# 第 3 部

# パケット交換網

# 第 1 章

# はじめに

近年、スタンドアロンで使用されていた計算機がイーサネットの発達によりネットワークに接続されるようになった。その数もここ数年の間に数台から数百台へと大規模になりつつある。これは、計算機の低価格化のみならず、UNIX などのようなネットワークオペレーティングシステムの発展が大きく関与している。情報に対するある種の付加価値が重要となってきた現在、ネットワークは高度情報化社会の中でなくてはならないものになっている。それにともない、計算機研究者の間で主に使われていたコンピュータネットワークも生活の一部として、一般社会に浸透し始めている。

最近では、複数の計算機に役割分担をさせ、それらを水平結合することによりあたかも一台の計算機であるかのような環境をユーザに対して提供する「分散 OS」の研究がさかんに行なわれている。ローカルエリアネットワークでは、通信速度が早く、時間差が少ないため分散環境を構築するのは容易である。これは、お互いの計算機同士が個々の役割を認識しているため、必要な情報が得られやすいためであり、必要な情報をどの計算機から収集してよいのか分からない時は、ブロードキャストを用いることにより要求を満足する計算機がどれであるのかを知ることができる。しかしながら、例えばイーサネットでの接続は、電気的な制約から最大長 2.5 Km までのネットワークしか構築できない。

それに対し最近では、ローカルエリアネットワーク同士を結合して新たに大きなネットワークを構築しようという動き (Internetworking) が盛んである [15]。ネットワークの広域化による最大の利点は、計算機環境の地理的、物理的拡大により、遠隔地にあるリソースを共有したり、数多くの人々の間でコミュニケーションを可能にしたりすることである。特に、電子メールは、不特定多数の相手と比較的短時間に文章のやりとりができるため、計算機研究者の間で活発な意見交換が可能である。アメリカなどでは、以前から電子メールを用いた議論が数多くなされ、その結果として、計算機科学はまれに見る急速な発展をとげている。その意味で、電子メールの果たした役割は非常に大きいと言える。

日本におけるネットワーク同士の接続は、電話回線による UUCP 接続が主流である。このプロトコルは、ファイル転送を基本としたプロトコルであり、一定時間ごとにゲートウェイが電話回線を利用して電子メールやニュースを転送する方式である。JUNET[28] では、電子メールサービスを 1984 年から行なっているが、メールを目的地に直接配送するのではなく、隣接ノードへ送ることにより、バケツリレー方式でメールやニュースを伝搬させていく。しかしながら、電子メールやニュースシステムなどの情報交換だけではなく、遠隔サイトにあるデータベースなどの計算機資源を対話形式で利用したいという要求がある。この遠隔データアクセスは、次世代 OS である分散 OS を構築する上で、なくてはならない基本要素でもある。広域分散環境下においては、遠隔地にある資源をアクセスするアプリケーションが、ローカルエリアネットワークと同等のスピードで透過的に実行されなければならない。そのためには、かなり高速な広域ネットワークの実現が必要とされる。

アメリカでは、全米で 7ヶ所あるスーパーコンピュータセンタを T1(1.544 Mbps) という高速専用回線で結合している (NSFnet)。これをバックボーンとして、各地域ごとのネットワークがこれに接続されている。地域ごとのネットワークは、専用線もしくは、公衆パケット交換網によりネットワークレイヤ (Internet Protocol) 的に接続されているのが一般的である。このようにして、全米中のネットワークを結合して、Internet が形成されている。

日本とアメリカのコンピュータネットワークを見た時、享受できるサービスの差異は大きい。アメリカでは、計算機同士がネットワークレイヤ的に接続されているので、IP データグラムが直接相手の計算機に届く。これにより、telnet や ftp などのようなリモートコマンドがローカルエリアネットワークと同等に利用できる。それに対し日本の場合、直接目的とする計算機とは接続されていないため、利用できるサービスがローカルエリアネットワークの時と異なり、かなり制約されてしまう。

そこで、日本でもアメリカの Internet のような広域ネットワークの構築の必要性が高まりつつある。広域ネットワークの構築方法としては、専用回線の利用、回線交換網の利用、パケット交換網の利用などが考えられるが、信頼性やコストパフォーマンスの点からパケット交換網を利用した通信が、最も適していると思われる。しかしながら、パケット交換は、一般的に通信速度が遅く、また、コネクションにもとづいた通信を行なっているために、回線設定までに時間的な遅延が生じるという欠点がある。そこで、いかにパケット交換において、通信速度を向上させるかが広域ネットワークを構築する上での重要な課題であると言える。

ここで、パケット交換において通信速度が遅い理由を考えてみると、

- 交換機の転送速度の限界
- パケットサイズの制限
- ウィンドウサイズの制限

などが挙げられる。このうち、交換機の転送速度に関しては、NTT の DDX-P では 9.6Kbps と 48Kbps の 2 種類のサービスが提供されているに過ぎない。しかし、ハードウェアの発展に伴いこれらは、徐々に改善されていくものと思われる。

次のパケットサイズとウィンドウサイズの制限に関しては、各仮想回線ごとに定められた値であり、フローコントロールおよび網内での輻輳制御のために設定されるものである。ところが、網および計算機の双方に計算機資源の余裕がある場合でさえも、この制約により通信速度がある一定限度に押えられてしまう。このパケットサイズとウィンドウサイズは CCITT の X.25 プロトコルで定められた値なので、この値を無視することはできない。

そこで、本研究では、同一通信相手に対して、仮想回線を複数本設立するというマルチリンク手法を導入し、パケットサイズやウィンドウサイズを全体的に増加させる。これにより、従来のプロトコルを破壊することなく、パケット交換の通信速度を向上させることが可能となる。

第 2 章では、パケット交換プロトコルについて説明し、特に X.25 プロトコルの問題点について言及する。また、広域分散ネットワークの構築をめざす **WIDE** プロジェクトの概要と本研究の位置付けに関して説明する。第 3 章では、**WIDE** /X.25 の全体的な設計思想について述べる。また、仮想回線の設立の手順、およびデータ転送手順に関して議論を行なう。第 4 章では、**WIDE** /X.25 の 4 つのサブレイヤの実装方法について詳しく解説する。第 5 章で、**WIDE** /X.25 の評価を行ない、第 6 章で考察および今後の課題について述べるものとする。

# 第 2 章

# パケット交換プロトコル

この章では、パケット交換プロトコルの特徴およびその問題点 [87][86] について述べる。

## 2.1 広域ネットワークを構築するための 3 つの方法

現在、広域ネットワークを構築するための基礎技術として確立されている伝送形態には以下のようなものがある。

1. 回線交換網の利用
2. 専用線の利用
3. 公衆パケット交換網の利用

第一の回線交換方式は、回線契約者が多いため比較的簡単な方法でネットワークを確立することができる。これは、従来の JUNET で広く用いられている方法である。回線交換網の利用は、ISDN の普及により最近注目されてきている。ISDN により回線品質もディジタル化され、従来のアナログ系の電話回線と比較してかなり向上した。また、データ転送速度も 64Kbps と高速になった。しかしながら、従来の電話回線によるアクセスでは、相手とコネクションを確立するためにダイヤリングという操作を必要とした。この時間的制約のため、回線交換方式による通信では、即答性を必要とするアクセスは極めて困難である。この制約を排除するためには、一度確立されたコネクションは二度と切断しないという方法が考えられるが、これは経済的に実用的ではない。また、回線交換方式では、通信中は物理的に回線を占有してしまうために、二ヶ所以上の相手と同時に通信することができない。このため、分散環境のように、いついかなる時でも任意の相手と通信する必要性がある通信には、この方式は向いていない。また、相手が通話中であったり、回線に空きがない時には通信する手段がないというのもこの方式の欠点である。

第二の方法は、近接したサイト間を専用線で接続し、それを結合して広域ネットワークを構築する方法である。この方法は、かなり実用的であり、かつ、高速通信を期待できる。しかし、各サイト間で専用線を維持するために、莫大な運用資金を要するであろう。そのため、バックボーンのようにトラフィックが大きいなところでないと効率が悪い。また、専用線はあくまで私的な利用目的のために構築されるため、そのサイトにとって無関係なパケットまで中継するかどうかは、そのサイト間の取り決めによる。したがって、それを基盤として公衆な広域ネットワークを構築することは、各サイト間で合意がない限り困難であると考えるべきであろう。

そこで、第三の方法として考えられるのが、パケット交換網を利用する方法である。パケット交換とは、データをある一定の規則によりパケットという形に区切り、それを回線上に流すことによって通信する方法である。これは Point-to-Point 通信である回線交換方式とは異なり、各パケットが多重化されているために、ユーザが回線を占有することがなくなる。このため、同時に複数の相手と通信を行なうことが可能となる。また、網はデータパケットを相手に確実に配達することを保証しているために、タイムアウトによる再送などを考慮する必要がなく、信頼性がかなり期待できる。課金に関しては、転送された情報量によって計算されるため、回線が保留状態であっても課金されない。また、通信距離による料金の格差がほとんどないのも特徴のひとつである。

以上のことをまとめるとそれぞれの利用形態について以下のようになる。

この図 2.1からも明らかな様に、平均呼率が低い場合には電話網が有利であり、逆に高い場合は、専用線が有利となる。また、平均保留時間が長い場合には、回線交換方式が有利であり、短い場合にはパケット交換方式が有利である。以上のようにトラフィック量と通信時間により最適な回線が異なってくる。不特定多数の相手と通信する可能性のある広域分散環境では、あらゆる使われ方を想定しなければならない。そのため、有効範囲の広いパケット交換網を有効に利用できるように改良することは有意義なことである。

## 2.2　パケット交換の特徴

パケット交換方式は、大別して仮想回線（バーチャルサーキット）方式とデータグラム方式に分類される。仮想回線方式は、データ通信に先だって論理的に回線を設立する方式である。回線交換のように物理的に回線を占有するのではなく、交換機内のメモリ上で仮想的に回線が管理される。このためデータが混信することなく相手に届けられる。また、パケットの

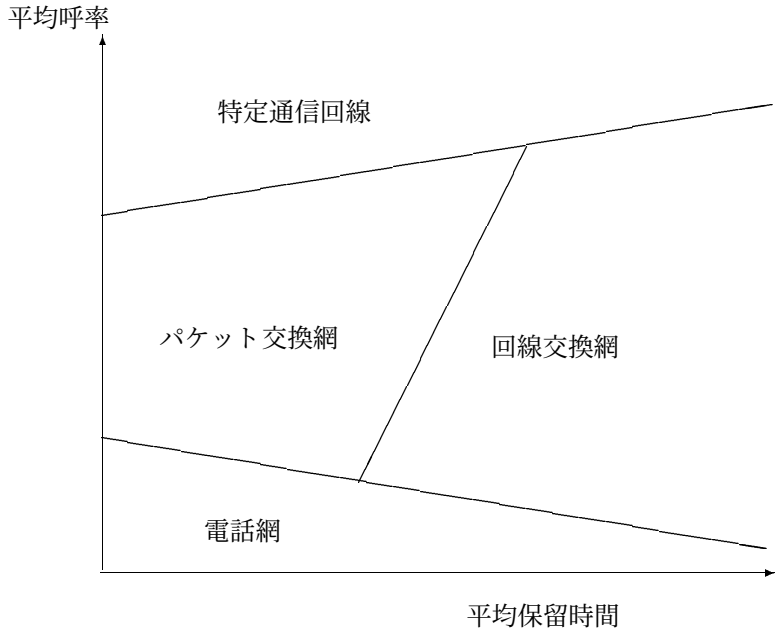


図 2.1: 伝送媒体における適応領域

順序制御は網が行なう。

一方、データグラム方式は、網がパケットをそれぞれ独立に扱い、各パケット間の連続性を考慮しない方式である。この方式では、混信防止や順序制御などをユーザ側で行なう必要がある。そのため現在の公衆パケット網では、バーチャルサーキット方式が広く採用されている。逆にデータグラム方式は、公衆パケット網としては例がない。

この章以降、パケット交換方式とは、バーチャルサーキット方式を意味するものとする。

パケット交換とは、データをパケットという一定の形式に変換し（図2.2参照）、それを交換機を経由して送信することにより通信する方式である。この時、通信に信頼性を持たせるため、パケットのヘッダ部には誤り符号や順序番号などが挿入されている。また、データ通信に先立ち、相手と仮想的に回線を設立するのもパケット交換の特徴である。仮想回線を設立する時に中継交換機が回線番号とアドレスの変換を行なう。データが流れる経路に関しては、この中継機が一切の責任を持つので、計算機は回線番号のみを意識すれば良い。

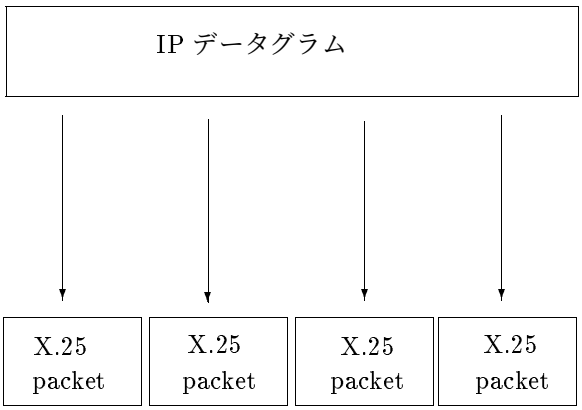


図 2.2: IP データグラムのパケット化

以下にパケット交換の特徴を示す。

**回線の有効利用** 回線を占有する回線交換方式と異なり、パケット交換ではあくまで仮想的に回線を設立しているに過ぎないので、実際にはデータを転送している時のみ交換機を占有している。交換機は、データパケットのヘッダに挿入された仮想回線の番号により、パケットを目的地に正しく転送する。DTE (Data Terminal Equipment) は同時に複数本の仮想回線を設立することができるため、複数の相手と同時に通信することが可能となる（図 2.3）。また、通信時間に関係なく、データ量によって課金することができるため、回線交換方式と比較してコストが安く済む。

**パケットの蓄積** DTE から送出されたパケットは、網内の交換機に一旦蓄積される。その後、転送される回線の速度に応じてパケットが送出される。したがって、端末間の物理的な回線に速度差がある場合でも、ここで吸収することができる。しかしながら、パケットが交換機に蓄積されるため、数百 msec の網内遅延が発生する。このため、パケット交換では、高速通信は期待できない。（図 2.4参照）

