第 5 部

トンネリング技術

第 1 章

はじめに

1.1 DDT WG の目標

近年、インターネット技術においてトンネリングの利用が多く見られるようになってきた。一方で、トンネリングは未解決な問題点を抱えた技術でもある。そのため、トンネリングは危険で避けるべき技術であると考えると考えるインターネット研究者も多い。そこで、DDT WG は次のふたつを目標として活動している。

1. トンネリングの汎用的な利用モデル “DDT” を提案・実装する。

2. トンネリングの問題点の解決方法や回避方法を示す。

本章では以下、DDT WG の報告書の導入として、トンネリングの定義、利用目的、問題点を紹介し、最後に DDT WG の報告書の構成を述べる。

1.2 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 x と y とが等しい場合を指す。

歴史的には、図 1.1 に示すように組織間を 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。

以下では、トンネルに入れるための操作を「カプセル化」または “encapsulation”、それを行なうルータを「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。トンネルから出すための操作を「脱カプセル化」または “decapsulation”、それを行なうルータを「トンネルの出口」と言う。

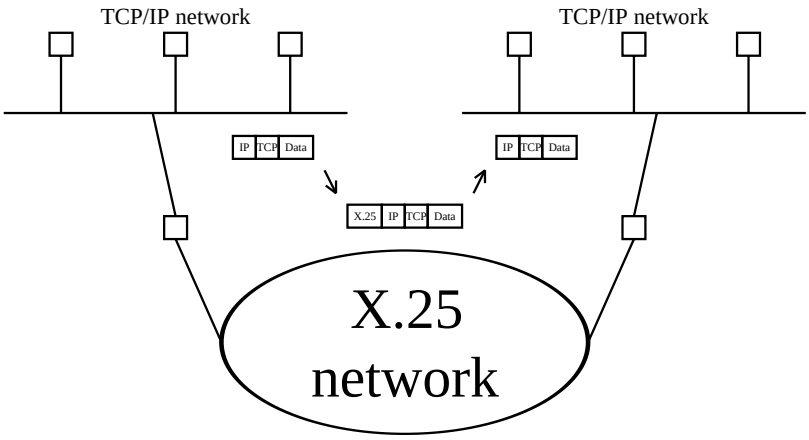


図 1.1: TCP/IP over X.25

1.3 トンネリングの利用目的

トンネリングはインターネット技術において様々な目的で利用されている。以下に具体的な利用例を目的別に紹介する。

1. 新しいプロトコルや異なるプロトコルを運ぶ

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を扱えないルータはまだまだ多い。マルチキャスト・パケットの配送時にそのようなルータをスキップするためにトンネリングが利用されている。トンネリングのプロトコルとして “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 (IPng) が議論されている。IPv4 から IPng への移行を円滑に行なうために、移行初期には “IPng over IPv4” の形で IPng 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 [64][65]。

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”[66], “OSI CLNP over IP”[67][68] 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)[69][70] が提案されている。

2. 移動ホスト支援

コロンビア大学方式 [71]
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は、“IP within IP” (ipproto=94) を独自に定義し、利用している。

3. 経路制御
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の仮想ネットワークを提供できるため、経路制御を簡単にする目的で利用することができる。

経路制御技術の制限の回避
ルータの forwarding 技術の制限 [72] もしくは経路制御プロトコルの制限 [73] 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。

IDRP
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トンネリングの利用を考えている [74]。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場合、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かどうかが問題になる。正式に割り当てられたアドレスではない場合、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 [75]。

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”[76] が提案されている。“PPP over TCP” も利用可能である [77]。

4. その他

IP option の拡張
現在の IP の IP option 長は最大 40 オクテットしかなく、暗号化のための情報等を送るには小さすぎる。そこでトンネリングの形を借りて、暗号化のための情報をトンネリング・プロトコルに持たせることが考えられている。

1.4　トンネリングの問題点

トンネリングは広く利用されているにもかかわらず、以下に示す未解決な問題点があることが知られている。以下では *TP-num* の形で参照する。

1. 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しなければならない。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。そのとき、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、ちょうど爆発したような状態になる [78]。

