コンピュータネットワーク教科書
(バージョン 0.99.80.08)
- Unix Lecture Trilogy 2 -

深町 賢一
http://www.nsrg.fml.org/

2019 年 6 月 4 日

はじめに

本ドキュメントは、“コンピュータネットワーク (2009〜)” の講義用に執筆した教科書です。

まぁ商業出版の予定は無いので同人誌ですが、この教科書だけで自習できるように作ってあります。

また、期末試験の持ち込みは「手書きのノート」のみとしますので、この教科書を使って、予習と復習をして下さい (要するに「ノートを作れ」)。

[注] この教科書は授業の間も手直しをしていくので、毎週バージョンが変わります。

印刷するなら、前年度の最終版か、今年度の最終版を印刷するのが、おすすめです。ちなみに、とても分厚いので一枚 2 ページ印刷をしてください。

また、参考書は付録 B を参照して下さい。

他の講義との相関について

本講義 (“コンピュータネットワーク (2009〜)” もしくは “ネットワーク基礎論 (〜2008)”) は、私が Unix[*1] 三部作 (Unix Trilogy) と (あくまで個人的に) 呼んでいる 3 つの授業

- プログラミング言語C。
- コンピュータネットワーク。
- オペレーティングシステム。

の二番目です。

この三つの授業内容は、たがいに深い相関関係にあります。

C 言語と Unix

C 言語は Unix オペレーティングシステム (OS: Operating System) を作るために設計されたプログラミング言語です。

C は、もっとも広く使われているプログラミング言語の一つで、OS や組み込み用 OS、ファームウエア (たとえば家電製品の OS)、サーバソフトウエアなど、ハードウエア制御や速度が要求されるソフトウエアの多くは C 言語で書かれています。

日本は製造業の国です。だから C が大事。

C は、もっとも基本的なプログラミング言語と言っても過言ではありません。

技術系の職につくなら、C 言語の心得があることは重要です。

C 言語の授業 (プログラミングスキル) で言ったとおり、「プログラマ需要のトップ 2 は C と Java です。ただ Java は明らかに C の影響を受けているので、(より難しい) C が書ける人は、たいていの言語が書けます (知らない言語でも、少し勉強すれば、すぐ書けるようになる) 。

インターネットと Unix

コンピュータネットワーク (Computer Network) と言えば、その代表はインターネット (Internet) です。

現在のインターネットの先祖は Unix オペレーティングシステムに試験実装されました。

そういった大学や研究所の実験機 (Unix) 同士が次第に相互接続した草の根的コンピュータネットワークが成長し、現在のインターネットと呼ばれるコンピュータネットワークの先祖になったのです。

Unix オペレーティングシステム

大学の PC 教室の PC は Microsoft Windows と Cent OS のデュアルブート (dual boot) です。Cent OS は Unix クローンと言われる OS にあ

[*1] Unix は「ゆにっくす」と発音します。業界の人は「うにっくす」と言うこともある:-) たとえば雑誌「Unix マガジン」(アスキー) は略して「うにまが」。

たります。

もうすこし詳しく言えば、現在、身の回りで使われているオペレーティングシステムは大きく分けて 3 つの系列に分かれます。

- Unix 直系の子孫ないしは Unix クローン。
  例: NetBSD OpenBSD FreeBSD Linux Solaris AIX HP-UX
  あと、Mac OS X のユーザランド (userland) は BSD からかき集めてて合体したものを元にしています。
- Mach[*2] の子孫ないしは Mach を手本にしたもの。
  例: Windows NT 系列、Mac OS X (の本体)。
- 組込み OS / リアルタイム OS。
  例: Simbian (Nokia の携帯で有名)、Tron (国産 OS だが死にかけ)、OS/9 など。

OS を理解する一番良い方法は、実際に動いている OS のソースコードを読むことです。つまり参照用ソースコードの入手しやすさが重要といえます。

そのため “コンピュータネットワーク” と “オペレーティングシステム” の授業では (無料[*3]で) 本物の動く OS と、そのソースコードのすべてが利用できるという理由により、Unix クローンを主に扱います。Simbian などの組込み系リアルタイム OS は扱いません。

対象者

本授業は選択科目ですが、できれば全員が履修することを推奨します。

[*2] 分散オペレーティングシステム。用語集も参照。くわしくは三年春の「オペレーティングシステム」の授業で。

[*3] フリーソフトウエア (free software) の OS 。なお、フリーソフトウエアの free は、無料 (free of charge) ではなく freedom の free です。詳しくはメディアリテラシーの授業「情報メディア社会」で説明します。

授業第一回「オリエンテーション」で、授業の目的、想定している対象者などについて説明します。詳細は 1 章を参照して下さい。

凡例

例の中で使う % や # はコマンドプロンプトです。これは OS がコマンドの入力を待っているという状態です。

```
% コマンド (Enter キーを押す)
# コマンド (Enter キーを押す)
```

例: ポータルサーバが動作しているか調査する時。

```
例: GNOME 端末を開き、% の右にある文字列を打ち込んで Enter を押す。

% ping -n portal.mc.chitose.ac.jp
```

モニター上では見分けづらいですが、各単語の間にはスペースが入っています。この例では、ping の左、ping と -n の間、-n と portal.mc.chitose.ac.jp の間にはスペースを入れて下さい。

また、入力したら最後に Enter を押すわけですが、それは常に省略です。

用語

付録 A に用語集があるので、それを参照して下さい。

問い合わせ先

各回の演習問題等は、それぞれの指示にしたがって下さい。

それら以外のもろもろの質問・改良案等については text-bugs@fml.org 宛に送って下さい。

FAQ

Q: fml.org って何ですか？ 何で ac.jp じゃな

いの？

A: 筆者の個人ドメインで、フリーソフトウエア開発プロジェクト用のドメインです。

メーリングリストドライバや one floppy NetBSD を筆頭に、こまごま、いろいろやってます。詳しくは http://www.fml.org/ を参照。

このドメインは 2028 年まで契約済みなので、安心してメールして下さい (そういう問題か？)

ac.jp でないのは… fml.org がインターネット業界的にトオリが良いからです。

あと、ソフトウエア開発プロジェクトは昔から .org と相場が決まっているのですよ。うむ。

Q: なんで「です、ます」調なんですか？

A: という質問？が以前アンケートにあったので解答しておきます。

(1) そもそも「です、ます」調だと、なんか困るのですか？

自習してもらうために自己完結している同人誌教科書ですから、これでいいと思いますけど？

なお「です、ます」と「ある、だ」どちらがいいということはありません。テクニカルライターも長年やってきましたが、出版社次第です。雑誌、出版社によっていろいろ。

(2) 学部用の教科書は、その分野について初心者向けのものになります。そのようなタイプの教科書は「です、ます」調が妥当です。

履歴

- バージョン 0.99.05。
  2014 年度版 (2014 年度の最初の版)。
- バージョン 0.99.04。
  2011 年度版。日本語の修整。
- バージョン 0.99.03。
  2010 年度版。日本語の修整。
- バージョン 0.99.02。
  章立てから「実習」「復習」を削除。
  これらは授業構成と合わせるために挿入されていただけの章のため、教科書としてダミーの章を入れているのは無意味という判断。
- バージョン 0.99.x。
  2010 年度版。
- バージョン 0.91.x 。
  2009 年度版。バージョンは毎週更新の予定。
- バージョン 0.90.01 。
  2008 年度講義版の最終バージョン。まだまだ日本語の修整が仕切れていない。
- バージョン 0.90 以前のもの。
  2008 年度講義用。たまに日本語がおかしいです。

目次

第 1 章

インターネットで何が出来るの？

はじめに

第一回はオリエンテーションです。

本講義の対象者と目的、次回以降の講義のために必要な若干の基礎知識について、具体例を元に解説します。

1.1 授業について

1.1.1 目的

コンピュータネットワーク、特にインターネットの基盤技術であるTCP/IPプロトコル体系について概観し、基本的な用語や知識を習得することが目的です。

教科書

本ドキュメント (PDF) が教科書です。

参考書

なお、参考書は付録 B を参照して下さい。

用語集

用語は初出の際に、講義の中で説明します。付録 A にある用語集も参照して下さい。

また、不足している部分については自分で参考書等を読み勉強して下さい。

1.1.2 授業の対象者

情報システム工学科では、本授業が必修になりました (再履修の人は選択科目の場合があります)。

いまや、世の中にある全ての機械をインターネット接続可能にしようとしている世の中です。極端な話をすれば、自動車や冷蔵庫、家まで対象にしようとしています。

日本は製造業の国です。

第二次産業 (製造業) に就職すれば、インターネットとまったく無関係な製品にたずさわる可能性は、まず無いでしょう。

第三次産業はサービス業つまりユーザと接する産業ですから、コミュニケーションの道具であるインターネットは当然という感があります。

いまや第一次産業でもインターネットを利用した効率的な運用方法や生産方式、情報交換が重要です。六次産業化などと言っていますね (1x2x3 で 6)。

つまり、どんな職についても、インターネットと無関係といったことは、まずありえません。そういった事情を考えれば、できれば全員が履修することを推奨するという理由が分かってもらえると思います。

なお、次のような条件を履修者の前提と考えています。「ふだんメールやウエブ (WWW) を使っている。」「N進数が分かる。」「コンピュータの基礎知識がある。」「オペレーティングシステムの基礎知識がある。」どの条件も皆さんは満たしているはずです。

また、難しい数学は出てきません。すべて算数のレベルです。このへん、歴史のない IT 産業のよいところです:-)

1.1.3 期末試験について

手書きのノートだけは持ち込み許可

です。もちろん、人の手書きノートのコピーは不可です。

試験直前にあわててノートを作るのではなく、ふだんから授業中にノートをまとめておきなさいね。

1.2 インターネットを利用したサービス

現代の身の回りにあるものの多くはインターネットを利用しています。特に通信関係でインターネットを利用していないものを探すことは難しいでしょう。

たとえば、毎日使っている大学のポータルシステムや各種ウエブページ、電子メールのシステムはインターネットを使って通信しています。

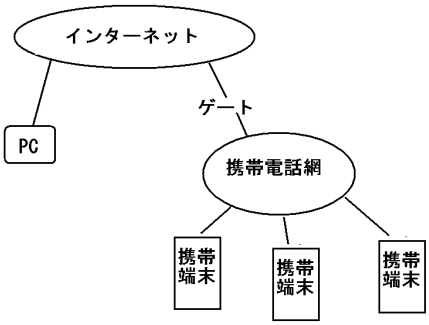

図 1.1 インターネットイメージ図: 携帯電話からもインターネットを利用できる。

携帯電話から “も” インターネットが利用可能です。たとえば携帯電話から検索したり、掲示板や blog に書き込んだりする人がたくさんいます。

正確には、携帯同士の通信は携帯会社独自のものですが、携帯電話の通信網からインターネットへの出口が用意されていて、インターネットが使えるようになっています。いまや「日本ではＰＣより携帯電話でインターネットを利用する人が多い」と言われているくらいです。

そういった身近な通信の事例以外にも、さまざまなものが裏側でインターネットを使っています。

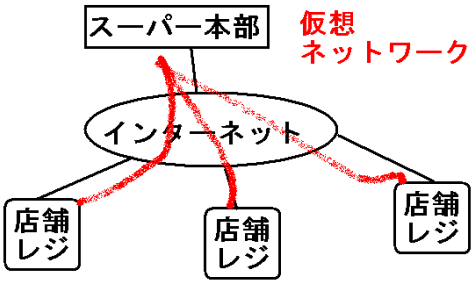

図 1.2 POS (Point Of Sales) システム。小売業では、インターネットを介して仮想的に社内ネットワークを構築している。

たとえばコンビニエンスストアやスーパーマーケットの各店舗にあるレジ (キャッシュレジスタ) は、その会社の中央にあるコンピュータと定期的に通信しています。これは中央で一括して在庫管理や売上を把握するためのシステムで POS (Point of Sales) システムと呼ばれています。POS を使っていない流通業や小売業などありえません。以前は、POS のデータ通信に専用回線を使っていましたが、最近ではコスト削減のためインターネットを利用しています。

製造業では、自動車や冷蔵庫本体のインターネット化というのは、これからなのでしょう (コラムも参照)。インターネットでのカタログの提供以外には、カーナビデータのアップデートくらいでしょうか？

現在、自動車や冷蔵庫、洗濯機などが、直接インターネット接続できるようにはなっていませ

ん。それよりも、「そもそも、接続して何をしたいんだ？」が問題です。しかしながら、自動車や家電、はたまた家までインターネット接続可能なようにしようという夢物語は、頻繁に唱えられています。

いずれにせよ、ユーザに見えている表舞台でも、裏側でも、インターネットは、すでに現代社会の重要なインフラストラクチャアとなっています。

コラム: 今日の不可能は明日可能となる。
「今日の不可能は明日可能となる」と言ったのはロケット工学の父ツォルコフスキーですが、インターネットの場合、妄想なのか詐欺なのか本気なのかよく分からない話がたくさんあります。
「家が Windows に操られる」とか、「冷蔵庫をインターネット化すれば出先から在庫の確認が出来る」とか、妄想たけだけしい話も多いです。
だいたい、冷蔵庫の中を確認するためには、いま何がどれだけ入れてあるとか、今日はどれだけ消費したとかを全部入力しなければいけないのに、それを誰が入力をするつもりでいるのでしょうね？ (まぁ、R 元年の今なら、画像監視できますよ！と言われますかね)
また、家の Windows ;-) OS が起動しなかったら閉じ込められたままになるのか？
そういうヨタ話も多いわけです。
まぁたんなる笑い話ならいいのですが、ヨタ話には実害もあります。
たとえばヨタ話をエサにして会社の宣伝をしているような例もありますし、「その会社の技術力と株価がどうみても釣り合っていない」という例は非常によく見かけます。
しかしながら、一般人にはそれが見抜けないので、だまされてしまうことも多いのが現実です。

以下、いくつかの例を少し詳しく見てみましょう。

1.2.1 ポータルシステム

まずは http://portal.mc.chitose.ac.jp/ を例に。

このシステムを使う時、みなさんは次のように操作するはずです。

1. OS (Windows や CentOS) を起動。
2. OS 上で WWW ブラウザ (Firefox) を起動。
3. Firefox で http://portal.mc.chitose.ac.jp/ を指定し、ポータルシステムへアクセス。
4. ポータルシステムへログイン (ユーザ名とパスワードを入力) → My Page が表示される。以下、略 (適宜スケジュールを見るなどの操作が続く)。

この場合、たとえば My Page が表示される時は、My Page の表示に必要なデータがポータルシステムから Firefox へ転送されています。そして、(生) データのままでは人間に読みづらいので Firefox が奇麗な画面に作り上げます。

コラム: 講義の責任分解点
この「データを転送する」部分が本講義の主題です。
Firefox の画面の見栄えや、ポータルシステム (サーバソフトウエア) の作り方は他の授業に、おまかせ。

コラム: 業界
この「データを転送する」部分を担当しているわれわれ ISP 業界は、大雑把にいえば「インフラ屋」に分類されるでしょう。
ちなみにリーマンショックの 2008 年、もっとも業績がよかった企業は、どこでしょうか？ (答えは、もちろんインフラ業界の某社です)
「インフラ屋」がいないと、社会が動かない。だから (一般論として) 不況にも強い。
携帯電話の機種変更は無理にしなくても生活に支障をきたしませんが、携帯電話の解約はできないでしょ？
もちろん不況でインフラ屋も業績が悪くはなったけど、製造業のように突然売上が三割減少とか半分になったりはしてない。
ただ、日本は製造業 (部品) の国なので、製造業が立ち直らないと、全産業がじわじわとボディブローをくらいまくっていく。もちろん、インフラ屋もそのうち影響を受けます (業績に時差があるということです)。

サーバクライアントシステム

これは “リクエストを出すもの” と “リクエストを受けるもの” からなるシステムです。

上のポータルシステムの例でいえば「Firefox が My Page を見たい」というリクエストを出し、ポータルシステムが、そのリクエストにこたえて My Page のデータを送り返します。

インターネットにおけるシステムのほぼすべてがこのような動作をしています。この “リクエストを出すもの”(クライアント) と “リクエストを受けるもの”(サーバ) からなるシステムはサーバクライアントシステムと呼ばれています。この例でいえば、ポータルシステムがサーバ、Firefox がクライアントです。

サーバクライアントシステムは身の回りにもたくさんあります。それを抽象化した概念にすぎません。

身近な例の一つはレストランです。この例では、お客がクライアント、給仕がサーバとなります。「サービス」とは給仕がクライアントにしてくれる仕事のことですよね？語源は同じです。

「マクドナルド」でも同じです。ハンバーガーの注文を受け、会計し、品物を渡してくれる人がサーバ、お客がクライアントとみなせます。

「吉野屋」や「みよしの」も同様です。

つまり、日常生活でも、多くの場面がサーバクライアントシステムから成っているわけです。インターネットにおけるサーバクライアントシステムとは、それを抽象化した概念と言えます。

コラム: サーバ
業界の人は、サーバという用語を複数の意味で使っています。

1. 本節のようにサーバの役割。つまりサーバクライアントシステムのサーバ側。
2. サーバの役割をするソフトウエア。たとえば WWW サーバ (ソフトウエア) の apache サーバソフトウエア。
3. サーバの役割をする PC のこと。これはサーバソフトウエアとPC、そして PC 上で走らせているOS 全部をさす。

慣れと文脈で意味を把握するしかありませんが、本講義に関していえば、サーバは、たいてい 1. か 2. の意味です。

コラム: テニスの「サービス」って何？
サーバクライアントシステムのサーバと同じ意味ですが、語源が忘れられてしまっているために現代では意味がとおらない例です。
初期のテニス期は貴族の遊びでした。その頃は、召使が最初のボールを旦那様に投げ (サービスし)、それを客つまりクライアントである旦那様 (プレイヤーつまりクライアント) がラケットで思いきり叩いて相手に打ち込んだのです。
今では自分でボールを投げ上げるため、最初の召使の部分がなくなってしまい、サービスの由来が分からなくなってしまっています。

プロトコル

「吉野屋」を例にとりましょう。この場合、牛丼の注文を受け、配膳し、会計してくれる「店員」が「サーバ」、「お客」が「クライアント」とみなせます。

しかしながら、サーバとクライアントの間には取り決めが必要です。たとえば客が「大盛り、つゆだく」という注文をしたとします。ところが「大盛り」や「つゆだく」が何を意味するか？を店員と客双方が分かっていないとリクエスト (注文) が成立しません。

こういった “サービスの取り決め” のことをプロトコル (Protocol) と呼びます。

日本語だと会話が出来て当たり前、プロトコルって何だ？と思うかも知れません。では、アメリカにいってマクドナルドでハンバーガーを買うことを想定してみて下さい。英語で何といえばハンバーガーにありつけるか分かりますか？

さらに、アメリカの吉野屋ではどうでしょう？

現実世界のプロトコルは、何語で会話するか？などボディランゲージのレベルから定義しなければうまくいきません。「相手に手をふる」という意味が挨拶ではなく、侮辱を意味する社会に行って、その取り決めを知らずに相手に手をふったら「逮捕される」とか「殺される」、そういう可能性も十分にありえます。

ただ、コンピュータの世界ではすべてが人工的な取り決めなので曖昧さはありません[*1]。その点、現実世界より若干、楽といえるでしょう。

なお、上述の「つゆだく」ですが、元々は正規メニューではないはずなのに、みんなが使っているうちに正規メニューとわれるようになったメニュー (というかオプション) です。

こういった押しつけられた取り決めではなく、なんとなく「草の根から業界の標準になってしまう取り決め」がよくあります。こういった取り決めのことを事実上の標準、デファクトスタンダード (defact standard) と呼びます。

ISO や JIS、ANSI のような規格 (スタンダード) は、たいてい国家が主導して業界団体が集まり規格を作っています。これが世間でいうところの (普通の)規格です。

一方、インターネットは、そういった上から押しつけられた規格ではなく、現場の研究者たちが自分たちで規格を決めてきました。そのため、インターネットの規格は、ほぼすべてがデファクトスタンダードといって差し支えありません。

インターネットにおける研究者や開発者の参加している代表的な団体がIETF で、IETF が発行しているドキュメントがRFC(Requests For Comments) です。RFC がインターネットの規

*1
ただし、プロトコルを定義する人が完璧でも、たいてい機械を作る方が失敗するので、メーカーが違うと通信できないということがよくあります。
そのため、強いメーカーのバグにあわせて修正しまうという悲しい現実もあります。
たとえば、インターネット業界では「CISCO のバグにあわせる」というのが普通です。

格文書となっています。

`http://www.ietf.org/rfc/`

これは、インターネットが草の根から作られていったことをよく示しているエピソードです。

1.2.2 携帯電話

携帯電話とインターネットに直接の関係はありません。携帯電話からインターネット “も” 使えるという話です。

たいてい、携帯電話 (mobile phone) は、携帯電話固有のサービス (携帯電話会社の独自部分) とインターネットが利用できるサービスの２つの要素から成り立っています。

たとえば、NTT ドコモの携帯から google (http://www.google.co.jp/) でインターネット検索するとか、掲示板サイトに接続して読み書きするとかできるわけですが、これは実際にインターネットへ接続しています。

実際にどういう動作をしているのか？というと、まず携帯電話は NTT ドコモの携帯電話専用の通信網というものに接続します。この NTT ドコモ通信網の中だけで閉じるドコモ固有のサービスもあります。携帯の設定変更や料金の問い合わせなどは、この通信網で閉じているはずです。

google などインターネットサイトに接続する場合には、ドコモ通信網のどこかにある出口 (この携帯通信網からインターネットへ接続するための口) からインターネットへ接続しています。

携帯のメールも同様です。username@docomo.ne.jp のようなメールアドレスを使って、携帯電話同士でも、普通のインターネットのメールアドレス宛へも、双方へ同じように電子メールが送れます。

携帯電話からインターネットへ電子メールを送信する際には上述の google の例と同様に、どこかの出口からインターネットへ接続してメールを送信しています。

ところで、この携帯電話の例で、どこからどこまでがインターネットというべきなのか？というと、その答は曖昧です。

どんな形であれ、通信方式が途中でどうなっているか？には関係なく、とにかくインターネットのサイトに接続できるとか、電子メールが相手に送信できるというのであれば、それは “インターネットを利用している” といえるのではないか？正直よくわかりません。

そこで、われわれコンピュータネットワークを専門に扱う技術者は、狭義の意味でインターネットという単語を使います。この狭い意味での “インターネット” とは “TCP/IP プロトコルを使うコンピュータネットワーク” と定義されます。本講義では、この意味でインターネットという単語を使いますので注意して下さい。

1.2.3 SNS

今は、コミュニケーション手段の多くが SNS と呼ばれる御時世ですが、この手のサービスのご先祖に BBS (掲示板) というものがあります。BBS は大昔、個人でコンピュータを所有できるようになった時代からありました。

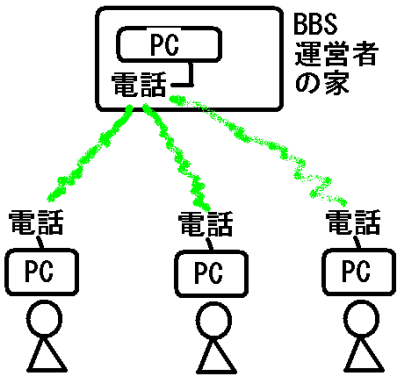

図 1.3 電話をかけていた時代の BBS のイメージ。

これは半世紀くらい前の話です。(BBS サーバ側) の家のパソコンに電話がつながっており、

BBS を読み書きしたい人は、そのパソコンへ電話をかけパソコン同士が通信したものです (図 1.3)。

もちろんアナログ電話の時代です。アナログ電話という通信回線の上でパソコン同士がデジタル通信をします。このためには、モデム (MODEM) という装置と、パソコン同士が会話するための取り決めが必要です。この取り決めもプロトコル (Protocol) です (「取り決め」は何でも「プロトコル」です)。

個人が運営している BBS も多かったものですが、日本で有名な商用 BBS の一つが NIFTY でした。

定額制の商用インターネットが普及するにつれ、こういった BBS はなくなり、インターネット版の BBS に置き換えられていきました。NIFTY も電話を使う型の BBS サービスは既に終了しています。

インターネット版の BBS は WWW (World Wide Web) を利用したものです。WWW については 3.2 節で取り上げます。

これら 2 つの BBS、電話版 BBS とインターネット版 BBS は、データ転送方式の相違を除けば、技術的な進化は本質的に無いといって差し支えありません。インターネット版 BBS しか生き残らなかったのは定額制の商用インターネットという要素が強いと言えます。

1.3 インターネットビジネス

一般ユーザに馴染みのあるインターネットを利用したビジネスというと、アマゾンに代表される通販 (通信販売[*2]、1.3.1 節参照) サイトやグーグルに代表される検索サイトの広告事業 (1.3.2 節参照) など、いわゆる “アプリケーション” 層のサービスです。

携帯向けのアプリや着メロのダウンロードなどプログラムや音楽データそのものの通販業もあります。これらはデータそのものの購入なので伝統的な通信販売とは異なりますが、それでも通信販売の一種です。

ユーザに見えやすい部分にかぎれば、インターネットには「広告業」と「通信販売」しかないと言いきることさえできそうです。

しかし、アプリケーションの前に、そもそも「インターネットへ接続」しなければインターネットは利用できません。

ユーザの目にふれることは少ないですが、インターネットという通信システムを安定運用する仕事をしている人たちがいます。それがインターネット接続を扱う業者です。この業者はインターネットサービスプロバイダ (ISP) と呼ばれています。

ISP はインターネットという通信のインフラストラクチャアを支えています。ISP の例は IIJ (日本で最初の ISP、http://www.iij.ad.jp/) や OCN、ぷららなどです。

NTT などの電話会社と ISP はどう違うのか？というと、おおまかに言って、土木工事までするのが電話会社、土木工事をしないのが ISP と考えるおけばよいでしょう。

ISP は光ファイバーの敷設などはしませんし、長距離回線 (札幌～東京、東京～アメリカ) の敷設や保守はしません。それを仕事にしているのは NTT などの、いわゆる電話会社 (キャリア) です。

なお、法律 (電気通信事業法) 上は、電話会社 (キャリア) を第一種通信事業者、プロバイダなどを第二種通信事業者と呼んでいます。

一般人がインターネットの物理的な装置を見ることはまずないので、IIJ と NTT、Amazon、Google はどこがどう違うのか？分かりにくいと思います。

それを順序立てて説明していくのが、この授業

[*2] もちろん、通信販売は「通信」の販売ではありません。対面での販売ではないということです。本来は遠隔販売とでもいうほうがわかりやすいような…

だと考えて下さい。

1.3.1 インターネット通販

Amazon.co.jp

インターネットで通信販売をしたことがありますか？

したことがある人は分かると思いますが、目的がはっきりしている買い物なら、とてつもなく便利です。

そして、数あるインターネット通販サイトの中で、ずばぬけて優秀なのがアマゾン (Amazon.co.jp、本家はアメリカの Amazon.com) です。

アマゾンは、この授業の観点では特別なシステムではありません。というのは、データ転送自体はごく普通だからです。アマゾンは、そのシステムの裏側のデータベース部分が非常にすぐれていますが、そこはデータ転送を担当するインターネットという仕組みと直接は関係ありません。

身近な例と比較すると分かりやすいでしょうか？

では、ポータルシステム[*3] と通販サイトは何が違うか分かりますか？

実のところ、この授業の観点では同じウエブのシステムです (サーバの中身は、まったく次元の違う技術ですが...)。

詳しくは次回かんがえてみましょう。

1.3.2 インターネット広告

Google

広告は多数の民衆が目にふれる場所に出してこそ効果のあるものです。

大昔でいえば「新聞」、２０世紀後半では「テレビ」が最も効果のある広告媒体でした。それが利権を生み、実体のない巨大企業群を栄えさせてきました。

それに代わったのがインターネット広告とソーシャルメディアです。ということは、インターネットでよく使われるサイトこそが最も強力な媒体（メディア）としての力を持つという意味になります。

つまり、何か有意義な情報を発信しているサイトであれば実在世界のマスコミと同様の力を持ち得ます。よって、フェイクニュースが問題になるわけです。

インターネット検索の最大手グーグル[*4] は最も強力なサイトの一つです。ブラウザを立ち上げたら、とりあえずグーグルという人も多いはずで、そのため最も目に触れるサイトがグーグルであり、グーグルが広告媒体としてパワーも持つことになります。

ただ、このビジネスモデルは諸刃の剣です。

実際、テレビＣＭに頼ったビジネスモデルであるテレビ局は、一世代前には栄華を究めました。が、いま、テレビ局の業績は落ちてきています。これが良い証拠です。

グーグルにも同じことがいえます。グーグルの収入のほぼすべてがインターネット広告です。かつて、最強のインターネット検索エンジンは Google か Yahoo か？と言われ続け、インターネット検索界の二強の栄華を誇ってきました。(アメリカ本家の)Yahoo は破綻したんですけどね…近代資本主義の根拠のない成長と、インターネット界の優秀なインターネット検索エンジンという２つの要素が破局した時、グーグルは潰れます。

音楽を売る

MP3 などデジタル化された音楽ファイルを売ることは単なる通信販売なので、この授業的には (技術的には) 何も面白いところがないのですが、ビジネスモデルとしては、どうなっていくのか？面白いかもしれません。

この分野でアップルが示した「ハードウエアとソフトウエアの両方を売るビジネスモデル」いわ

[*3] http://portal.mc.chitose.ac.jp/

[*4] http://www.google.co.jp/

ゆるプラットフォームを握った会社が圧勝するという手本が重要です。

ブルーオーシャンを全部さらっていくやり方ですね。

そして、日本の会社が苦手としているやり方でもあります。

動画を売る

これは DVD を通信販売で売るという意味ではなく、動画データをインターネットごしに転送する販売形式です。

動画も音楽もビジネスとして相違はありません。

ただ、転送量は桁違いです。

TSUTAYA や GEO で 100 円で借りられるものを、インターネットごしに買えるのは便利だといっても 10000 円でユーザが買うわけはありません。

Bフレッツとはいえ、DVD 並の質の動画データ転送には何時間もかかるし、それよりもサーバ側の設備が収入に見合いません。

このあたりの兼ね合いがつかないことが、うまくいっていない理由でしょう。

...