**ウィンドウサイズの制限** パケット交換では、網内で輻輳 (congestion) が起こらないようにあらかじめ、End-to-End で確認応答なしに送信で

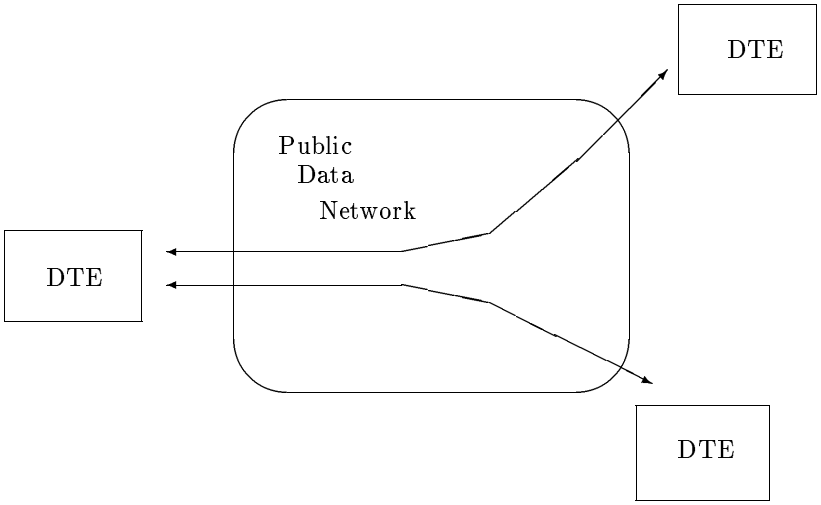


図 2.3: パケット交換網

きるパケット数を決めておく。これをウィンドウサイズと呼ぶ。このウィンドウサイズは、仮想回線確立時にお互いに取り決めが行なわれる。このとき、交換機が通信に必要なバッファを用意するためデータパケットを取りこぼすことがなくなる。このことによって、網内の信頼性が向上する。送信側では、確認応答を受け取ることによりデータを新たに一つ送信することが可能になる。このように確認応答を受け取るたびにウィンドウがスライドされていくので、このフロー制御はスライディングウィンドウと呼ばれる。

**パケットの到着順序の保証** データパケットのヘッダ中に順序番号が挿入されている。この順序番号は、ただ単に上述のウィンドウサイズを認識するためだけでなく、パケットの到着順序を正しく認識することにも利用される。これによりフラグメントされたデータを再び組み立てることが可能となる。また、正しい順序でパケットが到着しなかった場合は、エラーとみなし、正しいデータパケットの再送要求を行なう。これにより、高い伝送品質を保証している。

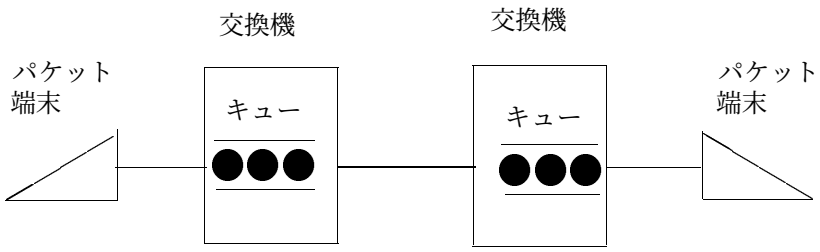


図 2.4: パケット交換の原理

## 2.3　X.25 プロトコル

X.25 プロトコルは、国際電信電話諮問委員会 (CCITT) が勧告しているパケット交換プロトコル [10] である。X.25 は、レベル 1 (電気特性の規定), レベル 2 (伝送制御手順), レベル 3 (パケット交換手順) の 3 つのレベルに分かれる。レベル 2 の HDLC (High-level Data Link Control protocol) は、ここでは説明を省く。

レベル 3 のパケット交換プロトコルでは、データを送信する前に、仮想回線の確立を行なう。この様子を図 2.5に簡単に示す [103]。

CR パケットのフォーマットを図 2.6に示す。

まず、データを送信する前に、CR (Call Request) パケットを用い、仮想回線の設立要求を行なう。この CR パケットには、今後この通信で使用される論理チャネルグループ番号および論理チャネル番号が指定されている。さらに、どの DTE と通信するかを X.121 アドレス形式で指定する。これは、通常 7 から 12 桁の数字で、電話番号に相当するものである。また、ファシリティフィールドを用いて、

- ウィンドウサイズ
- スループットクラス
- パケットサイズ

などを通信相手と取り決めることもできる。ただし、最終的に決定される値は、送信側、受信側、網の三者がすべて満足することのできる値である。

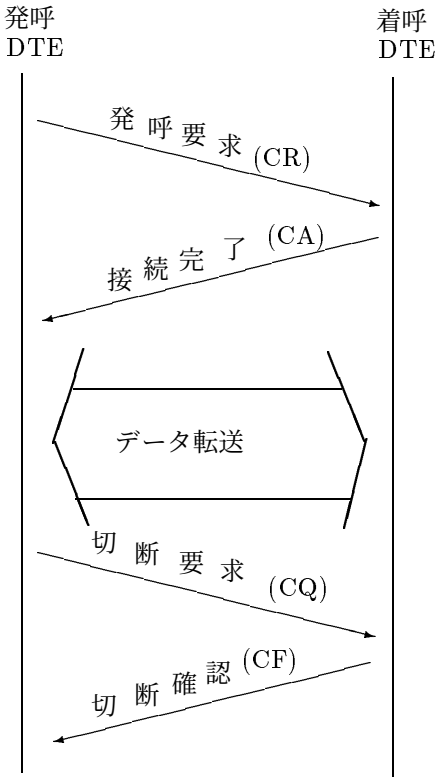


図 2.5: X.25 プロトコルにおけるパケット転送

これらの手順を踏んだ後に、初めてデータ転送が可能となる。データ転送が終了すると、切断パケットを送信し、仮想回線を閉じる。

### 2.3.1 パケット交換網の問題点

パケットをひとつ送信してから、その確認応答が返ってくるまでの時間 RTT (Round Trip Time) を計算してみる。(図 2.7)

図 2.7 においてパケット長を P, 確認応答 (ACK) 長を A, 回線速度を s, 網内の伝送遅延を T, 計算機内の応答遅延を D とすると、ラウンドトリップタイムは、パケットを送出するのに要する時間 (P/s)、伝送遅延 T、パケットを受信するのに要する時間 (P/s)、計算機内遅延 D、確認応答を送出する時間 (A/s)、伝送遅延 T、確認応答を受信するための時間 (A/s) の

| | |
|---|---|
| ゼネラルフォーマット識別子 | 論理チャネルグループ番号 |
| 論理チャネル番号 | |
| パケットタイプ識別子 | |
| 発呼アドレス長 | 着呼アドレス長 |
| 着呼アドレス<br>および<br>発呼アドレス | |
| ファシリティ長 | |
| ファシリティ | |
| コールユーザデータ | |

図 2.6: CR パケットフォーマット

総和になる。

$$RTT = (\frac{P}{s} + T + \frac{P}{s} + D + \frac{A}{s} + T + \frac{A}{s}) \tag{2.1}$$

この時間内に送出できるパケット数は、ラウンドトリップタイムをパケット送出に要する時間で割ることによって求められるので、

$$n = \frac{RTT}{P/s} \tag{2.2}$$

となる。この個数よりもウィンドウサイズが大きければフローコントロールは起きない。 故に

$$(\frac{P}{s} + T + \frac{P}{s} + D + \frac{A}{s} + T + \frac{A}{s})/\frac{P}{s} \leq W \tag{2.3}$$

(3) を T について解くと、

$$T \leq \frac{PW}{2s} - \frac{D}{2} - \frac{P+A}{s} \tag{2.4}$$

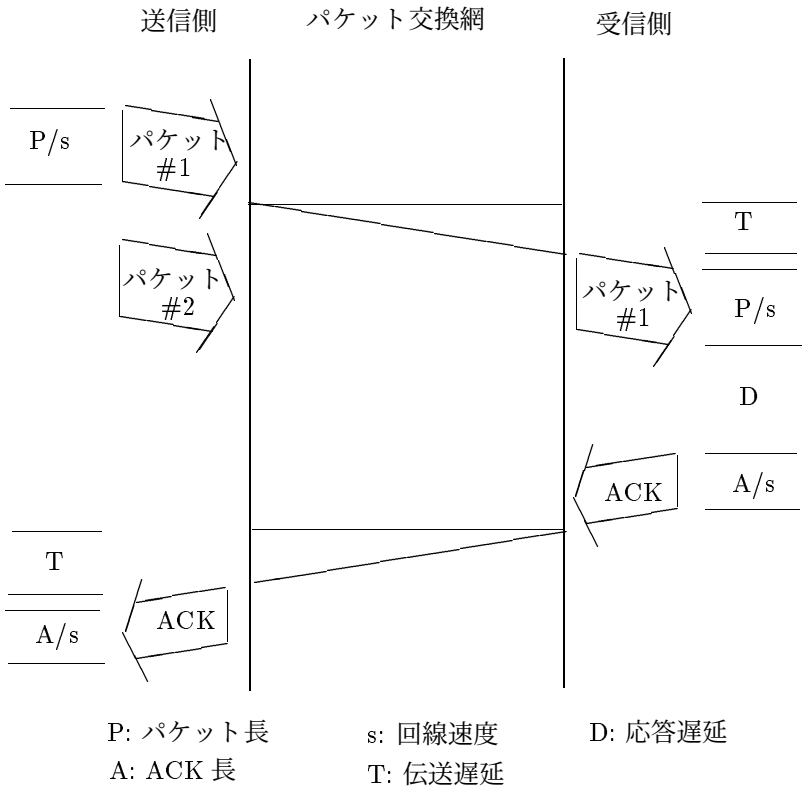


図 2.7: Round Trip Time と最小 window size

ここで、T > 0 となるための条件を計算する。簡単化のために計算機内の遅延 (D) を 0 と置くと、

$$(W-2)P > 2A \tag{2.5}$$

よって、

$$W > 2 \tag{2.6}$$

故に、パケット長や確認応答長に関係なく、最小ウィンドウサイズは 3 以上でなければならないことが分かる。すなわち、この値以下であると必ずフローコントロールが起き、データ転送が中断される。ところが、X.25 プロトコルでは、国際リンクは、ウィンドウサイズが 2 に固定されている。それにもまして、国際接続では、衛星回線を用いている場合がほとんどであるため、網内遅延 (T) がかなり大きいことが予想される。これは、人工

衛星が、赤道上 36000 Km の上空にあり、地上から発射されたパケットが人工衛星に反射されて再び地上に戻って来るまでに、270 msec の遅延を生ずるためである。先ほどの式 (3) にパケットサイズ = 259 オクテット、s = 9600 bps、 ack = 3 オクテット、T = 0.27 sec、D = 0 として値を代入すると、

$$W > 4.525 \tag{2.7}$$

となり、人工衛星を利用した通信では、ウィンドウサイズが 5 以上必要であることが分かる。

すなわち、国際接続では、このウィンドウサイズがボトルネックとなり高速な通信ができなくなっている。この問題を解決するためには、ウィンドウサイズを大きく取る方法が考えられるが、これでは X.25 プロトコルを無視してしまうことになる。実際には、仮想回線設立時にウィンドウサイズを規定よりも大きく要求しても網の制御により規定値以内に抑えられてしまう。また、通信中に故意にウィンドウサイズを超えてパケットを転送すると網が回線を切断してしまう。そこで、通信相手に対して、仮想回線を複数確立するといったマルチリンク手法 [32] を用いることにより、仮想的にウィンドウサイズを増やす方法が考えられる。X.25 プロトコルでは、同一通信相手に対する仮想回線の本数を規定していないため、このマルチリンク手法はプロトコルに違反しない。この概念図を図 2.8に示す。

しかしながら、既存のソフトウェアとの互換性を図るためには、効率やアルゴリズムの点で注意しなければならないことが多々ある。この方式では、X.25 のプロトコル体系をいかに崩さずに通信の高速化を図るかが重要である。

## 2.4　WIDE /X.25 の目標

現在、日本において、IP 接続を行なっているネットワークは、まだ数えるほどしかない。それもほとんどが、専用線を用いた Point-to-Point 接続である。この方式は高速通信には適しているが、費用の面から、このコミュニティに参加できる企業は限られてきてしまう。

わが国の公衆パケット交換網は、日本電信電話株式会社 (NTT) の DDX-P や学術情報センターの NACSIS などがあるが、これらの上では、ほとんど TCP/IP は稼働していない。 その理由としては、標準の UNIX は、X.25 プロトコルをサポートしておらず、X.25 網上で UNIX を利用するには、どうしても専用のボードを購入するか、特定の企業からソフトウェアを購入するしかない。これらは、稼働するマシンが限定されてしまったり、

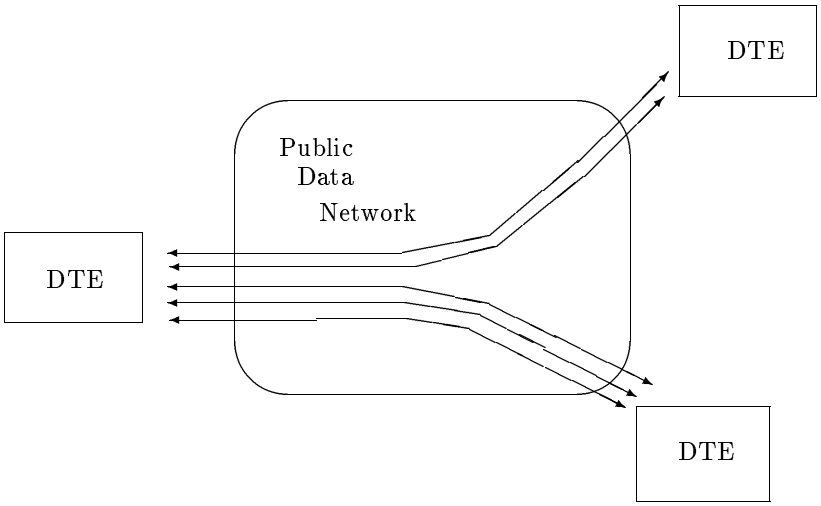


図 2.8: マルチリンク方式

ライセンスの問題でそれほど普及していない。また、ソースコードも入手できないため各種の実験などもできない。

そこで、慶応義塾大学、東京大学、東京工業大学、大阪大学などを中心に、広域分散ネットワークの構築およびその問題点の検討、新たな技術の確立を目的として **WIDE** (Widely Integrated Distributed Environment) プロジェクト [29] が 1988 年に設立された。現在は、広域分散環境における名前空間、ユーザ認証機構、経路制御問題、広域通信におけるプロトコルの研究が行なわれている。本研究で設計・実装されるソフトウェアを **WIDE** /X.25 と命名する。**WIDE** /X.25 は、研究目的だけではなく、実用的にも十分使えるように設計しなければならない。

**WIDE** /X.25 は、UNIX 4.3 BSD 上に X.25 プロトコルを構築し、Public Domain Software として無償配布することを目的としている。これは、すべてソフトウェアで書かれるだけでなく、UNIX 4.3 BSD 自体がマシンにほとんど依存しない OS であるために移植性も良い。これにより、**WIDE** プロジェクトの目標である広域ネットワークを構築することが可能となるであろう。

また、UNIX のアプリケーションは、既存のコマンドを組み合わせたものが数多くあるのが特徴であるが、これらのものが何ら変更なしに再利用

できた方が便利である。このことは、ユーザアプリケーションだけでなく、ネットワークコードを用いたアプリケーション全般について言える。これは、ネットワークプログラミングにおいて単純にレイヤリングを守るということだけでは不十分であり、より注意深い考察が必要である。そこで、**WIDE** /X.25 では、既存のものについてなるべく変更点を伴わないように設計する。今回の設計では、高速な通信の実現に重点を置くが、通常の X.25 をサポートしている専用ボードやサードパーティのソフトウェアと通信できなければ、**WIDE** プロジェクトの目標 == 異機種間接続をも考慮した広域ネットワークの実現 == を満たしているものとは言えない。

さらに、**WIDE** /X.25 は、ただ単に従来の X.25 プロトコルを高速化するだけではなく、先に挙げた経路制御問題や認証機構などの研究をする際の基盤として利用することを考えている。そのためには、拡張しやすい形で提供されなければならない。そこで、なるべくユーザがあらゆることを細かにしかも動的に指定できるような柔軟なインタフェースを持つことが望ましい。