2. 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 QoS が無視されたり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。

3. 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、ネットワークを評価する上で支障となる。

4. 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。

   - 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る (以下では「クラインの壷型ループ」と呼ぶ)。
   - 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。
   - トンネリングの cost/metric を正しく評価し、無意味な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。

5. 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元々のパケット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、如何にして元々のパケットの発信者に正しいエラー報告を返すかが問題となる。具体的な方法論はプロトコル・ファミリごとに考えねばならない。

- トンネル内のルータが元々のパケットの発信者に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の見つけ方と、エラー報告パケットを正しく組み立てる方法が必要になる。
- トンネルの入口が元々のパケットの発信者に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 [76] 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組み立てるための方法が必要になる。

6. 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。

7. 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。

8. 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。

9. モデルとの相性
   トンネリングは現在のプロトコル階層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、現在のプロトコル階層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モデルを当てはめることはできない。

10. トンネリング利用モデル
    トンネリングを利用する場合の汎用的な利用モデルが存在しない。そのため、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・実装しなくてはならない。

1.5 DDT WG 報告書の構成

以上で見たように、トンネリングは広く用いられているにもかかわらず未解決な問題点を多く抱えている。以下では、トンネリングの問題点の解決方法や回避方法を次の構成で示す。

2 章 — トンネリング利用モデル “DDT” の提案
TP-10 の解決のため、汎用的なトンネリングの利用モデルである “DDT” を提案する。“DDT” は本ワーキング・グループの名称でもある。

3 章 — トンネリングの問題点の解決
トンネリングの問題点はトンネリングを包含したモデルの上で議論すべきとの認識から、まず TP-9 の解決のために自由順序階層モデルを提案し、そのモデルの上で TP-1, TP-2, TP-3, TP-4 について考える。残った問題 TP-5, TP-6, TP-7, TP-8 については自由順序階層モデルとは独立に考える。

4 章 — DDT のネットワーク社会における影響に関する考察
“DDT” は、技術的な期待と不安の両方の目で見られている。そこで、“DDT” が持つ意味や、“DDT” を使ってネットワークを運用する事が内外部に与える影響などについてミーティングを開いて議論した。その結果を報告する。

5 章 — DDT WG の ’94 年度活動計画

第 2 章

トンネリング利用モデル

2.1 DDT の提案

トンネリングは基本的には仮想ネットワークを構築する手段である。Ethernet のような物理ネットワークと同様、仮想ネットワークもネットワーク・インタフェースを通してアクセスするのが自然であろう。その結果、仮想ネットワークを物理ネットワークと同様に扱うことができるようになる。例えば経路制御においても、トンネルも物理ネットワークも同じレベルで扱うことができることが期待される。以上の考察より、トンネリングを利用する際の汎用的なモデルとして “DDT”[1] を提案する。DDT をひとことで説明すると次のようになる。

> **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。

DDT モデルでは、仮想ネットワークは通常のネットワークと何ら変わりなく存在する。図 2.1 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。このアドレス付けの結果、仮想ネットワーク N_v はインターネットアーキテクチャにおいて通常のネットワークと同様に扱うことができるようになる。

図 2.1 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 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。

[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。

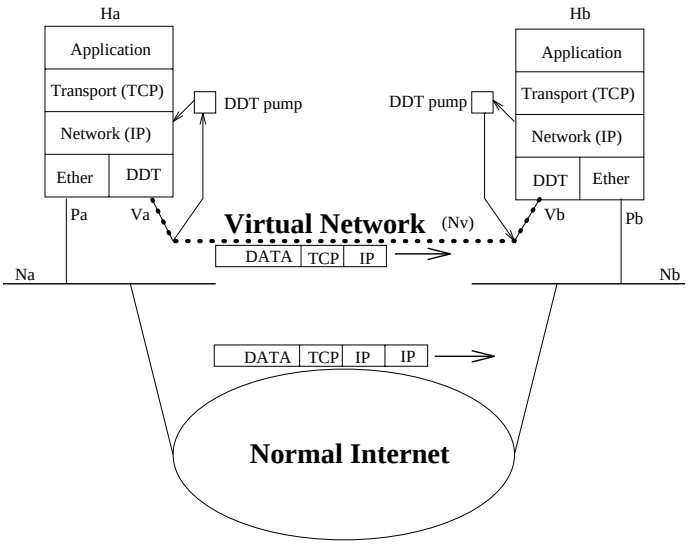


図 2.1: IP over IP の場合の DDT

図 2.1は、実際のパケットの動きをも示している。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 2.1 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。

