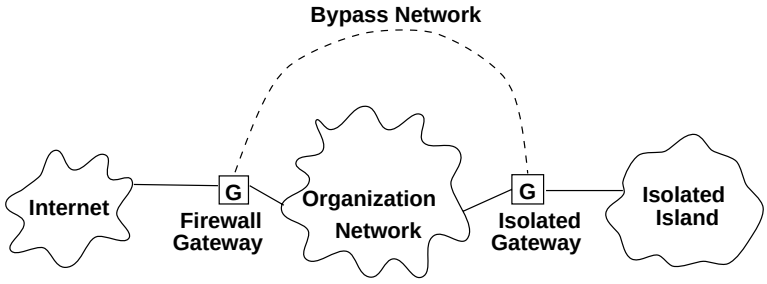


図 3.3: 離れ小島

3.2.2　アドレス空間の節約

組織がインターネットに接続するとき、限られた小数のホストしかインターネットに直接アクセスできないようにしている場合が多い。そころがそのような企業においても、社内のすべてのホストやサブネットに対してインターネットに到達可能なアドレスを割り振っているところが多く存在する。その理由のひとつに、社内のどのホストがいつインターネットに直接アクセスする必要が発生しても対応できるように備えておく、というのが挙げられる。その結果、企業は広いインターネット・アドレス空間を持っている割には実際にインターネットから見えるアドレス空間は狭く、アドレス空間枯渇の原因のひとつにもなっている。

DDT を用いれば、この問題も解決可能である。その場合、組織内のホストの IP アドレスは基本的にはインターネットとは独立なものを割り当てる。RFC1597[36] に従って割り当てるのがよいだろう。各ホストは、インターネットに直接アクセスするときだけ、firewall ゲートウェイとの間に DDT を用いて動的に仮想ネットワークを張るようにする。そしてその間だけ、インターネットに到達可能な IP アドレスを割り当てる。これはちょうど Dial-up IP を利用するときの典型的な状況に似ている。その結果、各組織は比較的小さなアドレス空間を取得するだけで、組織内の多くのホストのインターネットへのアクセス要求をさばくことができるようになる (図 3.4 参照)。

同じ目的をもつ技術に NAT (Network Address Translator)[43] がある。NAT の場合も、組織内のホストの IP アドレスは基本的に RFC1597[36] に従ってインターネットとは独立なもの (Local address と呼ぶ) を割り当てる。さらに組織外と通信するためにインターネットに到達可能な小さめの IP アドレス空間 (Global address と呼ぶ) を用意する。各ホストは、インターネットに直接アクセスするときだけ Global address を割り当ててもらう。NAT 自身は firewall のルータに組み込まれており、出ていくパケットの中にあるすべての Local address を対応する Global address に書き換え、入ってくるパケットの中にあるすべての Global address を対応する Local address に書き換える。これにより、NAT はインターネットに対してあたかも Global address を持つホストが組織内部に存在するかのように見せかけている。したがって NAT を実用レベルで実装するには、ftp などよく利用する

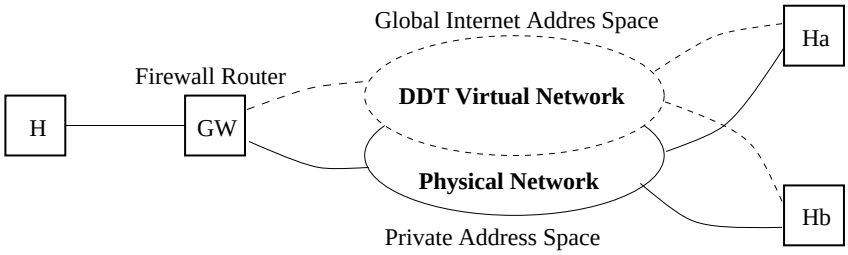


図 3.4: DDT による仮想ネットワーク

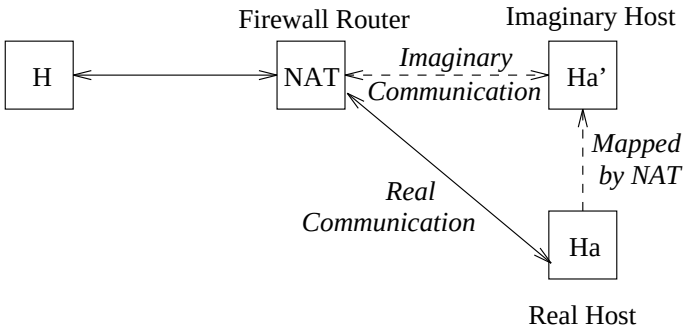


図 3.5: NAT

	DDT	NAT
変更の必要なもの	firewall のルータ 通信を行なうホスト	firewall のルータ
アドレス空間	(通常の) アドレス 仮想アドレス	Local address Global address
パケットの書き換え	必要なし	パケットに含まれる Global address 及び Local address すべて
プロトコルの知識	必要なし	利用するプロトコルのうちネットワーク・アドレスを含むもの (含: SNMP, DNS)
セキュリティ技術	利用に制限なし	暗号化部分にネットワーク・アドレスを含めることができない
トンネリング	必要	(パーティション間で. [43] 参照)

表 3.1: DDT と NAT の比較

プロトコルに含まれる IP アドレスの位置及びフォーマット、さらにアドレス書き換えに伴うチェックサムや TCP の Sequence Number の修正方法など様々な知識が要求される。プロトコルが複雑な場合は書き換えオーバーヘッドも大きく、暗号化されていた場合には手に負えないという問題がある (図 3.5 参照)。

一方 DDT は firewall のルータと、通信を行なうホストの両方に組み込まねばならない。しかし、DDT を用いる場合は通信を行なうホストが直接仮想アドレスを使用するため、プロトコルに含まれるネットワーク・アドレスを書換える必要はいっさいない。表 3.1 に示すように、DDT は実装は簡単、暗号化通信も利用可能、SNMP, DNS など書換えが複雑なプロトコルも問題なく利用可能など、NAT より優れている面が多い。

3.2.3 A Secure Virtual Network

インターネットで盗聴されるのを防ぐために、様々な層の様々なプロトコルに対して暗号化方法が提案されている。DDT は、ネットワーク層に暗号化機能を組み込む方法を提供する。このとき、トンネリングのためのプロトコルは暗号化機能を持っている必要がある。