でも、いまや Netflix や Amazon TV の時代です。この手の会社のコンテンツ制作費は、すでに従来のテレビ会社より上になりました。

おまけ情報ですが、Netflix のサーバは FreeBSD (BSD Unix 一族の中で最高速度を目指す一派) です。

1.3.3 動画共有サイト: YouTube

BBS は普通、テキスト (文字情報) だけをやりとりする仕組みですが、WWW (3.2 参照) はどんなデジタルデータでも転送できます。よって音声でも映像でも流せます。

もっとも有名で映像交換サイトのハシリとなったサイトが YouTube です。著作権違反コンテンツの問題で揉めていることでも有名。これについては、あいかわらずイタチごっこみたいです。

動画サイトは、技術的観点からみれば単なる WWW サーバです。文字で書かれたファイルが動画に置き換わっただけのつまらない技術です。

しかも経営的には非常に疑問のあるサービスです。

YouTube 自体のビジネスモデルは最初から不明瞭でした。ユーザがたくさん見てくれることを期待して、ビデオ再生画面に (インターネット検索サイトのように) 広告を出すくらいしか思いつきません。実際、Youtube が登場してから随分たちましたが、活気的なビジネスモデルの進化はありません。

その一方、著作権侵害でもめたり、文字ベースの掲示板に比べて非常に多くのリソース (回線幅、ハードディスク容量、WWW サーバのパワー) を消費するため、何一つついいことがありません。

広告程度では収入が足りずに、潰れかけたところを Google という超優良企業が買いとったので、今でも存続していますが、おそらく今でも YouTube 単体では赤字経営でしょう。

YouTube と涼宮ハルヒの憂鬱
(ハルヒ自体は古いネタですが、このコラムはビジネスの歴史的な転換点について語っているので、残しておきます)
YouTube のビジネスモデルには進展がありませんが、著作権問題というより利権構造には 2007 年に少し変化がありました。
きっかけは明らかにアメリカで「涼宮ハルヒの憂鬱」[a] の DVD が売れたことです。日本ですらローカル放送だったのに、まして、アメリカでは放映すらしていないのに、角川書店は何もせずにアメリカで 10 億円は儲けたでしょう。
この件以降、角川書店は YouTube の広告効果に好意的な印象を持っているようです。

[a] 放映: 2006/04 ～ 2006/07 まで全 14 話。製作: 角川書店、角川エンターテイメント、京都アニメーション、クロックワークス。
なお、原作はライトノベル (角川スニーカー文庫)。

1.3.4 IP 電話

インターネット電話とかインターネットフォンなどとも呼ばれていますが、言っていることは同じです。なお「ひかり電話」は NTT の商標で、中身は普通のインターネット電話と同じです。

IP 電話は、上で上げた例に比べると、少し複雑で、認証などを行なう「サーバクライアントシステム」部分とクライアント同士が直接通信する音声データ転送部分のハイブリッド構成です。

この「クライアント同士が直接通信する」ことを P2P (Peer To Peer) などと最近では言っていますが、昔のことを知らない勘違いな話です。

1.3.5 未知数: インターネット電話

かつて、Skype が、「Skype 同士なら世界中無料で電話がかけられます」で一世を風靡しました。それから幾年月、SNS のソフトウエアに電話機能がついて、インターネット電話は、もう普通ですね？いまや私用で電話会社の “電話” を使うことの方が珍しいのではないでしょうか。

データ転送の理屈は同じなので、音声だろうと文字だろうと動画だろうとデータ転送ができます。それを利用して、チャットや会議室の機能もあります。中小企業は金がないため、この手のソフトウエアでテレビ会議室をしている例もよく聞きます。

ソフトウエアのソースコードが非公開なので、技術的な詳細が分かりませんが、動作を見る限り、基本的な理屈は上述の IP 電話と同様なハイブリッド構成です。実際 Skype のサーバ (認証などをするだけのサーバ) は数台しか存在しないと聞いています。

ただ Skype は営利企業なので、どこかで利益を上げないといけないのですが、この点に関しては、うまくいっていなかったようです。最終的にマイクロソフトに買収されました。なんだか重たいし、マイクロソフトだし、みんなで Skype から脱出しました。エンジニアは SLACK あたりへ脱出しましたね。一般的には、自分のよく使う SNS アプリの通話機能や会議機能だと思います。

1.4 まとめ

- インターネットは、通信のインフラストラクチャアとして、現代では欠かすことの出来ないもの。
- サーバクライアントモデル。
- プロトコル。

第 2 章

ポータルシステムを例にして考えてみよう

前回までのあらすじ

コンピュータネットワークとは「コンピュータが接続し通信している」仕組みの総称です。

インターネットは、コンピュータネットワークという技術の一つの例に過ぎませんが、歴史上もっとも成功したコンピュータネットワークに成長しました (その勝因についての考察はコラムに譲ります)。

コンピュータ同士が接続している通信の仕組みであればすべてコンピュータネットワークのはずです。たとえば、少し昔のアナログ電話や携帯電話もコンピュータによる自動制御をしています。アナログ電話はともかくとして、携帯電話はデスクトップパソコンなみにいろいろなことができる立派なマイクロコンピュータですから、携帯電話とインターネットは何が違うのでしょうか？しかしながら、携帯電話のことをコンピュータネットワークと呼ぶ人は見たことがありません。逆に「携帯電話でインターネットする」という言い方も間違いです。

狭義の定義では、「インターネットとはTCP/IPプロトコルという通信方式を使うコンピュータネットワーク」のことを指します。正確には TCP や IP は通信体系の一部の名前なので、「TCP/IP と総称される技術体系」とでも呼ぶべきですが、とりあえず気にしなくてかまいません。

本講義では「TCP/IP と総称される技術体系」の基本的な動作の全てを概観します。授業で扱う範囲がインターネットの核心部分だと考えて下さい。

2.1 ポータルシステムを解剖してみよう

いつものように立ち上げてポータルシステムへアクセスして下さい。

2.1.1 画面のソースを見てみよう

ポータルシステムの画面のソースを見てみましょう。

- Firefox を立ち上げる。
- https://portal.mc.chitose.ac.jp/ へアクセス。
  ログイン画面が出ましたね？
- 画面上のメニューから「表示 → ページのソース」をクリック。

別のウィンドウで次のようなウエブページのソースが表示されたはずです。

```
<!DOCTYPE HTML ... 長いので省略 ... >
<html lang="ja">
<head>
   ... 以下、略 ...
```

これがポータルシステムで最初に出てくるログイン画面のHTML 言語で書かれたソースコードです。

この画面は HTML という言語で書かれた一種のプログラムと考えてください (普通の意味では、プログラムとは言わないレベルの画面記述の言語ですが…)。

Firefox は、このプログラムを解釈し、ログイン画面を生成しているわけです。

たとえば、上から 24 行目あたりを見て下さい。ここは最初の画面でユーザ名とパスワードを入力させる部分に相当しています。

```
注: 一行が長いので適宜折り返しています。

<form action="/portal2/j_security_check" method="post" >
<table>
  <tr>
    <th>ユーザ ID:</th>
    <td>
    <input type="text" name="j_username" size="20"/>
    </td>
  </tr>
  <tr>
    <th>Password:</th>
    <td>
    <input type="password" name="j_password" size="20"/>
    </td>
  </tr>
  <tr>
    <td colspan="2">
    <input class="button" type="submit" value="ログイン" />
    </td>
  </tr>
</table>
</form>
```

この form 文を使った HTML のプログラムを Firefox が解釈し、ユーザ名とパスワードを入力する画面を作っています。

つまり、この HTML 文は、ポータルサイトの製作者が「ユーザ名」と「パスワード」を入力させる画面を作りたいという意図を Firefox に伝えているわけです。

これも一種の取り決めですが、HTML をプロトコルとは呼びません。通常、プロトコルという単語は通信規約に対してのみ使うと考えていて下さい。

2.1.2 HTML はインターネットではない

では、上の HTML は狭義のインターネットですか？答は NO です。

「世間的にはインターネット」かもしれませんが、本講義は狭義のインターネット、通信方式としてのインターネットを考える授業です。だから答は NO です。

我々の観点からいえば HTML のソースコードは通信方式とは関係ありません。「HTML のソースコード」というデータを転送するのが “インターネット” の役割です。

このソースコードは、単にポータルサイトの製作者と Firefox の間の取り決めであって、Firefox とポータルサイトのサーバ (portal.mc.chitose.ac.jp) との間の通信方式とは何の関係もありません。

逆に言えば、転送するデータの詳細について “インターネット” は気にしていないので、HTML でも音楽でも映像でもどんなものでも転送が可能と言えます。

2.1.3 WWW

「HTML のソースコードを転送する仕組み」をインターネットが提供します。

ポータルサーバと Firefox クライアントの間のデータ転送はHTTP (Hyper Text Transfer Protocol) という名のプロトコルが行なっています。

HTTP は WWW (World Wide Web) と総称されるアプリケーションの転送担当部分のことです。WWW とは「HTML で書かれたソースコード」(データ) を、HTTP で転送を行なう仕組みと言えます。つまり、

```
WWW = HTML + HTTP
```

と考えて下さい。

ちなみに、上の例の場合、ポータルサーバは WWW サーバ、Firefox は WWW クライアントと呼びます。

2.1.4 土管としての HTTP

WWW の基本的な形は

```
WWW = HTML + HTTP
```

ですが HTTP 自体は、どんなデジタルデータで

も転送できます。文字、音声、動画、暗号化された謎のデータ、なんでも可です。

よって、正しくは

```
WWW = データ（何でも OK） + HTTP
```

と考えるのが正しいことになります。

データ部分の代表例は HTML ですが、HTML も標準の HTML をはじめとして拡張された言語 JavaScript や (Adobe) Macromedia Flash などのアドオン、JPEG や GIF といった画像、MPEG2 DivX Xvid WMV H.264 FMV などに代表される動画、さまざまなものが転送可能です。

ただし、転送できるからといって Firefox が画面に表示できるかどうかは分かりません。

はじめてＥラーニングシステムを使おうとした際に「Flash のアドオンがないので Macromedia からダウンロードしてきてください」という警告画面が表示されたことがありませんか？ (大学の PC 教室では、すでに flash がインストール済みの場合があるので、この画面が出ないかもしれません)。

これが「転送できることと Firefox が理解できることは違う」ということの代表事例です

ですが、前述のように、この Firefox が転送後のデータで困るとか困らないとかは、我々的には関係ありません。我々インターネット屋としては HTTP でデータ転送が確実に出来ていれば OK です。

そんなわけで、HTTP は「どんなデータでも流せるデータ用の土管」みたいなものだと考えて下さい。

2.2 インターネット的な住所

2.2.1 手紙

土管としての HTTP (2.1.4 節) はアプリケーションを書く人にとっては、とても楽な仕組みです。

たとえばプログラミング言語 Perl で書けば、実質、数行[*1] でポータルシステムの最初の画面の HTML ソースを転送するプログラムが書けます。

```
Perl の例（だいぶ簡略化してある）:
ポータルの HTML ソース（先頭 128 バイト）を転送し、画面に表示。

#!/usr/bin/env perl

use IO::Socket;
$socket = IO::Socket::INET->new('portal.mc.chitose.ac.jp:80');
$socket->sysread($buf, 128);
print $buf;

exit(0);
```

ここで通信のために (プログラマが) 指定している情報は portal.mc.chitose.ac.jp:80 の部分だけです。

- 相手のホスト名 = portal.mc.chitose.ac.jp
- ポート番号 = 80

正確には５つの情報が必要です。

その話の前に少し脇道にそれます。

現実世界の通信方式について考えてみます。たとえば郵便や宅急便です。以下では郵便を例にとりましょう。

手紙を出すことを考えてみてください。

相手に手紙を出す場合、

- 相手の住所
- 相手の氏名
- 自分の住所
- 自分の氏名

の４つは必須情報です。

郵便番号や電話番号は、あった方が確実に届きますが、必須情報ではありません。つまり「通信を特定する」には上の四つの情報だけで十分です。

さらに意識的に「宅急便ではなく手紙 (郵便) という通信手段を選んでいる」わけですから、その指定も必須情報です。つまり、通信するには少

[*1] ここで載せているプログラム例は ASSERT やエラー検出のコードが書かれていない不完全なプログラム。「実質、数行」と言っているのは、「転送プログラムの中核部分だけなら数行」ということ。

なくとも 5 つの情報を指定する必要があるわけです。

では、インターネット版です。

インターネット版でも同様で五つの情報指定が必須です。たとえば HTTP では、次の五つです。

- 相手の住所 = 送信先の IP アドレス
- 相手の氏名 = 送信先のポート番号
- 自分の住所 = 送信元の IP アドレス
- 自分の氏名 = 送信元のポート番号
- TCP を使う。

2.2.2 IP アドレスとポート番号

IP アドレスはインターネットにおける住所です。PC ごとに異なる IP アドレスがついています。住所に相当する情報なので PC ごとに固有の IP アドレスが割り当てられているのは当然ですね？

ポート番号はアプリケーションを特定するための識別番号と考えて下さい。

手紙でも同じ住所で複数の人がいたら住所だけでは届きません。そのために「相手の氏名」も書く必要があるわけです。

それと同様のことがインターネットでも言えます。同じ IP アドレス (住所) の PC には複数のアプリケーションが動いているため、それらを区別する必要があります。そのためにポート番号の指定が必要です。

なお、サーバ側はポート番号が固定です。たとえば WWW サーバはポート番号80 を使うことになっています。これが上のプログラム例に出てきた 80 の理由です。

各アプリケーションで使うポート番号、つまりサーバが使うポート番号は IANA 管理の規格として一覧表になっています。

PC 教室の PC についている IP アドレスはプライベートアドレスという特殊な使い方をしている IP アドレスです。詳しくは講義後半戦の NAT の回 (8.1 節参照) で説明します。

2.3 ハードウエア

ふたたび少し脇道ですが、ネットワーク機材について少しだけ見てみることにします。

ポータルシステムそのものではありませんが、まぁ似たようなものです。

(この節、未完成)

2.3.1 サーバ機材の例

2.3.2 機材同士をつなぐ配線

機材同士をつなぐ配線はイーサネットか光ファイバーです。

2.4 階層モデル

IP アドレスとポート番号といったソフトウエア、光ファイバーやサーバ機材といったハードウエアについて見ましたが、これらは互いにどういう関係にあるでしょうか？

ユーザに近い側から書いてみると、クライアント PC では次のようになります。

1. Firefox (WWW ブラウザ)
   ソフトウエア
2. オペレーティングシステム
   ソフトウエア
3. PC
   ハードウエア
4. イーサネット
   ハードウエア

サーバ PC では次のようになります。

1. ポータルシステム

ソフトウエア。WWW サーバ上で動作。
2. WWW サーバ
   ソフトウエア
3. オペレーティングシステム
   ソフトウエア
4. PC
   ハードウエア
5. イーサネット
   ハードウエア

WWW サーバは常に動作しているプログラムのことですが、そのサーバの設定をするのは人間なので、WWW サーバがユーザ (サーバの中身を作る人たち) に一番近い側といえます。

このように並べてみると階層の並び方が同じです。WWW サーバと Firefox は WWW というサービスのサーバとクライアントなので、同じ WWW というアプリケーション (サービス) と考えて下さい。

つまり、次のようになっているわけです。

1. アプリケーション
2. オペレーティングシステム
3. PC
4. イーサネット

こういった順番に重なっている構造を階層構造と呼びます。

この例ではソフトウエアとハードウエアが入り乱れていますが、ソフトウエアとハードウエアの境は曖昧ですし、複雑なモデルの考察をする際には詳細にとらわれず、動作や役割を元に考える方がすっきりします。

上の例をインターネット屋は次のように考えるのが普通です。

1. アプリケーションの階層 (HTTP)
2. データ転送をする階層 (1) (TCP,UDP)
   上図ではオペレーティングシステムの中に含まれています。
3. データ転送をする階層 (2) (IP,ICMP)
   上図ではオペレーティングシステムの中に含まれています。
4. イーサネットによるデータ転送。
5. 物理媒体。
   光ファイバーやイーサネット。

また、教科書的説明では、階層構造を七つの階層に分けて考えるのが普通です。この理論モデルはOSI 7 階層モデル (OSI 7 Layer Model)(図 2.1) と呼ばれています。

OSI 7 階層モデルは理論上のモデルなので、現実のネットワークが 7 階層でなくても気にする必要はありません。実際、上に示したように、インターネットは 7 階層のモデルになっていません。

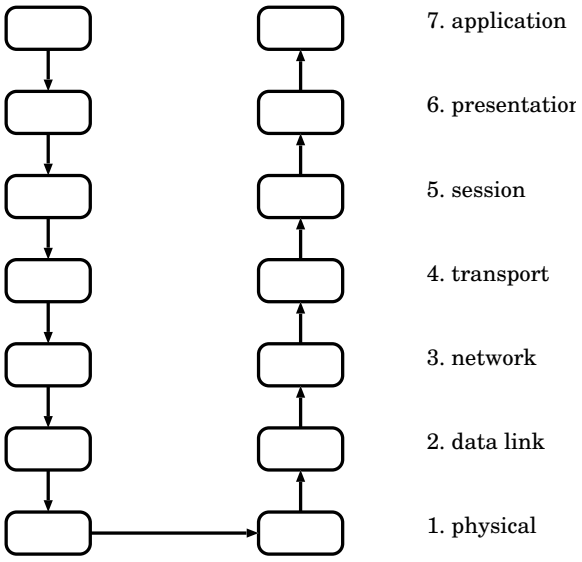

図 2.1　OSI 7 階層モデル

コラム: いまだに OSI モデルなの？
OSI モデルは商用インターネット以前に提案された理論上のモデルですが、ネットワークの教科書では、階層モデルの説明に OSI モデルを出してくるのが定番です。これは、教科書的説明に OSI より圧倒的に優れた「良いモデルがない」というだけのことでもありますが「とりあえずこの単語を知らないと現場で話が通じないので知っているべきもの」でもあります。
前のコラムで述べたように、 OSI 7 階層モデルは、ネットワークの説明において、必ず引き合いにだされる有名なモデルですが、あくまでも参照モデルであることに注意してください。 OSI の規格は、どれも不必要に複雑である傾向があり OSI 7 階層モデルにも、そのキライがあります。実際、インターネットの基盤技術である TCP/IP は OSI 7 階層モデルに完全には当てはまりません。
なお、OSI モデルは、某巨大企業 (IBM) の組織構造を模したものだという噂があります:)

社長→配送依頼→役員
役員→配送依頼→部長
...
社員→宅急便
宅急便屋が隣のビルへ郵便物を配送
宅急便→社員
...（逆順に繰り返す）
役員→配送→隣のビルの社長

社長　　社長
↓　　↑
役員　　役員
↓　　↑
部長　　部長
↓　　↑
課長　　課長
↓　　↑
主任　　主任
↓　　↑
社員　　社員
↓　　↑
宅急便屋 → 宅急便屋

社長　　社長　　ブラウザ、SMTP
↓　　↑
役員　　役員
↓　　↑
部長　　部長
↓　　↑
課長　　課長　　TCP
↓　　↑
主任　　主任　　IP
↓　　↑
社員　　社員　　イーサネット
↓　　↑
宅急便屋 → 宅急便屋　　銅線

2.5 パケット通信

2.5.1 パケットと通信

前節までで、インターネットという通信システムは階層化された構造を持つことを見ました。

光ファイバーの中を飛ぶレーザー光線や、イーサネットの中を流れる電流は、電磁波なので波としてアナログに振舞うわけですが、コンピュータが扱っているデータはデジタル、つまり 1 か 0 です。

どこかでアナログとデジタルが変換されているわけですが、それは授業の範囲外なので、他の授業におまかせします。

我々としては、デジタルデータしか扱いません。

2.5.2 パケット

デジタルデータは、小さな塊に分けて扱います。

この塊がパケット (packet) です。

インターネットにおけるパケットの代表的な大きさは 1500 バイトです。その理由は第 10 章で分かります。

1500 バイトは、とても小さな単位です。たとえば、DVD は 4.7 G (4,700,000) バイトですから、DVD のデータを転送する際には、約 300 万

パケットも必要になります。

2.5.3 WWW でのパケット通信例

2500 bytes の大きさの HTML ソース (データ) を転送する場合を考えましょう。

このデータを 1460 bytes と 1040 bytes の 2 つに分けます。

データが 1460 bytes と 1040 bytes の 2 つのパケット (図 2.2) を作り、これらを送信します。そして受信した相手が再構成し、2500 bytes の HTML ソースが Firefox へ渡されます。

分割と再構成をしているのは誰でしょうか？それは OS です。正確には OS の中にあるネットワークシステム中の TCP というサブシステムです。詳しくは第 5 章で説明します。

TCP サブシステムが仮想的なデータの土管を提供してくれるため、WWW サーバと Firefox は何も考えずに OS にデータを流したいと命令するだけでかまいません (プログラマは楽ちん)。

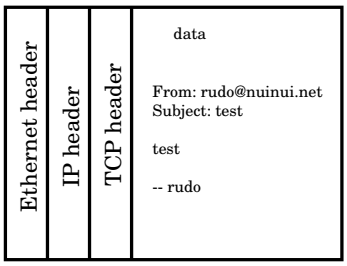

図 2.2 パケットの例:一つのイーサネットパケットを解剖した様子。この例では一つのパケットに一つのメール全体が入っている。

パケットは図 2.2 のように、データそのものとヘッダと呼ばれる制御情報の部分からなります。図 2.2 は TCP/IP モデルの一番下側の例なので、左側にはヘッダが 3 つ (イーサネットヘッダ、IP ヘッダ、TCP ヘッダ) 連なり、一番右側にデータ (例は電子メール) があります。

パケットのサイズはヘッダも含めた大きさになります。だから、上の例でデータを分割する際に 1500 ではなく 1460 などとなっていたわけです。

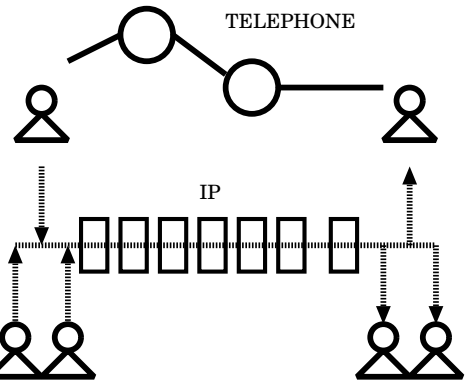

図 2.3 通信をパケット化することで得られるメリットの例:アナログ電話は一つの通信回線を二人が会話することしか出来ないが、パケット化すれば一つの通信回線を複数の人が複数の目的 (電話、メール、WWW など) で利用することが出来る。