これらのことを考慮した上で、**WIDE** /X.25 は以下のことを目標として設計する。

- 効率の良いパケット通信の実現
- 移植性の重視
- 様々な研究のための柔軟なインタフェースの提供
- 従来のアプリケーションに対する透過性
- 既存の X.25 ソフトウェアとの接続

ここで、パケット通信の高速化に関しては、前述のマルチリンク手法によって実現する。この章の後半で、効率の良いコネクションの張り方、データ転送方式などについて議論する。なお、この議論の中には通常の X.25 ソフトウェアとの互換性をも追求する。移植性および従来のアプリケーションの再利用に関する実現方法は、この後のレイヤリングの節で論ずることにする。様々な研究のために用意する柔軟なインタフェースは、ioctl() によって送受信したパケットもしくはフレームをコンソールに出力することができるとともに、各レイヤの情報を出力することができるようにする。

## 2.5　UNIX アーキテクチャの問題

ISO による OSI(Open System Interconnection) では、ネットワークアーキテクチャを全部で 7 階層 (physical, data link, network, transport,

session, presentation, application) に分割している。それぞれのレイヤの具体的な機能については、ここでは詳しく触れないが、UNIX のネットワークアーキテクチャと比較すると、図 2.9のように対応付けがなされる。

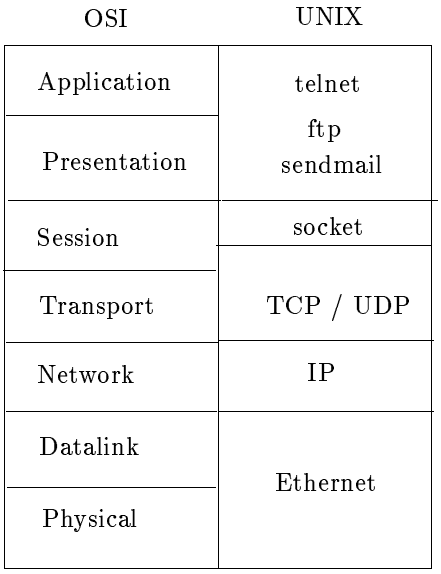


図 2.9: UNIX 4.3BSD のレイヤリング

ここで、X.25 を実装するにあたって、どのレイヤに X.25 の制御を任せるかが問題となってくる。考えられる方法としては、

1. アプリケーションとして構築する
2. ソケットレイヤの下に新しいプロトコルファミリを作る
3. TCP 内に X.25 の制御機能を持たせる
4. IP の下に新たに制御用のインタフェースを作る

がある。ここでは、上記の目標を達成するためにどの構築方法が一番優れているかを議論する [99][77]。

まず、第一の方法は、X.25 プロトコルをユーザプロセスとして実現する方法である。(図 2.10参照)

X.25 網を介した通信を行なう時にはまず、X.25 を制御しているユーザプロセスに要求を送る。この方式では、X.25 プロトコルは、UNIX のネッ

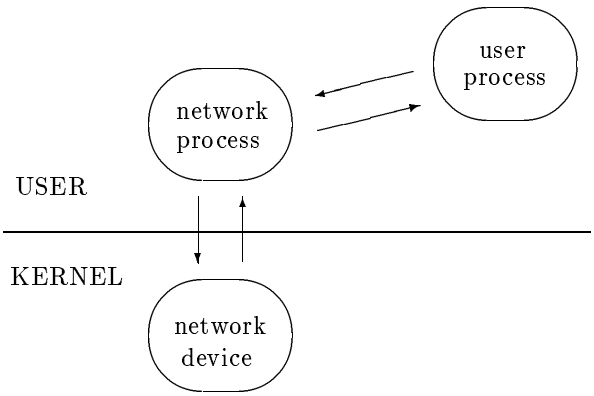


図 2.10: プロセスによる X.25 プロトコルの実現

トワークアーキテクチャと全く独立である。よって、比較的実現が容易で、UNIX 以外のオペレーティングシステム上にも移植しやすい。しかしながら、ソケットと同じインタフェースを用意するのが困難なため UNIX の従来のネットワークアプリケーションをすべて書き換えなければならなくなる。しかも、UNIX のユーザプロセスは、コンテクストスィッチによるオーバヘッドがあるため、リアルタイム性に欠ける。そのため、到着したパケットを頻繁に取りこぼすことが考えられ、通信効率が悪くなることが予想される。

第二の方法は、TCP/IP と同等のレイヤとして定義される（図 2.11）。

この方式は、ソケットと同じインタフェースを用意しなければならないが、ソケットのコネクションと X.25 のコネクションの制御方式が似ていることを利用して、網内に転送されるパケット数を減らすことができる。このことにより、転送効率を改善することができる。

すなわち、ソケットおよび X.25 プロトコルでは、通信に先立ち、呼を設定するためのパケットを送信する。これに対する確認応答が返り、仮想回線が確立されると、データ転送モードになる。データの転送が終ると仮想回線を解放する。実際には、ソケットと X.25 では、レイヤが異なるので、ソケットレイヤの仮想回線設定パケットやその確認応答などは X.25 レイヤでは、本来、データとして扱わなければならない。ところが、これら意味的に重複しているパケットをマッピング（図 2.12）することにより、送出されるパケットの個数を減らすことができる。

これは、第三の方式でも全く同じことが言えるが、コネクションを張

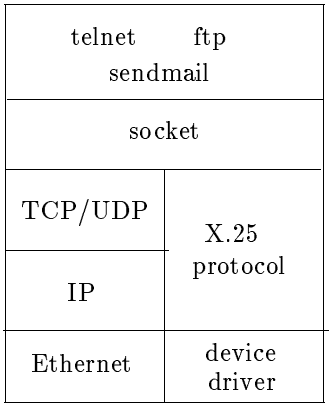


図 2.11: トランスポートレイヤでの実装

らないデータグラムの時は、マッピングが不可能なので、また別の機能を新たに作らなければならない。

第一から第三の方法で問題となるのが、ネットワーク間接続されている場合（図 2.13）である。

この場合、イーサネットのインタフェースと X.25 のインタフェースが異なるために、イーサネットから到着したデータグラムを X.25 網に中継することができない。そのためゲートウェイでは、アプリケーションレイヤで新たに網間接続のためのインタフェースを実現しなければならない。

そこで、X.25 網に対してもインタネットワークレイヤを提供する第四の方法が考えられる。

これは、X.25 を Ethernet と同等に考え、データリンクプロトコルとして扱う方法（図 2.14）である。本来、X.25 プロトコルは、OSI ではネットワークレイヤとして定義されているが、UNIX のネットワークアーキテクチャ上、データリンクレイヤに属することになる。この方式では、ソケットや TCP などのコネクションを基本とした通信における制御パケットもデータとして扱うために、パケット量が多い。また、すぐ上のレイヤは、データグラムを基本とした IP レイヤなので、コネクション管理を行なうレイヤを作らなければならない。しかしながら、従来のネットワークアプリケーションの変更をなんら必要とせず、また、ゲートウェイとしても利用できるという利点があるため、本研究では、この方式を採用する。

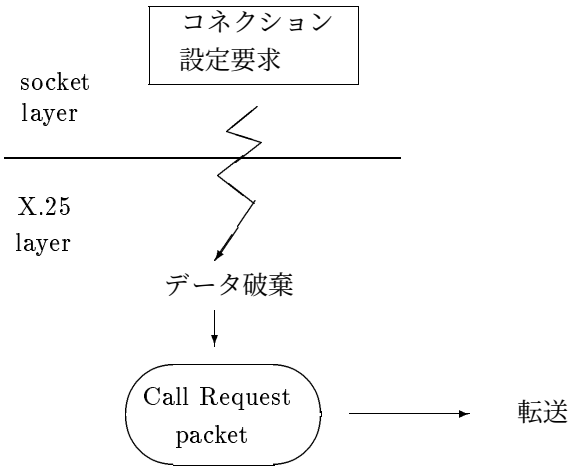


図 2.12: パケットのマッピング

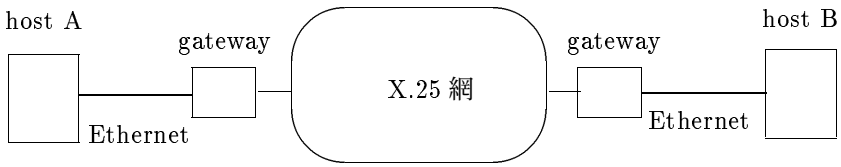


図 2.13: 経路中に X.25 網を含む場合

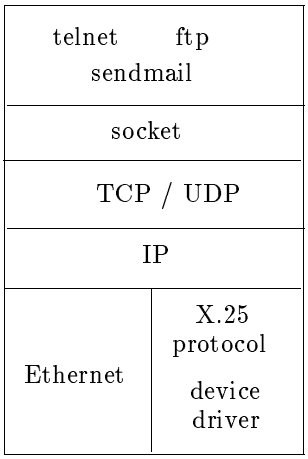


図 2.14: データリンクレイヤとしての実装

# 第 3 章

# WIDE /X.25 の設計

ここでは、**WIDE** /X.25 の設計について述べる。

## 3.1 WIDE /X.25 の 4 つのサブレイヤ

**WIDE** /X.25 は、全体で 4 つのサブレイヤ（HDLC, X.25, マルチリンク, インタフェースレイヤ）に分割することができる。図 3.1に **WIDE** /X.25 の全体像を示す。

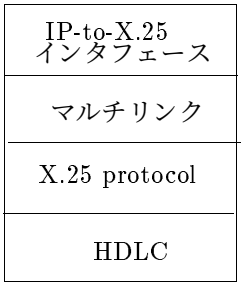


図 3.1: **WIDE** /X.25 の全体像

### 3.1.1 HDLC レイヤ

最下位のレイヤは、HDLC モジュールと呼ばれ、伝送制御手順を規定している部分である。 HDLC では基本機能として、DTE - DCE 間の回線の接続、切断、データフレームの送信を行なう。回線の設定など HDLC のコントロールに関しては、2 ウェイハンドシェークを行ない、回線の状態遷移に矛盾の生じないようにしている。その他、通信の信頼性を高める

ためフレーム毎にシーケンス番号を振り、この番号の入った確認応答をやり取りすることで、順序誤りやフレームの消失などのエラーから回復できるようになっている。また、DTE - DCE 間で輻輳を抑えるため、スライディングウィンドウというアルゴリズムを用いてフロー制御をしている。

### 3.1.2 X.25 レイヤ

その上のレイヤは、X.25 モジュールで、通信するマシン間のコネクションを End-to-End で管理する。通常、このレイヤでは、通信したい相手に対して 1 対 1 で仮想回線を張ることになる。このため、仮想回線が設立されていない相手に対しては、通信に先だって呼の設定が必要となる。この時、相手に対して、ウィンドウサイズ、スループット、その他のファシリティなどを取り決めることができる。また、HDLC と同様にスライディングウィンドウを用いたフロー制御を各回線ごとに行なっている。ここで、HDLC と異なるのは、X.25 レイヤがパケットの再送を行なわないということである。なぜなら、DTE - DCE 間は、HDLC がフレーム再送機能によって通信の信頼性を保証しており、DCE - DCE 間は、網自身が信頼性を保証している。結局、End-to-End でパケットが相手に到達することが保証されることになるためである。

### 3.1.3 Multilink レイヤ

その上位レイヤにマルチリンクモジュールを設ける。このモジュールは、一つの通信相手に対するコネクションを複数本管理する役割を持つ。すなわち、上位レイヤに複数の X.25 の論理回線を 1 本の回線であるかのように見せる。従来の X.25 では、各通信相手に対して仮想回線は 1 本であったが、これを複数本確立する。そして、このレイヤで、あるマシンに向かう IP データグラムを効率良く振り分けることによって、X.25 の高速化を実現する。また、回線の使用状況を見て、動的に論理回線の本数を増減する機能を持つ。これにより、バンド幅を広げ、スループットを向上させることが可能となる。ここで、注意しなければならないのは、IP レイヤから見るとただひとつの仮想回線が設立されているように見え、マルチリンクを意識させないことである。つまり上位層からは、回線が太くなったようにしか見えない。

### 3.1.4 インタフェースレイヤ

最上位のモジュールは、IP レイヤとのインタフェースをつかさどる部分である。このレイヤでは、IP アドレスから X.121 アドレスへの変換 [33]、その逆変換を受け持つ。そのための情報をテーブルとして持っており、ユー

ザレベルから追加、削除が可能である。また、IP リンクを統括的に管理するため、論理回線の接続、切断などの制御を指示する。

## 3.2　インタフェースの取り扱い

通常、UNIX では、ネットワークに接続されているデバイスに対してインタフェースが割り当てられている。具体的には、ifnet という構造体が割り当てられ、名前、ユニット番号、MTU などがインタフェースごとに区別される。この観点からすると X.25 網を一つのネットワークと考え、ただ一つのインタフェースを割り当てるのが自然であるように思われる。この場合、X.25 網に対してもネットワークアドレスをただ一つ決めなければならない。

しかしながら、このようにするとルーティングに関して不都合が生じる。なぜならパケット交換方式は、コネクションに基づいた通信を行なうため、ブロードキャストという概念が存在しない。現在、UNIX 上にあるルーティング情報を交換するアプリケーションとしては、routed や gated があるが、これらはブロードキャストもしくは Point-to-Point 通信でなければならない。したがって、X.25 網にインタフェースをひとつだけ割り当てる方式では、ルーティング情報の交換ができない。この問題に対処するためには、通信するすべての相手に対するエントリを計算機の起動時に静的に登録するという方法が考えられる。しかしながら、広域ネットワークでは計算機の故障やネットワークトポロジの変更などが頻繁に起こるので、これらの管理は動的に行なわれるのが望ましい。

そこで、通信する相手ごとにインタフェースを割り当てる方式が考えられる。この方式では、通信する相手とのリンクが Point-to-Point になるため、ルーティング情報の交換も従来通り行なえ、また、通信相手ごとの情報や回線そのものの管理も簡単になる。ところが、このようなネットワークアーキテクチャの構築を行なっていくと、X.25 網のゲートウェイではインタフェースが爆発的に増大してしまうことになる。ここで問題になるのが、到着したデータグラムがゲートウェイ自身宛なのか、それとも別のネットワークに転送しなければならないものなのかを判断する時に、膨大な時間を要することである。すなわち、データグラムが到着するたびに、データグラムの目的地のネットワークアドレスに一致するインタフェースを探そうとするためである。

この問題に関しては、インタフェースのリストの順番を意図的に操作することが必要となるが、この順番は、ゲートウェイ自身にどれくらいの割合でデータグラムが到着するかによっても左右されるものである。した

がって、何か新しいアルゴリズムが必要であるが、本研究ではネットワークレイヤまでの変更は考えないことにし、相手ごとにインタフェースを割り当てる方式を採用する。

## 3.3 コネクション管理

X.25 プロトコルでは、通信を開始するに当たって、仮想回線を設立しなければならない。このときに、DCE に対して使用する論理回線番号を指定しなければならないが、通常はこの本数に限度がある。NTT の DDX の場合、9600bps で 23 本、48Kbps で 256 本である。したがって、これらの本数分だけ論理回線を張ってしまうと、他の相手と通信を行なうことができなくなってしまう。そこで、コネクションの管理が重要になる。