2.2 応用例

2.2.1 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た

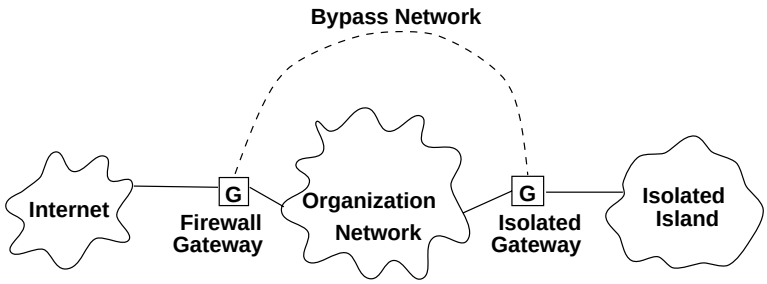


図 2.2: 離れ小島

め、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し、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。

2.2.2　アドレス空間の節約

企業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。にもかかわらず、企業は社内のすべてのホストやサブネットにインターネットに到達可能なアドレスを割り振る。その理由として、将来的に社内のすべてのマシンから安全にインターネットに直接アクセスできるようになったときへの備えがある。その結果、企業はインターネット・アドレス空間を広く消費し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。

この問題も DDT により解決可能である。インターネットに直接アクセスする必要ができたホストは、その時だけ動的に firewall ゲートウェイとの間に DDT による仮想ネットワークを張り、動的にアドレスを付けるようにすれば目的が達成できるからである。これにより、すべてのホストにインターネットのアドレスを割り振る必要がなくなるばかりか、アドレスの共有・再利用を行なうことにより比較的狭いアドレス空間で目的が達成できるようになる (図 2.3 参照)。