暗号化通信を行なう必要が発生したときは、まずホスト間で暗号化された DDT 仮想ネットワークを動的に張る。そしてアプリケーションは secure TOS[44] などを利用して、ホスト間通信トラヒックがすべて暗号化された仮想ネットワークを通るように指定すれば良い。

この方法は組織間通信へも応用が可能である。その場合は、両組織の firewall 間で暗号化された DDT 仮想ネットワークを張り、組織間トラヒックがすべて暗号化された仮想ネットワークを通るようにすることになる。実際に通信するホストはグローバル・インターネットのアドレスを持っていなくても良い。

3.2.4 実験ネットワーク構築

DDT は 2 点間リンクの他に多点間を結ぶ広域 Ethernet 的なネットワークなど様々なトポロジーの仮想ネットワークを提供できる。これらを組み合わせることにより、研究実験環境インターネットを構築できる。そこでは、運用ネットワークに影響を与えずに経路制御プロトコルの実験を行なうことができる。また、マルチキャストや OSI プロトコルなどの実験にも利用できる。

3.3 BSD における DDT の実装

以下の目標のもとで DDT の実装を行なった。

1. 4.3 BSD tahoe と BSD Networking Software Release 2 (NET2) の両方で同一ソースコードで動作すること。
2. ベンダーから提供されたオブジェクトファイルに手を加えない。
3. 特定のプロトコルに依存しない構造を取る。

現在次の OS での動作が確認されている。

- BSDI BSD386 (ただし numbered link のみ)
- DEC DECstation 5000/25, Ultrix 4.3
- OMRON LUNA2, Mach 2.5
- SONY NEWS, NewsOS 4.2.1
- SUN sun3/80, SunOS 4.1.1

3.3.1 4.3 BSD におけるネットワークの実装

4.3BSD におけるネットワークの実装は、図 3.6 に示すように階層的な構造になっている [45]。以下に各層の働きについて説明する[3]。

- カーネル入口
  ユーザプロセスからのシステムコールを適切な関数に振り分けている。ソケット関連のシステムコールではソケット層の関数が呼び出される。

[3] 一般に、出力関数は XX_output()、入力関数は XX_input() または XXintr()、入力キューは XX_intrq という形の名前を持つことが多い。

≪ カーネル入口 ≫ trap ⇒ ハンドラ呼出 ⇒ 引数チェック ⇒ ソケット層呼出
≪ ソケット層 ≫ ソケット・セマンティクスの提供
≪ プロトコル層 ≫ プロトコル処理
≪ インタフェース層 ≫ ネットワーク・インタフェース・ドライバ

図 3.6: 4.3BSD におけるネットワーク実装構造

- ソケット層
  ソケットのセマンティクスを提供している。ユーザ・アプリケーションからは、すべてのプロトコルをソケットを通してアクセスする。ソケット層はプロトコル処理のためにプロトコル層を呼び出す。

- プロトコル層
  OSI 7 層モデルで言うネットワーク層・トランスポート層のプロトコルの処理が行なわれる。TCP のように接続に関する状態を持つプロトコルの処理はソケット層と連係して行なわれる。実際のデータの送信が必要なときはインタフェース層を呼び出す。受信データはソケット層に引き渡している。

- インタフェース層
  物理ネットワークに対するパケットの送出、及び受信パケットのプロトコル層への引き渡しを行なっている。

3.3.2　実装の構造

図 3.7 に構造を示す。本実装は、仮想インタフェース機能を提供するモジュールと、実際にカプセル化を行なうモジュールの 2 つの部分からなっている。以下にそれぞれについて説明する。

1. 仮想インタフェース (インタフェース層)
   経路制御モジュールに対して仮想的にインタフェース機能を提供する。仮想インタフェースは利用すべきカプセル化プロトコルの種類とトンネルの両端アドレスを知っており、次の仕事をする。

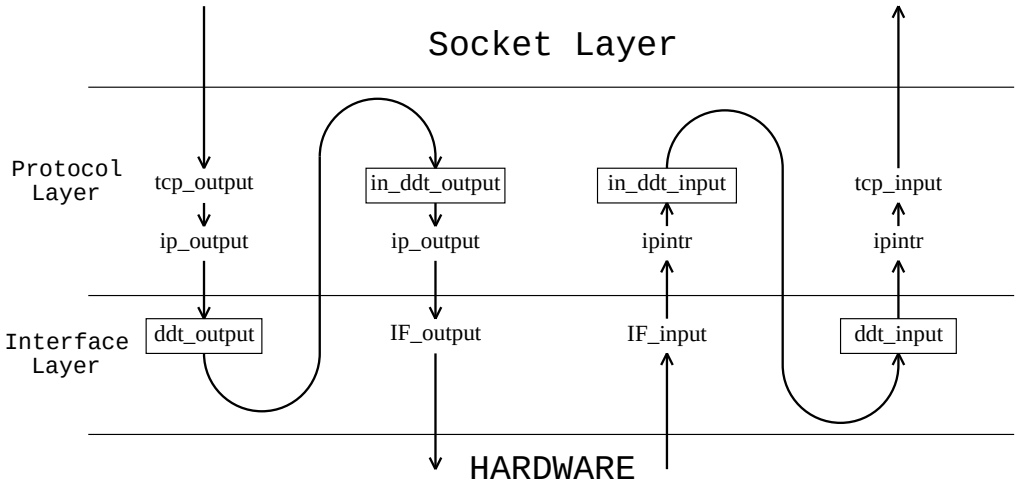


図 3.7: DDT の BSD 上の実装

- 経路制御モジュールからパケットが送られてくると、利用すべきカプセル化プロトコルに応じて最適なポンプ・モジュールを選択する。選択したポンプ・モジュールに対して、トンネルの両端アドレス情報とともにパケットを送り出し、処理を依頼する。図 3.7 では ddt_output() にあたる。
- ポンプ・モジュールからパケットが送られてくると、パケットのプロトコルに応じて最適なプロトコル処理モジュールに渡す。4.3 BSD ではパケット・キューにつなぐ場合が多い。図 3.7 では ddt_input() にあたる。