### 3.3.1 コネクションの確立

データを転送する一番簡単な方法は、データを送信するたびに仮想回線を設立する方法である。

```
send_datagram(X.121_addr, data)
{
        open Circuit for X.121 address;
        send data
        close Circuit
}
```

しかしながら、この解決方法は二つの面で良くない。第一は、データを送信する際に仮想回線を張るための遅延を生じることである。第二は、経済的に不利である点である。DDX では、コネクションの開設には基本的には課金されないが、その他の公衆パケット交換網（Telenet など）では課金されてしまうところもある。したがって、一度オープンした仮想回線はデータグラムの送出中はオープンしたままにしておく必要がある。そこで一般的には、以下のような方法が採られる。

```
if_output(IP_address, datagram)
{
        x25addr = convert_IP_to_X.121(IP_address);
        circuit = select_circuit(x25addr);
        x25_output(circuit,datagram);
}
```

関数 convert_IP_to_X.121() は非常に簡単で、32 bit の Internet アドレスから X.121 アドレスにマッピングするだけである。これはカーネル内のテーブル引きによってなされる。ここで、回線選択には、仮想回線の確立を含んでいる。すなわち、仮想回線が既に設定されていれば、適切な回線を選択する。また、コネクションが存在しない場合は、新たに仮想回線を設立する。

ここで、既に存在している仮想回線の数が限度以内であれば問題は生じないが、既に限度一杯に仮想回線が設立されている場合には、どれか一つ回線を選択し、それを切断してから、新しいコネクションを確立する必要がある。すなわち、次のようなアルゴリズムが必要である。

```
select_circuit(x25addr)
{
        if (Circuit is open to x25addr)
                return Circuit;
        else {
                if (all circuits are in use)
                        select and close existing circuit;
                open a new circuit to x25addr;
                return Circuit;
        }
}
```

ここで、コネクションが設立されるまでのデータグラムの取り扱い方としては、データを破棄してしまうやり方と、データを保存しておき、コネクションの確立後、転送するやり方がある。X.25 の上位レイヤが信頼性のない IP レイヤであることを考慮すると、前者でも全く問題ない。この回復は TCP もしくはそれ以上のレイヤが行なうからである。しかしながら、上位レイヤによるデータの再送までに遅延が生じる可能性があるので、本研究における実装については、データを保存しておく方式を採る。

### 3.3.2　コネクションの切断

前述のように既に論理回線の限度一杯にコネクションが設立されている場合には、どれかの回線を選択して、切断しなければならない。DDX や NACSIS では、データパケットを転送しない限り課金されることはないが、X.25 網によっては、保留時間に対しても課金されることがある。そのため、保留時間（アイドルタイム）がある一定値を越えたら積極的に切断する必要がある。

ところが、X.25 のレイヤとその上位レイヤとは基本的に独立しているため、どの契機でコネクションを切断すべきか判断できない。これは、ある一定時間アイドルしていたら、コネクションを切断するという単純な方法では決してうまく行かない。

たとえば、ある DTE に割り当てられている論理回線の本数を N, 通信相手の数が N+1 ある場合を考える。このとき、1 番から順に N+1 番の相手に対して定期的にパケットを送出するような場合、LRU (Least Recently Used) アルゴリズムによって、切断する回線を選択するものとする。このとき、任意の i ($1 \leq i \leq N+1$) に対して、i 番目の相手に対して通信を試みようとすると、LRU アルゴリズムによって最も長時間使用されていない回線、すなわち i+1 番目の回線を切断することになる。次に i+1 番目の相手に対してデータ送信をしなければならないので、今度は i+2 番目の回線を切断する。以後、この繰り返しを行なうと、FIFO, LRU どちらのアルゴリズムを用いても 1 パケット送出するために毎回コネクションの確立／切断をしなければならなくなる。実際問題として、この解決策としては、N 個以上の相手とは通信しないように限定してしまうか、最悪の場合、 1 パケット毎にコネクションの確立／切断を行なうかのどちらかであろう。

今回の実装では、保留時間を見て、ある一定時間以上経ていればコネクションを切断する方式を採用する。また、どのコネクションも一定時間内であれば、新しい相手とは通信をしないというポリシーで実装する。これは、既に設立されている回線に優先権を与えるものである。拒絶された通信は、上位レイヤによる回復が期待される。

ここで、アイドル時間のパラメータ IDLETIME は、high cost と low delay のトレードオフである。なぜなら、IDLETIME を 0 に近付けると、コネクションを設立する回数が増大し、不経済であるばかりでなく、データを転送するまでの遅延も増加する。また逆に、IDLETIME を大きく取り過ぎると、データ転送までの待ち時間は減少するが、限度以上の要求があった場合には、新たな回線を設立しにくくなる。これは、ネットワークの動的な負荷によるため予測が困難である。したがって、最適値とそれにともなう処理が課題である。**WIDE** /X.25 では、デフォルトで 1 分にしている。

## 3.4 コネクションの設立方法

X.25 プロトコルの欠点としては、ウィンドウサイズに制限があることである。これが、網内の遅延と比較して非常に小さいとフローコントロールが頻繁に起こってしまい、伝送速度が低速になってしまう理由となる。

これを避けるために、**WIDE** /X.25 では、仮想的にウィンドウサイズを大きくするために、一ヶ所の相手に対して複数の仮想回線を設立する。

コネクションを設立する最も単純な方法は、i) 送信側が希望する本数分の CR (Call Request) パケットを送信し、ii) さらにこれに対して、受信側が CA (Call Accepted) パケットで応える、というものである。X.25 では、コネクションを設定した側に課金されるので、この方式は課金論理との整合性もよい（図 3.2）。

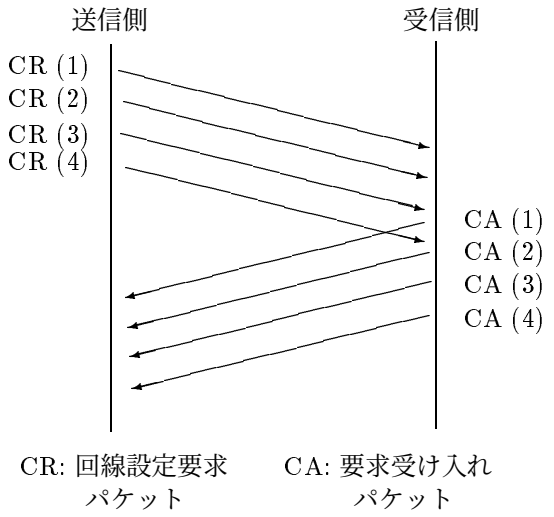


図 3.2: コネクションの設定方法 1

この方式は、2 フェーズでなされているため、シングルリンクの時とほとんど時間的な差はない。したがって、一見効率が良いように思える。しかし、送信側が希望した本数分のバッファを受信側が用意できない場合、もしくは、通常の実装のようにマルチリンクを認識しないものに対しては、非常に効率が悪い。例えば、図 3.3では受信側がマルチリンクをサポートしていない場合、複数のコネクション設定要求に対して、最初の一つしか受け入れることができない。それ以外のものについては、CQ (Clear Request) パケットによって着呼拒否をしなければならない。さらに、送信側では、CQ パケットに対する Ack として、CF (Clear Confirmation) パケットを送信しなければならない。

この図 3.3では、合計 11 個のパケットが網内に転送されたが、結果としては 1 本のリンクしか確立されていない。さらに、通信を行なうまでに

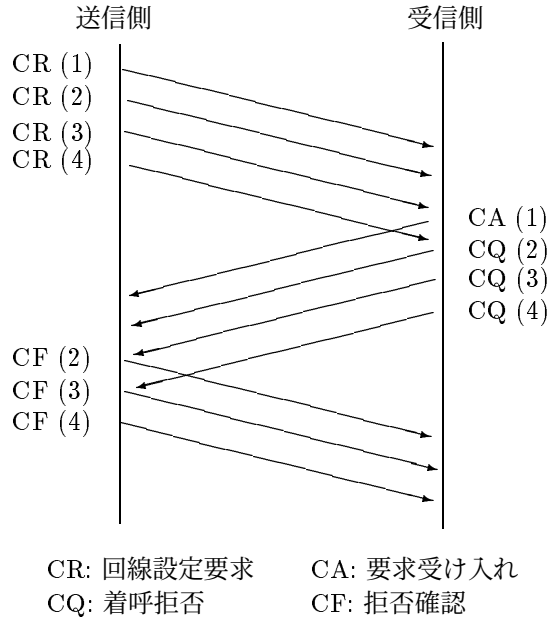


図 3.3: 通信相手がマルチリンクをサポートしていない場合

3 フェーズ必要である。無駄なパケットを網内に転送することは、むやみに網を混雑させるだけである。よって何か別のアプローチが必要となってくる。

今回の目標は、X.25 プロトコルの体系を崩さず、通常の実装とも通信ができることにあるので、以下のような方法を採る（図 3.4）。これは、まず、送信側が全体で何本のマルチリンクを張りたいかという要求を CR パケットの Call User Data フィールドに挿入する。受信側では、CR パケットに対する受け入れ応答を返した後、要求された値を満たせるかどうかを判断し、満足できる数だけの CR パケットを受信側から送信するというものである。この方式の利点は、複数の CR パケットを受け取った送信側では、着呼を拒絶することは絶対にありえないという点である。なぜならば、最初の CR パケットを送出した段階で送信側は、将来のために必要なリソースを確保しておくからである。また、受信側がマルチリンクを認識しない場合は、CR パケットに付加されている Call User Data を無視するので、それ以上受信側からコネクションを設立することはない。結果として 1 本のリンクを確立することができる。これもやはり無駄なパケットは

流れない。

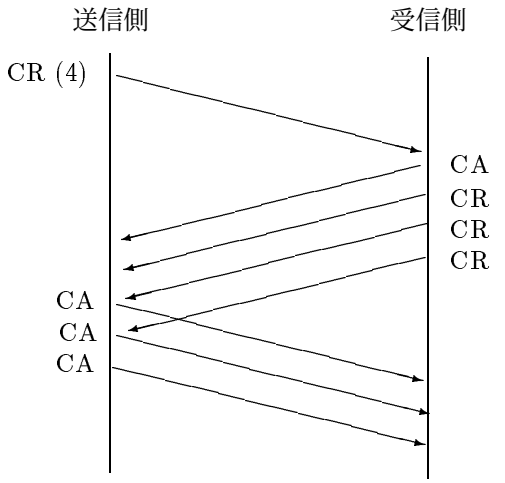


図 3.4: コネクションの設定２

この方式では、実際には送信側よりも受信側の方が多くの CR パケットを送出することになり、従来の課金体系をそのまま使用すると不公平が生じる。この問題は、着信課金サービスを利用することにより簡単に解決される。すなわち、受信側から送出される CR パケットはすべて着信課金要求を行なうようにする。また、着信課金サービスが提供されていない場合には、仮想回線を 2 本張ることで課金に対する平等性が保たれる。また、学術情報網 (NACSIS) では課金されないことから今回のような手順を採用した。

## 3.5　データ転送方式

IP データグラムを X.25 網を介して送信する際には、一度 X.25 パケットに分解しなければならない。これを受信側で正しく組み立て、再び IP データグラムにするためには、X.25 プロトコルの M bit (more bit) を利用する。(図 3.5参照)

さて、この IP データグラムをマルチリンク上に配送する方法として

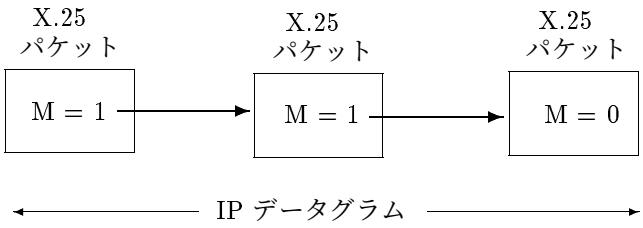


図 3.5: More bit によるパケットの接続

は、CSNET が INcard[1] のインタフェースとして採用している方法がある。これは、マルチリンクの各回線ごとに IP データグラムを割り当てていく方式である [13][12]。すなわち、最初のデータグラムは、1 番目の仮想回線に流し、次のデータグラムは、2 番目の回線に割り当てていく方式である（図 3.6）。この方式では、各仮想回線をそれぞれ全く独立なのものとして扱っている。このとき、X.25 の M bit を使うことにより、データの連続性を認識することができ、IP データグラムを組み立てることが可能となる。これは、通常の M bit の使用方法となんら変わりはない。また、IP レイヤでは、データグラムの到着順序などを一切保証していないために、送信側と受信側で IP データグラムの到着順序が狂っても問題はない。したがって受信側では、どの仮想回線からデータパケットを取り出そうと自由である。ちなみに、データグラムの到着順序であるが、この方式では、各仮想回線に設けられているウィンドウサイズの制限により、小さいデータグラムの方が早く到着しやすいのは容易に推測できる。また、データグラムを送出する回線を選択するアルゴリズムとしては、一番キューの短いものを選んでも良いし、ラウンドロビンで選択しても良い。基本的には、回線の使用率を平等にした方が効率が良いので、キューの短い回線を選択すべきであろう。

ところで、2 章のパケット交換網の問題点のところで述べたように、確認応答を返答するまでの計算機内の遅延 D は小さい方がフローコントロールが起きにくい。つまり、パケットが到着したときになるべく計算機が混んでいない方が望ましいわけである。

今、送信されるデータグラムのサイズがすべて同一であると仮定する。このとき、受信側では最初のパケットを受け取ってからはある一定の間、

[1]INcard は、Interactive Systems Corporation 社製の X.25 用インタフェースボードで、VAX UNIBUS を X.25 ネットワークに接続することが可能である。

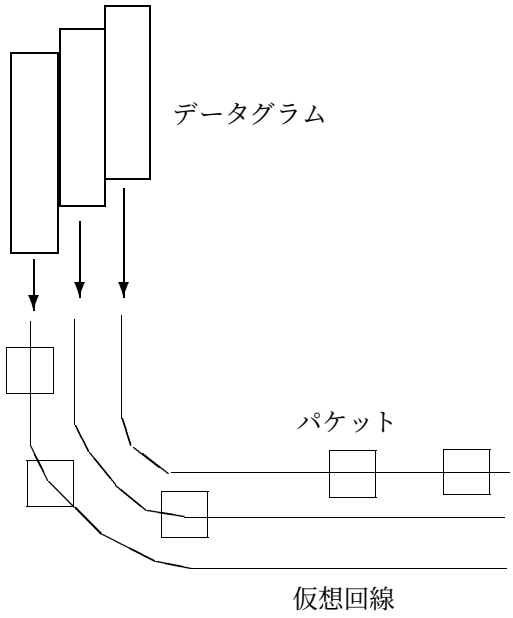


図 3.6: たてうなぎ方式

データグラムの組み立てを行なうことができない。そして、しばらくした後、到着したパケットをすべての回線上で一斉にデータグラムとして組み立てなければならなくなる。これでは、確認応答を送出するまでに時間がかかってしまい効率が良くない。

別のアプローチとしては、一つの IP データグラムを複数の回線にまたがって送出する方法（図 3.7）が考えられる。

このアプローチでは、IP データグラムを送信側と受信側で正しい順序で受け渡せるだけでなく、同一サイズのデータグラムを送信した場合でも、一定間隔でデータを受信することができる。これにより、一時的に計算機が混雑するということも防げる。ところが、この方式では、IP データグラムを組み立てることができない。なぜならば、X.25 の M bit はただ単にデータの連続性を意味しているだけなので、同一回線内でのみ有効である。ところが、回線をまたがってしまうと、M bit はもはや意味をなさない。それどころか、X.25 の各回線は全く独立しているので、パケットの到着順序は送信側と受信側では無関係であると考えた方が無難である。したがって、次はどの回線からデータを取り出して良いのか判断できない。