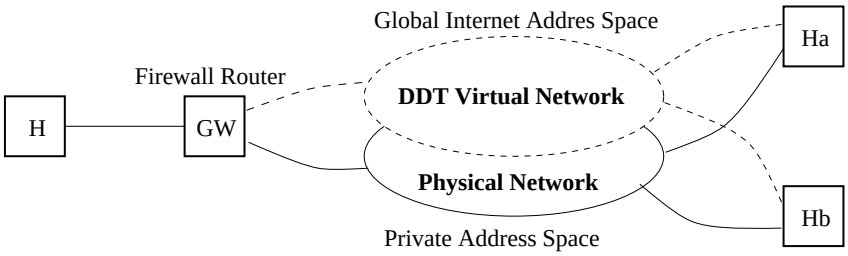


図 2.3: DDT による仮想ネットワーク

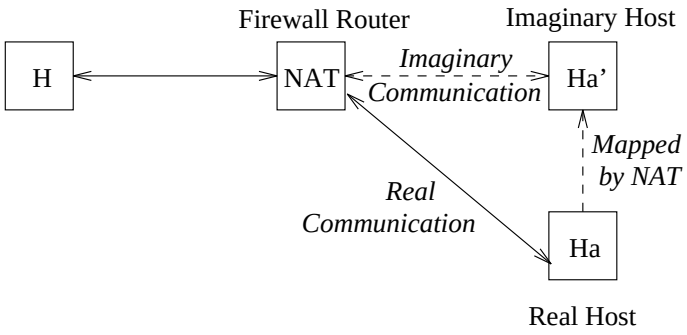


図 2.4: NAT

NAT (Network Address Translator)[79] もアドレス空間の節約のために利用可能であるが、DDT の方が実装が簡単である。NAT は firewall ルータに組み込まれ、出ていくパケットの中にあるすべての組織内ネットワーク・アドレスを疑似アドレスに書き換え、入ってくるパケットの中にあるすべての疑似アドレスを対応する組織内ネットワーク・アドレスに書き換える。これにより、NAT は組織内ネットワーク・アドレスを持つホストを、インターネットに対してグローバル・インターネット・アドレスを持つホストであるかのように見せることができる。NAT を正しく実装するには、アプリケーション・プロトコルなどに含まれるすべてのネットワーク・アドレスをも同時に書き換える必要があることから、対象となるプロトコルすべてのフォーマットを知っている必要がある。そのため完璧な NAT を実装することは不可能に近い。一方 DDT は、実装において他のプロトコルに関する知識は必要ない (図 2.4 参照)。

2.2.3 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[80] 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ことに注意して欲しい。

2.2.4 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。

2.3 BSD における DDT の実装

我々は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。

2.3.1 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 2.5 に示すように階層的な構造になっている [81]。

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。

≪ 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 2.5: 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。

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。

2.3.2 実装の構造

図 2.6 に構造を示す。我々の実装は、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ふたつの部分からなっている。

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。

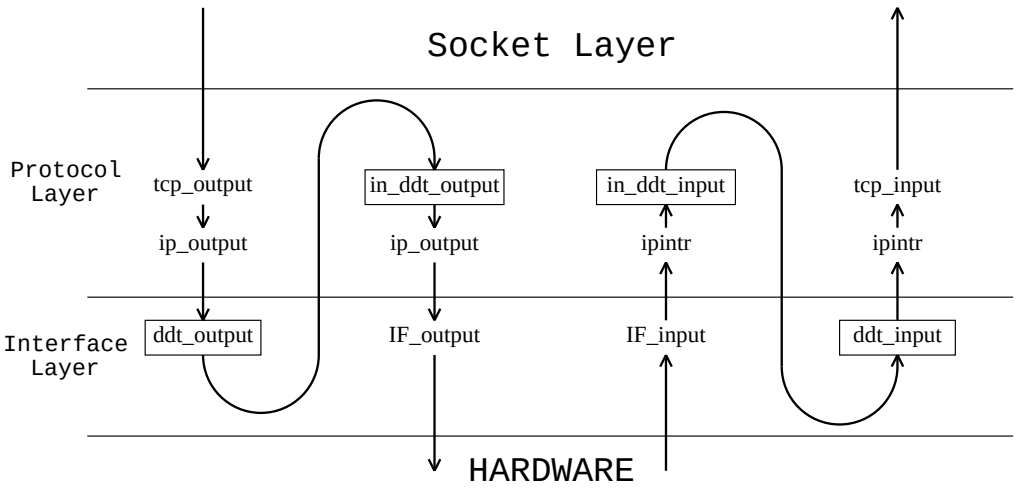


図 2.6: DDT の BSD 上の実装

- 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。

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。

2.3.3 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”[82][83] や “IP MTU Discovery options”[84] などを利用して発見する方が良いであろう。

2.3.4 An Unnumbered Link

DDT 仮想ネットワークを用いてふたつのネットワーク 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[85] 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。