仮想インタフェースはトンネル 1 本につきひとつ必要である。現在のバージョンのソースは 484 行である[4]。

2. ポンプ・モジュール (プロトコル層)
カプセル化及び脱カプセル化を担当し、次の仕事をする。

- 仮想インタフェースからパケットが送られてくると、指定されたカプセル化プロトコルでパケットをカプセル化する。このときトンネルの両端アドレスも仮想インタフェースから指定されたものを使う。カプセル化したパケットを、カプセル化プロトコルが所属するプロトコル・ファミリに対応した経路制御モジュールに渡す。図では in_ddt_output() にあたる。
- プロトコル処理モジュールからパケットが送られてくると、脱カプセル化して中身のパケットとトンネルの両端アドレス情報を取り出す。トンネルの両端ア

[4]C 言語、コメント行を含む

ドレスから最適な仮想インタフェースを選択し、それに対して取り出したパケットを渡す。図では in_ddt_input() にあたる。

ポンプ・モジュールはカプセル化プロトコルごとにひとつ必要である。複数のカプセル化プロトコルをサポートするポンプ・モジュールがあっても良い。現在のバージョンのソースは 471 行である[5]。

3.3.3　仮想インタフェース間でのループの回避方法

経路制御モジュールに手を加えずに DDT を実装したときの問題点として、仮想インタフェースから送出されたパケットが経路制御の結果再び同じ仮想インタフェースに投げられる可能性が挙げられる。あるいは、最初は異なる仮想インタフェースに投げられたとしても、経路制御モジュールと仮想インタフェースの間を何度か行き来するうちに、いつしか同じ仮想インタフェースに再び投げられる可能性もある。そのような場合、パケットは経路制御モジュールと仮想インタフェースとの間を無限にループしてしまう。

この問題を避けるために、仮想インタフェースの MTU の値として next hop インタフェースの MTU からカプセル化ヘッダの大きさを引いたものを用いた。この値は直観的にも最適な値である。ここで言う next hop インタフェースとは、仮想インタフェースからポンプ・モジュールに送られた後、経路制御モジュールによって選ばれるインタフェースを指す。これは経路制御表から容易に知ることができる。

上記のように MTU を選んだ結果、仮想インタフェース間でループが発生したときには、MTU の値は互いに影響しあって 0 に収束するようになり、問題が容易に発見できる。next hop インタフェースが物理インタフェースである場合には、最適な MTU を “The Path MTU discovery algorithm”[46][47] や “IP MTU Discovery options”[48] などを利用して発見する方が良いであろう。

3.3.4　An Unnumbered Link

DDT 仮想ネットワークを用いて 2 つのネットワーク N_a と N_b とを接続するときを考える。このとき DDT 仮想ネットワークを N_a の subnet とみなしてネットワーク・アドレスを割り振ったと仮定する。その場合、N_b にある DDT ゲートウェイはネットワーク N_a に直接接続していることになるため、他組織のネットワークである N_a の subnet の経路情報を知らないと N_a とはうまく通信できなくなる。これを避けるためには、(1) subnet default を実装する、(2) CIDR[49] を用いる、(3) ”unnumbered link” を実装する、のいずれかが必要となる。ここでは (3) について考える。

“unnumbered link” の場合、H_a の仮想インタフェースは、H_a の物理インタフェースが持つアドレス P_a を自アドレスとして、H_b の物理インタフェースが持つアドレス P_b を相

[5] C 言語、コメント行を含む

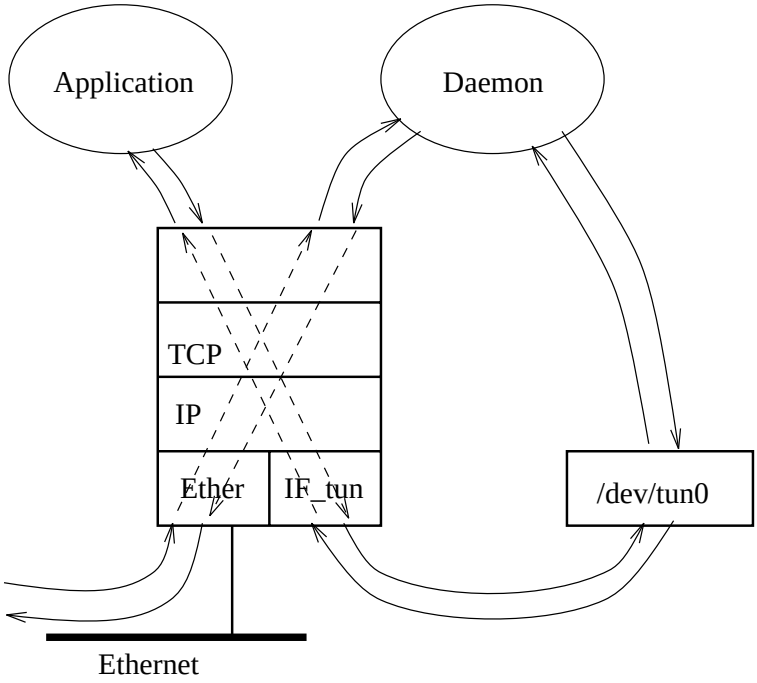


図 3.8: Pnet の仕組み

手アドレスとして設定 (ifconfig) されなければならない。また、経路制御表には宛先 P_b に対する経路として仮想インタフェースが指定されるよう設定されなければならない。ところがポンプ・モジュールによりカプセル化されたパケットの宛先も P_b にせざるをえないことから、そのままではパケットは再び同じ仮想インタフェースに投げられることとなり、パケットは無限ループにおちいってしまう。

したがって、unnumbered link が実装できるかどうかの鍵となるのは、ポンプ・モジュールが経路制御モジュールに対してパケットの発信依頼をするとき、そのパケットを送ってきた仮想インタフェースを経路として利用しないよう指示できるかどうかという点である。4.3BSD tahoe では、経路制御表から経路を選択する瞬間だけ、パケットが来た仮想インタフェースに簡単に「ダウン」マークをつけることができるため、unnumbered link が容易に実装できる。これはインタフェースの IFF_UP というフラグをその瞬間だけ off することで行なうことができる。

3.3.5 別の実装方法 — Pnet アプローチ

DDT の実装方法として Pnet[50] のアプローチもある。図 3.8 に Pnet の仕組みを示す。Pnet は tunXX という仮想インタフェースと、/dev/tunXX というデバイスとからなる。