これを解決するためには、IP データグラムを X.25 パケットに分割す

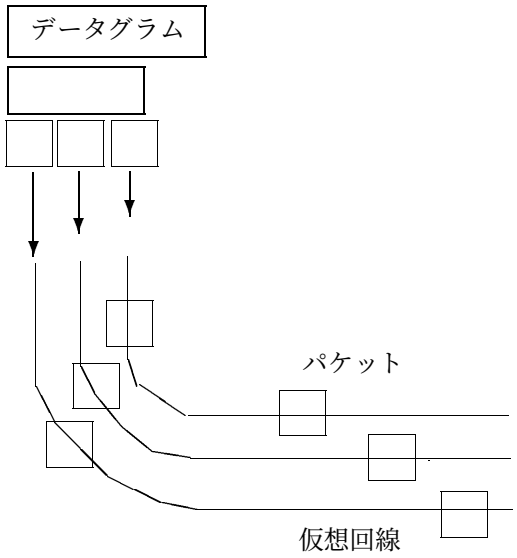


図 3.7: よこうなぎ方式

る際に順序番号の入ったヘッダを付加することである。このようにすれば、受信側では、順番通りにパケットを接続していき、M bit が 0 になった段階で、データグラムを IP レイヤの渡せば良い。この方式は、非常にシンプルで実装も簡単である。ところが、余分なヘッダを付加するために、通常のインプリメンテーションと通信ができない。これは、本研究の目標に反する。もしくは、シングルリンクのときは、ヘッダを付加しないようなアルゴリズムにしなければならなくなる。これは、実装を繁雑にするだけでなく統一性に欠けるので良い解決策とは言えない。

そこで、この問題を解決するためには、送信側と受信側で次にデータパケットを送受信すべき回線をあらかじめ認識してさえすればよい。これは、通信を開始するにあたってネゴシエーション用のパケットを送出しても良いが、コネクション設立時に Call User Data フィールドを用いて行なうことができる。**WIDE** /X.25 では、すでにマルチリンクの本数をこのフィールドに用いているが、その時に回線の番号も同時に取り決めるようにする（図 3.8）。このようにすれば、余分なヘッダを付加する必要もなく、受信側がマルチリンクをサポートしているか否かにかかわらず、アルゴリズムの変更なしに通信することができる。

ここで、IP データグラムがマルチリンク上を転送されて行く様子が、

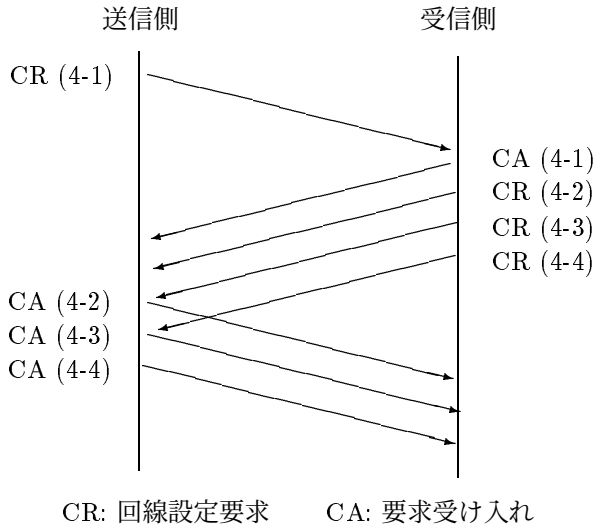


図 3.8: 改良されたコネクションの設定

あたかもうなぎの串焼きのような形をしているので、前者の方式を「たてうなぎ方式」、また、後者を「よこうなぎ方式」と呼ぶことにする。

最終的に **WIDE** /X.25 では、データ転送方式においては各仮想回線を平等に使用し、一時的な計算機の負荷の増大を防ぐために「よこうなぎ方式」を採用した。また、回線設立方式に関しては、回線設定要求パケット (CR) の Call User Data フィールドを用い、マルチリンクの本数および何本目の回線かを明示するようにした。これにより、「よこうなぎ方式」であっても特別なヘッダを付加することなく、データグラムの組み立てが可能となった。

# 第 4 章

# 実装

**WIDE** /X.25 では、下から HDLC, X.25, Multilink, IP-interface の 4 つの層からなる。どのレイヤも基本的には、3 つのモジュールから構成されており（図 4.1）、SONY NEWS (NWS 831) OS 3.2 (UNIX 4.3 BSD) カーネル内に C 言語で 6500 行で実装されている。

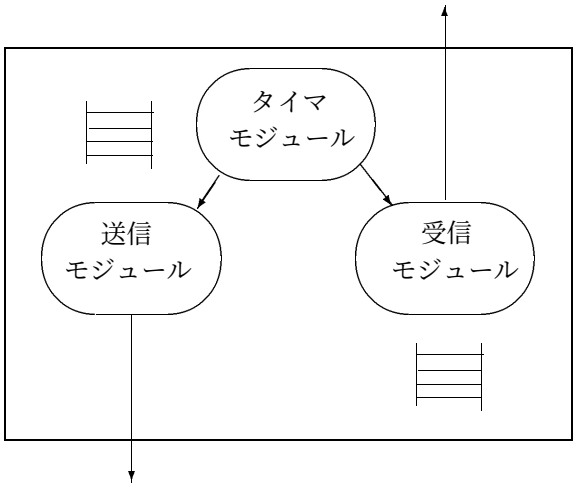


図 4.1: 各レイヤ内のモジュール構成

これらのモジュールのうち、送信モジュールは、上位レイヤから起動させられ、送信するデータのある間はループしている。逆に受信モジュールは下位レイヤから起動をかけられ、やはり処理すべきデータがある間は、ループして繰り返し処理を行なう。タイマモジュールは、オペレーティングシステムのタイマによって定期的に起動させられる。

送信モジュールは、データを上位レイヤとの間のキューから取り出し、そのレイヤに必要な情報をヘッダとしてデータに付加し、下位レイヤの出

力キューに入れる。

## 4.1　デバイスドライバ

### 4.1.1　受信ルーチン

SONY NEWS 831 では、デュアル CPU を採用しているため、IOP からの割り込みルーチンで呼び出される。この関数は、フレーム同期を取るための関数である。

```
syncrint(unit)
{
    if (データのサイズが 2 よりも小さい) {
        エラー処理;
        return;
    }

    while (データサイズ > 0) {
        MGET;               /*  mbuf を取ってくる */
        mbuf にコピー;
        データサイズを減らす;
    }

    IF_ENQUEUE();         /* 上位レイヤの入力キューに入れる */
    schednetisr(NETISR_X25);    /* スケジューリングする */
    ipc_send();          /* IOP 側に処理が終了したことを知らせる */
}
```

### 4.1.2　送信ルーチン

デバイスドライバの出力ルーチンは if_rsoutput() で、上位レイヤの HDLC モジュールとは、UNIX のインタフェースレイヤの形を取っている。そのため、引数としては、ネットワークインタフェースの構造体 ifnet へのポインタ、データ受渡し用の mbuf へのポインタ、カーネル内でアドレスを管理するための sockaddr 構造体へのポインタからなる。

```
if_rsoutput(ifp, m, dst)
{
    IF_ENQUEUEU();        /* 出力キューにデータを入れる */
```

```
    if (送信できる状態にない)
        return;

    IF_DEQUEUE();          /* 出力キューからデータを取り出す */

    bcopy();           /* mbuf --> buf コピー */
    m_freem();         /* mbuf の解放 */
    ipc_send();        /* IOP へデータ送信の依頼 */
}
```

## 4.2 HDLC レイヤ

### 4.2.1 受信ルーチン

受信ルーチンは、スケジューラから呼び出される。このルーチンでは、入力キューからデータを一つ取り出し、その時点での状態に即した処理を行なう。制御フレームを受信した場合は、状態遷移を行なった後、適切なフレームを返送する。データフレームであった場合は、上位レイヤに受け渡す。

```
ccittintr()
{
    for (;;) {
        IF_DEQUEUE();    /* 入力キューからデータを取り出す */

        if (データがない)
            return;

        アドレス部のチェック;

        switch (状態遷移) {

            case 状態１:

                フレームの認識;
                状態変数の更新;
                受信フレームに伴う Ack フレームの送信;
                break;
```

```
            case 状態２:

                処理２;
                break;

            default:
                エラー処理;
                m_freem();
        }
    }
}
```

### 4.2.2　出力ルーチン

このルーチンは基本的には上位レイヤから呼ばれる。すなわち、上位レイヤのパケットのみを HDLC レイヤのデータフレームに整形し、転送する。従って、HDLC レイヤ内の制御フレームの転送に関しては、このルーチンは責任を負わない。なお、データフレーム送信の際、網から再送要求が来る可能性があるので、フレームをコピーして格納しておく。これに対する確認応答を受信モジュールが受信したとき、このコピーは破棄される。よって、再送用のキューは、送信ルーチンからも入力ルーチンからもアクセスされることになる。

```
hdlc_output()
{
    while (ウィンドウサイズ < 最大ウィンドウサイズ) {

        HDLC_DEQUEUE();   /* 出力キューからデータの取りだし */

        if (データがない)
            return;

        フレーム送信;
        ウィンドウサイズの更新;
        状態遷移変数の更新;
    }
}
```

### 4.2.3 タイマルーチン

**WIDE** /X.25 では、AF_CCITT という新しいドメインを追加している。このドメインの追加により hdlc_fasttimo() は、200 msec 毎、hdlc_slowtimo() は、500 msec の割合で起動される。関数 hdlc_fasttimo() は、なんらかの原因によりリアルタイムで送信できなかったデータフレームを送信する。送信すべきデータが無い時は何もせずに終了する。

```
hdlc_fasttimo()
{
    if (送信可能状態) {
        while (ウィンドウサイズ < 最大ウィンドウサイズ) {

            HDLC_DEQUEUE();
            if (データがない)
                return;

            フレーム送信;
        }
    }
}
```

hdlc_slowtimo() は、タイマ変数を減少させていき、タイムアウトになったかどうかを判断する。タイムアウトになった場合は、状態に適した処理を行なう。基本的には、制御フレームを送信することになる。

```
hdlc_slowtimo()
{
    タイマ変数を 1 減らす;

    if (タイマ == 0) {
        if (タイムアウト > 最大タイムアウト) {

            switch (状態遷移) {

                case 状態１:

                    制御フレームの送信;
                    状態遷移変数の更新;
                    上位レイヤへ通知;
```

```
                    break;

                case 状態２:

                    処理２;
                    break;
            }

        }
    }
    タイムアウトの回数を 1 増やす;
}
```

## 4.3 X.25 レイヤ

ここでは、X.25 レイヤのモジュールについて説明する。HDLC レイヤと異なり、送信したデータを再送することはないので、再送用のキューは存在しない。

### 4.3.1 受信ルーチン

x25_input() は、下位レイヤからデータを mbuf を介して受け渡される。X.25 では、仮想回線という形で通信相手を管理しているため、まず最初に論理回線番号を調べ、対応するコントロールブロックを検索する。コントロールブロックとは、論理回線ごとの情報を管理しているデータの集合体のことである。

```
x25_input(m)
{
    LGCN と LCN から回線番号を調べる;

    回線番号からコントロールブロックを探す;

    パケットの種類を調べる;

    if (データパケット) {

        RR パケットを送信;
        X25_ENQUEUE();
```

```
        xml_input(x25cb);     /* 上位レイヤを呼び出す */

    } else {
        コントロールブロックの状態変数の更新;
        m_freem();          /* mbuf の解放 */
    }
}
```

### 4.3.2 送信ルーチン

この送信ルーチンは、上位レイヤからデータを直接受け渡されるのではなく、上位レイヤがデータをキューに入れた後、呼び出す。

```
x25_data_send(x25cb)
{
    if (HDLC 送信不可)
        return;

    if (X.25 送信可) {

        while (ウィンドウサイズ < 最大ウィンドウサイズ) {

            X25_DEQUEUE();
            if (データがない)
                return;

            データのパケット化;
            HDLC_ENQUEUE();
            hdlc_output();
            ウィンドウサイズの更新;
        }
    }
}
```

### 4.3.3 タイマモジュール

X.25 レイヤでは、500 msec ごとに x25_slowtimo() が起動する。X.25 レイヤでは、各状態のタイムアウトが長い (2 分から 3 分) ため、fasttimo() は必要ない。x25_slowtimo() では、出力キューに溜っているデータパケットを定期的に送信する役割をする。

```
x25_slowtimo()
{
    for (i = 0; i <= 最大コネクション数; i++) {

        コネクションごとのコントロールブロックを探す;
        if (データ送信可)
            データ送信;

        タイマ変数を 1 減らす;

        if (タイマ変数 == 0)
            回線の切断;
    }
}
```

## 4.4　マルチリンクレイヤ

X.25 レイヤでは、各仮想回線ごとにデータを管理していたが、マルチリンクレイヤでは、ホスト対ホストのデータを管理する。そのため、いくつかの仮想回線をまとめて管理する必要がある。

### 4.4.1　コネクションの設立

ここでは、マルチリンクのコネクションの設立方法に関する実装を説明する。

/*　上位レイヤからの接続指示 */

xml_connect() は、回線設定要求をするための関数である。この関数が呼ばれると、カーネル内にあるアドレスマッピングテーブルから設立したいリンク数を決定する。

```
xml_connect(xifcb)
{
    if (X.25 の論理回線が足りない) {
        どれかの回線を切断する;
        return;
    }

    status をコネクト中にする;
```

```
    タイマ起動;
    マルチリンクの本数を決定し、下位レイヤに伝える;
}
```

```
/*  接続要求に対する Ack */
```

xml_connected() は、上記の xml_connect() に対する応答を受け取った場合に下位レイヤから呼ばれる。ここで、1 回呼ばれる毎にリンクカウントを増やす。それが、要求した本数と一致すれば、回線設定が終了したことをただちに上位レイヤに通知する。

```
xml_connected(x25cb)
{
    if (status == コネクト中) {
        リンクカウントを 1 増やす;

        if (リンク数 == 要求した本数) {
            status = 使用中;
            上位レイヤに通知;
        }
    } else
       回線の切断指示;
}
```

```
/*  通信相手からの接続要求 */
```

xml_connectreq() は、網から回線設定要求が到着した場合の処理を行なう。ここで、通信相手が **WIDE** /X.25 であるかどうかを判断し、もしそうであればマルチリンクを何本要求しているかを確認する。相手の要求した値とこちら側の要求する値とを比較し、少ない本数を最終的なリンク数とし、残りの本数分の回線設定指示を下位レイヤに伝達する。

```
xml_connectreq(x25cb,dte,data)
{

    if (通信相手 == \WIDE /X.25) {
        要求された本数の確認;
        回線番号の認識;
```

```
    } else {
        要求は 1 本とみなす;
    }

    for (i = 0; i < 最大ホスト数; i++) {
        マルチリンクコントロールブロックを探す;

        if (X.121 アドレスが一致) {

            if (status == 使用中) {
                回線の切断指示;
                return;
            } else if (未使用)
                status = コネクト中;

            CA パケット送信指示;  /* CR パケットに対する Ack */

            if (回線番号 == 0) {

                if (要求された本数 == 1) {
                    status = 使用中;
                    上位レイヤに通知;
                    return;
                }
                残りの本数だけ CR パケット送信;

            } else {
                if (リンク数 == 要求された本数) {
                    status = 使用中;
                    上位レイヤに通知;
                }
            }
            return;
        }
    }
}
```