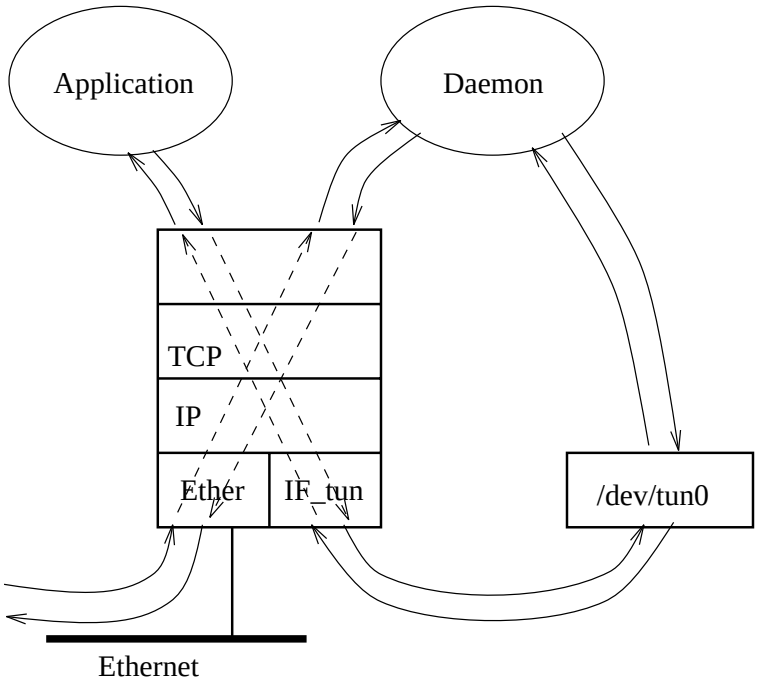


図 2.7: Pnet の仕組み

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。

2.3.5　別の実装方法 — Pnet アプローチ

DDT の実装方法として Pnet[86] のアプローチもある。図 2.7 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。我々の 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。
我々の 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。
- 2.3.4 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を必要だが、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 3 章

トンネリングの問題点の解決

トンネリングの問題点はトンネリングを包含したモデルの上で議論すべきことから、トンネリングの問題点のうち最も基本となる問題はモデルとの相性 (TP-9) であると言える。そこで本章では、まずトンネリングを包含するモデルを提案し、トンネリングの経路上での問題点及び経路制御の問題点の解決方法を提案する。さらに、モデルとは無関係な問題についての解決方法を提案する。

3.1 トンネリングを包含する階層モデル

3.1.1 自由順序階層モデル

本節ではトンネリングを包含する階層モデルとして「自由順序階層モデル」を提案する。

狭義のトンネリングが Layer Violation 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も提供している。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 3.1 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

[1] ここでは 1 章の 1.3 節 4 項で述べた方法での暗号化を仮定している。

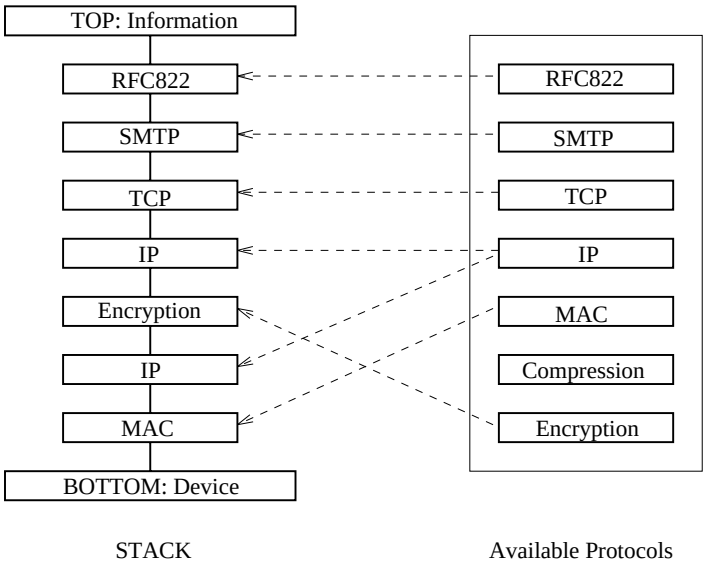
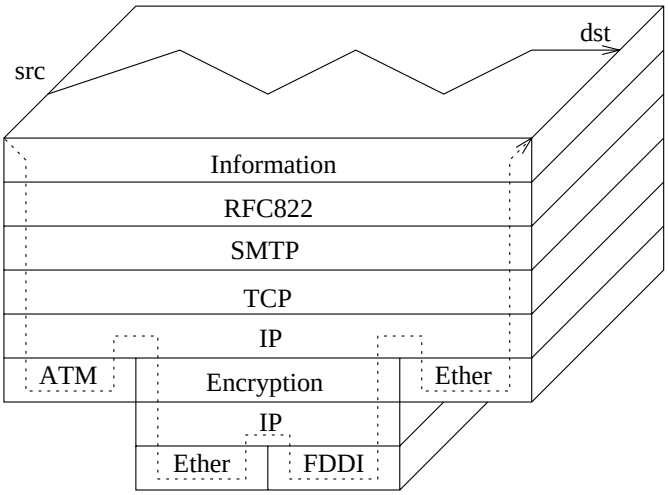