IP モジュールにより tunXX に送られたパケットは /dev/tunXX を通して読み出すことができ、/dev/tunXX に書き込まれたパケットは tunXX から IP モジュールに送られるようになっている。これはちょうど、仮想端末の裏表が ptyXX と ttypXX の組みで構成されるのに似ているところから Pnet という名前がつけられた。

Pnet では、トンネリングのプロトコル処理はユーザ・レベルのデーモンで行ない、カーネル内には仮想インタフェースと仮想デバイスの間の橋渡しを行なうモジュールだけしか置かれない。3.3 節で紹介した DDT の実装にたとえると、ポンプ・モジュールがユーザ・レベルに置かれている状態と言える。

3.3 節で紹介した DDT の実装と比較した利点・欠点として次が挙げられる。

- 利点
  - カーネル内のモジュールが小さいため移植性が高い。
  - 重要な部分はユーザレベル・アプリケーションとして動くのでデバッグが容易。
  - プロトコルはユーザレベル・アプリケーションが処理するので、様々な機能を持つプロトコルを容易に実装できる。
- 欠点
  - カーネル空間、ユーザ空間の間のコピーの回数が多い。
  - コンテキスト・スイッチが頻繁に起きる。
  - 3.3.4 節に書いた理由により “unnumbered link” の実装が難しい。

以上から Pnet は、トンネリングのために複雑なプロトコル処理を必要とするが、処理速度は厳しくは求めないときに便利な実装方法であると言える。

第 4 章

トンネリングの問題の解決方法

トンネリングには様々な応用が考えられているが、未解決な問題点も残されている。本章ではトンネリングの問題点を 2.3 節で行なった分類ごとに解決方法を提案する。

4.1 経路上の問題

トンネリングの問題点の内、経路上の問題及び経路制御の問題についてはトンネリングを包含したモデルの上で議論すべきである。そこで、まず自由順序階層モデルを TP-1 に対する解決案として提案する。そしてそのモデルの上で TP-2, TP-3, TP-4 について考える。

4.1.1 自由順序階層モデル

狭義のトンネリングが階層モデル違反を引き起こす原因は、従来の階層モデルでは階層間の序列、各層の役割分担、全体の深さが厳格に決められていたことにある。実はこれらは、複数のプロトコル・ファミリにまたがる広義のトンネリングを扱えなくする原因でもある。トンネリングを議論する上でベースとなりうる階層モデルでは、それらの制約をなくさねばならない。そこで本稿では次のような自由順序階層モデルを提案する。

- 各プロトコルの関係は階層的である。
- 目的に応じて自由な順序でプロトコル階層を組み立てることができる。
- ただし、各プロトコルが下の層に対して要求する制約条件は満たさねばならない。

自由順序階層モデルはトンネリングを議論できるだけでなく、暗号・認証・圧縮等の機能を持つ層を任意の場所に挿入を許す枠組みを提供することもできる。様々な階層を任意の順序で組み立てるためには、各層の SAP (Service Access Point) のインタフェースと制約条件記法が共通化されている必要がある。その意味では各ノードにおけるプロトコル選択機構は System V の STREAMS に似ている。図 4.1 にプロトコル階層選択の例を挙げる。右側の箱が利用可能なプロトコルの集合である。左側には、暗号化 IP を利用してメイルを送信するために組み立てたプロトコル階層である[1]。図の例のように、自由順序階層

[1] ここでは 2.2 節 4 項で述べた方法での暗号化を仮定している。

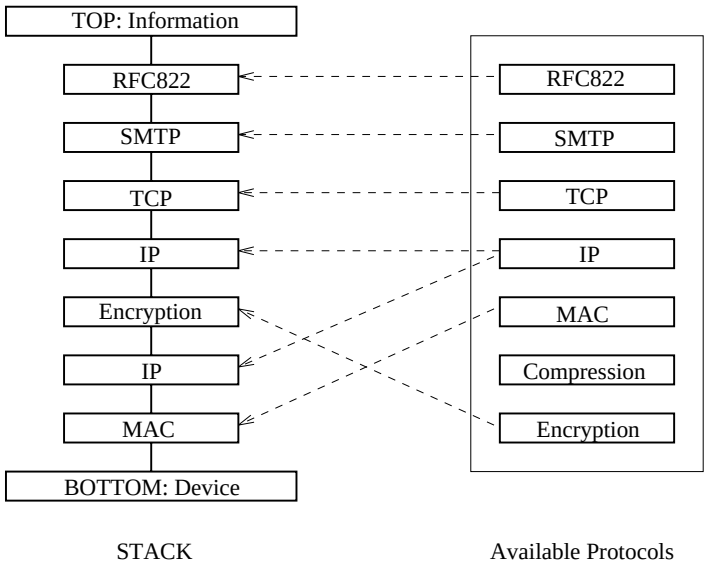
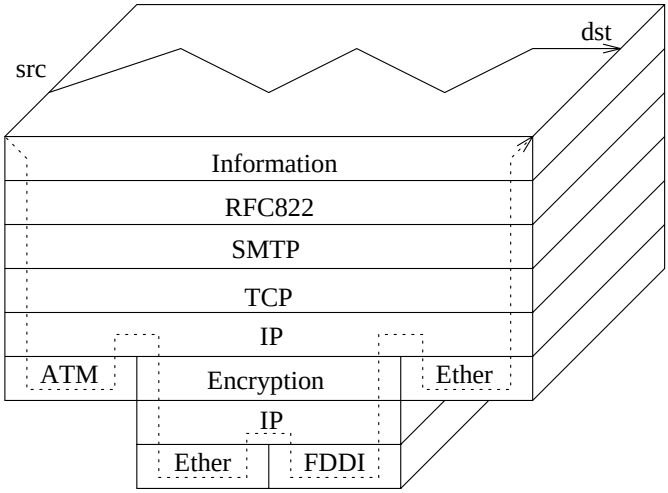


図 4.1: 自由順序階層モデルにおけるスタック

モデルでは “IP” を複数回プロトコル階層に登場させることができる。自由順序階層モデルはあくまでもプロトコル階層モデルなので、実装上どこまでをカーネルで行ないどこからをユーザ・アプリケーションで行なうかには関知しない。その点で、実装モデルである STREAMS とは異なる。