### 4.4.2 コネクションの切断

```
/* 上位レイヤからの切断指示 */
```

上位レイヤから切断指示があった場合は、該当する通信相手に対するマルチリンクの本数分だけ X.25 レイヤに切断指示を与える。

```
xml_disconnect(xifcb)
{
    for (i = 0; i < リンク数; i++)
        X.25 レイヤに切断指示;

    上位レイヤに通知;
}
```

```
/* 切断指示に対する Ack */
```

xml_disconnected() は、こちら側から送信した接続指示に対する応答である。この関数が呼び出される度に使用していたリンク数を減らしていき、0 になった段階で上位レイヤに切断終了を通知する。

```
xml_disconnected(x25cb)
{
    リンクの本数を 1 減らす;

    if (リンク本数 <= 0) {
        status = 未使用;
        上位レイヤに通知;
    }
}
```

```
/* 通信相手からの切断要求 */
```

xml_disconnreq() は、網側からの切断要求があった時に呼び出され、切断確認パケットを送信するように下位レイヤに指示を与える。リンクの本数が 0 になった段階で上位レイヤに通信終了を通知する。

```
xml_disconnreq(x25cb)
{
```

```
    CF パケット（切断確認）送信指示;

    リンクの本数を 1 減らす;

    if (リンクの本数 == 0) {
        status = 未使用;
        上位レイヤに通知;
    }
}
```

### 4.4.3　受信ルーチン

マルチリンクレイヤでは、「よこうなぎ方式」を採用しているために順番にパケットを取り込まなければならない。データグラムが完成するまでの間、データパケットを保持している必要がある。このレイヤでパケットの順序を入れ換えて組み立てるのは容易ではないため、この関数が呼ばれても期待した回線上にデータパケットが存在しない場合には、何もせずに処理を終了する。逆に、順序の正しい回線からパケットを取り出すことに成功すると、中途半端になっていたデータグラムの一番最後にこのパケットを追加する。その後、M bit により、データグラムが完成したかどうかを判断する。完成していれば、上位レイヤにデータグラムを渡す。

```
xml_input(x25cb)
{
    for(;;) {
        X25_DEQUEUE();  /* 入力キューから取ってくる */

        if (データがない) /* 期待した回線ではない */
            return;

        if (M bit == 0) {
            データグラムを完成させる;
            上位レイヤにデータグラムを渡す;
        } else {
            パケットを途中まで組み立てる;
        }
    }
}
```

### 4.4.4 送信ルーチン

上位レイヤから渡されたデータグラムはサイズが大きいので、X.25 のパケットサイズに分解する必要がでてくる。ここで、mbuf の区切りと X.25 のパケットサイズが必ずしも一致しないため、パケットの区切りで mbuf をコピーする。

```
xml_output(xifcb)
{
    if (status == コネクト中)
        return;

    for (;;) {
        IF_DEQUEUE();   /* 出力キューからデータを取り出す */

        if (データがない)
            return;

        for (;;) {

            if (データ長 > パケットサイズ) {
                パケットサイズ分だけ mbuf にコピー;
                X25_ENQUEUE();
                データ長からコピーした分を減らす;
            } else {
                データの残りを mbuf にコピー;
                X25_ENQUEUE();
                break;
            }
        }
        xml_data_send();
    }
}
```

## 4.5 IP-interface レイヤ

このレイヤは、IP レイヤとマルチリンクレイヤとのインタフェースをつかさどるレイヤである。このレイヤは、目的ホストに対してコネクションの存在性だけを管理しており、その他の情報は一切持たない。

### 4.5.1　受信ルーチン

基本的には、下位レイヤから渡されたデータをキューに入れた後、スケジューリングする。

```
xif_input(xmlcb,m)
{
    IF_ENQUEUE();    /*  データを入力キューに入れる */

    schednetisr(NETISR_IP);   /* スケジューリングする */
}
```

### 4.5.2　送信ルーチン

telnet などのキータイプによる小さいデータや TCP などの回線を管理するための制御データは、緊急性を要す場合が多いので優先的にキューに入れる。また、回線が設立されていなければ、接続指示を出す。接続中であれば、ただちに送信指示を下位レイヤに対して行なう。

```
xioutput(ifp, m, dst)
{
    小さいデータグラムを優先的にキューに入れる;

    if (status == 未使用)
        xml_connect();    /* 接続指示 */
    else if (status == 使用中)
        xml_output();     /* データ送信指示 */
}
```

## 4.6　タイマルーチン

このルーチンは、回線がある一定値以上保留状態が連続した場合、コネクションの切断指示を発行する。xi_timer() は、OS のタイマによって 15 秒に 1 回の割合で起動される。

```
xi_timer()
{
    for (i =0; i < 最大ホスト数; i++) {
        コントロールブロックを探す;
```

```
        if (status == 未使用)
             continue;

        else if (status == コネクト中) {
            if (回線設定タイムアウト)
                xml_disconnect();     /* 切断指示 */

        } else if (status == 使用中) {
            if (保留時間タイムアウト)
                xml_disconnect();     /* 切断指示 */
        }
    }
    timeout(xi_timer, 0, TIMERINT*hz);   /* 関数の再設定 */
}
```

# 第 5 章

# 評価

**WIDE** /X.25 は、本来のプロトコル体系をくずさないように設計・実装されたため、公衆パケット網である NTT の DDX-P や学術情報センターの NACSISnet 上で、Sun Micro Systems 社の SunLink および SONY ISDN board との接続に成功した。

## 5.1 実験環境

今回の実験は、学術情報網 (NACSIS) の国際衛星回線を利用して、東京大学とワシントン D.C. にある National Science Foundation（米国科学財団）との間で **WIDE** /X.25 同士の接続を行ないマルチリンクの性能測定を行なった。

| | 東京大学 | 米国科学財団 |
|---|---|---|
| 使用機種 | SONY NEWS-831 | SONY NEWS-830 |
| オペレーティングシステム | NEWS OS 3.2b | NEWS OS 3.2b |
| インタネットアドレス | 192.41.192.23 | 192.41.192.22 |
| X.121 アドレス | 9315800 | 9415010 |

表 5.1: 国際回線を使用した実験環境

その他の情報

**交換機** 富士通 FETEX

**ネットワーク** 学術情報網 (NACSIS/X.25) 9600bps

**転送データ長** 12288 バイト (/bin/echo)

**パケットサイズ** 128 オクテット

| | ウィンドウサイズ | | | | | |
|---|---|---|---|---|---|---|
| リンク数 | 2 | 3 | 4 | 5 | 6 | 7 |
| 1 | 2240 | 3200 | 2880 | 5200 | 5280 | 6240 |
| 2 | 2320 | 6240 | 6720 | 7040 | 5680 | 5760 |
| 3 | 3520 | 7120 | 7120 | 7600 | 7200 | 7280 |
| 4 | 7120 | 6880 | 6880 | 7120 | 7360 | 7360 |
| 5 | 6960 | 6880 | 7200 | 7200 | 7360 | 7120 |