コラム: パケット通信は手紙だと思え

「電話とは違う理論の上に成立する通信体系を考えろ」という依頼を受けた TCP/IP の設計者は「電報」を思い浮かべたと言われています。

みなさんは電報など見たことがない世代かも知れませんが、郵便だと思えば分かりやすいでしょう。

一枚の葉書きには 1500 文字しか書けないとします。もし 2500 文字の文章を送りたいなら「1460 文字の葉書」と「1040 文字の葉書」の二枚の葉書を書いて相手に送ればよいわけです。これがパケット化とパケットの分割です。

葉書を受けとった側は二つの葉書きを続けて読めば意味が分かります。つまり 2 つのパケットから元の内容を再合成できたわけですね。

パケット通信とは、これだけのことです。

コラム: アナログ電話をパケット化する (?)
(かなり無理矢理な例ですが) 電話でも同じようなパケット処理ができます。
昔のアナログ電話の場合、「あ、い、う、え、お」を相手に伝えようとすると、

1. 相手との間に電線 (銅線) が物理的に接続される。
2. 「あ」の電気信号の電磁波が電線の上を進み、相手側の電話機で音声が再生される。
3. 「い」の電気信号の電磁波が電線の上を進み、相手側の電話機で音声が再生される。
4. …

のようになっています。つまりアナログです。
これを次のようにしても相手に「あ、い、う、え、お」を伝えることは出来るわけです。

1. 「あ」をカセットテープに録音
2. 封筒に入れ (カプセル化)
3. 郵便で送る
4. 受けとった側はそれを再生
5. 「い」をカセットテープに録音
6. … 以下、同じように繰り返す。略。…

無駄に複雑ですが、電話とは異なるメリット (図 2.3) があるために、わざわざ複雑なことをしているわけです。それを説明するのが、この講義と言えます。

2.6 まとめ

- 階層モデル
- パケット通信
  - 小さく分割して送信。
  - 受信側で再構成。
  - 途中の経路は、よきにはからわれる。
    → 詳しくはルーティングの回。
- インターネットの住所
  - IPv4 アドレス。
- アプリケーションの区別
  - ポート番号。
- 通信の特定
  - IPv4 アドレス x2 + ポート番号の組 x2 + プロトコルの指定。

コラム: OSI 陣営 vs IETF という政治力学
OSI (Open System Interconnection) は ISO という業界団体が提唱した理論モデルです。
コンピュータが高価な時代のコンピュータメーカーが集まって作った業界団体なので、既存利権勢力側の団体と考えてもらってかまいません。かつて (1960 年代あたりまで) の業務用コンピュータの主流はレンタルで、主に使っていたのは巨大企業や政府です。大学にはコンピュータが一つしかないといった時代の話です。業界団体が利権をもつ巨大企業の集まりなのは致し方ないでしょう。
かたやインターネットの規格を決めてきた IETF は現場の技術者や研究者の集まり、いわば草の根団体です。Unix コミュニティも同様の草の根団体が強力です。
いずれにせよ、全く出自の異なる OSI 陣営と IETF 陣営は政治的対立関係にあります。IPv6 規格の成立過程などが、その良い例です。

第 3 章

アプリケーション

前回までのあらすじ

ポータルシステムを例にして、プロトコル、サーバクライアントモデル、階層モデル、パケット通信、IP アドレスとポート番号などについて概観しました。

今回は、アプリケーションレベルでの転送方式について詳しく見てみることにします。

3.1 プロトコル

3.1.1 はじめに

今回はプロトコルに慣れることにします。

通信システムのあらゆる部分に、ハードウエアに近い側 (ハードウエア側) からユーザに近い側 (ソフトウエア側) までプロトコル (決めごと) があります。

ユーザに近い側からプロトコルを見てみましょう。ハードウエアに近い側から説明をはじめてもいいのですが、ユーザに近い側のアプリケーション層 (レイヤー 7) プロトコルの多くは可読性が高く、理解が容易です。

よって、本教科書ではレイヤー 7 からレイヤー 2 へ向かって説明していきます。

今回はウエブやメールといったアプリケーションのプロトコルを見てみましょう。

なお、実際のアプリケーションの動作では DNS というプロトコルも考慮に入れないといけないのですが、DNS は複雑なため、今回は DNS の部分を抜かして (DNS 処理が終った後の部分だけを) 説明します。DNS についての詳細は次回 (第 4 章)。

プロトコル	ポート番号
FTP	21
SSH	22
TELNET	23
SMTP	25
HTTP	80
POP3	110

表 3.1　ポート番号の例

3.1.2 凡例

前回、説明した通り、通信には少なくとも五つの情報を指定する必要があります。特に断らないかぎり、本章の通信は次の値とします。

- サーバの IP アドレス = 192.168.0.1。
- サーバのポート番号。
  アプリケーションごとに異なるので、そのつど説明します。例: HTTP なら 80 など。
- クライアントの IP アドレス = 10.0.0.1。
- クライアントのポート番号。
  通信のたびに OS が適当に番号を割り振ります (本教科書の図表では、たいてい 1025 番です)。
- 通信方式は TCP。

3.1.3 プロトコルのいくつかの例

代表的なアプリケーションプロトコルの例をいくつか簡単な方からあげると、次のようになります。

- HTTP
  “Hyper Text Transfer Protocol”。
  もっとも簡単なプロトコルの例。
- POP3
  “Post Office Protocol Version 3”。
  HTTP より、少しだけ複雑。認証あり。ステートフル (statefull) なプロトコル。
- SMTP
  “Simple Mail Transfer Protocol”。
  さらに複雑。ステートフル (statefull) なプロトコル。RFC822 (RFC2822, RFC5335, RFC5336) を参照。
  オリジナルの SMTP には認証がありませんが、現代の多くのプロバイダは、メール送信時に認証を要求するSMTP-AUTH を使っています。
- FTP
  “File Transfer Protocol”。
  これも代表的なプロトコルですが、複雑なので省略します。
- TELNET
  操作が複雑なので省略します。

3.2 HTTP

ポータルシステム、つまり WWW が転送に使っているHTTP が一番簡単なアプリケーションプロトコルの一つです。

これを例にとり、プロトコルというものに慣れることにしましょう。

3.2.1 WWW の復習

WWW ブラウザ (WWW クライアント) で URL (例: http://portal.mc.chitose.ac.jp/) を指定すると、ブラウザは URL のサーバから指定されたファイルをダウンロードし、表示します。

ファイルの形式は任意です。また URL が / で終る場合、サーバのデフォルトファイル名 (例: index.html) がダウンロード対象となります。

ただし、ファイルの形式に関わらず転送は可能ですが、その転送したファイルの形式をブラウザが理解できない場合、うまく表示できません。

つまり “転送する” という動作と “画面に表示する” という動作は全く別の “取り決め” になっているということを意味します。そして、我々は転送の部分だけに着目していることに注意して下さい。

3.2.2 URL

WWW ブラウザに入力する文字列 http://portal.mc.chitose.ac.jp/ は、URL(Uniform Resource Locator) と呼ばれます。フォーマットは「プロトコル://ドメイン/パス」です。「パス」部分が省略されることもあります。

Locator という名前の通り、URL はインターネットという情報空間の中での位置を示すものです。

URL のプロトコル部分では HTTP が最も有名ですが HTTPS や FTP などさまざまなプロトコルが指定可能です。

```
http://portal.mc.chitose.ac.jp/
https://portal.mc.chitose.ac.jp/
telnet://host.example.org/
ftp://ftp.fml.org/pub/fml8/
rsync://rsync.fml.org/
... その他いろいろ ...
```

なお、現代では URI が正式な名称です。

コラム: URL? URI? URN?
これらの用語は紛らわしいですが、URI (Uniform Resource Identifier) が正式で URI の一部が URL と URN ということになります。
WWW を設計した当時は URL しかなかったのですが、のちに「情報空間で識別可能なものすべてを xxx:yyy 形式で表現したい」という欲求が生まれ、URI URL URN の三つの概念が定義されました。
URL は Locator です。つまり、あくまでも一種の場所を示しているため、「サーバ + ファイルパス」形式です。
一方、場所とは無関係な「電話番号」や「ISBN (本の番号)」も考えられます。これらの概念には「場所」という意味がないので、これらを表現しようとすると URL 形式では困るわけです。
そこで URL から URI へ概念が拡張されました。逆に言えば、URL を考えた頃には、そこまで考えてなかったということでしょう。
参考文献: URI は RFC2396 を、URN は RFC2141 を参照。

URL の例

```
http://www.chitose.ac.jp/
ftp://ftp.iij.ad.jp/
mailto:user@example.co.jp
```

URN の例

```
tel:012-345-6789
isbn:1-23-45678
```

3.2.3 サーバの IP アドレス

portal.mc.chitose.ac.jp の IP アドレスは 172.16.3.45 です。

WWW ブラウザは、どのように、その IP アドレスを知るのか？

については次回DNS の回で説明します。

以下では WWW サーバの IP アドレスが 192.168.0.1 であるとして話を進めましょう。

3.2.4 WWW ブラウザの動作

WWW ブラウザのデータ転送の動作は、おおまかにいえば次のようなものです。

1. ブラウザは、WWW サーバ (192.168.0.1) と通信したいと OS に依頼します。
2. OS が TCP の通信路 (いわばデジタル土管ですね) を作成します。
   専門用語では「socket を open する」などと言います。
3. WWW サーバ (192.168.0.1) と (TCP で) 通信できる準備ができたら、HTTP で WWW サーバとデータのやりとりをはじめます。
   HTTP の命令を送り、データを受けとります。ここの部分だけが HTTP と呼ばれるプロトコルです。
4. ブラウザは、WWW サーバ (192.168.0.1) との通信を終了したいと OS に依頼します。
5. OS が TCP の土管を廃棄します。
   専門用語では「socket を close する」などと言います。

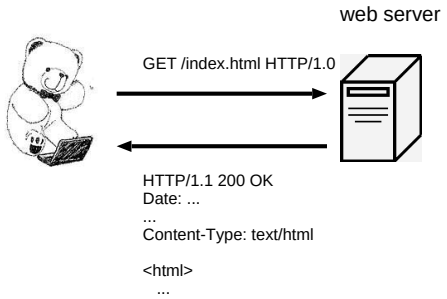

図 3.1 HTTP の概念図

3.2.5 HTTP

では、HTTP の詳細を見てみましょう。

1. クライアントからサーバへリクエストを送信します。
   リクエストとは「ファイル index.html が欲しい」といった内容です。

   ```
   GET ファイルのパス HTTP/1.0

   例:
   GET /index.html HTTP/1.0
   ```

   HTTP のリクエストでは「命令 ファイル名 HTTP/バージョン番号」(それぞれの文字列の間にはスペースがある。スペース区切り) というフォーマットを使っています。
2. サーバからクライアントへデータ転送
   サーバからの返事としてデータ (この場合は index.html というファイルそのもの) が送られてきます。
   ただ、データ本体以外にもクライアント (WWW ブラウザ) に伝えるための制御情報がヘッダとして送られてきます。
   そのため返事全体としてはヘッダ、一行の空行、データというパケットに似た形で送られてきます。
   命令を「スペースで区切る」「空行でデータを区切る」「: でデータを区切る」というテクニックは Unix では非常にありふれた文法です。多くのプロトコルやファイルフォーマットが、このフォーマットを採用しています (実際には、もちろん逆で、Unix の真似をして HTTP を設計したわけです)。
   - HTTP ヘッダ + 空行 + ファイル
   - HTTP ヘッダの先頭はステータスコードでスペース区切り
   - HTTP ヘッダの形は「ヘッダフィールド:値」形式
     これは電子メールフォーマットの真似です。最終更新日時、サイズなどの付加情報が入っており、WWW ブラウザに対するヒントとなります。

3.2.6 演習: HTTP

HTTP を手動で実行してみましょう (ソフトウエア工学概論でやっているので省略します)。

1. CentOS でターミナルを開いてください
   Windows の場合は、アクセサリ→ MS-DOS プロンプトで同じことが出来ますが、この後の動作が違うので CentOS 上で作業することを推奨します。
2. 「telnet 172.20.172.30 80」を打ち込んで ENTER キーを叩いてください。
3. 次の文字列が表示されたら TCP で「接続した」状態となったという意味です。

   ```
   Connected to 172.20.172.30.
   Escape character is '^]'.
   ```

4. 「GET /index.html HTTP/1.0」(注: 二箇所スペースがあります) を入力し、ENTER キーを二回叩いてください。
5. index.html が表示されましたか？
   「いろいろ〜 Congraturations! いろいろ〜」と画面に出ていたら成功です。

3.3 POP3

HTTP より少し複雑なプロトコルとして、「メールを取り込む」プロトコルPOP3 を見てみることにします。これは Mozilla Thunderbird や Microsoft Outlook などのメールソフト (メールリーダ、メーラー) が裏側で行なっている動作の説明です。

POP3 は Post Office Protocol Version 3 の頭文字で、インターネット標準規格 0053 (RFC 1725 と RFC 1939) です。

POP バージョン 1 は 1984 年というインターネット初期に設計された由緒のある古いプロトコ

命令	引数	概要
USER	ユーザ名	
PASS	パスワード	
QUIT		
CAPA		機能一覧を表示
STAT		統計情報を表示
LIST		メールの一覧を表示
RETR	数字	「数字」番目のメールを取り寄せる
DELE	数字	「数字」番目のメールを消す

表 3.2　POP3 プロトコルコマンド (一部)

応答	引数	概要
+OK		もしあれば、情報つき
-ERR		もしあれば、情報つき

表 3.3　POP3 プロトコルの応答コード

ルですが、今でも広く使われています。いわゆるデファクトスタンダードです。

同時期に設計されたメール取り込みプロトコルとして、POP3 の対極にあたるIMAP4 (Internet Message Access Protocol, Version 4) というプロトコルがあります。商用インターネット時代になっても、しばらく IMAP4 は POP3 ほど広くは使われていませんでした。スマートフォンの時代になり、やっと IMAP4 の前提条件が普通になりました。今では広く使われています。ですが、IMAP4 は非常に複雑なプロトコルなので本書では割愛します。

伝説によれば、初代の POP サーバは 2 時間で書けたそうです。
確かに、最低限の機能だけをもつプロトタイプのサーバなら 2 時間でかけるのは間違いありません。それくらい簡潔なプロトコルです。
ぜひ、君も (プロトタイプでいいから) 作ってみよう。
→ 演習問題

3.3.1　プロトコル

- クライアント → サーバへのリクエスト: 四文字 + 必要な情報
  四文字コマンドの例は表 3.2 を参照してください。
- サーバ → クライアントへの返事: 情報、メールそのもの、ステータス
  ステータスは 2 つしかありません (表 3.3 参照)。

用例:

>サーバからの返事
<クライアントからサーバへ送り込む命令

以下に POP3 プロトコルの様子を示します。ただ、実際にメールソフトが実行している様子を元にしているため、最小構成ではありません。そのため少し長いですが我慢して読んでみて下さい。

最初は「挨拶」から始まります。

```
< +OK Qpopper (version 4.0.5) at
    メールサーバ starting. <12600.1081091433@メールサーバ>
> CAPA
< +OK Capability list follows
< TOP
< USER
< LOGIN-DELAY 0
< EXPIRE 0
< UIDL
< RESP-CODES
< AUTH-RESP-CODE
< X-MANGLE
< X-MACRO
< X-LOCALTIME Mon, 5 Apr 2004 00:10:33 +0900
< IMPLEMENTATION Qpopper-version-4.0.5
< .
```

この挨拶の部分は付加情報や制御情報が交換されるフェイズです。ここは、おまけ部分といえます。

次にメールソフトが POP3 サーバへログインします。ここからがメインの部分です。

```
> USER ユーザ名
< +OK Password required for ユーザ名.
> PASS パスワード
< +OK ユーザ名 has 1 visible message (0 hidden) in 434 octets.
> STAT
< +OK 1 434
> LAST
< +OK 0 is the last read message.
> LIST
< +OK 1 visible messages (434 octets)
< 1 434
< .
```

POP3 サーバへのログイン操作をメールソフトが裏側で行なっています。

なお、メールソフトがユーザに対してパスワード入力を要求するのは、この PASS 命令をサーバ

へ送信する際です。

次は「1 番目のメールを取得し、メールサーバから削除する」例です。

```
> RETR 1
     ... 略（メールが表示されている部分）...
< +OK 434 octets
> DELE 1
< +OK Message 1 has been deleted.
```

メールが複数あるなら、2 番目 (RETR 2)、3 番目 (RETR 3)、…と、N番目のNを変えながら同じ操作を繰り返します。

作業を終えたら POP3 サーバからログアウトして終了です。

```
> QUIT
< +OK Pop server at メールサーバ signing off.
```

3.3.2 まとめ

簡単でしたね？

3.4 サーバ設計思想の変遷

> どんな理論にも「適応可能な範囲」があり、与えられた環境においてのみ「妥当性」がある。

少し脇道にそれますが、せっかくなので「サーバ設計思想変遷の歴史」について説明しようと思います。

サーバや OS の設計 (design) には OS やサーバの設計技法からプログラミング言語まで、実に多くの事柄について理解している必要があります。 もっと腕の立つレベルになると、デザイナーではなくアーキテクト (Architect) と呼ばれる人になりますが、David Cutler [*1] クラスじゃないとアーキテクト “様” と言ってはいけないものでしょう。

それはさておき、サーバ設計思想変遷の歴史に戻ります。

POP3 を使うメール環境はサーバクライアントモデルを実現した典型例です。

時代は移り変わり、POP3 が否定され IMAP4 がもてはやされたり、また流れが逆になり POP3 が優勢になったり、さまざまに変遷してきました。

歴史はスパイラル。“少しだけ違う”[*2] 同じような場所へ定期的に戻ってくるようです。

3.4.1 POP が想定していた環境

POP1 (POP version 1) が設計された時代 (1980 年代前半) とは次のような環境でした。そして、この環境においては、サーバクライアントモデル、POP1 の設計が当然であり、IMAP は数十年先を見ていた進歩的すぎた設計でした。

- コンピュータの性能は現代のコンピュータに比べ非常に非力です
  - 速さは現代のマシンの数千分の 1 です 。ちなみにマシン速度の基準になっている 1 MIPS という単位は、この時代の代表的なマシン (VAX) の速度が基準となっています。ちなみに、現代の CPU は数万 MIPS クラスです。
  - メモリは十数Mバイト程度でした。単位を良く見てください。M (メガ) です。
  - ディスクは数十Mバイト程度でした。単位を良く見てください。M (メガ) です。
- サーバは、(当時としては) かなり強力なマシンですが少数でした。

[*1] 元 DEC で VMS の設計チーム、のちに Microsoft へ移籍しました。

Windows NT のメイン設計者です。もっとも Cutler の手によると言えるのは NT バージョン 3 だけなので、NT 3.51 信者というのが存在します (本当)。

ちなみに David Cutler については「闘うプログラマ 上・下」(日経ＢＰ) を参照してください。プログラマの必読書ですね。

図書館にあります。

[*2] “少しだけ違う” という、ここが大事です。良く似た状態が定期的に再現されているように見えますが、全く同じ環境へ戻ってくることなどありえません。

サーバは高価なため、なかなか買えません。学科に一台というのは普通でした。

メールサーバを動かすには、かなりのパワーが必要です。たとえば、速い CPU、大容量のメモリとディスクが必要になります。しかしながら、そういったサーバは高価なため、あまり買えません。

たとえば、当時の代表的なマシン DEC VAX11/780 は、“安くても” 今の価格に換算して 1000 万円くらいはしたはずです。

それでも VAX が高いとはいえ、スーパーコンピュータのように大学で一台などということはなく、学科で一台くらいならなんとか買える値段でした。

DEC VAX シリーズの一つ前の傑作 DEC PDP シリーズも同じような値段でしたが、そういったマーケット向きの製品を作ってきた会社が DEC です。DEC がなければ、現在の Unix とインターネットは無かったでしょう。

偉大な会社でした (注: DEC は COMPAQ に買収され、COMPAQ は HP に買収された)。

- クライアントは、そこそこの性能で、一人に一台くばられます。

  ちなみにクライアント PC ではありません。この頃のクライアント向けコンピュータはワークステーションと言いました。

  1980 年代前半には SUN [*3] に代表されるワークステーションメーカーが現れ、性能 (CPU もメモリもディスク) もそこそこ、値段もそこそこというコンピュータが売られるようになりました。

  これらのコンピュータは、サーバとして使おうとすると厳しいスペックですが、個人の端末としては十分でした。これが、ワークステーションと呼ばれている機材です。代表的なマシンが SUN 3 で、値段は現在の感覚で 200 万円相当だと思います。

この前提条件は、特に価格性能比 (コストパフォーマンス) という点ではだいぶ変わりましたが、現在でも同様にサーバ用とクライアント用という PC が売られています。

そのため、サーバクライアントモデルが今でも多用されているわけです。

このモデルでは、いろいろなサービスや資源 (ディスクなど) がサーバに集中しています。よって、サーバが壊れると事実上ネットワークが使えなくなります。

そのため、サーバ用機材として高価なハードウエアを購入し、さらに、サービスの中断を防ぐために機材の冗長化やクラスタリングといった構成を考えるのが普通です。

逆に、クライアント PC は安価なハードウエアでかまいません。

低機能なディスクすら持たないクライアントを、最近ではシンクライアント (thin client) と呼んだりします。

現在では PC 自体が安い[*4]ため、シンクライアントが流行していませんが、シンクライアントの究極は SUN の Sun Ray でした。これは液晶とキーボード、マウスしかない端末で、実体はサーバ側にあります。つまり Sun Ray 1 はサーバ側で行なっている作業の画面を表示しているだけの家電製品です (突然電源を落しても大丈夫です)。

[*3] あ～ Java の会社か？とか言っている君は若過ぎです orz 偉大なベンチャー企業だったんですぞ。SUN を知っているか？と聞かれたら、「SUN といえば面白いほどよく落ちる Sparc ですよね？」と答えてみましょう。面接官にウケルこと間違いなしです！なお、上の格言 (?) の意味については Junet という単語を知っている年寄りを見つけて聞いてみてください。

[*4] 中古の数千円の PC で十分な速度、新品なら数万円の PC でも速くておつりが来る

3.4.2 サーバクライアントモデルの利点と弱点

すべての理論には適用可能な範囲があります。当然、サーバクライアントモデルにも、モデルがうまく適用できる場合とそうでない場合があります。

つまるところ、モデルの利点と弱点です。

これは、立場によって、利点と弱点が入れ替わり、どちらが正解というものではありません。

ただ「ほとんどすべての人はコンピュータ (OS やサーバの中身) に詳しくないし、まめな管理もしない (できない)」という条件が成立し続けているので、サーバクライアントモデルを使うことには、いつの時代でも利点があったし、これからもそうに違いない、と言えると思います。

いつか、コンピュータが家電なみになる時が来たら変わるでしょうが、当面、そうなる見込みは薄いようです。

なにはともあれ、いくつかのケースを検討してみましょう。それぞれのケースについて利点と弱点、このモデルが流行する環境と廃れる環境について考察してみて下さい (もちろん、どの解答も正解といえてしまうとは思います)。

3.4.3 サーバクライアントモデルの利点と弱点: 管理コスト

工業化が進んだ社会では、コストの主要な部分は人件費です。

「管理コスト」という点から言えば、サーバクライアントモデルは正しいモデルです。利点は欠点を凌駕しています。

そのため管理者という特殊技能職の人件費をいかに抑えるか？が主要な課題です。

利点

サーバクライアントモデルでは

> 面倒をみる必須の機材が少数のサーバ

だけでよいということになります。

これは貴重なサーバ/ネットワーク管理者という人的資源 (==コスト) を最小に出来るモデルです。よってコストの面で最も優れています。

面倒を防ぐため、クライアント PC も、どうしても必須のものだけを除いて、特別なアプリケーションなどを入れてはいけません。

たとえば PC 教室の PC は、素の状態の CentOS に「Firefox + Flash と Java のプラグイン」をインストールすれば、普段の使用には十分です。一方、ポータルや E ラーニングシステムの複雑な部分は、すべてサーバ側で持っている機能です。

これ以上ふつざつにしてはいけません。コストが跳ね上がります。

コラム: アウトソーシング

管理能力を他の会社から買うということも可能です。これはアウトソーシングと呼ばれ、日本企業が lay off [a]にあけくれた 1990 年代以降、流行しています。

その一方でアウトソーシングしすぎて、つまり外部に “丸投げ” し過ぎて、何かあった時に誰も社内システムが分からないといった “祭り” が起きている会社もあります。

ま、結局、管理職が駄目という話なのですが、これまたよくある話らしい。

どっちにしても困ったものです。

[a] リストラなどといったマスコミに踊らされたカタカナ英語を使うのはやめましょう。日本企業はリストラクチャリング (←これは本来はちゃんとした経営戦略用語) なんてしていない。あれは、単なるレイオフ (馘首切り)。

弱点

クライアント側に何の問題がなくても、サーバ側で障害が起きると何もできなくなります。

これは説明しなくても、どういう状態か分かりますね？;)

また、管理者の側から理想をいわせてもらえば、クライアント PC をユーザにいじられたくもありません。

クライアント PC にどんなソフトウエアを入れても初期化できる製品というのもあるのですが、Windows が相手の場合、どんな製品を使っても面倒です (コラムも参照)。

正直、Windows なんて使いたくありません。

コラム: Mac OS X サーバ

Mac OS X にはサーバから OS のコピーを送り込める仕組みがあります。別売ではなく、標準搭載です。ここが話のポイント。

クライアントは起動する際にサーバへリクエストを送り、OS のイメージをダウンロードする仕組みと考えて下さい。

一昔前ではこんな仕組みは考えられませんが、これだけネットワークが高速になると、OS の CD-ROM イメージを毎回ダウンロードさせるのも “アリ” となります。

しかも、別途有料ではなく Mac OS X サーバには初めから装備されている仕組みです。

こんな素敵な OS なのに、なぜ大学のコンピュータセンターに Mac OS X が並んでいないのか？というと、Apple が保守サービスの仕組みを作らないから、ただそれだけのことです (保守できる SI 業者がいない)。