通信経路を通しての階層の関係は、横軸に始点から終点までの経路、縦軸に各点における階層構成を取ることにより表すことができる。このとき、縦軸方向の再上層には伝えるべき情報が位置し、再下層には物理媒体が位置する。その間には、ネットワーク機能やトランスポート機能等を持つ層が任意の順番で任意の数並んでいる。階層の深さは経路上で一定ではなく、ところによって異なっている可能性がある。横軸の通信経路は一般にトポロジーを持つので 2 次元平面と考えることができることから、縦軸の階層を含めた全体では 3 次元空間と考えることができる。図 4.2 は自由順序階層モデルにおけるパケットの経路の例である。横軸は (特に OSI 7 階層で言うネットワーク層の機能を持つ層では) 図の上面に実線で示すように 2 次元のネットワーク・トポロジーが存在する。ところが、縦軸方向に注目するとパケットは破線で示す経路を通るとみなすことができる。したがって、縦軸・横軸を合わせて考えると、パケットは 3 次元空間で動いていると考えることができる。

自由順序階層モデルでは、従来の階層モデルはプロトコル階層の構造を理解するための “view” とみなされる。従来の階層モデルによる view がひとつだけ定義できるとき、そのモデルが保証する安全性を得ることができる。逆に複数の view が存在するとき、個々の view の前提が矛盾して問題が発生する可能性がある。トンネリングが行なわれるときは、図 4.3 に示すように view が複数存在している。



実線　：　横軸方向の軌跡 (2 次元平面での軌跡)
破線　：　縦軸方向の軌跡
(注意)　　実線と破線の交わり部分が 3 次元空間での軌跡である。

図 4.2: 自由順序階層モデルにおける経路

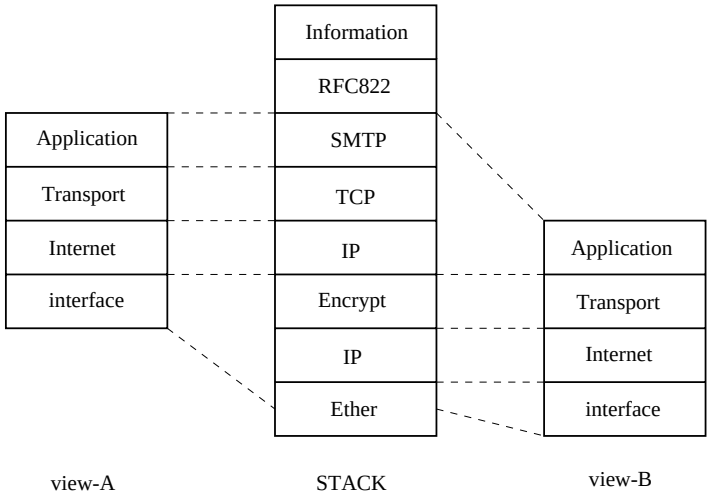


図 4.3: 自由順序階層モデルに対する “view”

4.1.2　自由順序階層モデルにおけるトンネリング

自由順序階層モデルにおいてトンネリングを再定義する。

- 広義には、ネットワーク層の機能を持つ層が複数存在する階層構造を作ることである。
- 狭義には、アドレス空間を共有するネットワーク層の機能を持つ層が、複数存在する階層構造を作ることである。

このように自由順序階層モデルはトンネリングを自然に記述できる (TP-1 参照)。

従来のネットワーク技術が 2 次元平面のトポロジーを対象としたものであることを考えると、それをそのまま 3 次元空間の現象であるトンネリングに適用するのは危険であることが容易に想像がつく。また、自由階層モデルはトンネリングを包含していることから、トンネリングの経路上の問題点 TP-2, TP-3, TP-4 及び経路制御の問題点 TP-5 は自由順序階層モデルにおいても同様に問題となる。逆に、自由順序階層モデル上でこれらの問題に対する一般的な安全性保証手法を構築し、その手法をトンネリングの利用場面の状況に応じて適用するようにすれば、それはそのままトンネリングにおける上記問題点に対する解決方法にもなる。

そこで以下では、自由順序階層モデルにおいてこれらの問題の解決方法を考える。ここで取り上げる問題点はネットワーク層の機能を持つ層における問題なので、以下では特に断らない限り自由順序階層モデルの中でもネットワーク層の機能を持つ層にだけに注目して議論する。解決方法の提案は 2 次元平面におけるネットワーク技術を 3 次元空間のネットワーク技術に拡張する形で行なう。

4.1.3　経路上の問題点

自由順序階層モデルにおいて経路上の問題点 (TP-2, TP-3, TP-4) を解決するには以下を実現せねばならない。

1. 目的地に到着できないパケットを消滅させる。(TP-2)
   (2 次元平面のネットワークでは主に TTL が利用されている。)
2. 目的地まで可能な限り希望に沿った経路を通す。(TP-3)
   (2 次元平面のネットワークでは QoS にしたがった経路選択がなされている。)
3. ネットワーク中のパケットを識別可能にする。(TP-4)
   (2 次元平面のネットワークではプロトコルを見れば簡単にわかる。)

これらを 3 次元空間のネットワークで達成する方法は、階層を縦方向に移動する時のカプセル化の型によって異なる。考察すべきカプセル化の型がどのくらいあるのかはまだわかっていない。ここでは「Connectionless 型」と「Connection-Oriented 型」の 2 つを提案する。

- Connectionless 型
  Connectionless 型とは、パケットをひとつひとつカプセル化して別々に送るものを言う。パケットの中のデータに注目すると、移動範囲が 2 次元平面のネットワークに閉じず、3 次元空間のネットワークに広がったと考えることができる。したがって、基本的には TTL, QoS, ID (識別情報) などの属性がパケットの中のデータ自身に付随するものであるとみなし、それらの TTL, QoS, ID などの属性を一番外側のカプセル化ヘッダに書くようにすることで、3 次元空間のネットワークにおいても 2 次元平面のネットワークと同様の手法で上記の目標が達成できる。以下に目標別に解決方法を提案する。