表 5.2: スループット（単位は bps）とマルチリンク数の関係

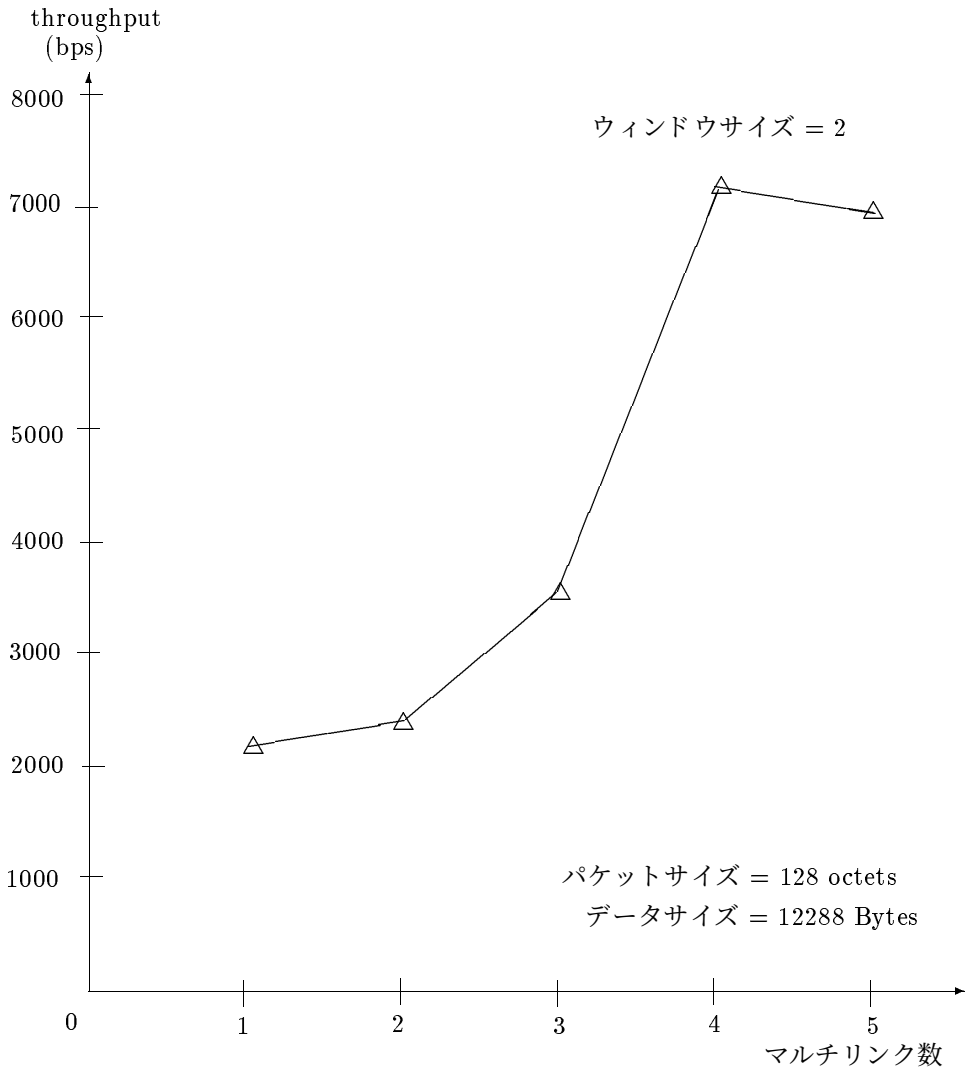


図 5.1: ウィンドウサイズ 2 におけるスループットとマルチリンクの関係

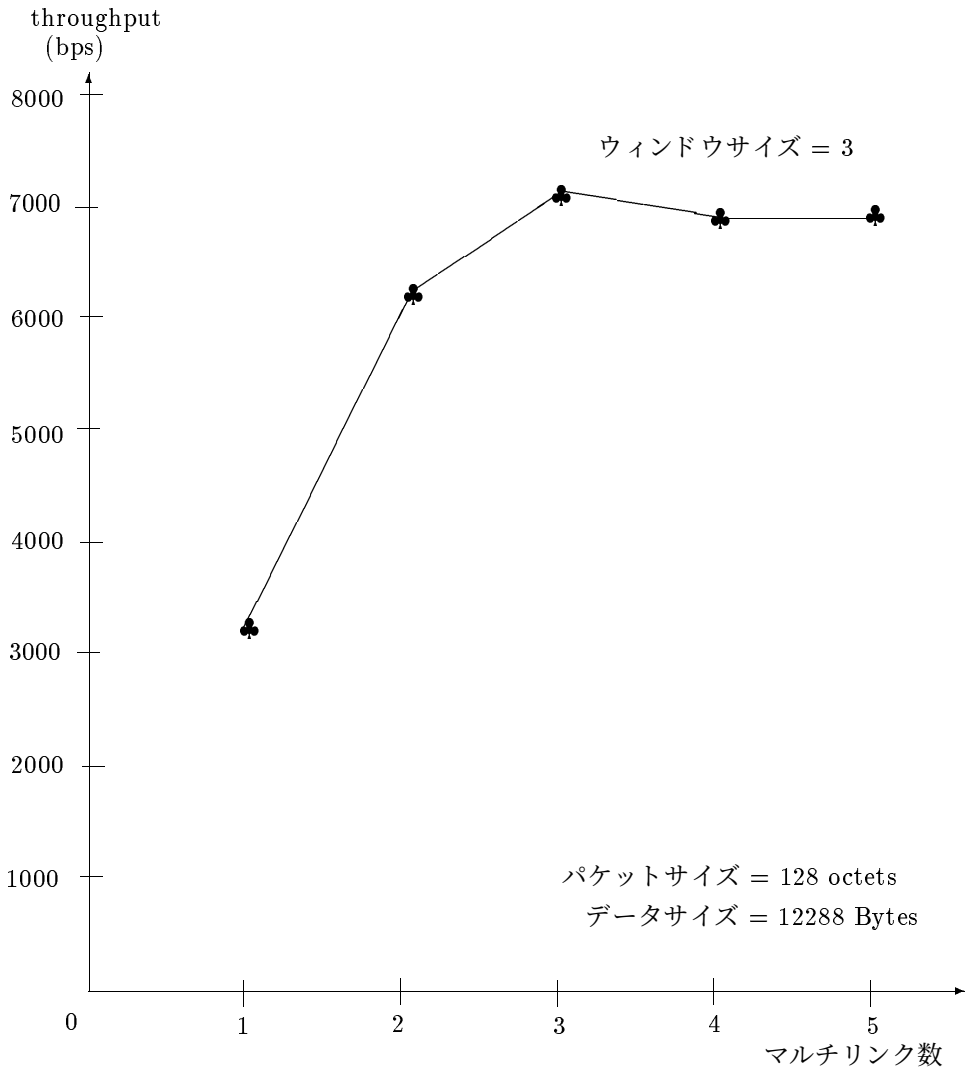


図 5.2: ウィンドウサイズ 3 におけるスループットとマルチリンクの関係

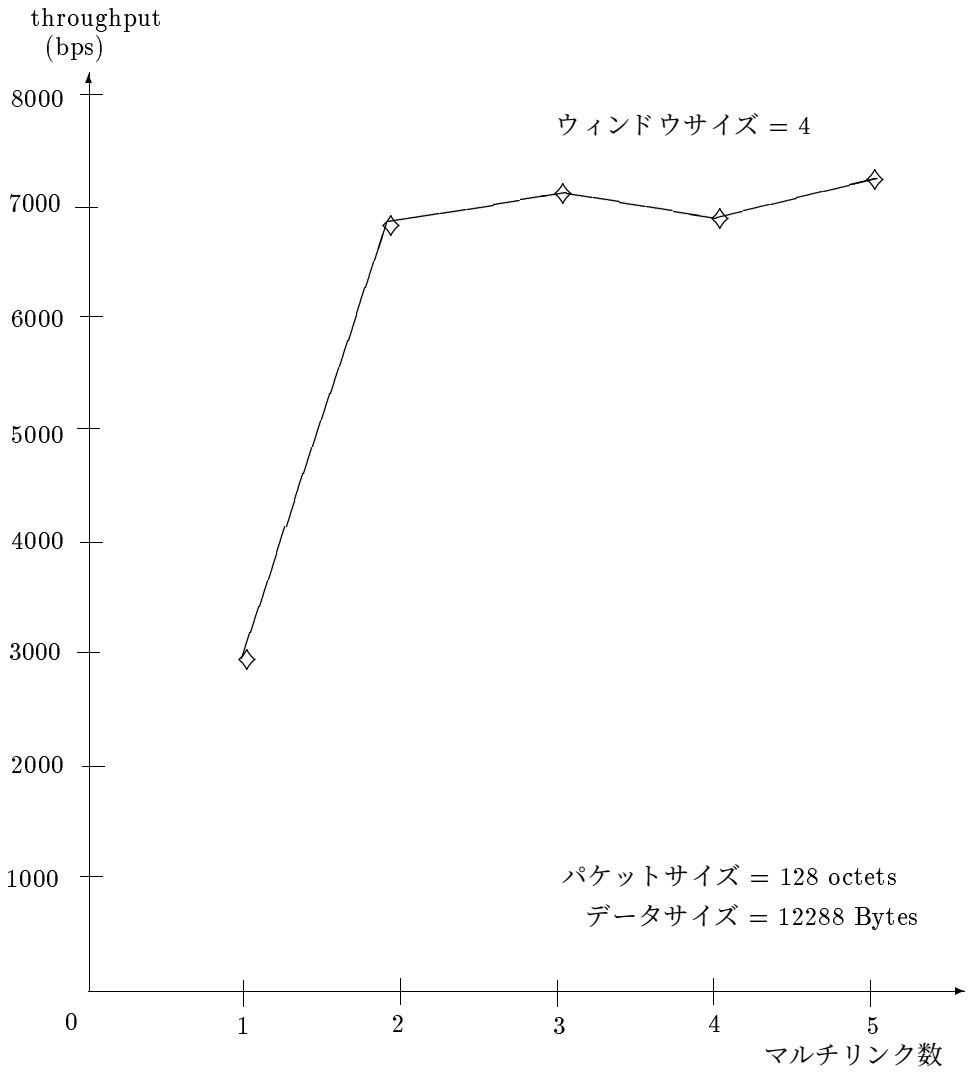


図 5.3: ウィンドウサイズ 4 におけるスループットとマルチリンクの関係

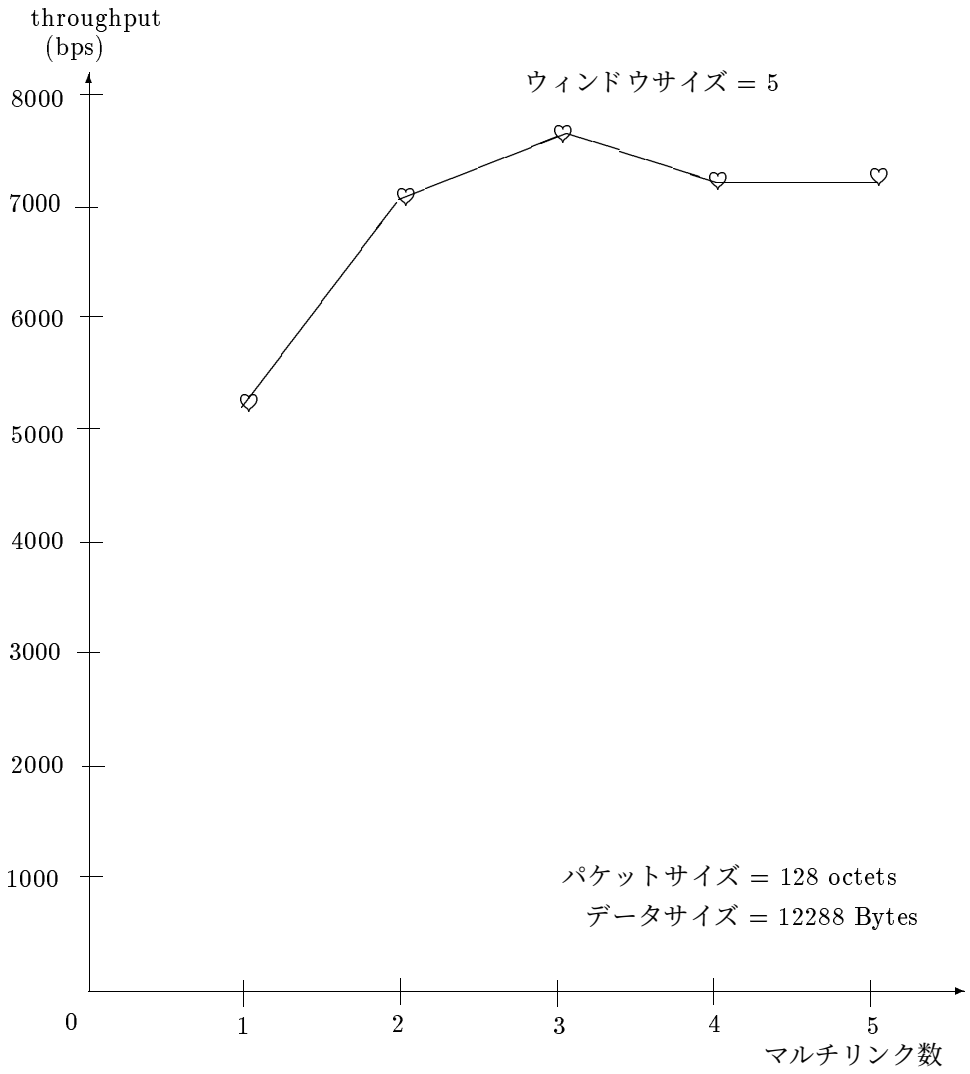


図 5.4: ウィンドウサイズ５におけるスループットとマルチリンクの関係

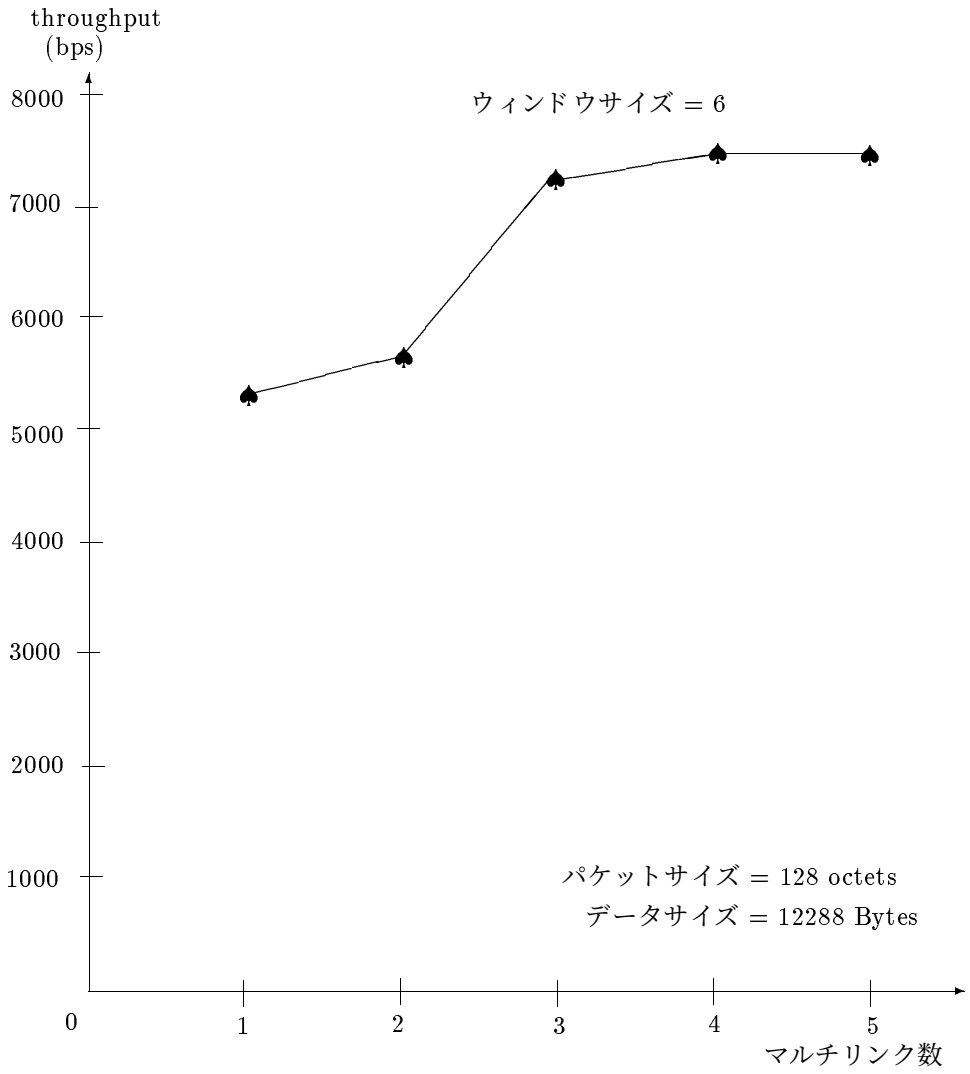


図 5.5: ウィンドウサイズ 6 におけるスループットとマルチリンクの関係

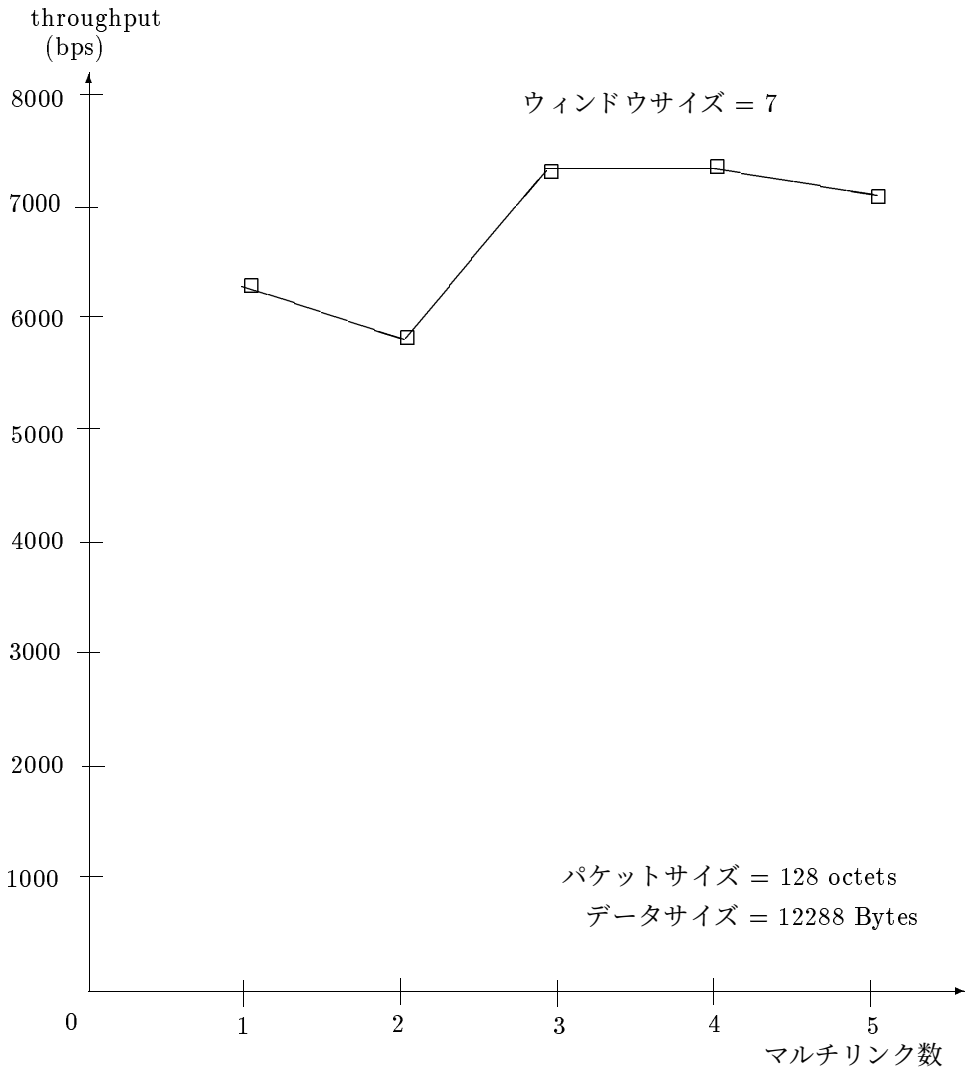


図 5.6: ウィンドウサイズ 7 におけるスループットとマルチリンクの関係

# 第 6 章

# 考察および今後の課題

## 6.1　転送速度について

前章からも明らかな様に、ウィンドウサイズが十分にとれない時は、本研究で提案したマルチリンクの手法により、通信の高速化が実現された。具体的には、衛星回線を利用した通信では、従来の方式と比較して 3.18 倍通信速度が向上した。本来、この研究で最も効果が現れる状況としては、衛星通信などのように、ウィンドウサイズが小さく、なおかつ遅延が大きいものに対してである。したがって、学術情報網などの遅延の小さい地上リンクに関しては、物理的な伝送速度を満たしてしまい、マルチリンクの効果が現れにくい。

## 6.2　リンク数の動的な変化

今回の実装では、データ転送に先だつコネクション設立時にマルチリンクの本数を静的に決定していた。ところが、この方式では、新たに別のリンクを張りたいという要求が上位レイヤから来た時に、論理回線の制約から、この要求を拒絶する可能性があった。また、この値は、あらかじめシステム管理者が相手との平均データ転送量を考慮して決めるものである。ところが、ネットワークの負荷は動的に変化するものなので、マルチリンクが必ずしも効率良く利用されているとは限らない。そこで、データ転送量にあった動的なリンク数の変更がなされるべきである。

さて、これを今回の実装を変更して実現するためには、各リンクごとにタイマ変数を変えて、動的にリンク数を減らしていく方法が考えられる。データ転送量が減れば、自動的にリンク数が減少していくはずである。ところが、よこうなぎ方式では、すべてのリンクを平等に使用しているため、徐々にリンク数を減らしていくことは不可能である。また、データ転送中にリンクの本数を変化させることは、送信側と受信側でのネゴシエーションが取れないので矛盾を生じる危険がある。したがって、リンク数を増加

していく場合でも、データ転送を一時中断して、送信側と受信側で回線の順序番号をネゴシエーションしなおす必要がある。このことは、よこうなぎ方式が、余分なヘッダを付加しない代わりに、各リンク毎に順序番号をつけたことに起因する。

これに対し、たてうなぎ方式では各回線をまったく独立に使用しているので、タイマ変数を変えることにより、動的にリンク数を減らすことも可能であるし、また、過負荷になったときに新たにリンクを設立しても他のリンクに影響を与えることはない。

**WIDE** /X.25 の目的として、色々な実験をするためのテストベッドとして利用することが今後あるので、たてうなぎバージョンも作る必要がある。だが、これもマルチリンクレイヤにおいて、パケットをどの回線に流すかというアルゴリズムの変更だけで済むので、簡単に実現できる。