世の中、技術問題により…ではない障壁がたくさんあります。これは、そんな話題の中の一つです。

3.4.4 サーバクライアントモデルの利点と弱点: 自由度

現代では数万円の PC や数千円の二世代くらい前の中古 PC ですらサーバとして十分な機能を持っています。ですから、サーバとクライアントという区別に意味はなく、我々は、すべてがサーバと言ってもよい世界にいるわけです。

よって、すべての PC がサーバであり、だれもが自由にサーバを作り、情報を発信することが可能です。こういった環境が P2P (Peer to Peer) モデルと言えます。

この「自由度」を前面に押し出した観点から見れば、サーバクライアントモデルは不利なモデルです。

たしかに実際に、こういった環境が実現しているわけですが、サーバ化した PC でウィルスに感染していたり悪事に荷担しているものが無数にあります。

どこの国とは言いませんが、インターネットには不正 Windows が何百万台もつながっており、それらは crackers (他人のコンピュータを不正利用する人たち) にのっとられ、攻撃や SPAM の送信に使われています。

ここで、話は前の問題に戻ります。「管理能力があるユーザは限られているため、P2P な世界は成立しない」ということです。

最新の OS といえど家電製品には遠く及びません。素人には OS の面倒は無理です。

3.4.5 サーバクライアントモデルの利点と弱点: アプリケーション

管理コストから考えれば、サーバクライアントモデルは正しい。すべてシンクライアントに移行するべきである。

弱点

シンクライアントにしてもいいけど、Microsoft の office とか powerpoint は、ちゃ

んと使えるの？

毎回、顧客に聞かれるセリフです。

シンクライアントでは、しょせん画面が出ているだけなので、office や powerpoint を動かす場合、サーバ側でプログラムを動かして、その画面を表示することになります。

よって、PC 教室などでは、一つのサーバ上で何十個もプログラムが走るわけです。これが激しく重たいのです。よって、だいたいイマイチと言われます。

そんなわけで、シンクライアントは、いまいち流行しません。クライアントも普通に PC を買って、ソフトウエアをインストールすることになります。つまり PC の数だけ Microsoft は儲かるという構図です。

アプリケーションでシェアを取った企業が勝利するというわけ。

3.4.6 サーバクライアントモデルの利点と弱点: セキュリティ

これも前述のように「管理者の能力がある人は限られている」ため、サーバクライアントモデルの方に利点があります。

情報漏洩を防ぐためには、各ユーザに「限定された権限」を与えるべきです。管理もさせなければ、余分のハードウエアも与えません。そして余計な操作を一切させてはいけません。

たとえば、PC には普通 USB インターフェイスがありますが、そんな PC を与えてはいけません。個人情報を PC の画面に表示し、それを USB メモリにダウンロードできるからです。

よってセキュリティを重視するなら、クライアント PC は何の機能も持たないシンクライアントが正しいということになります。

もちろん、抜け道はあります。たとえば携帯電話です。最近の携帯電話はカメラつきなので PC の画面の個人情報をカメラで撮影することが出来ます。よって、セキュリティの厳しい職場では携帯電話も持ち込み禁止が当たり前です。

こういった抜け道は、運用や管理といったレベルで回避するべきテーマといえます。

3.4.7 サーバクライアントモデルの利点と弱点: 資本主義

ハードウエアメーカはサーバクライアントモデルに賛成でしょう。というのは、サーバは強力で高価で利益も高く、クライアント PC の利益は低いからです。

クライアントも昔はまだそれなりの値段でしたが、いまやクライアントは悲しくなるくらい安いです。49,800 でも速すぎるくらいの能力があります。一台売って、いくらの利益も出ません。

つまり、サーバ製品を主力とするメーカは、このモデルが望ましいわけです。

一方、サーバも強力、クライアントもそれなりに強力であるべきだと主張する陣営があります。

それは OS を売る人たちです。といっても、事実上マイクロソフト社だけですが、この陣営はハードウエアの数だけ OS が売れるので、OS の出荷数が重要です。また OS 自体は低価格でもよく、ほぼ抱き合わせ販売のマイクロソフトオフィスを高価格で売りさばけば笑いたいほど利益が出ます。

これがマイクロソフトの基本営業戦略です。

サーバとクライアントといった区別はない方が良く、管理しやすくするためにサーバからクライアントへ OS のコピーを送る (前述) といった仕組みを売ることもありません。いえ、“ありえません”。

一方、Apple がマイクロソフトと違う戦略がとれるのは、「ハードウエアと OS 両方をセットで販売している」ことと「Mac OS X の多くがフリーの Unix ベースである」ため、そしてブランドの力です。

ブランディングも一種の宗教ですが、Apple のハードウエアはリビングにおいてもよいと思えるデザインセンスであることも事実で、そのへんは

ソフトウエア専業のマイクロソフトとは一線をかしています。

このどちらでもない第三の選択もあります。かつて、ネットスケープ社が進もうとしてマイクロソフトに会社ごと潰され、今、グーグルが再び進もうとしている戦略です。

それは

> ユーザが使っているのは WWW とオフィス製品だけである。サーバ側であらゆる機能を提供できるなら、ユーザは WWW ブラウザ一つあれば OS など不要であろう。

つまり、WWW ブラウザさえあれば何も要らないという道です。

WWW ブラウザがあれば、OS やハードウエアは何でも良い。これは現在の利権を握っている旧勢力が没落するシナリオです。当然、この路線を広めようとした会社は全て旧勢力に叩き潰されました。

世界的に景気後退の中、グーグルの経営も微妙ですが、今、マイクロソフトと闘える資本があるのはグーグルだけです。

3.4.8 まとめ: 歴史的展望

さまざまなサーバクライアントモデルの利点と弱点を見てきましたが、これらのどの論点も、ある時代には讃えられ、ある時代には軽視されて来ました。

どちらが正しいと言うことはありません。その時の予算や既存の構成などに応じた最善の提案があるだけです。

○経××などの雑誌を筆頭に、いろいろたわごとが流布しますが、そういったことにまどわされないようにしよう。

3.5 SMTP

最後に、もう少し複雑なプロトコルを紹介しましょう。

本章の最後を飾るアプリケーションプロトコルは、40 年近く生き延びてきたメール転送プロトコル、SMTP (Simple Mail Transer Protocol) です。

SMTP は Simple という名の通り、いたって簡単なものです。POP3 (3.3 節参照) とさして変わりません。もっとも、その Simple さゆえに SPAM を許してしまっているともいえますが…

3.5.1 想定

以下では、次のようなメールのやりとりを考えます (図 3.2)。

- 送信者のメールアドレスは rudo@nuinui.net つまり nuinui.net ドメインのユーザ rudo さん。
- 受信者のメールアドレスは kenken@fml.org fml.org ドメインのユーザ kenken さん。
- 通信内容 (メール本文) は「ミーティングのお知らせ」

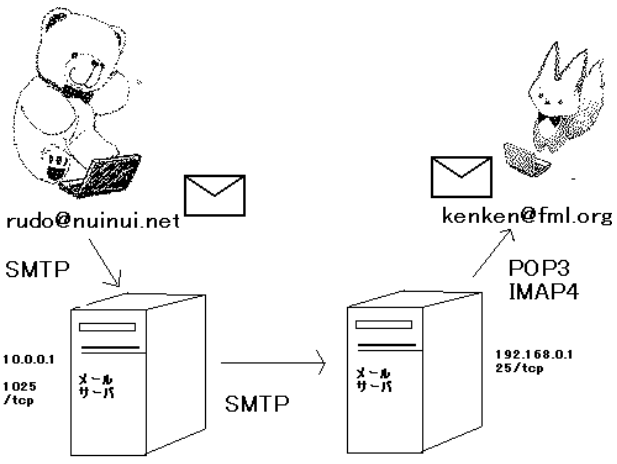

図 3.2 メールの送信、概念図

3.5.2 メールアドレス

あらためて説明する必要はないでしょうが一応説明します。なお、正確な定義は RFC を参照してください。

メールアドレスは b999999@photon.chitose.ac.jp といった

「ユーザ名」と「ドメイン名」を @ でつないだ文字列です。

この例ではユーザ名が b999999 です。通常、この文字列はコンピュータにログインする際に使うユーザ名 (ログイン名) と同じです。photon.chitose.ac.jp はドメイン名 (くわしくは次回 DNS で取り上げます) と呼ばれます。

ユーザ部分は大文字小文字を区別します (case sensitive)。つまり b999999 と B999999 は違うユーザです。

と、RFC には書いてあるのですが、sendmail という最も古くから使われているデファクトスタンダードなメールサーバソフトウエアが大文字小文字を区別しない動作をするため、事実上大文字でも小文字でも同じに扱われる場合がほとんどです。

一方、ドメイン名は大文字小文字を区別しません (case insensitive)。photon.chitose.ac.jp も PHOTON.CHITOSE.AC.JP も同じ意味になります。これは RFC で定義されているとおりです。

3.5.3 SMTP の概要

1. ユーザがメールソフトを起動し、
2. メールを編集します。
3. ユーザは、メールを送信します。
   メールソフトは、最寄りのメールサーバへ SMTP で接続します。
4. 最寄りのメールサーバ → 相手のメールサーバSMTP
   本節では、この部分を考えています。最寄りのメールサーバが 10.0.0.1、相手のメールサーバが 192.168.0.1 としましょう。
5. 送った先のユーザがメールを読みます。
   メールサーバから POP3 や IMAP4 でメールを取り込み表示します。ここが前節 (3.3 節を参照) にあたります。最近では、ブラウザで読む場合も多いですね。

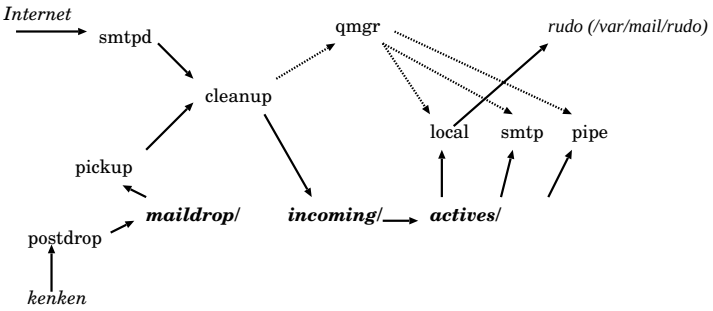

図 3.3 Postfix の内部構造: 本書では SMTP の概略だけを説明していますが、本物のメールサーバの内部構造は、これくらい複雑です。確実なメール転送を行なうために必要な要素は無数にあり、また安全なソフトウエアの設計というのも非常に難しいものです。なお、C 言語の勉強をするなら、この Postfix のソースコードがお薦めです。

命令	説明	備考
HELO ホスト名	自己紹介	
MAIL FROM:<メールアドレス>	送信者	
RCPT TO:<メールアドレス>	受信者	複数可
DATA	データ開始	
.	データ終了	
QUIT	SMTP 終了	

表 3.4 SMTP コマンド

コード	説明	備考
220	rudo.home.fml.org ESMTP Postfix	
250	ok	
354	Start mail input; end with <CRLF>.<CRLF>	
452	Requested action not taken: insufficient system storage	
554	Transaction failed	

表 3.5 SMTP リプライコードの例: サーバは数字 (+ 人間が読める文章) で返事 (コード) を返す

3.5.4 メールの転送

挨拶

最初は挨拶から始まります。

挨拶に対するサーバからの返事として「サーバがサポートしているオプション」が教えてもらえます。

```
[用例]
>     メールサーバ（受信側）からの返事
<     メールサーバ（送信側）からの要求


>     220 mta.fml.org ESMTP Postfix（サーバ側の挨拶から開始）
<     EHLO client.nuinui.net
>     250-mta.fml.org
>     250-PIPELINING
>     250-SIZE 10240000
>     250-ETRN
>     250 8BITMIME
```

220 の行はサーバ側から送られてきたものです。SMTP では TCP 的にサーバへ接続が確立した際に「サーバ側からの挨拶」で始まります。

「EHLO 自分のホスト名」がクライアントからの挨拶に相当します。挨拶を送るとオプション等が教えてもらえます。オプションは参考にするだけです (使っても使わなくてもかまいません)。

250 のあとにスペースが続く行がサーバ側からの最終行です。Unix ではスペース区切りというテクニックを多用することを思い出して下さい。250- の行は「スペースで区切らない」ことで継続行があることをクライアントへ教えています。

250 のあとにスペースがあることで返事の最後であることがクライアントに分かります。

送信者と受信者の情報を渡す

誰から誰へメールを転送したいのかをサーバへ教えます。

```
<     MAIL FROM:<rudo@nuinui.net>    (rudo@nuinui.net が送信者だ)
>     250 Ok
<     RCPT TO:<kenken@fml.org>       (送りたい先は kenken@fml.org だ)
>     250 Ok
```

リクエストに問題がなければ 250 が返ります。Ok の部分は人間が分かりやすいように表示しているだけでプロトコル的には無関係 (クライアントが見ているのは 250 のところだけ) です。

データを渡す

SMTP 的にデータを渡すというのは、メール (822 形式のテキストファイル) を相手に渡すことです。このデータはユーザがメールソフトで編集した画面 ($+\alpha$) に相当します。

DATA 命令を送り、サーバへ本文を渡すことを宣言します。サーバから 354 が返ってきたらサーバが本文を渡し始めてよいという合図です。データ (メール) をサーバへ渡し、. で終了を知らせます。250 が返れば、サーバが無事に受けとったという意味です。

```
<     DATA
>     354 End data with <CR><LF>.<CR><LF>
<     From: rudo@nuinui.net
<     To: kenken@fml.org
<     Subject: Todays's meeting
<     Mime-Version: 1.0
<     Content-Type: text/plain; charset=iso-2022-jp
<
<     今日のミーティングは 15:00 からだ
<
<     るど隊長@ぬいぬい
<     .
>     250 Ok: queued as 9BB9F86640
```

メールソフトの画面はユーザに優しくなっており、生ヘッダが表示されていません。

(通常ユーザに見えていない) インターネットメールの実体は、このように「識別子:値」形式で情報を並べたヘッダ (制御情報)、空行、本文が並べられたテキストファイルです。: 区切り、空行区切りといった Unix 王道のテクニックが満載であることに注意して下さい。

また、HTTP ヘッダも同じ形式でしたね？ (3.2.5 節参照)

終り

礼儀正しく SMTP を終了します。

```
<     QUIT
>     221 Bye
```

3.5.5 SMTP のまとめ

状態

他のプロトコルと同様に、 SMTP はコマンドを送りステータスコードを受けとります。そしてステータスコードによって、適宜、対応を変えます。

状態 (state) があるわけです。

SMTP の重要な点

- わりと単純です
  しかし、動きます！だてに長く使われてきているわけではありません
- 人間が手で再現できます
  プロは、この動作を手で再現できなければいけません。デバッグや障害対応が楽になりま

す。これはプロトコル設計において重要なことです。

- プロトコル設計における典型的な見本として重要です。
  類似プロトコルがたくさんあります。SMTP が出来れば、ほぼ何でも分かるといっても過言ではないでしょう。

簡潔さ

SMTP 自体には複雑な処理が定義されていません。

たとえば、SMTP 処理の途中にエラーが起きたらどうなるでしょうか？エラーがあったことを教えるステータスコードは定義されていますが、そのあと、どのように再送処理を行なうのか？といったことはメールサーバの設計者に巻かされています。

```
エラーの例:
452 Requested action not taken: insufficient system storage
```

```
Q: 192.168.0.1 のサーバが受信に失敗したら？
A: 10.0.0.1（送信側メールサーバ）が再送処理を頑張ります。
```

10.0.0.1 (送信側メールサーバ) は、452 というコードを受けた場合、即座に SMTP 処理を中断し、送信に失敗したメールを自分でしばらくあづかることにします。これを「キュー (queue) に入れる」などと呼びます。

何分かしたら、もう一度 SMTP の再挑戦をします。それにも失敗したら、もう少し長く時間をおいて再挑戦を行ないます。以下、繰り返し。

何分おきに何日まで挑戦するか？はメールサーバの設計者次第です。ただ、多くの場合、メールサーバソフトウエアは sendmail のパラメータを真似るようにしています (例: 最大五日間再配送を試みる)。

第 4 章

DNS

今回は、第 3.1 節で省略したDNS (Domain Name Service) について取り上げます。

DNS は名前解決を行なうシステムです。

インターネット技術の中で、ほぼすべてのアプリケーションが使う最も重要なサービスの一つが DNS です。

4.1 前回までのあらすじ

よくよく考えてみると今までの話には大きな飛躍があります。

まず、コンピュータの動作は二進数ベースです。そして、コンピュータは互いの住所を IP アドレス、つまり数字として認識していると説明しました。

たとえば (二進数を四桁ずつに分けて表示すると)、コンピュータＡは 1100 0000 0000 0000 0000 0000 0000 0001、コンピュータＢは 0000 1010 0000 0000 0000 0000 0000 0001 です。コンピュータどうしは互いをそういった数字とみなしています。

この数字は一種の住所とみなせるため、この数字つまり IP アドレスを住所と思って下さいと説明しました。

一方、人間は、こんな長い数字を覚えられないので「WWW ブラウザに http://www.chitose.ac.jp/ が見たい」とか「b999999@photon.chitose.ac.jp へメールを送信」といった具合にコンピュータに指示を出しています。

明らかに、これらは全く異なる表現形式です。

ということは、システムのどこかで「www.chitose.ac.jp というコンピュータは 1101 0010 1000 0000 0011 0100 0010 0000 という数字 (IP アドレス、つまり住所)」であると翻訳してくれる仕組みがないといけません。

この仕組みがDNS (Domain Name System) です。そして、この人間が使う文字列 www.chitose.ac.jp とコンピュータの使う数字 1101 0010 1000 0000 0011 0100 0010 0000 との間の変換を名前解決 (e.g. resolve domain names) と呼んでいます。

今回は DNS の仕組みについて概観し、アプリケーションが DNS をどのように使っているかについてを考えてみることにします。

4.1.1 凡例

特に断らないかぎり、通信は次のようなものとします。

- サーバの IP アドレス = 192.168.0.1
- サーバのポート番号
  アプリケーションごとに異なるので、そのつど説明します。
- クライアントの IP アドレス = 10.0.0.1
- クライアントのポート番号
  通信のたびに OS が適当に番号を割り振ります。本教科書の図表では、分かりやすくするために 1025 番を使うように統一してい

ます。

- 通信方式はUDP がデフォルト。
  今回は UDP です。注意！

変数	例	備考
メールアドレス	b999999@photon.chitose.ac.jp	@ の右側がドメイン名
URL	http://www.chitose.ac.jp/	:// 直後がドメイン名

表 4.1　ドメイン名を使う例

4.2　DNS の概要

DNS は名前解決のための仕組みです。

ほぼ全てのアプリケーションが必ず使うインターネットで最も重要な技術の一つで、まさにインターネットの基盤技術と言えます。

また、歴史上最も成功した分散システムとして DNS の仕組みを知ることは工学的センスをみがく上で重要です。

そして DNS は主にUDP (第 6.1 節参照) を使う例としても重要といえます。というのも UDP をメインに使うシステムは、現在のインターネットでは少数派なのです。

4.3　ドメイン名

4.3.1　IP アドレスから名前へ

コンピュータは IP アドレス (例: 192.168.0.1) を 32 ビットの二進数の数字として扱うのが便利だと思っています。一方、人間は、とてもこんな長い数字を覚え切れません。1、2台なら覚えられなくもありませんが、数十台あったらもはや無理でしょう。

そこで「この PC はメールサーバだから mail という名前をつけよう」と思うわけです。そして、コンピュータには「mail という名前は 192.168.0.1 と翻訳したまえ」と教え込みたいわけです。

インターネット黎明期、インターネットではなく ARPANET と呼ばれ、接続しているコンピュータが数十台だった頃なら、このやり方でも問題はありませんでした。実際、このやり方をしていました。

しかし、mail という安直な名前の付け方をしていれば、すぐに名前の重複が起きることは自明ですね？

というわけで、より計画的な名前の付け方をすることにしました。

それが DNS で使うドメイン名です。

4.3.2　ドメイン名

PC につける名前は「マシン名．組織名．組織種別．国」といった構造を持った形式とします。正確には、アルファベット、数字、– (ハイフン、語をつなげる短いマイナスのような記号) からなる文字列をピリオド (.) でつないだ文字列です。

この文字列はドメイン名と呼ばれています。歴史上のしがらみがいろいろあって話がややこしい (コラム参照) のですが、

> すべてのコンピュータには home などといった安直な名前ではなく、mail.chitose.ac.jp のように「マシン名．組織名．組織種別．国」を全て表現したドメイン名をつける

と覚えて下さい。

この「PC につけるような長い」ドメイン名は FQDN (Fully Qualified Domain Name、日本語では「完全修飾されたドメイン名」) と呼ばれています。PC に mail などと短い名前をつけるのは DNS 的には間違いです。

> コンピュータには FQDN をつける

と覚えて下さい。

4.4　名前解決

4.4.1　正引きと逆引き

IP アドレスとドメイン名の間の変換は名前の解決 (resolve の訳) と呼ばれています (面倒なので、そのままリゾルバと呼んでしまうこともあります、でも英語としてはオカシイね)。

そして、ドメイン名から IP アドレスへの変換は正引き、IP アドレスからドメイン名への変換は逆引きと呼びます。

正引き

```
210.137.170.32 <- www.chitose.ac.jp
```

逆引き

```
210.137.170.32 -> www.chitose.ac.jp
```

つまり、DNS に www.chitose.ac.jp の IP アドレスが知りたいと尋ね、解答として IP アドレスをもらう場合が正引き、その逆で IP アドレスを尋ねてドメイン名を知るのが逆引きです。

4.4.2 アプリケーションと名前解決

では HTTP (第 3.2 節) を例にとり、第 3.1 章で省略した DNS (Domain Name Service) の部分を説明しましょう。

大雑把に見れば、WWW ブラウザの動作とは次のようなものです。

1. WWW ブラウザに URL(例: http://www.chitose.ac.jp/) を入力します
2. ブラウザが HTTP でコンテンツをダウンロードします
   これが第 3.2 節で説明した HTTP の役割です
3. ダウンロードしたコンテンツをブラウザが解釈して表示します

この 1 番目と 2 番目の動作の間に、ユーザに見えないところで名前解決が行なわれています。

名前解決の動作を付け加えた正しい動作は次のようになります。

1. WWW ブラウザに URL を入力する。
   たとえば URL は http://www.chitose.ac.jp/ とします。
   (a) ブラウザは OS の名前解決システム (リゾルバ) を呼び出し
   (b)「www.chitose.ac.jp の IP アドレスが知りたい」と問い合わせます。
   (c) OS の名前解決システムが 210.137.170.32 と教えてくれます。
   (d) IP アドレスが 210.137.170.32、ポート番号が 80 の PC へ TCP で接続。
2. ブラウザが HTTP でコンテンツをダウンロードする。
3. ダウンロードしたコンテンツをブラウザが解釈して表示する。

なお、この DNS の問い合わせの際、転送方式として UDP を利用しています。もし、うまくいかない場合には UDP でリクエストの再送を試み、それでもダメな場合は転送方式を TCP へ自動的に切替えて試みます。

この後、名前解決について説明しますが、何度も何度も問い合わせを行なうプロトコルなので、動作が軽いことが望ましいと考えられます。よってプロトコル設計時には、処理が重たい TCP (第 5 章) よりも UDP (6.1 節) が望ましいと考えたわけです。

4.4.3 レコード

DNS の設定では各データがレコード (record) と呼ばれています。

A レコード

A はアドレスの A です。

4.4.2 節で説明した動作は、ドメイン名 www.chitose.ac.jp から IP アドレス 210.137.170.32 を知るという動作でした。この動作は「A レコードを引く」と呼ばれています。

これは「IP アドレス」というデータが A レコードと呼ばれているためです。

A 以外にも、さまざまなレコードがあります。

PTR レコード

ポインタ (pointer) の省略形です。

ポインタレコードとはドメイン名データのこと

です。

IP アドレス (210.137.170.32) からドメイン名 (www.chitose.ac.jp) を知りたい時には PTR レコードを検索します。つまり逆引きとは PTR レコードを引くことです。

MX レコード

ドメイン宛メールサーバのデータは MX レコードと呼ばれます。MX は Mail eXchanger の頭文字です。

たとえば、b999999@photon.chitose.ac.jp へメールを送信したい場合、メールサーバは photon.chitose.ac.jp ドメインの MX レコードを検索し、そのメールサーバへ SMTP (3.5 節) によるメール配送を行ないます。

この場合、photon.chitose.ac.jp 宛の MX レコードを検索すると、メールサーバ nurikabe.chitose.ac.jp が教えてもらえます。メールサーバは nurikabe.chitose.ac.jp へ SMTP します。

NS レコード

ネームサーバ (DNS サーバ) のデータは NS レコードと呼ばれます。NS は Name Server の頭文字です。

NS レコードは分散システムとしての DNS および DNS の階層構造において決定的な役割をもつデータです。

この後、詳しく説明します。

4.4.4 演習: 名前解決

CentOS の「アプリケーション」から「ターミナル端末」を開いて、次のコマンドを叩き、問い合わせの答え (下側) を見てみてください (注: 現在は、ソフトウエア工学概論の中で実際にやるので、この節は省略します)。

```
% nslookup www.chitose.ac.jp
% nslookup 210.128.52.32
% nslookup -q=mx photon.chitose.ac.jp
% nslookup -q=ns photon.chitose.ac.jp
```

それぞれ何が表示されましたか？

なお、同じ動作をするが表示が少し異なる dig コマンドも試して下さい。こちらの方が新しいコマンドで便利。なお ; で始まる行はコメントです。

```
% dig www.chitose.ac.jp
% dig -x 210.128.52.32
% dig photon.chitose.ac.jp mx
% dig photon.chitose.ac.jp ns
```

4.5 DNS の階層構造

4.5.1 史上最大の分散システム

前述のように、DNS では各 PC に「組織名や国名をつけたドメイン名」を割り振ります。これにより系統的な検索が出来るようになりました。以下、その理屈を説明しましょう。

まず、地球の人口を考えてみれば分かる通り、一人一台の PC を持っているとすると、何十億台分のドメイン名があり、それぞれに IP アドレスがあるわけです。

DNS は、これらの何十億レコードのデータの間で変換を行なうシステムです。では、その設定は？更新は？維持管理は？どうすれば良いのでしょうか？

何十億台分の変換表を持つ “一つのマスターファイル” を作って管理しますか？数十台規模ならともかく、数十億エントリもある巨大なデータベースを一つのマスターファイルで管理は出来ないでしょう。もっとも現代のマシンパワーでは出来てしまうかもしれませんが、当時のコンピュータパワーでは到底無理です (参考: いま、みなさ