  1. 目的地に到着できないパケットを消滅させる。
     2 次元平面のネットワークと同様に TTL で行なう。横軸方向の移動時には従来と同じく移動のたびに減少させれば良い。縦軸方向に階層を降りる時にも TTL を減少させる。これは無限に降り続けるのを防ぐ意味がある。縦軸方向に上がる時には、無限に上がり続ける心配はないので TTL を減少させる必要はない。縦軸方向に移動した先の層の TTL 値の範囲が移動前の層のものと異なる場合には、TTL 値をマップしなければならない。
  2. 目的地まで可能な限り希望に沿った経路を通す。
     2 次元平面のネットワークと同様に QoS で行なう。横軸方向の移動時には QoS は従来通り値をそのまま保存する。縦軸方向に降りる時には、次の層に合わせて QoS をマップしなければならない。
  3. ネットワーク中のパケットを識別可能にする。
     2 次元平面のネットワークと同様、パケット自身が識別情報 (以下 ID と書く) を持っていなければならない。横軸方向の移動時には ID は従来通りそのまま保存する。縦軸方向に降りる時には、なんらかの形で ID を提示しなければならない。

  TTL, QoS, ID をマップするための具体的な手法、及び 3 次元空間のネットワークにおける ID の提示方法と詐称防止方法は今後の研究課題である。

- Connection-Oriented 型
  Connection-Oriented 型とは、パケットをストリーム内に直列に並べて、フロー制御をしながら送受信するものを言う。ストリームは同じ層内でしか作成できないため、フロー制御に関しては 2 次元平面のネットワークと同じ手法が利用できる。以下に目標別に解決方法を提案する。

  1. 目的地に到着できないパケットを消滅させる。
     2 次元平面のネットワークと同様、フロー制御で行なう。ストリーム自身を構成するパケットが目的地に到着できない場合に消滅させることは、フロー制御プ

ロトコルの責任で行なわねばならない。フロー制御プロトコルが TCP のようにパケットをベースとしたものであった場合、Connectionless 型トンネルの手法が利用できる。

2. 目的地まで可能な限り希望に沿った経路を通す。
ストリームを通る個々のデータは一般に同質であると仮定されているため、ストリームを複数用意して QoS に合わせてストリームを選択しなければならない。例えば ftp のように従来から存在するプロトコルにおいても、制御情報用のストリームとデータ転送用のストリームを別々に用意して、異なる port と TOS をつけている。これと同じ考え方である。

3. ネットワーク中のパケットを識別可能にする。
上記 QoS と同じ手法が利用できる。ただし、ID は QoS に比べて種類が非常に多いため、実際にはすべての ID ごとにフローを用意するのは難しい。パケットの完全な識別はあきらめるのが現実的と考える。
異なるアプローチとして、フロー制御プロトコルがパケット・ベースのものであった場合に、プロトコルを拡張して個々のパケットの中身の ID 情報を提示する機能を付加する方法も考えられる。

4.2　経路制御の問題

4.2.1　経路制御における課題

ここでは、トンネリングによる仮想ネットワークを含んだネットワークにおける経路制御を考える。その場合の経路制御は、通常のループを避ける以外に次の点に注意しなければならない。

- 定常状態の維持
トンネリングを正しく機能させるため、トンネルの出入口への経路を確保しなければならない。IP 層における経路制御を考えた場合、以下のような手法が考えられる。

  1. トンネルの入口から、トンネル出口またはトンネルの出口を含むネットワークの経路情報を流さないようにする。
  2. トンネルの出口から、自身の host route を低い metric で流す。トンネルの出口が他の AS にあるときは、その AS に最も近い border gateway からトンネルの出口の host route を低い metric で流す。
  3. source route を利用して、トンネルの出口にたどりつくように設定する。
  4. “A except B” 型の経路情報を導入する。これは、ネットワーク A への経路がトンネルを経由しているとき、ネットワーク A に関する経路情報に、トンネルの出口 B への経路はそれには含まない旨を明示的に書くものである。

- 障害時のフェイルセーフ
  トンネルを抜け出ることなく再び同一トンネルに入るような無限ループを避けなければならない。これは障害発生時にのみ起きる問題である。そのためには次を考える必要がある。

  – 障害発生時に問題を起こさないための方法。
  – 障害発生時に仮に問題が発生しても、それを素早く回避する方法。
  – 障害発生後、定常状態に早く回復する方法。

- 最適な経路選択
  経路制御の方針通りにトンネリングを利用しなければならない。また、トンネルの cost/metric を物理ネットワークの状態に応じて正しく評価し、意味のないトンネリングは避けなければならない。

これらの問題に対して、まだ完全な解決方法を提案できていない。しかし、次のように場面を限定すれば現在でも経路制御の問題を避けてトンネリングを利用することができる。

- エンド・エンドの通信で利用場合には経路制御の問題は発生しない。
- source route を利用して、トンネルの出口までの経路を確保する。
- トンネリングによる仮想ネットワークに関する経路情報を流さないようにして、代わりに管理者が慎重に静的な経路設定を行なう。例えば 3.2.2 節で述べたものを含めて、比較的小規模なネットワークではほとんどの場合矛盾のない設定が可能である。しかし、何に注意してどの程度慎重になれば良いかという指針はまだわかっていない。

トンネリングの応用に際して以上のような制約があるのは望ましくない。どのようなトポロジーでも専門的知識なしでトンネリングが利用できるようにするための経路制御の確立が望まれる。トンネリングによる仮想ネットワークを含んだネットワークにおける経路制御を自動的に行なうためには、取り組み方がいくつか考えられている。

1. 問題となるネットワーク・トポロジーの適切な表現方法を考えて、そのトポロジー上で経路制御のアルゴリズムを考える。
2. 意味論や証明論を応用してループを検出する。

以下に、前者の方針で取り組んだ結果を書く。ただし、ここに書いた方法はまだ完全ではなく、今後改良が必要である。トポロジーの表現方法としては、前節の自由順序階層モデルを利用した。

4.2.2　自由順序階層モデルの元での解決方法案

自由順序階層モデルでは、同じアドレス空間を持つ層がプロトコル階層上に複数回出現する可能性がある。そのため各層の経路が複雑に依存し合う可能性がある。そこでまず、従来からある 2 次元平面のネットワークの経路制御が適用できる条件、及び 3 次元空間のネットワークの経路制御が新しく必要となる条件を考える。これは、各層間の依存関係の有向グラフの性質によって分類できる。ここでは依存関係を次のように定義する。

定義: 層の依存関係

- ある層 x に存在する経路のトラヒックを下の層 y が運んでいるとき、層 x は層 y に依存しているとする。
- 同じアドレス空間を持つ層は互いに依存し合っているとする。

この定義にしたがって依存する層から依存される層に矢を描けば、各層間の依存関係を表す有向グラフが作成できる。できあがった有向グラフ上のループに注目して、ループに含まれている層とループには含まれていない層に分類する。