## 6.3 パケットの優先順位について

UNIX のインタフェースレイヤでは、上位レイヤおよび下位レイヤとの間に一つずつ
キューを持つことで、上下のレイヤとデータを受渡ししていた。ところが、X.25 のようにコネクションを基本とした通信では、パケットの種類にもデータとコントロールの 2 通りある。コントロールパケットは、基本的になるべく早く効力を発揮しなければならない。すなわち、普通のデータパケットに比べ、優先順位が高いパケットである。また、telnet や rlogin といったユーザのキータイピングを転送するときにもスピードが要求される。これらのパケットに共通していることは、サイズが小さいということである。

そこで今回の実装では、送信の時にのみ、データサイズをみて、それが 2 よりも小さいならば、優先的にキューの先頭に割り込むのを許可するというものであった。これにより、稀にしか発生しないコントロールパケットやキータイピングも比較的早くネットワークに転送され、ftp と同時に telnet などを実行しても、さほど遅さは感じられなくなった。

## 6.4 X.121 アドレスと IP アドレスのマッピングテーブルの更新

実装の章で示した通り、**WIDE** /X.25 では、起動時にアドレスマッピングテーブルから X.121 アドレスと IP アドレスの静的なテーブルをカーネル内に作っている。これは、
**WIDE** Internet に参加するホストが増えるたびに全サイトで一斉に更新

されなければならない。このデータベースは、ネットワークインフォメーションセンターなどが一括管理して、定期的に配布すべきであろう。ところが、これでは通信する全ての相手のエントリが必要とされるため現実的ではない。アクセスする相手がデータベースになかった時のみ、named のように知らない相手が誰なのかを尋ねる作業をする方が実用的である。パケット交換には、ブロードキャストという概念が存在しないので、別のアプローチが必要かも知れないが、ネットワークが大きくなるとテーブルの管理やアップデートを頻繁に行なうことが、現実的に不可能になってくるので、X.121 アドレスと IP アドレスのマッピングをどのように動的に行なっていくかは、今後の重要な課題である。

## 6.5 Maximum Transfer Unit について

Maximum Transfer Unit(MTU) とは、インタフェースレイヤで扱える最大データ長のことである。これよりも大きいデータは、インタフェースレイヤで細分化（フラグメント）される。さて、広域ネットワークでは、データグラムは、Ethernet, X.25 網, 専用線などあらゆる伝送媒体を経由して転送される。このとき、MTU をいくつに取ると最も効率の良い転送ができるかという疑問が生じる。

たとえば、X.25 のパケットサイズがあらかじめ 256 バイトだと分かっていたものとすると、End-to-End での MTU を 256 バイトに設定しておけば、X.25 レイヤによるフラグメントは起こらない。これにより、フラグメントおよびリアセンブルによるオーバヘッドを減少させることができる。しかしながら、MTU を小さくとることによりデータグラムが細分化され、それに付加されるヘッダが多くなくので、結果としてパケット中の有効データ量は減ることになる。

逆に大きく取り過ぎると、複数の伝送媒体を経由する間に何回もフラグメントが起こる可能性が高くなる。また、最終目的地に到着した段階でリアセンブルすることになるが、このとき一つでもパケットが到着しないと、IP データグラムを組み立てるために、すべてのデータを再送しなければならなくなる危険性も高い。

このように、MTU はヘッダを付加することによるオーバヘッドとパケット喪失に伴う再送のオーバヘッドとのトレードオフである。これは、ネットワークの形態にも左右されるので一概にはどの値が最適かは言えないが、これも今後の **WIDE** プロジェクトのなかで研究されていくであろう。

## 6.6 経路選択に関する問題

ネットワークが拡大すると、ある目的地にデータを転送するのに経路が複数存在する場合がでてくる。今回 **WIDE** /X.25 のテスト段階で東京大学–東京工業大学間がこのような状態になった。東京大学 – 東京工業大学間には、既に 64Kbps の専用回線が敷設されており大部分のトラフィックがこの上を流れている。

さて、このように経路が複数あった場合、現在はそれぞれのインタフェースに IP アドレスが割り当てられており、ユーザが経路選択を行ないたい場合には、この IP アドレスを明示しなければならない。これは、複数のインタフェースを持つマシン上でのみ可能である。しかし、これらのマシンを経由した通信を考えた場合、中継ノードのインタフェースの IP アドレスまでは指定することはできない。したがって、どちらの経路を通るかはゲートウェイに一切任されることになる。

ところが、最近ではマルチメディアの研究が盛んになり、通信データもテキストファイルから音声、画像データと変化してきており、データ量も増加傾向にある。画像通信では、動画を実現するためには、大量のデータ転送をしかも瞬時に行なわなければならない。それに対して、ユーザがどこのマシンにログインしているかといった情報は、それほど高速通信を必要とはしていない。このように、各アプリケーションによって必要とする、通信量、遅延、スループット、信頼性、安全性などが異なる。そのため、アプリケーション自体が経路選択に関する要求をデータパケットに挿入する必要がある。

この問題を柔軟に解決する方法としては、IP データグラムのヘッダに Type Of Service フィールドが用意されているので、これを利用する方法が考えられる。現在、TOS フィールドは、D (遅延), T (スループット), R (信頼性) の 3 ビットが用意されている。

このうち D ビットは、 delay を小さく抑えるためのビットであり、T ビットは、高い throughput を、R ビットは、高信頼性をそれぞれ要求するためのビットである。これらのビットを上位レイヤでセットすることで、ポリシーを伝達することができる。このアプローチは、最適経路を選択する上でかなり有効である。

たとえば、今、あるサイトと通信するための経路として、衛星通信と専用線があるものとする。ただし、衛星通信は throughput は高いが、衛星と地上の間を電波が伝搬するために 270ms かかるため、delay が大きい。それに対して専用線は、キャパシティが低いものとする。このとき、phone のようにユーザからのキーストロークを送る場合、delay を抑えるために D ビットをセットしたとすれば、専用線が選択される。また、ファイル転

送のように高い throughput を必要とするのであれば、T ビットをセットすることにより、衛星通信が経路として選択される。

また、特に ISDN を意識した場合には、回線交換とパケット交換の選択が迫られる。 まず、あるサービスにおいて回線交換を必要とするか、パケット交換を必要とするかを考えてみる。その交換方式を決定するための要素として、

- Round Trip Time はどれくらい必要か
- throughput はどれくらい必要か
- packet size
- window size

が挙げられる。このときも、ファイル転送などのように、throughput を必要とするものは、T ビットを立てることで回線交換を選ぶことができる。

それに対し、rwhod や routed などのように、常に小さなデータを送信するアプリケーションは、packet size が小さくて良い。また、それほど throughput や round trip time を必要とはしていないので、パケット交換の方が適している。これは、R ビットを立てることで選択される。

伝送媒体の選択においては、ゲートウェイのインタフェース部で、このフィールドを見て決定することになる。最適経路が、使用できない時に行なわれる経路変更（バックアップリンクの利用）もこの部分で行なわれる。そのため、この部分では、常に各伝送媒体の状態を保持している。

アプリケーションによる経路選択に関する要求を IP ヘッダの TOS フィールドに挿入できるので、今後はいかに伝送媒体の混雑度などの状態をルーティングに積極的に反映させられるかが重要な課題である。**WIDE** /X.25 では、リンクの本数やパケットサイズ、ウィンドウサイズなどをユーザが簡単に変更できるインタフェースを提供しているので、これらの研究にも利用できる。

## 6.7　WIDE /X.25 を用いた研究

今後、**WIDE** /X.25 を用いてどのような研究ができるのかを考察する。**WIDE** /X.25 では、各仮想回線ごとに

- window size
- packet size

- timeout

が設定できるように作られている。これらを組み合わせることにより様々な実験環境をシミュレートすることが可能となる。

例えば、2 本の仮想回線において、一方だけウィンドウサイズとパケットサイズを共に小さくしておく。このようにすると、同一目的地に対して速い回線と遅い回線がある状況をシミュレートすることができる。そのため、アプリケーションによってどちらの回線を選択するべきかといった上述の経路選択問題を実験する上で、非常に役立つ。

また、やはり 2 本の仮想回線を用い、片方のタイムアウト値を小さく設定しておく。すると一方だけ回線が切断されやすくなるので、故障率の大きい伝送媒体をシミュレートできる。この値を変更すれば、平均故障間隔 (Mean Time Between Failures) や平均修復時間 (Mean Time To Repair) を調整することができ、動的な経路選択やルーティング情報の更新、データグラムの再送などを実験することができる。

さらに、信頼性の低い伝送路をシミュレートするためには、通常のマルチリンクを「たてうなぎ方式」で使用し、すべての回線上に同じデータを転送する。こうすることによって、例え信頼性が低くても多数決を行なうことにより冗長性を持たせることができる。また、バックアップリンクとしての利用も考えられる。

現在の **WIDE** /X.25 では、データの転送効率から「よこうなぎ方式」を採用したが、上記の 3 例では、すべて「たてうなぎ方式」によって実現される。このように「たてうなぎ方式」もマルチリンクでは重要な役割を演じていることからも、両方の方式をサポートすることが将来の課題である。

## 6.8 課金問題について

X.25 プロトコルでは、仮想回線を設立した側にすべて課金される体制になっている。このため、「よこうなぎ方式」では、送信側よりも受信側にほとんど課金さてしまい経済的に不公平が生じる。現に X.25 プロトコルでは、受信側に課金する手段として、「着信課金サービス」が用意されている。これは、仮想回線設立時に、CR パケット中に着信課金ファシリティを挿入することによって実現可能である。公衆パケット交換網では課金されるのが普通なので、着信課金機能を **WIDE** /X.25 に追加するべきであろう。

しかしながら、「たてうなぎ方式」においても回線が切断される前に、受信側から大量のファイル転送などを実行された場合は、やはり一方だけが不利となる。この問題は、着信課金では解決しないであろう。

また、課金を両方で二分したいといったケースも考えられるので、これらの要求に応えられるような機能を今後用意するべきであろう。

# 第 7 章

# 結論

高度情報化社会を迎え、日常生活においてもコンピュータネットワークの重要性が認識されてきている。次世代 OS である「分散 OS」を実現するためにも、遠隔地に存在している計算機資源をいかに高速にアクセスするかが鍵となっている。

広域分散ネットワークを構築するには、電話網、公衆パケット交換網、ISDN、専用線、イーサネット、FDDI など様々な伝送媒体を利用した通信を考慮しなくてはならない。これらを統一的に利用するだけではなく、その媒体の持っているキャパシティを最大限に利用しなければならない。データ転送量や回線保留時間、回線速度、転送遅延、課金体系など様々な観点からすると、各々の伝送媒体は一長一短である。

本研究では、広域分散ネットワークを実現する時に、最も汎用的に利用されるであろうパケット交換について、その高速化の研究を行なった。パケット交換方式では、網内でパケットを蓄積することによる遅延が非常に大きい。また、輻輳制御のため、パケットサイズやウィンドウサイズに制限が課されている。特に国際接続では、人工衛星までの物理的な距離が遠いため、網内遅延も大きい。そこで、仮想的にウィンドウサイズを増やすために、同一通信相手に対して、複数の仮想回線を設立する手法（マルチリンク手法）を導入した。

通信相手が通常の実装形態であることも想定して、網内に無駄なパケットが流出しないような回線設定アルゴリズムを提案した。これは、送信側から複数の仮想回線を張りに行こうとせず、最初の回線設定要求パケットに全体で何本リンクを張りたいかという要求を便乗させる方式である。受信側では、残りのバッファや論理回線数を考慮に入れながら、可能な本数分だけ、逆に仮想回線の設立を試みる。ここで送信側では、最初の回線設定要求を行なった段階で、あらかじめ希望した本数分の資源を確保してあるため、受信側から来た回線設定要求を拒絶することはない。この方式は、すべての回線が設立されるまでに 3 フェーズ要するが 2 本目以降の回線設定要求が拒否されることがないなど、効率面で優れている。

また、データ転送方式においては、マルチリンクをそれぞれ独立に扱い、IP データグラム単位で仮想回線に流入する「たてうなぎ方式」とパケット単位でマルチリンクを並列に使用する「よこうなぎ方式」を提案した。本研究では、データ転送の効率面から、「よこうなぎ方式」を採用した。この時、マルチリンクのためのヘッダを付加すると、通常の実装方式と通信するのが不可能になる。そこで、回線を設定する段階で、通信相手とパケットを送受信する順番などを取り決めておくことにより、マルチリンクを意識しない通信が実現された。

**WIDE** /X.25 は、SONY NWS-831 NEWS OS3.2 (UNIX 4.3BSD) 上で開発された。パケット交換網としては、学術情報センターの NACSIS、および、NTT の DDX との接続が確認された。また、**WIDE** /X.25 以外の製品として、Sun Micro Systems の SunLink、
SONY の ISDN board との接続試験を行ない、いずれも成功した。

**WIDE** /X.25 の性能評価を行なうために、NEWS をワシントン D.C. にある National Science Foundation（米国科学財団）に設置し、人工衛星を経由して東京大学との間でファイル転送を行なった。この結果、パケット交換方式において、ウィンドウサイズが小さく、かつ遅延の大きい通信においては、同一相手に対して仮想回線を複数設立するマルチリンク手法が効果的であることが実証された。今回の測定では、ウィンドウサイズ 2、パケットサイズ 128 オクテットとしたときに、従来の方式の 3 倍以上の通信速度の向上が確認された。

現在、**WIDE** /X.25 は、東京大学および東京工業大学において NACSIS/X.25 網を利用して稼働中である。1990 年 3 月をめどに α バージョンを配布予定である。また、3 月からは、Sun への移植も行ない、今後は 4.4 BSD への移植も行なわれる予定である。

# 付録 A

# 付録

**WIDE** /X.25 とマルチリンク接続をするためには、CR パケット中のコールユーザデータフィールドに以下のようなデータを挿入する必要がある。

## A.1 発呼

送信側では、マルチリンクの本数を決定後、CR パケットをただ一つ送信する。このときコールユーザデータの最初の 1 オクテットは、RFC 877 に基づき 0xCC を挿入する。これは、Internet Protocol の識別子である。**WIDE** /X.25 では、次の 1 オクテットは **WIDE** /X.25 の識別子として 0x33 を挿入するものとする。また、第 3 オクテットの上位 4 bit に要求されたマルチリンクの本数、下位 4 bit は何番目のリンクかを表すものとする。したがって **WIDE** /X.25 では、要求できるマルチリンクの最大数は、15 本までである。また、送信側におけるリンクの順番は、必ず 0 でなければならない。

## A.2 着呼

受信した CN パケットのコールユーザデータの値によって受信側は、異なった動作を行なう必要がある。

### A.2.1 WIDE /X.25 識別子があるとき

コールユーザデータの第 2 オクテットに **WIDE** /X.25 識別子がある場合は、以下の手順を踏まなければならない。

#### リンク番号が 0 のとき

受信された CN パケットのリンク番号が 0 のときは、

1. CA パケットを返す。
2. 要求された本数とアドレスマッピングテーブルから最終的なリンク数を決定する。
3. （リンク数 - 1) 本分の CR パケットを送信する。

ただし、CR パケットには IP 識別子、**WIDE** /X.25 識別子、最終的なリンク数、リンク番号が挿入されていなければならない。

#### リンク番号が 0 以外のとき

CA パケットを返し、最終的なリンク数およびリンク番号を格納する。最終的なリンク数だけ回線が確立された後、データ通信を開始する。

### A.2.2　WIDE /X.25 識別子がないとき

通信相手が、既存の X.25 プロトコルの実装と判断し、CA パケットをただ一つ返し、ただちにデータ通信を開始する。