んが普段使っているコンピュータは、当時のサーバ機材の一万倍くらいの速度です)。

そこで、DNS を設計する際には、階層構造を持つドメイン名を採用し、次の二つの問題を同時に解決しました。

1. 何十億台にも対応可能。
2. 分散管理が可能。

設計した当時には、よもやインターネットがこれほど巨大になるとは思っていなかったと思うのですが、現在でも DNS は動作しています。しかも歴史上、世界規模で動作しているほぼ唯一の分散システムです。

見事な設計でした。

俗にドッグイヤーといわれるインターネット業界は世間の 6 から 7 倍速くらいで動いている気がします。半世紀前に現在を見通して設計したものが正しかったというのは、江戸時代に 21 世紀の世界を予想して見事に的中させたくらい凄いことです。そのくらい偉大なシステムだと思ってください。

4.5.2 分散管理

分散管理とは管理責任の委譲 (delegation) を意味します。

ドメイン名が表しているものは組織構造です。その構造の各部分部分ごとに、つまり各組織ごとにデータベースを分散 管理してもらう仕組みを考えます。

www.chitose.ac.jp を例にとりましょう。この文字列を意訳すると「日本にある、学術組織の、千歳科学技術大学の、WWW サーバ」という意味になります。ドメイン名は逆順に読むのがポイントです (アメリカ人が考えたからね、日本とは住所の書き方が逆なの:)。

というわけで、「日本にある、学術組織の、千歳科学技術大学の、WWW サーバ」は

- まず「日本」の管理下の、
- その下位組織「日本にある学術組織」の、
- そのまた下の組織「日本にある学術組織の千歳科学技術大学」管理下の
- WWW サーバ

という意味になります。

この管理権限の細切れ化を実現するために、次のようにします。

- 分散管理を実現するために、データを階層構造 (木構造) 化 (図 4.1)。
- DNS のサーバは各組織ごとに作る (図 4.2)。
- DNS のデータは各組織ごとに自分の分だけを維持管理する。

つまりインターネットにはドメインの数だけ DNS サーバが存在します。数百万台必要だろうが必要なだけサーバを作る必要があるわけです。

もっとも実際には一つのサーバが複数のドメイン分の仕事をするので、サーバの実数はドメインの総数に比べればかなり少ないです。そして自分で作らなくてもプロバイダからサービスを購入することも出来ます。

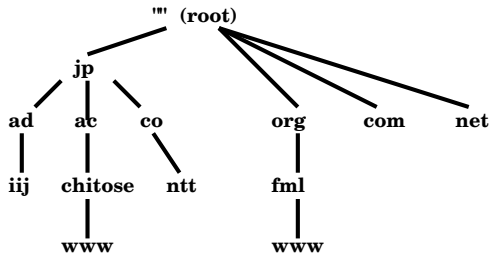

図 4.1 木構造

4.5.3 木構造

図 4.1 のような構造を木構造 (tree structure) と呼びます。

図の一番上、頂点部分は空文字列 ("") です。ここは root (根[*1]) と呼ばれています。一番上が

[*1] 図の一番上にある部分を「根」と呼ぶのは逆では？と

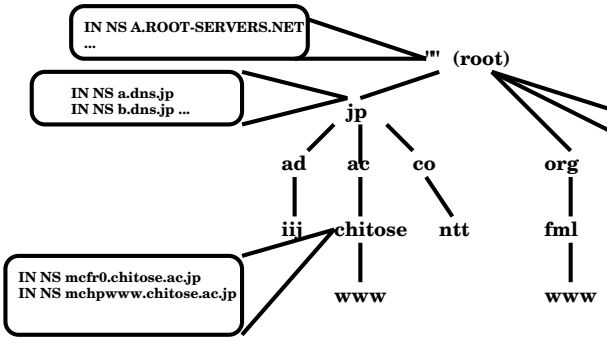

図 4.2 木構造の各レベルに複数の DNS サーバが配置されています

もっとも大きな組織ですが、ここの "" は仮想的な組織と考えて下さい。

二番目に大きな組織は国名 (たとえば日本は jp) か特別なサービス (例: info や tv など) です。最近はもっと大きな範囲であるアジア asia なども出来ました。この階層が現実社会との対応を考えると一番大きな単位ですので、これらをトップレベルドメイン (TLD) と呼びます。

アメリカは国名ドメインの us 以外に、国名のつかない gov edu org com net も使っています。最初にインターネットを始めた国の特権です。

なお、1990 年代半ばに org com net は誰でも利用可能となったので、お金さえ払えば誰でも org com net ドメインを取得することが可能です。

よって org com net ドメインについては、ドメインの取得者やそのドメイン組織がアメリカとは何の関係もないことが多いです。たとえば fml.org がそうですね？

なお、これらのドメインの取得が容易になった頃、この 3 つのドメイン取得ブームというのがありました。企業は、より短いドメイン名を求めて「会社名.com」を欲しがりました。このブームの時に「.com」(ドットコム) という単語が生まれました。

日本のドメイン屋の大手に「お名前ドットコム」という会社がありますが、そのドットコムという単語がソレです。今となっては恥ずかしい感じ ;-)

4.5.4 木構造と名前解決

図 4.2 のような木構造の各部分部分、つまり各組織に DNS サーバがあります。

名前解決とは、この木を上 (root) から下に向かって検索していくことに他なりません。

木構造と正引き

「www.chitose.ac.jp の IP アドレスを検索する (A レコードを引く)」を例にとりましょう。

図 4.2 を見て下さい。

どんな時でも、DNS の検索は一番上の階層から始まります。つまりルートの階層です。この階層のネームサーバ群をルートサーバ (root server) と呼びます。

このサーバ群は全世界から常に検索され続ける運命にありますが、DNS の仕様上 13 台しか作れません (演習問題も参照)。よもや DNS がこんなに巨大なシステムになるとは予想していなかったのでしょう。間違いなく設計ミスと思われます。

ルートサーバへ「www.chitose.ac.jp を知っているか？」と尋ねると、「自分は知らないけど、jp のことは a.dns.jp に聞いて見なさい」という答がもらえます。実際には解答の際に複数の候補を教えてもらえるので、問い合わせ側で適当に一つ選んで問い合わせを送ります。

この時に教えてもらえる「jp の問い合わせ先」が NS レコードに登録されている (複数の) サーバ名です。

さて、jp ドメイン管理側も、jp ドメインあての問い合わせを受けるために jp ドメインのネームサーバ群 (a.dns.jp 他) を用意しておきます。そして、この問い合わせ先 (a.dns.jp など) をあらかじめ上位のネームサーバ (この場合はルート

いう疑問があるかもしれませんが、この用語は植物の「木」ではなく数学のグラフ理論に由来しているので間違っていません。

サーバ) に教えておかなくてはいけません。そうでないと上位のサーバが解答しようがありませんから。

この「上位の組織に連絡する」作業は新しいドメインを取得ないしは作成する際に常に必要な作業です。この作業を一般人がすることはまずないと思いますが、もし行なうことがあったら、詳しい作業手順は上位組織の管理者の指示にしたがって下さい。

以下、検索は同様に続きます。

jp のネームサーバに「www.chitose.ac.jp を知っているか？」と尋ねると、「自分は知らないが、chitose.ac.jp の情報は mcfr0.chitose.ac.jp と mchpwww.chitose.ac.jp が知っている」と教えてもらえます。

そこで mcfr0.chitose.ac.jp か mchpwww.chitose.ac.jp のどちらかに「www.chitose.ac.jp を知っているか？」と尋ねると、その IP アドレスは 210.128.52.32 だと教えてもらえます。

ここで一段階、 ac.jp の階層が飛んでしまっていることに気づきましたか？

ルート、jp、ac.jp、chitose.ac.jp と順番に四回たずねてあるくのが本来の動作なのですが、ある階層のサーバが情報を持っている場合には、より細かい単位の組織に情報を一気に教えてもらえることもあります。これはサーバの設定の問題なので、この事例では、たまたまそうなったということに注意して下さい。全ドメインが同じように動作するとは限りません。

木構造と逆引き

210.128.52.32 という問い合わせを送り、www.chitose.ac.jp という答を得たい。そういう話です。

検索手順は同じですので、検索の詳細は演習問題にしましょう。考えてみて下さい。

逆引きのポイントは２つあります。

まず IP アドレスは左側が大きな単位です。そして、生引きと逆引きで別のプログラムを書くのは面倒なので、同じような動作をさせれば、プログラムの変更が最少ですむはずです。

そこで、IP アドレス 210.128.52.32 は 32.52.128.210.in-addr.arpa. という名前のドメイン名だとみなして、DNS の設定を書きます。データは PTR レコードです。

こうすると、ルート直下に仮想ドメイン arpa、その下に in-addr と来て、大きな単位から順に下へ向かって木構造が作れることが分かります。

図 4.3 をよく見て下さい。

図 4.3 のように仮想的なドメイン名を作り PTR レコードとして扱えば「210.128.52.32 という問い合わせを送り、www.chitose.ac.jp という答をもらう」検索、を生引きと同様に上から下へたどることで実現することが出来ます。

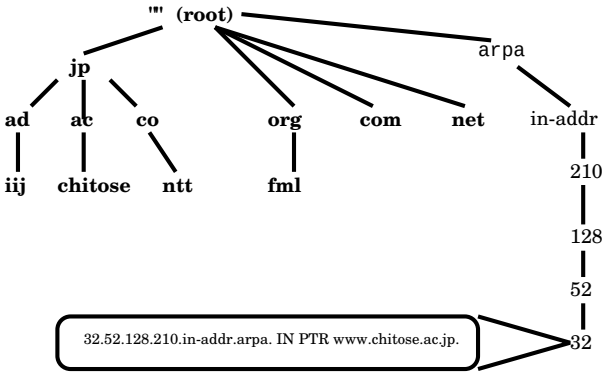

図 4.3　木構造に仮想ドメイン in-addr.arpa を追加し、逆引きのためのドメイン名 (PTR レコード) を木構造として配置しています。一番右の枝です。これを上から下へたどると 210 128 52 32 と読めることが分かります。

4.5.5　ケーススタディ

dig コマンドを使って正引きの各動作をエミュレートし、インターネットの木構造を確かめてみよう (注: 残念ながら、学内からは出来ないので自宅からやってください)。

```
例: ルートサーバを探す。
% dig . ns
```

複数のネームサーバ (NS レコード) が表示されます。適当に一つを選んで、それに問い合わせを送ってください。ここでは日本にあるネームサーバ m.root-servers.net. (202.12.27.33) を選択しましょう。

問い合わせを指定するには@サーバ オプションを使います。

```
例: jp ドメインのネームサーバ (NS レコード) を検索。
% dig @202.12.27.33 jp ns
```

複数のネームサーバ (NS レコード) が表示されます。適当に一つを選んで、それに問い合わせを送ってください。ここでは a.dns.jp とします。

```
例: ac.jp ドメインのネームサーバ (NS レコード) を検索。
% dig @203.119.1.1 ac.jp ns
```

おや？出ませんね。じゃ次に行ってしまえ。

```
例: chitose.ac.jp ドメインのネームサーバ (NS レコード) を検索。
% dig @203.119.1.1 chitose.ac.jp ns
```

mcfr0.chitose.ac.jp と mchpwww.chitose.ac.jp の IP アドレス (A レコード) が表示されましたね？

mcfr0.chitose.ac.jp に www.chitose.ac.jp の IP アドレスを聞いてみましょう。

```
例: www.chitose.ac.jp の IP アドレス (A レコード) を検索。
% dig @210.128.52.11 www.chitose.ac.jp a
```

IP アドレスが 210.128.52.32 であると表示されましたか？

4.6 DNS サーバの冗長化構成

4.6.1 プライマリとセカンダリ

上の例では、問い合わせに対して複数のネームサーバ (NS レコード) が候補として解答されることを見ました。

複数のネームサーバをどうやって運用しているのか？というと、答えは簡単で、一台がマスターデータを持つメインのネームサーバ、それ以外はマスターデータのコピーを持っているだけです。

業界では、DNS に限らず、メインのサーバをプライマリ (primary)、予備のサーバをセカンダリ (secondary) と呼ぶのが習慣です。DNS の場合、メインのサーバはプライマリネームサーバ、予備はセカンダリネームサーバと呼んでいます (長いので、たいていは、プライマリ、セカンダリとしか呼びませんが)。

4.6.2 プライマリとセカンダリ間のシンクロ

セカンダリネームサーバは定期的にプライマリネームサーバからデータをコピーして全サーバが同じデータを持つようにしています。

このコピー操作は転送 (transfer) と呼ばれていますが、実際の作業をしているコマンド名 (/usr/libexec/named-transfer) にちなんで named transfer と呼ぶことも多いです。

なお、このコピーは、確実にデータ転送を行うために、UDP ではなく TCP を使って行なわれます。

4.6.3 伝搬の時差

セカンダリネームサーバは定期的にプライマリネームサーバからデータをコピーしています。

つまり、セカンダリ側からプライマリにコピーをしに来てくれないと、セカンダリとプライマリで異なるデータを持っていることになります。

実際、DNS は、そのように動作しています。

DNS データの伝搬には時間がかかる、などと言っていますが、設定変更の際にはこの伝搬時差があることを想定した計画と操作がエンジニアには要求されます。[*2]

[*2] 定型サービスなら問題内でしょうが、少し複雑なシステムになったとたんに、DNS の操作手順をきちんとこなせる技術者の数は激減します。困ったもんだね。

4.7 まとめ

以下の点についてまとめてみてください。

- 分散システム
- ドメイン名
- 木構造
- レコード (A PTR NS MX ...)
- 階層構造
- 名前解決
  正引き、逆引き
- 名前解決をするには、どのように階層をたどるでしょうか？
- 転送方式の相違
  名前解決は UDP がデフォルト、ネームサーバ間のデータコピーは TCP です。

DNS の例から、良い意味でいい加減な設計 (失敗したら再送すればよいとか、複数のサーバがあれば一つくらいは動くだろう) 、まぁ力の抜き加減と入れ加減というか、その設計センスを見てとって欲しいと思います。

また、これは机上の空論ではないところが重要です。

「話を聞く分には美しいけれど実際に作ろうとするとうまくいかない」デザイン論などではなく、実際に数億台規模のコンピュータが接続されたネットワークで 40 年近く動作し続けてきました。

DNS から学ぶべきものは非常に多いはずです。

第 5 章

TCP

第 3.2 章と第 4 章では、HTTP や DNS といったレイヤー 7 つまりアプリケーション層の実例を見ました。

たとえば HTTP は「GET /index.html HTTP/1.0」といったリクエストを送り、コンテンツ (index.html) をダウンロードするといった WWW 固有の処理をするだけです。

実際のデータ転送はレイヤー 4 の TCP というシステムに頼っています。

このデータ転送も、HTTP 側 (たとえば WWW ブラウザの Firefox) からは、OS の TCP システムに対して「192.168.0.1 の WWW サーバと通信したい」といった指示を出すだけで、TCP の細かな処理は一切気にしませんし、そもそも TCP システムの詳細は WWW ブラウザ側からは見えません。

自分の管轄の部分だけを責任を持って処理すればよいわけです。これが階層モデルの利点といえます。

今回は、データ転送をする TCP について、少し詳しく見てみます。

以下、具体的にアプリケーションについて考える際は、HTTP を例に取り上げます。HTTP については、適宜、第 3.2 節を見直してください。

5.1 TCP の概要

今回はTCP (Transmission Control Protocol) について概要します。難しいというより、ややこしい話です。慣れれば基本動作は大したことをやっていないことがわかるでしょう。

5.1.1 今回の主な内容

- 信頼性のあるデータストリームとは何か？
- TCP プロトコルの動作原理
  - 接続 (初期化)
  - データ転送
  - 接続断

TCP の良さはプログラマにとって扱いが容易なことです。アプリケーション製作者からみると実に使いやすい。パケットを気にする必要もなければ、TCP の詳細を気にする必要もありません。TCP は非常に複雑で高度な処理を行なっています。処理も重たいです。しかしながら、それらは OS の中で自動的に処理される話であって、プログラマが意識する必要がありません。

一方、TCP 以外の、たとえば簡潔で動作も軽い UDP を使うには、プログラマがパケットを意識する必要があります。

つまりトレードオフです。

各アプリケーションは必要に応じて TCP を使うか否かを決めるべきなのですが、実際には、ほぼすべてのアプリケーションが TCP です。これには主に 2 つの理由が考えられますが、詳しくはファイアウォールについての解説 (付録 B.4) を参照してください。

5.1.2 凡例

以前、説明した通り、通信には少なくとも五つの情報を指定する必要があります。特に断らないかぎり、本章の通信も次のようなパラメータとします。

- サーバの IP アドレス = 192.168.0.1
- サーバのポート番号
  アプリケーションごとに異なるので、そのつど説明します。HTTP なら 80 ね。
- クライアントの IP アドレス = 10.0.0.1
- クライアントのポート番号
  通信のたびに OS が適当に番号を割り振ります。本教科書の図表では、たいてい 1025 番を使っています。
- 通信方式は TCP

HTTP などアプリケーションの側では、これら 5 つのパラメータより細かい部分は知る必要がありません。つまり通信路の詳細はブラックボックス (図 5.1) です。これが階層モデル (2.4 節を参照) のよいところです。

5.2 データストリーム

パケット (2.5 節) の復習から始めましょう。

Firefox で、ある URL を開いた時、2500 バイトの大きさの HTML ソースをダウンロードすることになったとします。

図 5.1 ソケット: Firefox と apache (WWW サーバ) の間にあるデータ土管の端がソケット。Firefox は土管の中がどうなっているかは気にする必要はない。単に土管の端からリクエストを流し込めば良い。

5.2.1 ソケット

Firefox が指示 (GET / HTTP/1.0) を与える際、また、データが出力されるのを待つ時、データの入出力[*1] に使う口のことをソケット (socket) と呼んでいます (図 5.1)。ソケットとは、電源コードを挿すあのソケットと同じ意味で、ようするに何かの出入口のことです。

たとえば Perl でソケットを使うコードを書くと、次のようになります。これはデータを受けとる部分の例で、ソケットから一行ずつ読みとっていくという風に書いています。ようするに普通のファイル入出力と同じように書けばよいだけです。

```
while ($data = $socket->getline()) {
    print $data;
}
```

この例では getline() メソッドを使って一行ずつ読んでいますが、(動作が遅いですが) 1 バイトずつだらだら読んでも、いっきに 8192 バイト読み込んだりしてもかまいません。

5.2.2 データストリーム

上の話で大事なところは、ソケットという出入口に出入りするデータには “特別な型” が仮定されていないということです。

違いを理解するために、日常の例で考えてみましょう。

たとえば大学ノートは、幅が 21 cm くらいで、紙の上には線が引いてあります。普通に書くと 1 ページには最大 30 行しか書けません。1 行に書ける文字数も、1 行には 80 文字までといったぐあいに限界があるので、各行の最後まで来ても文章の切れ目が来ない場合は次の行まで文章が続くわけですから、改行して次の行に続きを書き

[*1] ファイルシステムインターフェイスと異なり、ソケットはデフォルトで「入出力」です。ソケットは Unix 本来の思想からずれてしまっていて、そこが困ってしまうところでもあるのですが、詳細は OS の講義にまわします。

ます。

しかしながら、元の文章は一つながりですよね？

ようするに、大学ノートに文字を書く場合、縦 30、横 80 という形を持った “型形” にあてはめる必要があると考えられるわけです。

実際、大学や研究所のコンピュータセンターや銀行などで使ってきた大型計算機と呼ばれる類のコンピュータではデータにそういったデータの “型” を仮定しています。

しかし、コンピュータが理解できるデジタルデータは紙を使うわけではないので、無理に 30x80 といった型にこだわる必要などありません。それに行が 80 文字ぴったりでない場合には無駄な空白が出来るだけです。

プロトコルを思い出して下さい (1.2.1 節参照)。コンピュータどうしのやりとりでは、取り決めさえあればどうとでもなるのです。

そういうわけで TCP/IP ではデータに形を仮定しないと取り決めます。

データとは単なる二進数の列です。普通、8 ビットを一つの単位として 1 オクテットないしは 1 バイト[*2] と呼んでいます。

この単なるデータ列のことをデータストリーム (stream) と呼んでいます。ストリームとは流れのことです。

では Unix では改行をどうするのか？というと特別な “改行コード” というものを入れて表現しています。あとは各アプリケーションが改行コードを見て画面を作成するという仕組みです。

たとえばデータ

```
Rise of Endymion 改行コード by Dan Simons 改行コード
```

を Firefox で見ると、次のように二行に表示されるといった具合です。

```
Rise of Endymion
by Dan Simons
```

よって、改行コードの取り決めが違うと、画面が乱れます。これは Unix と Mac OS、Windows の間でテキストファイルのやりとりがうまくできない理由の一つです (うまくできる場合もありますが、それはアプリケーションのおかげ)。

5.2.3 復習: パケットの分割と再構成

Firefox で、ある URL を開いた時、2500 バイトの大きさの HTML ソースをダウンロードすることになったとします。

代表的なパケットデータサイズは1460 バイトなので、サーバ側では、これを 1460 バイトと 1040 バイトの 2 つに分けます。

データが 1460 バイトと 1040 バイトの 2 つのパケット (図 2.2) を作り、サーバ PC からクライアント PC へ転送されます。そして受信したクライアント PC でパケットから再構成し、2500 バイトの HTML ソースが Firefox へ渡されます。

この分割と再構成をしているのは誰でしょうか？それが今回の主題 OS の中にあるネットワークシステム中の TCP サブシステムです。

5.3 TCP

5.3.1 TCP の歴史

1960 年代、アメリカ国防総省の研究機関である ARPA の元、既存の通信網と異なる通信方式の研究が始められました。

これが現在のインターネットの祖先です。

当時の通信システムの代表はテレビや電話です。事実上、電話以外の通信方式の探求といって良いでしょう。

ARPA の委託を受けた BBN による研究成果を受け、ARPA はパケット通信である TCP/IP を採用しました。BBN による試験実装も行われましたが、BBN の実験コードは性能が悪過ぎ[*3]

[*2] 1 バイトが 8 ビットでない場合もあるのです。

[*3] そもそも BBN 社内が 56kbps だったそうなので、そ

たため、後述する BSD Unix 版に載せる際には Bill Joy [*4] が三週間で速度を 10 倍速にしたという逸話が残っています。

７０年代後半、すでに ARPANET は試験運用されていましたが、本格的な TCP/IP への移行をひかえ、各組織のハードウエアの買い換え時期、32 ビット世代へのハードウエアの移行など、さまざまな課題があり、実際に (多くの人が使うハードウエア上で) 動く OS に TCP/IP を載せる必要がありました。そういうわけで、バークレイ (カルフォルニア大学バークレイ校、University of Californiaat Berkeley) が ARPA から資金提供を受け、BSD Unix に TCP/IP を実装することになったわけです。TCP/IP を載せて最初に広く出荷された Unix バージョンは 4.1c BSD、ターゲットマシンは DEC VAX11/780 でした。

PDP11 と同様に VAX11 は多くの大学や研究機関が持っていましたし、買い換え先候補としても VAX11 は最有力候補だったため、BSD Unix に TCP/IP が搭載されると、これらの組織が BSD Unix を入手し、BSD ベースの TCP/IP ネットワークが大学、研究機関に導入されました。そして、それらを中心に草の根的にネットワークが構築されていったわけです。これが現在の商用インターネットの先祖になりました。

コラム: 1960 年代

1960 年代のアメリカとは、冷戦構造の中、ソビエト連邦に宇宙開発で先を越され (スプートニクショック)、アポロ計画で月に到達することにアメリカの威信回復をかけていた時代でした。

当時の人は、本当に核戦争が近いと思っていました。

アポロ 11 号はで月に到達し、アメリカの科学力と資金力と意地を見せつけましたが、アメリカ帝国の栄光も長くは続きません。

1960 年代も終りになると、ベトナム戦争も膠着化し、アメリカ帝国の正義がゆらぎはじめます。その反動やゆらぎは、それまでの文化や科学万能主義、西洋文明至上主義に対する反動として社会に現れてきます。

コンピュータネットワークの歴史において、こちらの負の側面は直接の影響を与えていません。まだネットワークの黎明期だったからでしょう。

第二次世界大戦後のアメリカでは、科学とは個人プレーではなく、集団による巨大科学となり、多くの予算が科学研究に注ぎこまれました。また、アメリカでは有名な大学のほとんどすべてが、国防総省など軍関係から研究資金をもらっています。

実のところ、60 年代の TCP/IP のスポンサーになった部局は、実験心理学の人たちが率いていて、彼らの興味はコンピュータによるコミュニケーションでした。

そのため、軍事技術と言うと少し違うのですが、お金の出所が DARPA つまりアメリカ軍がスポンサーだったことは確かなことです。

こまでしか実権もできなかったし、そこまでしか頑張らなかったという推測があります

[*4] Unix 業界の伝説的ウイザード。vi や csh の作者です。BSD 4.1b までの BSD Unix のアーキテクトでプログラマ、今で言うエヴァンジェリストでもあり、カスタマーサポートまで何でもこなした伝説の人です。

SUN の創設時にバークレイを去り SUN 四番目の社員として参加し、その後の SUN の技術面での最高指導者 (Chief Technology Officer) として 20 年間 SUN を率いた後、2003 年に SUN から去りました。その後、SUN は Oracle に買収されてしまいました。

いまでは SUN というと Java の会社か？と言われてしまいそうですが、SUN のワークステーション (ハードウエア) と SUN OS および Solaris (ソフトウエア) など、ハードとソフト両面においてさまざまな貢献をしてきた Unix とインターネットの歴史でもっとも重要な会社の一つです。

NFS や VFS なども SUN 由来です。これまた、詳しくは OS の講義だ。

あと、近年の SUN 絡みで燃える話題といえば ZFS ですな。

コラム: 軍事技術と科学研究
詳しくは述べませんが、ローレンスによる加速器 (サイクロトロン) 研究からマンハッタン計画あたりまでの時代を境にして、それ以前と以降における科学は大きく変わります。
それらの変化についても、しっかりとした理解が望まれます。
科学とは何か？を考える授業は「科学哲学」です。
今のところ「科学哲学」の授業は、開講予定にありませんが、行なうべきだと考えてはいます。
むかしは「メディアリテラシー」の授業で少しだけ取り扱っていたのですが、残念ながら、この授業は終わってしまいました。