1. ループには含まれていない層
   これはちょうど既存の 2 次元平面のネットワークにおける経路制御技術が仮定した状況である。この場合は、各層ごとに既存の経路制御技術を適用すれば良い。

2. ループに含まれている層
   ループに関係する層全体で経路制御を考えなければならないため、3 次元空間のネットワークにおける経路制御技術が必要になる。なお、以下では同じループに含まれている層同士を「相互依存関係」にあると言う。

以上より、層間に相互依存関係があるときには、相互依存関係にある複数の層全体に対して 3 次元空間のネットワークのための経路制御技術を適用しなければならないことがわかった。

次に、ループに含まれる層全体における経路制御方法を考える。3 次元空間のネットワークにおける経路制御の問題点 (TP-5) の解決のためには、2 次元平面のネットワークにおける経路制御に加えて次の 3 点を解決しなければならない。

- クラインの壷型ループ[2]を避ける。
- トンネルの出口への経路を確保する。
- 最適経路選択

[2]「クラインの壷型ループ」の用語の意味は TP-2 参照

これらの問題を解決するには、経路情報プロトコルは 3 次元空間のネットワークにおける経路の依存関係を処理しなければならない。ここでは経路の依存関係を次のように定義する。

定義: 経路の依存関係

- 経路 x を通過するトラヒックが実際にはその下の層にある経路 y を通過しているとき、「経路 x は経路 y に依存している」と言う。

  例えば、ふたつのホストが Ethernet で結ばれていてその上に IP を通しているとき、ふたつのホストの間の IP の経路は Ethernet による経路に依存していると言う。

この定義を用いて、3 次元空間のネットワークにおける経路制御プロトコルのために必要となる情報と、その情報を用いた経路制御アルゴリズムを次のように提案する。

- 経路情報プロトコルに新たに必要となる各経路に関する情報

  1. 依存している経路のうち、相互依存関係にある層に含まれる経路。
  2. 両端のルータのアドレス
     ただし、アドレスがインタフェースに対してつけられるプロトコルの場合、相互依存関係にない層を下の層として持つアドレスを利用する。
  3. cost/metric や up/down など他経路に依存する情報は必要ない。

- 経路制御表作成アルゴリズム

  1. クラインの壷型ループ[3]を検出して利用しないようにする。
     1-1 経路利用の依存関係を有向グラフで表現する。
     経路 x が経路 y に依存しているとき、x から y に矢印を描く。
     1-2 できあがった有向グラフのループに含まれる経路に “down” マークをつける。
     1-3 依存する経路がひとつでも “down” になっている経路には “down” マークをつける。
  2. トンネルの出口への経路を確保する。
     トンネルを含む経路を利用するときは、その経路の出入口及び依存している経路の出入口に対する経路情報を追加する。
  3. 最適経路選択
     通常のアルゴリズムを各層毎に適用する。

[3]「クラインの壷型ループ」の用語の意味は TP-2 参照

以上は、Distance Vecotr 型、Link State 型いずれの型の経路制御プロトコルであっても適用可能であると考えている。特に層の依存関係のループが同じアドレス空間を持つ層に限定されているときには、既存の経路制御プロトコルを拡張することで比較的簡単に実現できるであろう。ただし、拡張の対象となる既存のプロトコルは可変長の経路の属性が扱えなければならないであろう。

層の依存関係のループが異なるアドレス空間にまたがる場合には、相互依存関係にあるすべての層をひとつの経路制御プロトコルで処理しなければならない。そこで用いられる経路制御プロトコルは、アドレス情報として、プロトコル・ファミリ、アドレス空間、アドレスの組を交換しなければならないが、それ以外は大枠としては既存の経路制御プロトコルと大差ないであろう。マルチプロトコル環境では相互依存関係を作りたい場面が多々あるかもしれないことを考えると、このような経路制御プロトコルも必要になってくるであろう。

以上の方法の検証と改良は今後の研究課題である。

4.3　プロトコルに関する問題

エラー報告方法 (TP-6) の問題及びオーバーヘッド (TP-7) の問題は個々のプロトコルに依存する度合が大きいため一般的な解決方法を述べるのは難しい。解決のためには、プロトコル・ファミリごとに、カプセル化プロトコルを工夫してバンド幅や CPU の消費の少なくしていく地道な作業となると考えられる。

4.3.1　IP の場合の解決方法案

IP データグラムがトンネル内でエラーに遭遇したとき、トンネルの入口に対して ICMP が送られる。トンネルの入口は ICMP の種類を見て、その ICMP を無視すべきか元々のデータグラムの発信者にネットワークのエラーを伝えるべきか判断しなければならない [37]。

トンネルの入口が元々のデータグラムの発信者に対してネットワークのエラーを伝えるためには、トンネル内のルータから送られてきた ICMP から、元々のデータグラムの発信者にとって意味のある ICMP を生成する必要がある。ところがトンネル内で生成された ICMP には、エラーに遭遇した IP データグラムのヘッダ (カプセル化ヘッダ) とデータの先頭 8 オクテット (カプセル化プロトコルの先頭 8 オクテット) のコピーしか書かれていない。つまり、元々のデータグラムの発信者にとって意味のある ICMP を生成するにはカプセル化プロトコルを工夫しなければならない。

そのためのプロトコルを 2 つ提案する。

opt	len	type	proto
Source Address			
Destination Address			
First eight octets of data			
part of IP option if possible			

図 4.4: IPOPT_TUNINFO

4	2	Dep	HL	Identifier
DF,MF,and Offset				IP opt if any
src/dst pairs if any				
IP data without first eight octets				

図 4.5: IPPROTO_DDT の例

IPOPT_TUNINFO

IP option は IP データグラムのヘッダに含まれるため、トンネル内から返ってくる ICMP に必ず含まれる。図 4.4 に示すように、IPOPT_TUNINFO には元々のパケットの識別のための情報が入っている。データ部の先頭 8 オクテットには TCP, UDP のポート番号のようにパケットを識別するための重要な情報が入っていることに注意して欲しい。この IP option は、connectionless 型のカプセル化を行なう時にルータに対して元々のパケットの識別情報を与えている。