図 3.1: 自由順序階層モデルにおけるスタック

層モデルでは “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、縦軸方向の最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 3.2 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 3.3 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 3.2: 自由順序階層モデルにおける経路

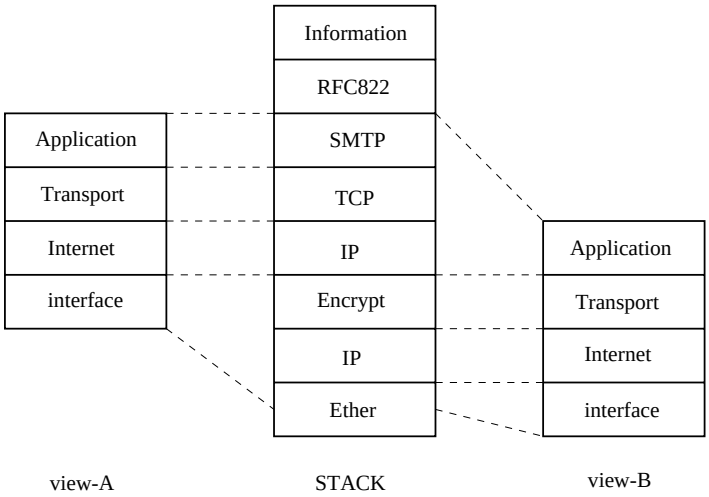


図 3.3: 自由順序階層モデルに対する “view”

3.1.2 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-9 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。トンネリングの問題点を解決するためには、自由順序階層モデルにおける一般的な安全性保証手法を構築する必要がある。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れば、トンネリングの問題点の多くは解決可能である。

3.2 モデルを用いた問題点の解決方法

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-1, TP-2, TP-3) 及び経路制御の問題点 (TP-4) 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においてこれらの問題点を解決すれば、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。

そこで本節では、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。

3.2.1 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-1, TP-2, TP-3) 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-1)
   (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-2)
   (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-3)
   (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」のふたつについて考える。

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。フロー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。フロー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・プロトコルの責任で行なわねばならない。フロー・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。
   フロー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、フローを複数用意して QoS に合わせてフロー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、制御情報用のフローとデータ転送用のフローを別々に用意して、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・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。

3.2.2 経路制御

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-4) 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

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。

[2]「クラインの壷型ループ」の用語の意味は TP-4 参照

- 経路制御表作成アルゴリズム

  1. クラインの壷型ループ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。

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。

以上の方法の検証や詳細化は今後の研究課題である。

3.3 その他の問題点の解決方法

本節では TP-5, TP-6, TP-7, TP-8 の解決方法を考える。

エラー報告方法 (TP-5) の問題及びオーバーヘッド (TP-6) 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であろう。IP の場合の提案は [87] にある。

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-7)。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 とするのが良いと考える [82][83][84]。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 にするのが良いと考える。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。

制御不能なトポロジー (TP-8) にどのように対処するかについては未解決である。

第 4 章

DDT のネットワーク社会における影響に関する考察

現在のネットワークは、細心の注意が払われた経路情報の交換、緻密な相談の上で決定された経路の人手による設定によって機能している。我々は、このような状況下で DDT が持つ意味、 DDT を使ってネットワークを運用する事の内外部に与える影響などについて考察した。

4.1 トンネリングの必要性

現在、様々な場面で、トンネリングと呼ばれる手法によって実際に問題が解決されている。本年度の報告書でも次のような応用例を挙げた。

- 経路制御に関する理由を伴う応用例
  - 離れ小島問題の解決
  - トポロジの補助線
  - 仮想専用ネットワーク
- 新技術/実験に関する理由を伴う応用例
  - 移動ホストのサポート
  - 実験ネットワークの構築
  - 新しいプロトコルの利用