5.3.2 TCP の設計目標

TCP の原典であるドキュメント RFC793 を元に解説しましょう。

TCP の目標は大きく分けて次の 3 つといえます。

- 信頼性
- 安全な論理回線
  注: 現代的な意味での「安全」とは、かなり異なる「安全」です。
- コネクション指向

もう少し詳しく TCP の特徴を見ると次のようになります。

- データ転送
  TCP はデータストリームの連続転送を行ないます。上の層は TCP 層を単なるストリームの入出力として扱うだけです。アプリケーションの側からは 5 つのパラメータを指定して TCP 接続 をすれば、あとは入出力の口 (ソケット) に対して読み書きするだけです。3.1 節も参照してください。
- 信頼性 (reliablility)
  パケットの紛失、重複、乱れた順序などの回復を行ないます。[*5]
  注: (実際にそれを行なうことは難しいとしても) "原理的"には「データを横取りして書き換えてしまう」といったことが可能ですが、ここで言う信頼性は現代語の「安全」「確実」ではなく「データ転送については確実に行ないます」という枠組の提供を指す程度の意味と考えられます。
- フロー制御 (flow control)
  データの流れ方を制御します。たとえば、受取側が手一杯であれば、送信側は少し待つといったぐあいの動作です。
- 多重化 (multiplexity)
  コンピュータ間で複数の TCP コネクションを使えます。各コネクションはポート番号と IP アドレスにより区別しています。すでに 3.1 節で説明しました。
- コネクション (connection)
  データ転送において状態の情報が維持管理されています。そうでないと、信頼性やフロー制御が実現できません。このため、TCP は最初にコネクションを確立する必要があります。
- Precedence and Security
  現代的な意味の「安全」とは違うので、この話は省略します。

5.3.3 TCP パケット

パケットとは情報の固まりのことです。ヘッダと呼ばれる制御情報部分とデータ本体の 2 つから構成されています。

パケットの制御はヘッダに制御情報を書くこと

[*5] ちなみに、TCP 直下のレイヤー 3 (IP 層) は転送を行なうだけで、パケットの紛失、重複、乱れた順序などは一切気にしません。

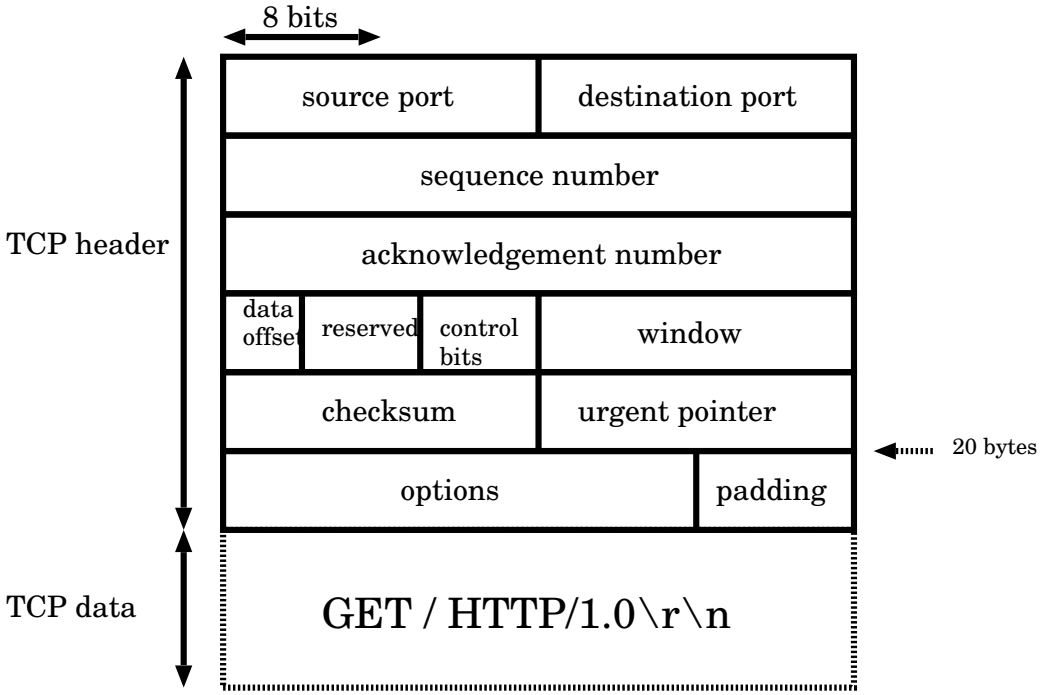

図 5.2　TCP パケット: 横一列では書き切れないので 32 ビットごとに改段しています。また、データ部分は HTTP のリクエストを格納している様子です。

で行ないます。つまり、ポート番号や IP アドレスをヘッダに書くわけです。

TCP のパケットもヘッダとデータから成っています (図 5.2)。

TCP ヘッダには、指定するべき 5 つのパラメータ (5.1.2 節) のうち、ポート番号 2 つ分が書かれています。

では、残りの 3 つは？ (IP アドレス 2 つ分と TCP の指定) というと、TCP の下で TCP パケットの転送を行なう IP パケットに書かれています (図 5.3)。

図 5.3 は、IP パケットという封筒の中に TCP の封筒が入っていて、TCP 封筒の中に HTTP のデータがあるという構造と思うと分かりやすいでしょう。

封筒の表に書く宛名などがヘッダにあたります。また、封筒の中に入れるデータサイズは可変です。

IP 封筒の大きさは 1500 バイトが代表的な値です。これは IP 封筒を入れるイーサネット封筒つまりイーサネットパケットが運ぶデータサイズが 1500 バイトだからです。詳しくは、イーサネットの回 (第 10 章) を参照してください。

実際に封筒の中に入れることが出来るデータの大きさは、ヘッダの分 (例: 20 バイト) を差し引いた残りです。そのためパケットの最大データサイズが 1480 バイトなどとなります。

通常、ヘッダのオプションも使うことが多いため 1480 より小さいことが普通ですが、本書では簡略化のためオプションは無視し、IP パケットのデータサイズは最大 1480 バイトとしています。

同様に TCP パケットの最大値が 1480 バイト、TCP ヘッダが 20 バイト分として、TCP データ部分の最大値が 1460 バイトとしています。

実際の通信ではヘッダオプションを使っているので 1460 より小さいことが普通です。

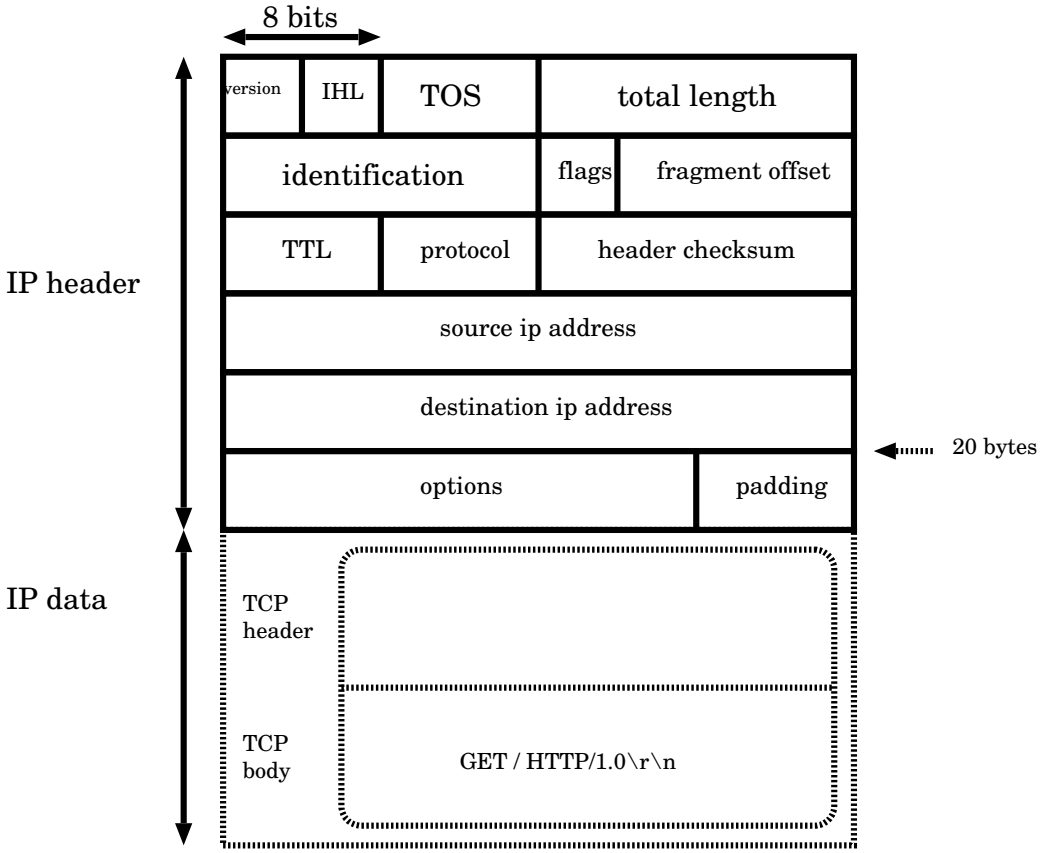

図 5.3　IP パケット: 横一列では書き切れないので 32 ビットごとに改段しています。また、データ部分は HTTP のリクエストを格納している様子です。

5.3.4　TCP ヘッダ

図 5.2 の各要素は次のような意味です。

source port

ソース (source) ポート番号。16 ビットの長さ (0 〜 65535) で、送信元のポート番号です。

destination port

デスティネーション (destination) ポート番号。16 ビットの長さ (0 〜 65535) で、受信先のポート番号です。

sequence number

シークエンス番号。32 ビットの数字で、「初期シークセンス番号」か「先頭データのオフセット (ずれ)」という意味です。送ったデータ量分、数字が増えていきます。詳しくは後述します。

acknowledgement number

アクノリッジ番号。32 ビットの数字で、「次に期待されているシークエンス番号」の数字が書かれています。言い替えると「この番号以前の番号のデータは受けとった」という意味になります。詳しくは後述します。

なお acknowledge は確認という意味です。

data offset

データオフセット。大きさは 4 ビットですが、単位が 32 ビットであることに注意が必要です。

データ (TCP パケットが運ぶデータ) が TCP ヘッダ後のどこから始まるかを示す数字です。

reserved

予約。6 ビット分ありますが予約となっていて、使われていません。

control bits

コントロールビット。6 ビットあります。

1 ビットずつそれぞれに意味があり、TCP の制御に使うフラグです。左から順に 1 ビットずつ URG ACK PSH RST SYN FIN と呼ばれています。

window

ウインドウサイズ。大きさは 16 ビットです。ウインドウのサイズを表しています。

checksum

チェックサム。大きさは 16 ビットです。データが壊れていないかを検証するためのデータです。

ヘッダとデータ部すべてのチェックサム[*6]をとっています。

urgent pointer

大きさは 16 ビットです。急ぎのデータが、どこにあるかを意味するオフセットの値で、URG フラッグの時にだけ意味があります。

options と padding

オプションの指定です。

今時の TCP では常に timestamp オプションなどを使っていたりするのですが、長くなるので省略します。

5.4 TCP 接続

TCP によるデータ転送は次のように大きく三段階に分かれます。

- 接続
  TCP コネクションの確立です。
  図 5.4 の ESTABLISTED 状態になるところまでが該当します。
- データ転送
  コネクションの確立後、データ転送です。転送時における信頼性の確保やフロー制御も自動的に行なわれます。
  図 5.4 の ESTABLISTED 状態で行われる動作です。
- 切断
  データ転送が終ったら、明示的に TCP コネクションを切ります。

5.4.1 接続 (初期化)

HTTP の回 (第 3.2 節) で「telnet 172.20.172.30 80」を打ち込み、ENTER キーを叩いて

```
Connected to 172.20.172.30.
Escape character is '^]'.
```

が出たら「TCP で接続した」という意味であると説明しました。

この ENTER を叩いてから「Connected to 172.20.172.30」が出るまでに、OS が裏で行なっている作業が、この接続ないしは初期化です。TCP の初期化 (initialization) とか TCP コネクション (connection) の確立と呼ぶ場合も多いです。また、コネクションが確立した時は図 5.4 の ESTABLISTED 状態にあるので、業界では ESTAB などと呼びます。

この TCP 初期化プロセスが有名な3 way handshake(三回の握手) です。

制御情報の使い方

TCP ではコネクション指向通信を実現するために、TCP 層が信頼性のあるデータ転送を保証し、TCP 層自身がデータ転送における状態の維持管理を行ないます。

このために TCP ヘッダ (5.3.3 節) には、たくさんの要素があるわけですが、3 way handshake で最も重要な要素が次の 3 つです。

- シークエンス番号 (sequence number)
- アクノリッジ番号 (acknowledge number)

[*6] アルゴリズムは BSD Unix kernel の */usr/src/sys/netinet/in_cksum.c* を参照してください。

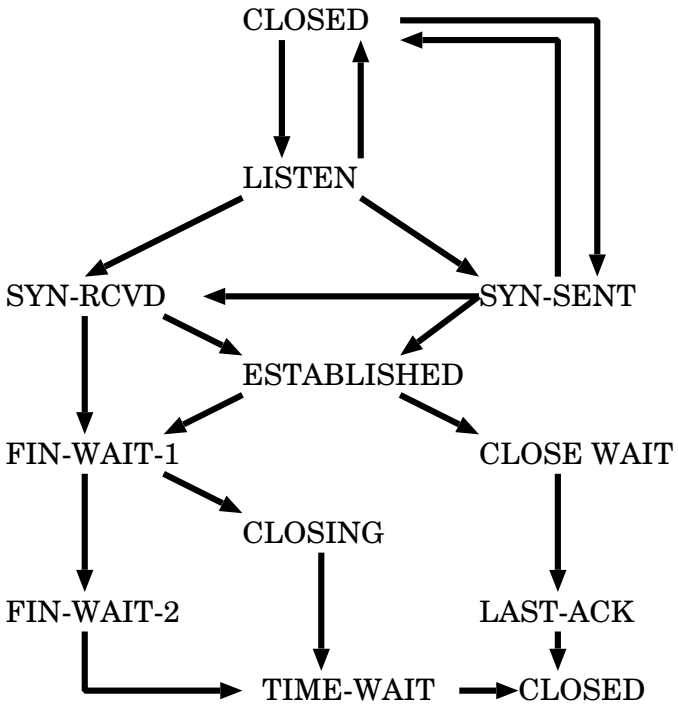

図 5.4　TCP の状態遷移

- コントロールビット (control bits)

図 5.2 のとおり、シークエンス番号とアクノリッジ番号は、各 TCP ヘッダにある制御情報です。

シークエンス番号とアクノリッジの番号は無関係です。ただ、TCP パケットがやりとりされるにつれて、シークエンス番号とアクノリッジ番号の値が順番に変わっていくところに注意して下さい。詳しくは以下の事例で見てみましょう。

コントロールビットは命令や確認の相槌を意味する制御情報で、コネクション状態の維持管理に使われます (表 5.1)。

コントロールビットでは、データ量の節約のため、1 ビットで 1 命令です。アプリケーション層の分かりやすいプロトコルとは大きく変わりましたね？

ビット	意味
URG	Urgent Pointer field significant 急ぎのデータ転送 (略)
ACK	Acknowledgment field significant 受領を確認したと知らせる
PSH	Push Function 受取側プロセスに即座に渡すことを強制する。 バッファリングなし。 送信プロセスから受信プロセスまでの間で 確実にデータを渡す。
RST	Reset the connection コネクションのリセット。
SYN	Synchronize sequence numbers コネクションの確立を要求する。 最初に使われる。
FIN	No more data from sender コネクションの切断を要求する。 TCP の終る時。

表 5.1　TCP ヘッダのコントロールビット

3 way handshake

TCP コネクションの確立は 3 way handshake (三回の握手) と呼ばれる有名な技法です。

コントロールビットのやりとりで handshake を実現します。

```
送信者  <-------> 受信者
        -- SYN ->
        <- ACK --
        <- SYN --
        -- ACK ->
```

実際には 4 つの処理 (4 つのパケット) から構成されていますが、真中の 2 つの処理は 1 つのパケットにまとめることができるので、合計 3 回となっていることに注意してください。この 2 つの処理を 1 つのパケットにまとめる処理は、相乗り (piggyback) と呼ばれます。TCP 以外でも、相乗りというテクニックはパケットの往復回数を減らすために重要です。

図 5.5 が初期化の際のパケットのやりとりです。

1. 3 way handshake の 1 つ目の握手
   10.0.0.1 から 192.168.0.1 へパケットを送信します。
   SYN コントロールビットをセットし、「シークエンス番号 100 から転送をしたい」というリクエストを送っている状態です。
2. 3 way handshake の 2 つ目の握手
   192.168.0.1 から 10.0.0.1 へのパケット送信 (返信) です。次の 2 つの処理を一つのパケットへ相乗りさせています。
   - ACK
     「了承した」という印に番号を +1 します。また、返事なので ACK コントロールビットをセットし、「シークエンス番号 101 からの転送を期待します」というリクエストを送っています。
   - SYN
     SYN コントロールビットをセットし、「192.168.0.1 側はシークエンス番号 3000 から使いたい」というリクエストを送ります。

   前のパケットとは、TCP ヘッダのシークエンス番号とアクノリッジ番号が入れ替わったことに注意しましょう。
3. 3 way handshake の 3 つ目の握手
   10.0.0.1 から 192.168.0.1 へのパケット送信です。
   「了承した」という印に番号を +1 しています。また ACK コントロールビットをセットし、「シークエンス番号 3001 からの転送を期待します」という返事を送ります。
   前のパケットとは、TCP ヘッダのシークエンス番号とアクノリッジ番号が入れ替わったことに注意してください (つまり一つ目のパケットと同じ順に戻りました)。

この時点で双方が ESTABLISHED 状態へ遷移し、実際のデータ転送を始める準備が出来たということになります (TCP 状態遷移図 5.4 参照)。

5.4.2 転送

図 5.6 は TCP のデータ転送の様子を表しています。簡単化のため、データの流れが一方通行 (10.0.0.1 → 192.168.0.1) の例 (たとえばクライアントからファイルをアップロードしている場合) です。

この図 5.6 は次のように読みます。

転送 (1)

10.0.0.1 から 192.168.0.1 へパケットを送信。

10.0.0.1 から 192.168.0.1 へ 100 バイト分のデータ (注: TCP のデータ部の大きさが 100 バイト) を送ります。

シークエンス番号は 101 から始めます (seq=101)。初期化の最後のシークエンス番号が 101 ですね？

返事のシークエンス番号は seq=3001 を期待

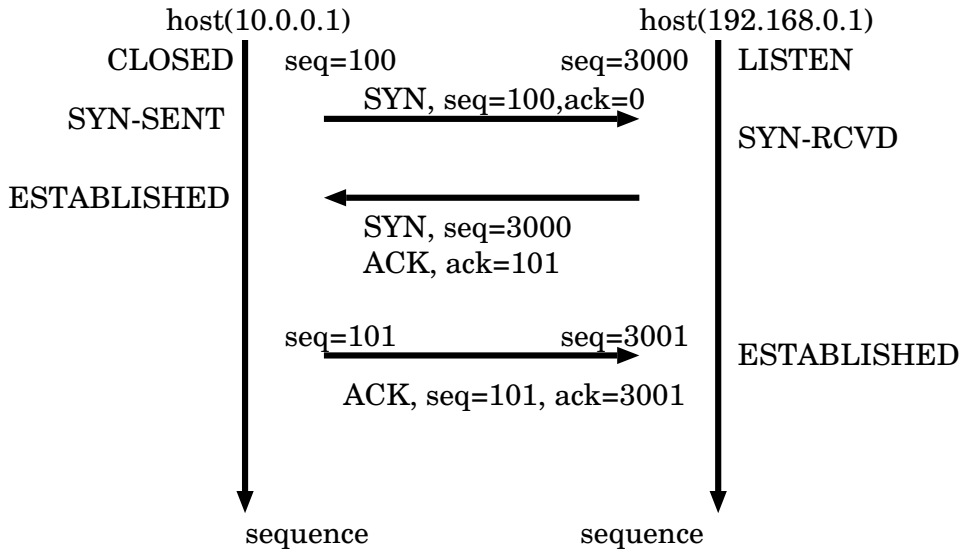

図 5.5　TCP の初期化: 3 way handshake

しています (ack=3001)。初期化の最後のアクノリッジ番号が 3001 ですね？

これらは初期化の際にそう取り決めたのだから当然です。初期化のところを見直して下さい。

また、返事の際にはシークエンス番号とアクノリッジ番号の役割が反対になっていることに注意してください。

転送 (2)

192.168.0.1 から 10.0.0.1 へパケットを送信。

つまり返事ですね (図 5.6 では返事に当たる部分は赤くしてあります)。

アクノリッジ番号が 201 (ack=201) なのは、「シークエンス番号 101 からはじまり 100 バイト受けとりました。次回 (返事) はシークエンス番号 201 (201 = 101 + 100) の転送を期待します。」という意味です。

シークエンス番号は 3001 です (seq=3001)。転送するデータがないのでシークエンス番号は変わりません。

この例は、もっとも簡単な「クライアントからサーバへ一方的にデータが流れている」ケースなので、たまたま「転送するデータがない」だけです。注意してください。一般には、なんらかの返事を (たとえば「OK」や「220」を) 送るでしょうから、シークエンス番号もアクノリッジ番号もともに変化していきます。

転送 (3)

10.0.0.1 から 192.168.0.1 へパケットを送信。

100 バイト送ります。相手の期待通り、シークエンス番号は 201 (seq=201)、アクノリッジ番号は 3001 (ack=3001) のパケットになっています。次の返事のパケットでは seq=3001 を期待しているということです。

OK？

転送 (4)

192.168.0.1 から 10.0.0.1 へパケットを送信。

「100 バイト分を受けとりました」という返事なので次回のシークエンス番号は 301 (201 + 100 = 301) を期待します。

今回もシークエンス番号は 3001 です (seq=3001)。転送するデータがないのでシークエンス番号は変わりません。

転送 (5)

10.0.0.1 から 192.168.0.1 へパケットを送信。

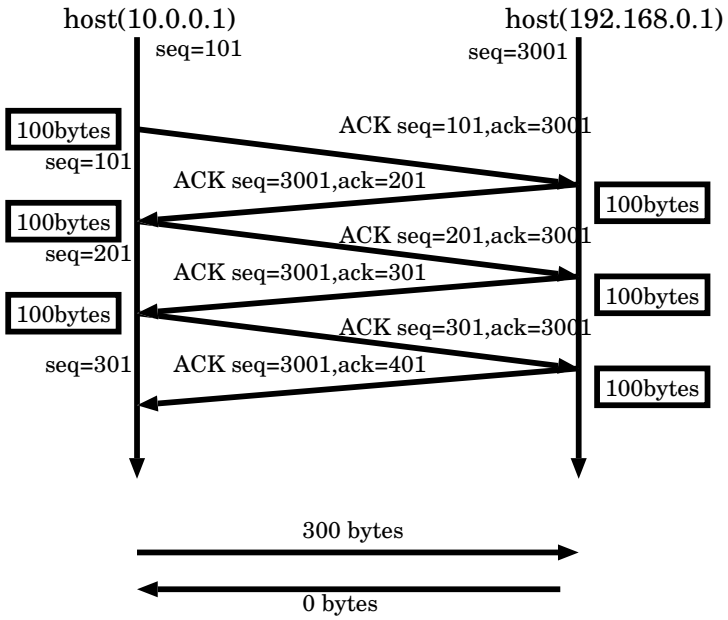

図 5.6　TCP のデータ転送: 簡単化のため、データの流れが一方通行 (10.0.0.1 → 192.168.0.1) の例。

100 バイト送ります。相手の期待通り、シークエンス番号は 301 (seq=301)、アクノリッジ番号は 3001 (ack=3001) のパケットになっています。次の返事のパケットでは seq=3001 を期待しているということです。

OK？

転送 (6)

192.168.0.1 から 10.0.0.1 へパケットを送信。

「100 バイト分を受けとりました」という返事なので次回のシークエンス番号は 401 (301 + 100 = 401) を期待します。

今回もシークエンス番号は 3001 です (seq=3001)。転送するデータがないのでシークエンス番号は変わりません。

OK ?