IPPROTO_DDT

カプセル化プロトコル IPPROTO_DDT のフォーマットを図 4.5 に示す。IPPROTO_DDT は IPOPT_TUNINFO と同時に用いることを想定している。そのため、IPOPT_TUNINFO が持つ情報はプロトコルから省略してある。また、トンネルの入口が元々のデータグラムの発信者にとって意味のある ICMP を生成できるよう、カプセル化プロトコルの先頭 8 オクテットに重要な情報を集めている。さらに、他のフィールドから計算可能なフィールドについても省略することでバンド幅の消費を抑えている。

IPOPT_TUNINFO と IPPROTO_DDT を組み合わせて利用すれば、トンネル内で発生

Field	From where
VERS	IPPROTO_DDT
HL	IPPROTO_DDT
TOS	Encapsulation header
LEN	Encapsulation header
ID	IPPROTO_DDT
OFFSET	IPPROTO_DDT
TTL	Encapsulation header
PROTO	IPOPT_TUNINFO
Check sum	Computable
SRC, DST	IPOPT_TUNINFO
IP options	IPOPT_TUNINFO if possible
first 64bits	IPOPT_TUNINFO

表 4.1: 意味ある ICMP の生成方法

した ICMP を元に表 4.1 に示す方法により元々のデータグラムの送信者に対して意味ある ICMP を生成できる。

異なるアプローチとして、各パケットの ICMP を生成する上で必要な情報をトンネルの入口が記憶して、そのインデックスをカプセル化プロトコルのヘッダに書く方法がある。この方法の場合、ギガビット・トラヒックのような大量のトラヒックをさばくには大量の記憶域が必要となるという問題がある。

4.4　実装に関する問題

実際には厳格な MTU は存在しない層であっても、実装時上 MTU が必要になる場合ある (TP-8)。その下の層が物理ネットワークである場合は、トンネルの経路における最小の MTU をその層の MTU とするのが良い [46][47][48]。そうでないときは、その下の層の MTU からその層におけるカプセル化ヘッダの大きさを引いたものを MTU にするのが良い。その結果、各ノード内において層間で依存関係のループができたとき MTU が 0 に収束するため、パケットがループに飛び込むのを防ぐことができる。

4.5　ネットワーク管理上の問題

管理者がトポロジーを把握できなくなる問題 TP-9 は、トンネリングの利用以外にも、ISDN などの公衆網を利用してユーザが管理者に無断で他サイトと IP 接続を行なった時に

も起きうる。つまり TP-9 はトンネリング固有の問題ではない。ただし、公衆網を利用する場合には接続機器の購入や回線利用料金が必要なため、金銭的な歯止めがかかりうる点で異なる。また機器を利用するため、各ユーザがそれによる影響を推測できる可能性が高いことも歯止めとなっている。ネットワーク利用規約などで扱いを明文化している組織も大いであろう。一方、トンネリングによる仮想リンクは馴染みのない概念であり、ユーザ、管理者ともにそれによる影響を想像するのは難しい。従って、トンネリングに関する「常識」を作っていくことが大切である。例えば、他組織との間にトンネリングを張る場合は firewall を構築しなければならないとか、トンネリングを張った場合には必ずネットワーク管理者に報告しなければならないなどである。

トンネルを通過中のパケットの種類はトンネルの外からわかりにくいという問題があるが、これもトンネリング固有の問題ではない。トンネリングを使わなくても始点と終点が協調すれば port 番号はいつでも任意のものを使うことができるからである。また、ftpmail, telnetx のように外から見ると確かにメイルであり telnet であるものでも、実際にはファイル転送であったりするものもある。

この問題を考える前に、まずネットワークを通過中のパケットの種類の情報として外から見えるべきものは何かを議論せねばならない。候補としては中に入っている情報の種類、プロトコルの種類、QoS が挙げられる。このうち中に入っている情報の種類については、「情報の種類」を数え挙げることが難しいこと、ひとつのパケットもしくはフローに複数の「情報の種類」が混在する可能性があることから、意味のある情報としてパケットに付加するのは不可能である。したがって、パケットにはプロトコルの種類と QoS が正しく付加され、ルータではそれらが正しく扱われなければならない。

4.6 まとめ

以上、2.3 節に列挙した問題点に対する解決方法の提案を述べた。このうち、TP-1, TP-5, TP-9 については、完全な提案がまだできていない。

このうち経路制御については、場面を限定すれば現在でもトンネリングは問題なく利用できることがわかっている。例えば、end-to-end の通信では経路制御上の問題は発生しない。また、トンネリングの小規模な利用であれば、慎重に静的な経路設定を行なえばほとんどの場合矛盾のない経路設定ができるようだ。3.2.2 節の「アドレス空間の節約」で述べたような利用方法の場合は、簡単に矛盾のない経路設定が可能である。しかし、何に注意してどの程度慎重になれば良いかという指針はまだわかっていない。どのようなトポロジーでも専門的知識なしでトンネリングが利用できるようにするための経路制御の確立が望まれる。

第 5 章

DDT WG の活動のまとめ

トンネリングは仮想ネットワーク構築手段として注目を集めている。ところが汎用的な利用方法が存在しなかったため、トンネリングを必要とするモジュールは目的に合わせて個々に設計・実装しなければならなかった。そこで、トンネリングを仮想ネットワークとして抽象化し、それを仮想インタフェースを通してアクセスするという、汎用的なトンネリングの利用モデル “DDT” の提案を行なった。

DDT は既存のトンネリングの応用分野のほとんどに適用可能であるばかりか、トンネリングの応用の設計を容易にする。さらに、DDT はアドレス空間節約のために応用可能であることを示した。DDT による解決方法は、既存の解決方法である NAT よりも優れていることも示した。また、DDT の実装の目標と現在の構造、及び実装上問題となる部分についての回避方法を示した。

DDT の実験を通して、トンネリングの問題点が明らかになった。そこで問題点の多くに対して解決方法の提案を行なったが、経路制御などいくつかの問題点についてはまだ有効な解決方法を提案できていない。しかし、場面を限定すればトンネリングは問題なく利用できることも示した。

トンネリングに関してまだ問題点は残されてはいるが、以上の成果をおさめたので、DDT WG としての活動はこれで終了する。

ネットワーク技術には、仮想的なネットワークを利用してこそ実現できるサービスや機能が存在する。トンネリングは仮想ネットワークを構築する有効な手段であることを考えると、トンネリングの残っている問題点の早期解決が望まれる。