これらの例は DDT を使っているとは限らないが、全て、実際に トンネリングが問題解決の手法として使われている例である。これを見てもわかるように、トンネリングは Short Term での問題解決方法として非常に優れている。長期的な解決にあまり用いられないのは、目的がトンネリング自体にあるのではなく、他のところにある事、トンネリング自体がモデリング等を含めて技術が確立していない (トンネリングに関する研究があまり行なわれていない) 事が大きな理由のようである。その他にもインプリメントが統一的でない等の理由もある。

トンネリングの有用性は明らかである。さらにトンネリング自体に関する問題が解決されれば、長期的な解決作としても有効な手段となり得る。

我々は、有用な技術であるトンネリングを安全且つ簡単に利用できる様に DDT を設計しなおす必要がある。

4.2 DDT の問題点

現在、多くの管理者は DDT を用いる事に少なからず不安を抱いている。我々はこの「不安」とは何か、と言う事に興味を持ち、ここから考察をはじめる事にした。管理者が持つ「DDT を使う場合の不安」に対する少しばかりの調査[1]を行なった結果、幾つかの不安要素が浮き彫りになった。これらは次のような 3 つのカテゴリに分類する事が出来る。

- 仮想ネットワークに伴う問題

運用者がトポロジを把握できなくなる
: DDT によって簡単にネットワークトポロジを操作する事が出来る為、トポロジが繁雑になる事が予想される。また、ユーザによって勝手にトポロジを変えられてしまい、管理者が把握できない構成になりかねない。

歯止めが無い
: 部署内に張られる電話回線を用いたプライベートリンクと異なり、金銭的な歯止めが無い為に、安易にリンクが張られてしまう可能性がある。また、DDT によってリンクを張る事によってマルチホームサイトになるが、あまりに簡単にリンクを張る事が出来る為、経路制御技術などが未熟な管理者が気軽に手を出してしまう可能性がある。

大規模になると経路制御が難しい
: 既存のルーティングプロトコルを使うとループが起きる等、経路制御が難しい。現在、管理者がネットワークトポロジを理解して、運用を行なっているが、他のサイトで同じようにトンネリングが行なわれると、このような運用形態では破綻を来す。

DDT の出入口に防火壁が必要
: 現在、企業などでは防火壁 (Firewall) を用いる事によって、サイト内での安全性を確保しているが、トンネリングによって、外とのリンクが張られるとそこにも防火壁が必要になり、ユーザに勝手に運用させるわけには行かない。

不確定ネットワークである
: リンクを張ったり、切ったりが簡単に出来る為、不確定なネットワークになりやすい。

[1] WIDE のメンバのうち、自分のサイトを管理している立場にある方を対象として口頭で答えて頂いた。

同種の問題を持つ技術: ISDN による間欠リンク等

- カプセル化に伴う問題

  防火壁でのフィルタリングが出来ない
  従来の防火壁技術の多くは始点アドレス、終点アドレス、ポート番号等を利用していた。しかし、パケットがカプセル化される事により、これらの番号を調べる事が難しくなる。更に、DDT では 2 段、3 段のカプセル化が行なわれる可能性が有り、このことが更に問題を難しくしている。

  統計情報、ログ等を取る事が出来ない
  防火壁でフィルタリングが出来ないのと同じ理由で、統計情報、ログ等を残す事が難しい。

  パケットの爆発に対する歯止めがない
  現在の IP では TTL による経路ループに対する歯止めがある。しかし、カプセル化する事により、これらの機構が動作しなくなる。また、ループが起こった場合、カプセル化を行なうと、ヘッダの分、パケットが大きくなり、そのうち MTU を越え、分裂が起きる事が予想される。