小まとめ

これで基本は終りです。

送るデータがある限り、同じ動作を繰り返します。

明らかに、たくさんのパケットが往復し続ける様子は無駄に見えますね？

そのとおりです。

そのために効率をあげる相乗り (5.6.1 節参照) やフロー制御 (5.6.2 節参照) がありますが、それらは付録に譲ります。

5.4.3　再送

前節 (5.4.2 節) はデータ転送が順調にいっている例ですが、何らかの理由でパケットが届かない場合があります。その場合、届かなかったパケットを再送しなければなりません。

図 5.7 はシークエンス番号 201 のパケットが紛失したという想定です。

10.0.0.1 側は 192.168.0.1 からの返事パケットのアクノリッジ番号をみることで「次回シークエンス番号 201 を期待します」の分までしか返事が来ていないことが分かります。ということはシークエンス番号 201 の分のパケットが届いていないに違いありません。「では、もう一度、seq=201 のパケットを送り出そう」と考えるわけです。

順番が狂っていても 192.168.0.1 側は受けとり続けます (注: OS が受けとるだけでアプリケーションにはデータを渡しません)。

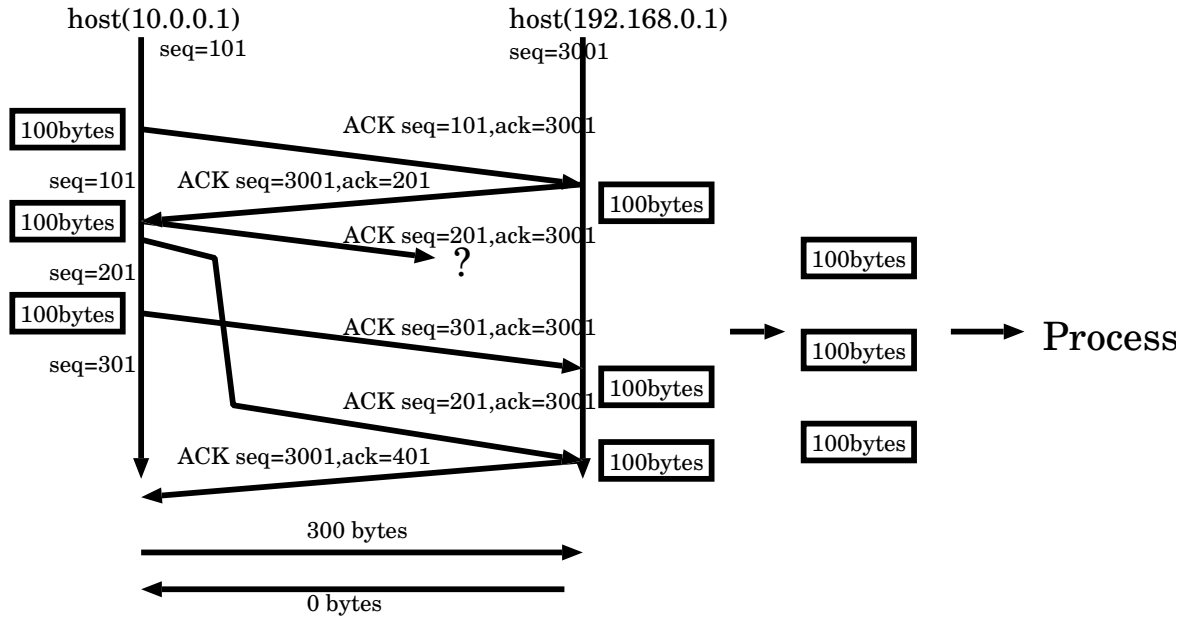

図 5.7　TCP のデータ再送例: 簡単化のため、データの流れが一方通行 (10.0.0.1 → 192.168.0.1) の例。

やがてシークエンス番号 301 201 の順に到着します。

順番が乱れていますが、受けとった側でシークエンス番号 101, 201, 301 まで受けとったことを確認できたら、アクノリッジ番号 401 (次回はシークエンス番号 401 を期待) を 10.0.01 へ送信します。

ここでシークエンス番号 101 201 などのパケットについて個別の受領確認は送らない (送る必要がない) ことに注意してください。

転送がどこまで進んだか？を相手に知らせることが出来れば十分なのです。

なお、この「まとめて返事を送る」動作が相乗りです。詳しくは 5.6.1 節を参照。

5.4.4　接続断

互いに FIN 命令を送り、接続を切断することに合意します。

これは各自の宿題としましょう。3 way handshake とたいして変わりません。自分で調べてみてください。

5.5　まとめ

5.5.1　まとめ

自分でまとめてみよう。

5.5.2　原典

RFC793 (インターネットスタンダード 0007) http://www.ietf.org/rfc/rfc793.txt "Transmission Control Protocol. J. Postel. Sep-01-1981."

5.5.3　参考書

"TCP/IP Illustrated Volume 1-3", W. Richard Stevens, Addison-Wesley.

"Unix ネットワークプログラミング", W. Richard Stevens, Prentice Hall.

日本では「マスタリング TCP/IP」あたりが有名（らしい。日本語の本は読んだことないから、おすすめできるのかどうか分からん)。

5.6　付録: TCP やや高度な話

5.6.1　転送: 効率をあげる相乗り

再送の話 (5.4.3 節) で見た通り、「返事をまとめて返す」と、パケットの数が減ります。

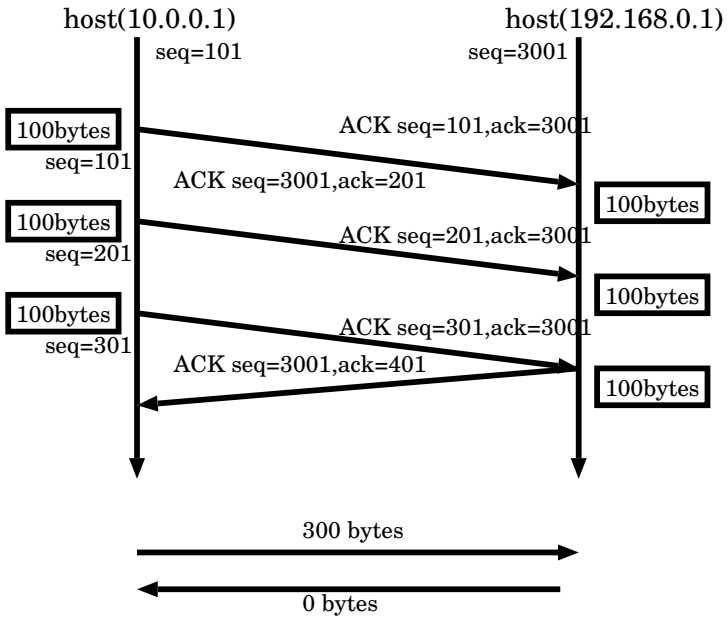

図 5.8 TCP の相乗り

これを piggybacking と呼びます。相乗りとか便乗という意味です。

「返事をまとめて返す」メリットを考えてみましょう。

まず、PC の能力を考えます。

現代の PC のパワーをもってすれば 1 Gbps (1 G bits per seconds) のデータ転送は可能です。実際、最近の PC のネットワークインターフェイスは 1 Gbps 対応が普通です。

Pentium III 世代ですら、100 Mbps のデータ転送は可能です。ちなみに、こういった文脈では転送するデータ量のことをトラフィック (traffic) と呼びます。

つまり 1500 バイトのパケットを一秒間に 10000 発近く送信可能なわけです。1 パケットあたり、送信にかかる時間は 0.1 ms (マイクロ秒) と見積もられます。

次に、パケットの往復時間を見積もりましょう。

10.0.0.1 から 192.168.0.1 へ同一 LAN (後述) の中でパケットがやりとりされる場合、片道 0.15 ms、パケットの往復に 0.3 ms くらいかかります。まぁまぁです。

ところが、インターネットでは、さらにひどく、現在でも自宅から大学まで往復 50ms くらいかかるのは普通です。

つまりパケットの往復時間は同一 LAN や TCP の送信タイミングに比べ 500 倍は遅い動作ということになります。

TCP では「きちんと送れたのか？」の返事を待つわけですから、パケットが往復すればするほど足を引っ張るということです。

たとえば、同一 LAN では、

- 時刻 = 0 ms
  seq=101 を送信。
- 時刻 = 0.1 ms 後
  ack=201 が到着、そくざに seq=201 送信。

これは可能でしょう。つまり 100 Mbps が実現できます。

しかし、インターネットでは？

- 時刻 = 0 ms

seq=101 送信。

- 時刻 = 50 ms 後
  ack=201 が到着、そくざに seq=201 を送信。

これを単純計算すると LAN の 1/500 の速度、つまり 200 kbps の速度しか出せないことになります。

つまるところ、返事を待っているのが悪いわけですね？

だから相乗りをして、往復するパケットの回数を減らすことに意味があるわけです。また、待たずに、先出し送信などもうまくいけばパケットの待ち時間を減らせる可能性があります。

実際の効率化技術については RFC や参考書を読んで下さい。たとえば、RFC1323 "TCP Extensions for High Performance" (1992) では、timestamp オプション、パケットの往復時間の測定など転送速度を最適化するために微調整を繰り返すテクニックが提案されています。

5.6.2 転送: フロー制御

フロー制御 (flow control) とは、流れ方の制御です。データ転送量を増やせないだろうか、流れが詰まっているようなら少しデータ転送量を減らすべきか？そういったことを考えることです。

TCP のフロー制御

TCP の場合、おおざっぱにいって次のようなことがおきえます。

- ACK を待っていると、遅くなる。
  5.6.1 節で説明したように、パケットが往復する時間は PC にとって長すぎます。待っているだけでは駄目です。
- あるていど ACK を待たずにパケットを先に送信してしまう。
  先出しで送信するということです。
- だが、うまく相手が受け切れないと、再送するハメになる。
  先出しで送信するのはいいのですが、大量に先出しをして一気に ACK をもらおうとすると、相手が受け切れません。

そういうわけで、パケットの流れ方の様子を見つつ、送信タイミングやどれくらいの量を先出しするか？を微調整する必要があるわけです。

ウインドウ

TCP の場合、そういった処理をウインドウ (窓、window) というパラメータで制御しています。

window は TCP ヘッダにある 16 ビットの大きさの数字です。この数字は一気に受け切れる量の目安を意味しています。

なお、以下の例のように TCP ヘッダのウインドウサイズはパケットごとに異なる可能性があります。

図 5.9 が一例です。これは次のように読んでください。

初期段階、受信側はウインドウサイズ 200 でスタートするつもりです。おおまかにいえば 100 バイトの大きさのパケットを 2 つくらいなら一気に受けとれるという意思表示を相手にしていると考えて下さい。

送信側ではシークエンス番号 101 の ACK が返ってきた時にウインドウサイズが 200 であると知ります。つまり相手が一気に 200 バイトは受け切れるらしいと分かったわけです。そこで、一つ一つの ACK を待たずにパケットを 2 つ (シークエンス番号 201 と 301 を) 立て続けに送信してしまいます。

2 つのパケットを送信して、受領確認の ACK のパケットを 1 つ受けとることで、一つ分のパケットの待ち時間が節約できました。

この論法をもっとつき進め、もっと大量のパケットを送り、一気に返事をもらったり、タイミング調整をもっと絶妙にすることでデータ転送速度をあげていくことが出来ます。

この例では簡単化のためにパケット一つあたりのデータサイズを 100 バイトにしていますが、実

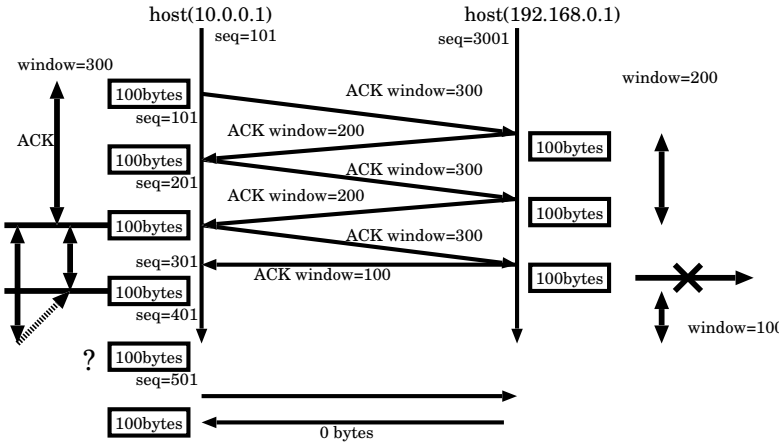

図 5.9 TCP のフロー制御とウインドウサイズ

際のパケットデータサイズは 1400 バイト強で、ウインドウサイズの初期値は 16384 です。

さて、では、受信側でパケットの処理が滞り、つまってしまった場合、どうすれば良いでしょうか？

結論をいえば「次のパケットを送信しないで、少し待って欲しい」わけです。そこでウインドウサイズを小さくして相手に「あまり一気に送信しないで欲しい」ということを伝えます。

図 5.9 の下の方を見て下さい。

送信側は、シークエンス番号 301 分の ACK までは返って来たが、ウインドウサイズが 100 になった。そこで一気に 2 パケットずつ送ることにしていた予定を変更し、一つずつ送り一つずつ返事 (ACK) を待つことにします。

やがて、受信側の処理がスムーズに進むようになれば、再びウインドウサイズを大きくしていくことになるでしょう。

5.7 ツールの使い方: パケットを見る

5.7.1 tcpdump

この tcpdump コマンドを使うと生のパケットが見えます。普通に使うと以下のようにヘッダ情報だけですが、もちろんデータ部分も見えますので、パスワードでも何でも見えてしまいます。

そのため OS の管理者権限がないと tcpdump コマンドは実行できません。よって、残念ながら大学の PC 教室では実行できません。

portal.mc.chitose.ac.jp (210.128.52.45) とのやりとりを表示する例。1 行が長いので適当に折り返し、1 パケットごとに改行を入れて分かりやすくしてあります。

```
# tcpdump -n port 80 and host portal.mc.chitose.ac.jp
tcpdump: listening on fxp0

12:09:43.483361 10.0.0.5.59921 > 210.128.52.45.80:
 S 2100231749:2100231749(0) win 32768
 <mss 1460,nop,wscale 0,nop,nop,timestamp 0 0> (DF)

12:09:43.483678 210.128.52.45.80 > 10.0.0.5.59921:
 S 2359084301:2359084301(0) ack 2100231750 win 32768
 <mss 1460,nop,wscale 0,nop,nop,timestamp 0 0> (DF)

12:09:46.484109 210.128.52.45.80 > 10.0.0.5.59921:
 S 2359084301:2359084301(0) ack 2100231750 win 32768
 <mss 1460,nop,wscale 0,nop,nop,timestamp 6 0> (DF)

12:09:46.490991 10.0.0.5.59921 > 210.128.52.45.80:
 . ack 1 win 33580 <nop,nop,timestamp 6 6> (DF)

12:09:46.491516 10.0.0.5.59921 > 210.128.52.45.80:
 P 1:180(179) ack 1 win 33580 <nop,nop,timestamp 6 6> (DF)

12:09:46.581063 210.128.52.45.80 > 10.0.0.5.59921:
 P 1:331(330) ack 180 win 33580 <nop,nop,timestamp 7 6> (DF)

12:09:46.583292 210.128.52.45.80 > 10.0.0.5.59921:
 P 331:1094(763) ack 180 win 33580 <nop,nop,timestamp 7 6> (DF)

12:09:46.583801 210.128.52.45.80 > 10.0.0.5.59921:
 F 1094:1094(0) ack 180 win 33580 <nop,nop,timestamp 7 6> (DF)

12:09:46.585780 10.0.0.5.59921 > 210.128.52.45.80:
 . ack 1094 win 32817 <nop,nop,timestamp 7 7> (DF)

12:09:46.585932 10.0.0.5.59921 > 210.128.52.45.80:
 . ack 1095 win 33580 <nop,nop,timestamp 7 7> (DF)

12:09:46.586926 10.0.0.5.59921 > 210.128.52.45.80:
 F 180:180(0) ack 1095 win 33580 <nop,nop,timestamp 7 7> (DF)

12:09:46.587099 210.128.52.45.80 > 10.0.0.5.59921:
 . ack 181 win 33580 <nop,nop,timestamp 7 7> (DF)
```

5.7.2 wireshark

GUI でパケットの中身を見るソフトウエアに wireshark (旧名称 ethereal) があります。

tcpdump コマンドと同様に OS の管理者権限がないと wireshark コマンドは実行できません。よって、残念ながら大学の PC 教室では実行できません。

第 6 章

レイヤー 4 その他

6.1 UDP

6.1.1 UDP の概要

UDP (User Datagram Protocol) はパケットを転送する最低限の仕組みを提供します。

代表的な UDP アプリケーションは次のものでしょう。

- DNS (Domain Name System)。
  第 4 章参照。
- TFTP (Trivial File Transfer Protocol)。
  ルータなどのネットワーク機材との間で設定のやりとりやファームウエアのアップデートをする際に使うプロトコル。
- インターネット中継やインターネット電話。
  映像や音声は、少しくらいパケットがなくなっても困りません。
  そういった、少しくらいコマ落ちしても困らない映像系や IP 電話などで使われます。

UDP は動作が軽いかわりに、TCP のような「転送の保証」や「重複回避の保証」はありません。

6.1.2 UDP パケット

図 6.1 は UDP パケットの概念図です。TCP パケット (5.3.3 節) の図と同様に、横幅は 32 ビットで、32 ビットごとに改行してあります。

UDP ヘッダは最初の 2 段分だけです。その後 (図 6.1 の data 部分) は UDP データ部で、情報本体が続きます。

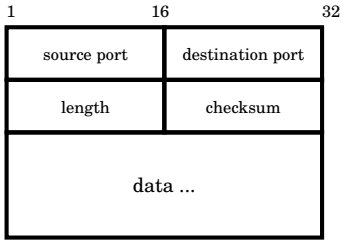

図 6.1 UDP パケット

最初の一段目は TCP と同じく、送信元と受信先のポート番号です。大きさは 16 ビット。それに続いてパケットの長さの情報と、パケットが壊れていないかを確認するための最低限のチェックサムがあるだけです。

非常に単純であることが分かりますね？

6.1.3 事例: DNS

第 4 章を復習して下さい。

DNS では、何度も問い合わせが発生し、多くのパケットがサーバクライアント間を往復します。ただ、小さなパケットが大量にやりとりされることが普通です。

そのため DNS の設計時には「動作が軽い UDP を使う」ことが選択されたと考えられます。

6.1.4 事例: IP 電話

インターネットを使った電話のことです。

電話のような音声会話では、人間が全部の音を聞きとれなくても問題ありません。少しくらいなら、聞きとれない部分を人間が自動的に補間します。

逆に、確実な転送を行なおうとすると、TCP (第 5 章) で見たように再送処理が発生し、その待ち時間の間、音声が途切れてしまいます。

音声は、少しくらいデータが欠落しても、正しいタイミングで音が届けば、人間にとって自然に聞こえます。

そのため、IP 電話 (もしくはインターネット電話) では UDP が適しているというわけです。

インターネット中継などの映像系のアプリケーションでも同様の理屈が成立します。

6.1.5 事例: TFTP

TFTP (Trivial FTP) はネットワーク機器 (ルータやスイッチ、ファイアウォールなど) との通信に使われる UDP ベースのプロトコルです。

主にファームウエアの更新や設定のバックアップなどに使われています。

通常、これらの作業はインターネットごしではなく PC と機器をほとんど直結させた状態で行なうため、通信回線の品質は高いと考えて問題ありません。そのため、TCP を使わなくとも UDP で十分と判断できます。実際、UDP で十分です。

また、PC などとは異なり、家電製品の一種であるネットワーク機器は CPU も遅く、メモリ量も少ないことが普通です。ファームウエアの更新や設定のバックアップの仕組みは必須ですが、本業ではないそれらの仕組みのために、ファームウエアが肥大化してはいけません。それでは本末転倒です。

そこで TFTP という簡単な仕組みが実装されているわけです。

コラム: 家電製品

「家電製品」といっても冷蔵庫や洗濯機のことではなく、ネットワーク屋が言う「家電製品」というのは

> いきなり電源を落しても壊れないように作られている機械。

といった意味合いです。

PC は家電製品ではありません。普通の PC は、突然電源を落すと壊れる可能性があります。壊れるものの代表例が PC のハードディスクです。

たとえばハードディスクのような高速回転体を正しい手順で止めなかった場合、どうなるか分かりません。運悪く、ちょうど書き込みをしようとしていた瞬間であれば、ハードディスクを物理的に傷つけるかもしれません。

ソフトウエアのレベルでも問題があります。OS のファイルシステムは、正しい手順で書き込みをしなければなりません。書きかけの時に電源を落されると中途半端な状態になってしまいます。だからハードディスクとちがい回転してしない仕組みのフラッシュメモリを使っていたとしても、とつぜん電源を落せば、壊れることがありえます。

コラム: tftp を実演するか？

DNS も目に見えないし、やはり、ルータのアップデートデモでもするかね (準備が間に合えばやろう)。

6.1.6 参考文献

RFC768 J.Postal, ”User Datagram Protocol”, 1980.

6.2 HTTPS

6.2.1 事例: アマゾン

インターネット通販の代表例ということで、アマゾン (Amazon [*1]) を例に出しますが、まともな通販サイトであればどこでも同様の動作になっています。

インターネット通販をする際には、よく URL の先頭部分 (左端の http:// や https:// の部分) を見ていて下さい。

アマゾンのページをめぐり、商品をショッピングカートに入れている間、URL は HTTP (http://...) です。最後に「レジに進む」をクリックし自分のアカウントでログインすると URL が HTTPS (https://...) に変わっていますね？

ブラウザの中には HTTPS に切り替わるとブラウザの隅にある鍵マークが「鍵がかかっている」状態へ変化して「安全なモードです」ということをユーザに知らせるものもあります。

この安全な状態になってから、はじめて「商品の発送先指定」や「クレジットカード情報」などの “個人情報” をアマゾンへ伝える仕組みになっています。

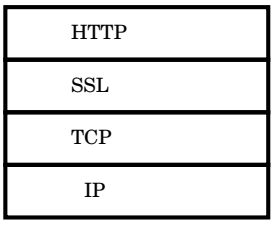

図 6.2 HTTPS の階層図: HTTP と TCP の間に SSL の層が挟まっている。

6.2.2 安全な HTTP

HTTPS は「安全な通信」を実現するために拡張された HTTP です。

もともと HTTPS はインターネット通販などで安全な通信を行なうため、ネットスケープ (Netscape) 社のブラウザ (Firefox の先祖) に実装されました。

HTTPS は HTTP over SSL (Secure Socket Layer) となっていて、図 6.2 のように HTTP と TCP の間に「暗号と認証の層」を行なう SSL 層を入れます。

こういった拡張が階層モデル (2.4 節) の御利益です。TCP/IP プロトコル体系では、こういった実装の拡張が簡単に実現できることに注目して下さい。

この仕組みは広く使われ、のちに IETF で SSL 3.1 をベースにした TLS (Transport Layer Security) という規格にまとめられました。そのため、ほぼ TLS と SSL は同義ですが、微妙な違いがあります。使っているアプリケーションによって用語が TLS だったり SSL だったりと若干の混乱が見られることもありますので注意が必要です。

TLS 自体は汎用的な考え方です。階層モデル (2.4 節) なので、どんなプロトコルでも同様に TLS 層を挟み込めば「暗号と認証の層」を行なうプロトコルがすぐに実装できます。

実際 HTTP 以外のプロトコルでも SMTP over TLS や POP3 over TLS などが使われています。

6.2.3 奇麗なインターフェイス

きちんと作られた階層モデル (2.4 節) でなければ、うまく TLS を挟み込むことが出来ないことに注意して下さい。

各階層間 (HTTP と TCP の間、TCP と IP の間など) で「インターフェイスが奇麗に作られていれば簡単に積み重ねることが可能」というところが重要です。

[*1] http://www.amazon.co.jp/

コラム: HTTPS なら安全か？
HTTPS を使いさえすれば安全か？というと、そんなことはありません。
認証の手続きがいいかげんであれば、やはり危険です。
また、暗号レベルが低いなら、やはり危険です。いまどきの PC の演算能力を持ってすれば暗号など瞬時に解かれてしまいます。

コラム: 自衛
インターネットセキュリティの回もあるので、簡単に:

- つねに最新版の OS を使う。
  「ファイルを開く」だけなら問題ないだろうということはありません。ユーザが見ていなくても、OS が勝手にファイルを調べて、その際に感染したりします。もともとはユーザの利便性のために事前調査をしているつもりなのでしょうが、大きなお世話です。そのせいで、メールを開かなくてもウィルスに感染したり、WWW もトップページにアクセスしただけでウィルスに感染したりします。
- ウィルスチェッカは必須。
- メールや WWW に書かれている URL を開かない。
  もちろん SPAM メールや怪しげな WWW サイトであればなおさら信じないこと。
- インターネット通販などは慎重の上にも慎重に。
  インターネット通販サイトは、誘導されていった URL ではなく、明示的に URL をブラウザに打ち込む。
  取り引き関係がめったにない通販サイトや初めての通販サイトでは、クレジットカードを使わないこと。荷物を受けとる際に代引を使うのがよい。300 円くらいの代引手数料は保険料。
- 個人情報をもらさない。
  通信販売に必須な情報以外は嘘を書いておくのもよい。たとえば、通販サイトのアンケートなどで本当の生年月日などを書く必要はない。また、サイトごとに少しずつ違う情報を書いておくと、どこから洩れたのか追跡することが出来るのでお薦め。

6.3 まとめ

1. 階層モデルの各階層を、別のものに入れ替えたり、階層の間に新しい階層を追加して、通信プロトコルを拡張できる。
2. 信頼性を求めるなら TCP だが、信頼性よりも動作の軽さや実装の容易さを優先する場合がある。
   (a) 動作の軽さを優先した DNS
   (b) 実装が容易で、安定した LAN の中でしか使わない前提: TFTP, DHCP など
   (c) 少しくらい欠落しても人間に分からないたぐい: 電話や動画
3. 元々の WWW は、HTTP を使うが、これでは安全なインターネットショッピングが出来ない。そこで、暗号化と認証を行う階層 (SSL) を追加することで HTTPS を作成した。

第 7 章

ネットワークの構造

今回から話はレイヤー 3 に入ります。

学内ネットワークを例に、レイヤー 3 で使われる機材、それらの機材を使ったネットワークの構築について概観しましょう。

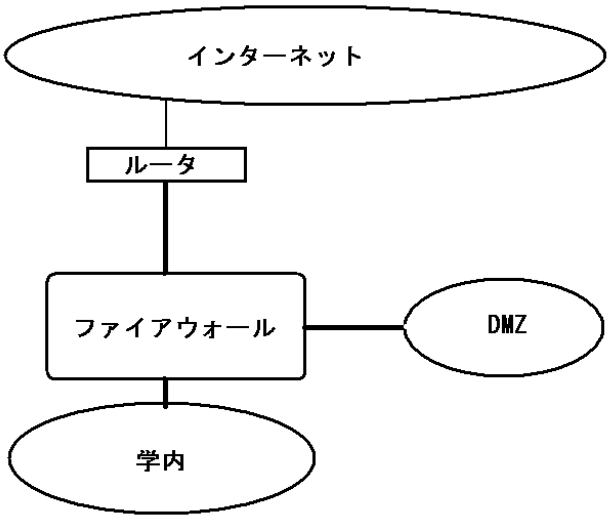

図 7.1　学内ネットワークの概念図: ファイアウォールを境界として、学外 (インターネット)、DMZ、学内が接続しあっている。なお、ファイアウォールだけではインターネットへ接続できない。インターネット接続のためには別途ルータという装置が必要。

7.1　用語

組織の内側 (学内や社内) にあるネットワークはLAN (Local Area Network) と呼ばれています。

「同じ建物の中にある機器が接続したネットワーク」のように、文字どおり「ローカルにある」ネットワークが LAN です。

逆にインターネットのように大きく広がったネットワークのことをWAN(Wide Area Network) と呼びます。Local の反対で Wide 。これも文字どおりですね。

地理的に離れたところにある機器を接続したネットワークで、LAN 同士を”専用線”などでつないでいます。たとえば、東京オフィスと千歳オフィスを (インターネットを越えて) つなぐ[*1] といった大きさのネットワークが WAN です。大学とプロバイダとの接続なども WAN と呼びます。