  ICMP の問題
  トンネリング技術を使うと、ICMP を正確に扱う事が出来ない。トンネル内でエラーが起きると、ICMP はトンネルの入口に帰って来る。しかし、本来ならばトンネルに入る前のパケットを送信したホストに帰るべきである。

  同種の問題を持つ技術: telnetx、ftpmail 等

- 階層モデル違反に伴う問題

  従来の安全保証機構が役に立たない
  現在の IP では経路ループの為の安全策として TTL を利用している。しかし、トンネリングでは層が上下するので、このような機構は役に立たない。

  トンネリングを扱う技術が確立していない
  トンネリング自体の研究が進んでいない為、経路制御においてトンネリングを扱う方法がない。この為、既存のルーティングプロトコルをそのまま適用する事が出来ず、新しい手法を開発する、もしくは運用で問題を解決する必要がある。

  同種の問題を持つ技術: udprelay 等

上記のように、これらの問題は、全て、DDT 以外の現存の技術の中にも存在する問題である。DDT が管理者に不安を抱かせるのは上記のような問題がまとめて存在しており、その不安要素の多さにあるようである。DDT を安心して使う為にはこれらの問題にに一つ一つ真剣に取り組み、解決していく必要がある。

4.3　問題点の解決

トンネリングの重要性/必要性を 4.1節で示した。我々は、このような需要を前述したような問題点を解決した上で満たすことを考えなければならない。問題を解決する上で上記の問題を更に別の視点から 2 つに分ける。

1. 他の技術/運用との接点における問題
    - 不確定ネットワーク
    - カプセル化
2. 独立に解くべき問題
    - 階層モデル違反
    - ループなどでの安全保証機構

1に関しては他の分野の研究者と共に問題を解いていくべきであろう。統計に関する問題等ではお互いに譲り合う事である程度解決する事が出来ると考えられる。また、トポロジ的な問題もネットワーク管理者グループに意見を聞く事により、DDT が機能としてどのあたりまで提供できれば、管理に支障をきたさないような機構を用意できるかが明らかになるであろう。

一方、2の方は独立に解く事が出来ると思われる問題である。階層モデル違反に伴う問題に関しては、現在のネットワークモデルを見直し、きちんとモデリングの中にトンネリング機構を組み込む事により、よりエレガントな解が見つかるものと思われる。また、安全保証機構に関する問題も、DDT の枠組の中である程度解決できる問題であり、停止条件を定義し、その為に必要な機構を現在の DDT に組み込む必要がある。

以上の事より、DDT を現在のネットワーク環境で安全、且つ、簡単に使えるようにする為に必要な事は

- DDT の管理に関して、ユーザフレンドリなインターフェイスを作成し、管理を簡略化する。
- モデリングをきちんと行ない不安要素を取り除く。
- 安全保証機構を組み込む。
- 運用ガイドラインを用意する。

等である。また、

- セキュリティ
- 本格的な移動体のサポート

等の付加価値を付ける事により、より多くのユーザに利用してもらい、実際の危険度の少なさをアピールする必要もありそうである。

4.4 結論

これまで、見て来たように DDT の全てが新しい技術ではない。DDT に存在している問題点の多くはこれまでに問題解決の為に使われてきた手法の中にも潜在的に残っている。我々は、DDT によって顕在化した、これらの残されている問題点について改めて研究し、解を得る必要がある。問題点の多くが解決された時に DDT は安全、且つ、エレガントに問題を解決する手法として用いられるようになるであろう。

第 5 章

DDT WG '94 度計画計画

1994 年度も DDT WG は研究の柱をふたつ考えている。それぞれ、次の活動を計画している。

- トンネリングの利用方法を考える。
  - セキュリティへの応用を考える。
  - CIDR を知らないルータを skip するのに使えるよう考える。
  - データ圧縮、ヘッダ圧縮を考える。
  - DDT インタフェースの β リリース。
- トンネリングの問題点の解決。
  - 3 章で提案した解決方法を再チェックする。
  - 3 章で提案した解決方法を IP に適用し、実装する。
  - 3 章で提案した解決方法を検証する。
  - 残った問題点 (TP-8) の解決方法を探る。