7.2　学内ネットワーク構成

本学のネットワーク構成は 1990 年代なかば以降の典型的な設計方針にもとづいています。

この設計方針は一般企業のものと同じです。というのは、大学ネットワークを設計した当時、すでに企業向けのデファクトスタンダードなネットワーク設計方針が確立されていたため、その方針をそのまま採用しています。

[*1] 東京オフィスの社内と千歳オフィスの社内をつなぐ時は、インターネットを越えて、仮想的に社内ネットワークを広げます。これがVPN (Virtual Private Network) です。

7.2.1 ネットワークの各部分

ネットワークは次のような 4 つの部分から構成されます (図 7.1)。

- インターネット (接続部分)
- 境界
- 学内
- DMZ

「境界」は「学外」(インターネット) と「学内」の境目となる部分で、ファイアウォール (B.4 節) を設置します。

ファイアウォールは「 DMZ 」(後述) と呼ばれる特別な構造を持つ場合があります。一般企業や大学では、インターネット向けの WWW サーバを用意する場合、DMZ に設置するのが普通です。

つまり、ファイアウォールは最低 3 つの部分「インターネット (接続部分)」「DMZ」「学内」とつながりを持つことになります。

なお、ファイアウォールがインターネットへ直接つながっているわけではありません。ファイアウォールのインターネット側インターフェイスはインターネットの一部ですが、もっと低いレベル (つまりレイヤー 1 とかレイヤー 2 の意味) でインターネットと通信するために別途ルータという装置が必要です。

ちなみに、安価な製品ではファイアウォールとルータの役割が一つの機械の中に入っているものもあります。

7.3 ネットワーク機材

ネットワークの各部分で用いられる機材について説明します。

7.3.1 用語

先に進む前に、二つ三つ用語を説明します。

スイッチ (switch) と PC だけで構成された部分 (コンピュータネットワークの一部分) をセグメント (segment) と呼びます。

このセグメントは実際に機械が物理的に線 (たとえばイーサネット) で継っている範囲のことです。そのため物理セグメントとも呼ばれます。

スイッチはセグメント内での通信を行なう機械です。この授業の場合、通信とはパケットのやりとりのことですから、スイッチは PC から PC へ IP パケットを転送する装置と言えます。

IP パケットの転送は IP フォワーディング (forwarding) と呼ばれます。

なお、スイッチの動作についてはイーサネットの回で説明します。

注意: なお、ネットワークという単語は曖昧で便利な単語です。本章のような学内ネットワークやネットワークの構築といった文脈では “ネットワーク” という単語がセグメントを意味することが多いと思います。いろいろとまぎらわしいですが、文脈をよく考えて見分けるしかありません。

7.3.2 インターネット (接続部分)

“インターネットへ接続する” という表現も曖昧模糊です。

前述の通り、一般の人は WWW が使えていれば「インターネットに接続している」と思うでしょう。この授業では違いますね？

9.3 節で解説しますが、インターネットの実体は、数万の「プロバイダ」群とそのプロバイダのいずれかに接続する組織 (大学、企業、個人) の集合体です。なおプロバイダのほとんどが商用プロバイダつまり営利企業で、ごく一部 SINET のように国営の非営利プロバイダなどがあります。

「インターネットに接続する」ためには、いずれかのプロバイダと大学が接続契約を結ぶ必要があります (図 7.1 でいえばルータとインターネットをつないでいる部分)。

プロバイダと大学間は NTT などのキャリア (第一種通信事業者) から借りた通信回線でつなぎます。その通信回線の両端にはルータ (router) と呼ばれる装置が必要です。

この通信回線の物理媒体には光ファイバーが主に使われています。個人向けサービスでも光ファ

イバーが多くなってきましたが、ADSL や ISDN を使っている家庭もまだまだあります。

ルータもスイッチも同じ IP フォワーディングを行なう装置ですが、スイッチは同一セグメントの中の IP フォワーディングを、ルータは異なるセグメント間の IP フォワーディングを行なうことが主目的です。

7.3.3 境界

境界にはファイアウォール (B.4 節) を設置します。

ファイアウォールは複数のネットワークインターフェイスを持ち、アクセス制御をしたり、ファイアウォールを通る通信の記録を取るための装置です。

アクセス制御

ファイアウォールでは、必須のサービスだけをとおすようにします。

図 7.1 の場合、通信方向は次の数だけあります。

- 学内 → インターネット
- 学内 → DMZ
- インターネット → 学内
- インターネット → DMZ
- DMZ → インターネット
- DMZ → 学内

各方向それぞれについて、最大限に厳しい設定をするべきです。

ログ

通信記録を取っておくことは重要です。

第一に、インターネットからの不正なアクセスがあった場合に、どこからの攻撃であったのかを突き止める手がかりになることがあります (ファイアウォールにも侵入されて記録を消されていたら追跡できませんが;-)。

第二に、学内にある PC のどれかがウィルスに感染し、学内からインターネットへ攻撃やウィルスの送信をしてしまった場合に、その犯人を突き止めることが可能です。

そういった攻撃の追跡以外にも、通信記録から、どのような通信が多いのか？を分析し、ネットワークの構成を最適化するといったことにも応用できます。

ログの保存、そしてログの分析テクニックはネットワーク技術者にとって最も重要です。

防壁？

ファイアウォールを置いておけばインターネットから攻撃されない、というのは間違った思い込みです。

確かにファイアウォールがあれば「インターネットから学内の Windows の Microsoft Network へ直接攻撃」といったつまらないミスは防げます。

しかしながら、ファイアウォールに出来ることは、こういった単純ミスのようなものに対しての攻撃を防げるだけで、いやがらせであるサービス不能攻撃 (Denial of Service) や、HTTP プロトコルよりも上位な世界での攻撃 (例: RDBMS を使っている学内ポータルシステムへ SQL injection をかけてサーバへ侵入する) といった複雑な攻撃を防げるわけではありません。

ファイアウォールとは、あくまでも最低限必須な装置にすぎません。逆に、サーバや自分の PC の設定に自信があればファイアウォールなど不要です。

しかしながら、一つでも穴があればセキュリティが守れません。そして、穴がないという自信は全く持てません。しかたがないので、学内全体をファイアウォールでおおうのです。そうやって、弱点をインターネットにさらさないようにする装置がファイアウォールと考えて下さい。

ファイアウォールとは、その程度のものなのです。

7.3.4 DMZ

DMZ はインターネットに公開するサーバを設置するセグメントという特別な役割をになっています。

ファイアウォールを経由しなければ DMZ と

の通信は不可能です。サーバの通信にはファイアウォールによるアクセス制限をかけ、最低限の通信しか許可しないようにします。

これにより、つまらないミスを覆い隠すことが可能です。

逆に言えば、それしか出来ません。もちろん高度な攻撃をされれば、突破されます。逆に、きちんとしたサーバが作れるなら DMZ に置く必要はありません。

たいていの人は、きちんとサーバを作れないし、つねにサーバを正しい状態に維持し続けることが難しい、といった話の前提条件があることに注意してください。

7.3.5 学内

学内もセグメントですが、かなり大きなセグメントであることが普通です。

学内に数百台の PC がある以上、一つのセグメントでは使いづらいので、複数のセグメントがあります。また、セグメントが仮想的に建物を越えて広がっています。

前述のように、本来のセグメントとはスイッチと PC だけで構成されたもので、一つの部屋の中とか、大きくても一つの建物の中に広がっているものです。

ただ、これでは使いづらいということで、仮想的に建物を越えて広がったセグメントが使えるように拡張されました。この論理的なセグメントは VLAN (Virtual LAN) と呼ばれています。

コラム: スイッチとハブの違い

現在ではスイッチと呼ぶことが普通ですが同じ役割をする装置には、リピータ、ブリッジ、ラーニングブリッジ、ハブ、スイッチングハブなどがあります。

それぞれ機能が少しずつ異なりますが、使い方はどれもスイッチと同様です。

スイッチという用語はパケットフォワーディングを専用 LSI が行なうものを指すことが普通です。この専用 LSI は特定用途向け集積回路 ASIC (Application Specific Integrated Circuit) と呼ばれています。

コラム: L ◎スイッチ ？

最近では何にでもスイッチという単語をつけるので、何だか良く分かりません。本来は L2 スイッチのことですが、L3 L4 L7 スイッチと呼ばれている装置もあります。また、同じ単語でもメーカーによって微妙に異なることが普通です。

「L2 スイッチ」はレイヤー 2 で動作するスイッチのことで、いわゆるスイッチに相当するものです。ヨドバシなどの家電量販店で買える安物のスイッチから高価なものまで色々あります。

「L3 スイッチ」はレイヤー 3 (つまり IP アドレス) 情報も扱えるスイッチということで、ルータ機能もあるスイッチを指すのが普通です。どのメーカーでも、たいていは L2 スイッチの上位機種に L3 スイッチの製品があります。基本的にルータ (後述) と同じものですが、ルータはルーティング機能を重視し、L3 スイッチはルーティングよりパケットフォワーディング性能を重視している機器というのが用語の慣習になっているようです。

「L4 スイッチ」はレイヤー 4 (つまり TCP や UDP のポート番号) 情報を扱えるスイッチです。たとえば「HTTP は、特定の方向へフォワーディング (物理ポート B へ曲げる)」などの機能があるものをさします。

「L7 スイッチ」は、さらに特殊で、レイヤー 7 つまりアプリケーションプロトコルの詳細を理解できる機械です。たとえばインターネット接続回線が 2 つ、通信回線 A (高価) と B (安価) がある時、 HTTP の URL を見て、通常は A を使いますが、通信相手が youtube なら回線 B を使うといった制御ができます。

いずれにせよ L4 スイッチや L7 スイッチはプロバイダなどが使う特殊な機材です。

7.4 VLAN

前述のように学内ネットワークは複数のセグメントから構成されています。セグメントの例は「PC 教室のセグメント」「研究室で使っているセグメント」「事務のセグメント」「ネットワーク管理セグメント」などです。

たとえば「PC 教室のセグメント」ですが、PC 教室は G201 と G202 だけではありません。G202 の前にある PC コーナーも PC 教室と同じセグメントですし、大学院棟の F104 にある PC 部屋も同じセグメントです。

7.4.1 セグメントの表記

PC 教室のセグメントは 172.23.0.0/16 です。詳しくは後述しますが、このセグメントの PC につけてよい IP アドレスは「172.23. 数字. 数字」で、セグメントの大きさが /16 の 16 です。

いまのところは

セグメントごとに IP アドレス帯が異なる

ことだけ覚えておいてください。とりあえず VLAN の話を先にしましょう。

7.4.2 LAN と VLAN

LAN

元々の LAN は、一つのスイッチにつながった PC 群のことと考えてもらってかまいません。

もっとも一つの机の上サイズの LAN ばかりではありません。大きな部屋ないしは一つのフロア全体、一つの建物全体といった LAN もありえます。必要に応じて複数のスイッチを使ったり電気的な増幅が必要です。

それでも、これらは物理的に一つながりの一つのセグメントでした。

建物を越えるには？

普通の LAN でもセグメントごとに通信回線を用意すれば出来ます。

つまり建物間に 10 本の光ファイバーを引いてしまえば、物理的に建物を越える 10 のセグメントを構築することが出来ます。

しかしながら、この構築方法は改造するのが大変です。

11 番目のセグメントを新設する時は、もう一

本、光ファイバーを敷設しなくてはなりません。土木工事です。大変な費用がかかります。

もっと小さな変更、たとえば部屋を引っ越す場合、引っ越した先が違うセグメントだったとしたら、回線と機材のつなぎ替えなどが必要になります。

いずれにせよ、いろいろと面倒です。

VLAN

では、建物間にある一本のファイバーが仮想的に 10 セグメント分の役割をしてくれたらどうでしょう？

実際にそういったものを構築する際には、一本の光ファイバーと、そういった特別な機能を持ったスイッチを使い、スイッチがあたかも論理的には 10 セグメントあるかのように幻影を見せてくれます。

これなら、11 番目のセグメントを新設する際に、土木工事は不要であり、スイッチの設定をするだけですみます。容易ですし費用も段違いにかかりません。同様に部屋の引越しも大半はスイッチの設定だけで終ります。

このような仕組みをVLAN(Virtual LAN) と呼びます。一つのスイッチの中に「複数の仮想スイッチを作る」と考えても良いでしょう。

この場合、セグメントは建物を跨いで論理的に広がっており物理的な配置はよくわかりません。このようなことがセグメントは論理セグメントとも呼ばれています。

7.4.3 ポート VLAN

VLAN の使い方にもいくつか種類がありますが、基本となる考え方は「スイッチの物理ポートが、どの VLAN に属するか？」という使い方です。

スイッチの物理ポートごとに VLAN を指定します。たとえばポート 1 番は「PC 教室のセグメント」、ポート 2 番は「研究室のセグメント」といった具合です。つまり挿したポートによって VLAN つまりセグメントが異なります。

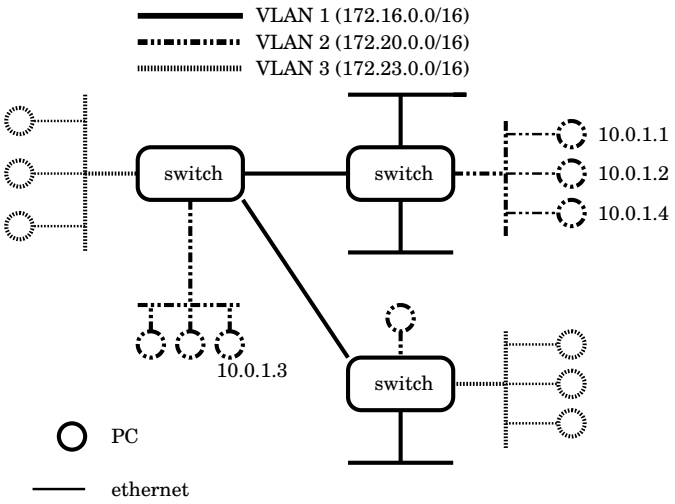

図 7.2 VLAN 概念図

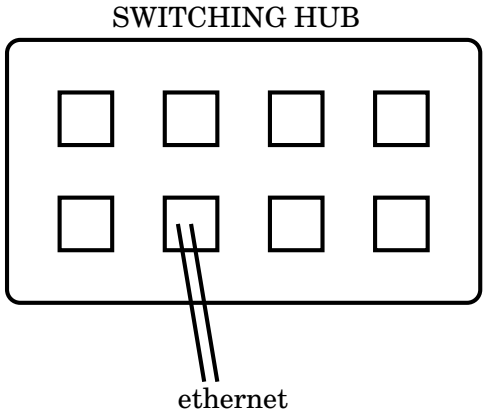

図 7.3 ポート VLAN 概念図: スイッチにポートが 8 つあるとして、4 つずつ別のセグメントにする設定を入れる。赤いポートにイーサネットを接続すると赤のセグメント、緑のポートにイーサネットを接続すると緑のセグメントとなる。

これをポート VLAN(Port VLAN) と呼びます。

```
CISCO の設定例

... 略 ...

interface GigabitEthernet0/1
 switchport access vlan 100
 no ip address
!
interface GigabitEthernet0/2
 switchport access vlan 100
 no ip address
!
... 略 ...
interface Vlan100
 ip address 10.0.100.1 255.255.255.0
!
ip classless
ip route 0.0.0.0 0.0.0.0 10.0.0.1
no ip http server
... 略 ...
```

コラム: カタログに書いてあるからといって使っていいわけではない。

「一つの物理ポートに複数の VLAN を割り当てる」方法もありますが、複雑になるのでやめた方が良いでしょう。

ポート VLAN ですら使いたくないのに、こんな複雑な設定をされたら障害時にデバッグできません。

無数にある「『カタログで出来ること』と『やって良いこと』は別」という例の一つです。

7.5 VLAN のメリット/デメリット

7.5.1 VLAN のメリット (1)

まず VLAN を使うと、ネットワーク機器が物理的な配置と無関係になります。たとえば PC の IP アドレスを変えずに機器を移動できます。よって、部屋の引越しや機器の追加、ネットワークの物理的な変更があっても、それはスイッチの設定変更のみで対応可能です。作業は数分ですみます。

例: 階をまたいだ LAN

部屋 E101 (フロアは一階)、E201 (二階)、E304 (三階) は同じ研究グループだから同じセグメントにするといったことがスイッチの設定変更だけですぐできます。

例: 引越し

このグループが部屋の引越しをして部屋 E101 E210 E211 を使うことになったとしてもスイッチの設定変更だけで対応可能です。

7.5.2 VLAN のメリット (2)

Windows の「マイネットワーク」は同一のセグメントでないと利用できません。この (つまらない) 点で VLAN は Windows の人には重要です。

正直、われわれ Unix の人はどうでもいいとおもっているメリットなのですが、世間的には「マイネットワーク」が使えないか否かは重要 (だそう) です。

Microsoft Windows Network では「マイネットワーク」をクリックすると隣にある PC の一覧が分かりますが、これは Windows がブロードキャストを使っているためです (詳しくは 10 章を参照)。

実際には同一セグメントでなくても「マイネットワーク」を使えなくはないのですが、マウス一つで設定できるというものではなくファイルの編集等が必要なので素人には難し過ぎます。

7.5.3 VLAN のデメリット

物理的な構成と実際の構成が異なるので、管理する上では直観的に分かりにくいネットワーク構成が作られてしまいます。

物理的な構成と無関係であるため、なにかと管理しにくいし、抜け穴が出来てしまいます。セキュリティ上、大問題です。

と現場の人間 (ネットワーク屋) は思うのですが、実際には VLAN は多用されています (VLAN を設計するのは SI 屋だから)。

VLAN のメリットを乱用しがちなため、どんどん分かりにくいネットワークになってゆきます。いざ障害が起きた場合に故障箇所が特定できません。

物理的な構成と異なるので、セグメントの出口で防衛するといった概念が適用できないのも頭が痛いところです (実話)。

図 7.4 がその実例です。赤いセグメントだけにウィルスチェック (透過型) をかけたいとします。物理的なセグメント構成であれば、チェッカーをセグメントの出口におけばよいのですが VLAN ではどうしたらよいでしょうか？

答えは「不可能」です。

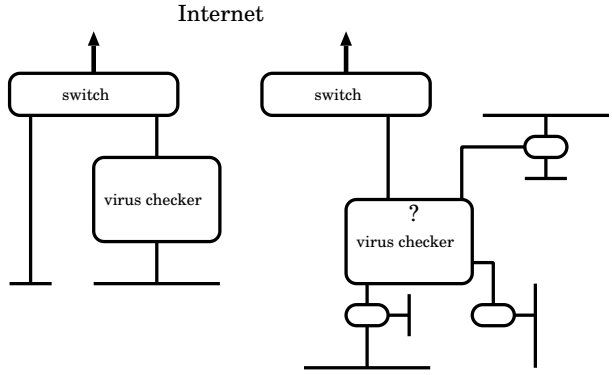

図 7.4 構成例: 透過型ウィルスチェッカーを置きたいが置けない。

7.6 IP アドレス

前述のとおり IP アドレス[*2] はインターネット上の住所に相当するものです。

その正体は 32 ビットの数字、たとえば二進数表現で 1100 0000 1010 1000 0000 0000 0000 0001 ですが、人間には扱いづらいので、8 ビットずつに区切り十進数 4 つで表現することにしています。

```
11000000 10101000 00000000 00000001
                ↓
          192.168.0.1
```

7.6.1 正確には住所ではない

正確にいえば IP アドレスは住所ではありません。出口につける印と考える方が正しい理解です。

家に 3 つ入口があったとしても住所は同じです。しかしながら、インターネットでは家の入口を区別します。その入口とはネットワークインターフェイスのことです。

たとえばファイアウォールは DMZ も入れると最低 3 つのネットワークインターフェイスを持っています。そしてそれぞれのインターフェイスは異なるセグメントにつながっています。ということは、それぞれのインターフェイスが違う IP アドレス帯の IP アドレスを持つということでしょうか？

答えは YES です。

ネットワークインターフェイスごとに異なる IP アドレスを持たないと通信ができません。

ただ、多くの PC は一つのネットワークインターフェイスしか持たないので、一つの PC には一つの IP アドレスが対応します。そのため、普通は IP アドレスを住所と考えていて問題ありません。

つまり、家に入口が一つしかない場合は、住所と「入口につけた印」は同等の意味になるというわけです。

7.6.2 IP アドレスの種類

- ユニキャスト (unicast)
  一対一の通信のこと。IP アドレスといえば、普通ユニキャスト。ただし特別な意味のあるユニキャストもある。
  - ループバックアドレス
  - プライベートアドレス
- マルチキャスト (multicast) 一対多の通信
- 予約されている (使わない) 範囲

7.6.3 IP アドレスの表記法

IP アドレスは「数字. 数字. 数字. 数字」か「数字. 数字. 数字. 数字/数字」で表現します。

たとえば PC 教室の PC には 172.23.1.190 のような IP アドレスがついています。

[*2] 注: 正確には IP バージョン 4 のアドレスです。

演習: CentOS 上で /sbin/ifconfig -a コマンドを実行して確認してみよう。

この PC 教室のセグメントは 172.23.0.0/16 です。

この /16 部分は「ネットワークの範囲」ないしは「セグメントの大きさ」を意味しています。「セグメントの大きさ」とは利用可能な IP アドレスの数です。

利用可能な IP アドレスの数は「32 - 数字」ビット分です。つまり /16 では 32 - 16 = 16 ビットつまり 65535 個の IP アドレスが利用できます。

「/数字」は大きな数字ほど利用可能な IP アドレスの数は少なくなります。最大は 32 で、利用可能な IP アドレスは一つだけです (/32 では 32 - 32 = 0 ビットつまり 1 個)。

たとえば PC についている 172.23.1.190 は 172.23.1.190/32 です。通常は /32 を省略して表記します。

7.6.4 例: 172.23.0.0/16 セグメント

172.23.0.0/16 のセグメント内にあるネットワーク機器へつける IP アドレスは 172.23.0.0/16 (172.23.0.0 ～ 172.23.255.255) の中から選ぶ必要があります。

このセグメントの IP アドレスは 172.23.1.190 や 172.23.200.10 のようになります。左側の 172.23 部分は固定で、IP アドレスの右側部分だけが可変です。

この固定されている 172.23 の部分を IP アドレスのネットワーク部分と呼びます。ネットワークアドレスと呼んでいます。逆に 172.23 より右側の可変部分はホスト部と呼ばれます。

この 172.23 は IP アドレスの先頭 (左側) 16 ビット分にあたります。これが/16 の 16 です。利用可能な IP アドレスは残りの部分つまり IP アドレスの右側部分は 32 - 16 = 16 ビット分です。

「千歳市本町一丁目 1 番地」を例にとり、やや強引な類似をすると「千歳市本町一丁目」全体が「ネットワーク」「千歳市本町一丁目 1 番地」まで特定するのが「IP アドレス」と考えてください。

利用可能な IP アドレスの範囲の端にある 172.23.0.0 と 172.23.255.255 は一般のネットワーク機器には使えません。172.23.0.0 はネットワークアドレス、172.23.255.255 はブロードキャストアドレスと呼ばれています。ブロードキャストがどのような仕組みになっているのかについては 10 章で説明します。

7.6.5 ネットワークアドレス

セグメントの先頭 IP アドレスのことです。二進数表現ではホスト部分のアドレスを全部 0 にしたものといえます。

たとえば、172.23.0.0/16 なら 172.23.0.0 がネットワークアドレスになります。10.0.0.0/24 なら 10.0.0.0 がネットワークアドレスです。

ネットワークを意味する (代表する) アドレスですが、何にも使っていません。もったいない気もしますが、使いません:)

BSD の初期の実装では、ブロードキャストを意味しました。ものすごく古い機器では、これをブロードキャストとして使うことがあります (さすがに最近は見ないかもしれない)。ネットワークアドレスを PC などに使わないのは、この問題を回避するためという意味もあります。

注: 前述のブロードキャストの説明でも 172.23.0.0 が出てきていることに注意してください。

7.6.6 ブロードキャストアドレス

そのセグメントで利用可能な IP アドレスの最後のものです。

たとえば 10.0.0.0/24 なら 10.0.0.255 がブロードキャストアドレスです。二進数表現ではホスト部分のアドレスを全部 1 にしたものになっています。

すべての機器 (10.0.0.1 ～ 10.0.0.254) に通信

を行ないたい時にブロードキャストを使います。詳しくは 10 章で取り上げます。

7.6.7 演習: 10.0.0.0/24

確認してみて下さい。

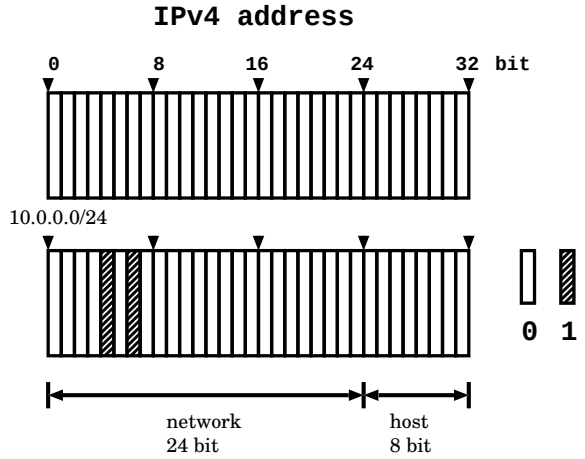

図 7.5 例: 10.0.0.0/24

- 10.0.0.0 から始まる /24 の大きさのネットワーク。
- 10.0.0.0 ～ 10.0.0.255 まで 256 個分の IP アドレスが利用可能。
  32 - 24 = 8 ビットつまり 256 個であるという意味
- 先頭から 24 ビット (つまり /24 部分) がネットワーク部。
- また、最後の 8 ビット (つまり 32 - 24 bit) がホスト部。
- 10.0.0.0 がネットワークアドレス
- 10.0.0.255 がブロードキャストアドレス
- PC などに利用可能な IP アドレスの範囲は「ネットワークの大きさ - 2」。
  先頭と最後の 2 つのアドレス、ネットワークアドレス (先頭) とブロードキャストアドレス (最後) は利用できない。

なお、この例では、たまたま . 区切りと一致するので、10.0.0 の部分がネットワーク部、一番右の「. 数字」部分がホスト部ということになります。これは /24 という区切りの良い例であるためで、/25 などでは奇麗な形にはなりません。これは十進数表記をしているためです。二進数表現で見れば別に不思議なことは何もありません。

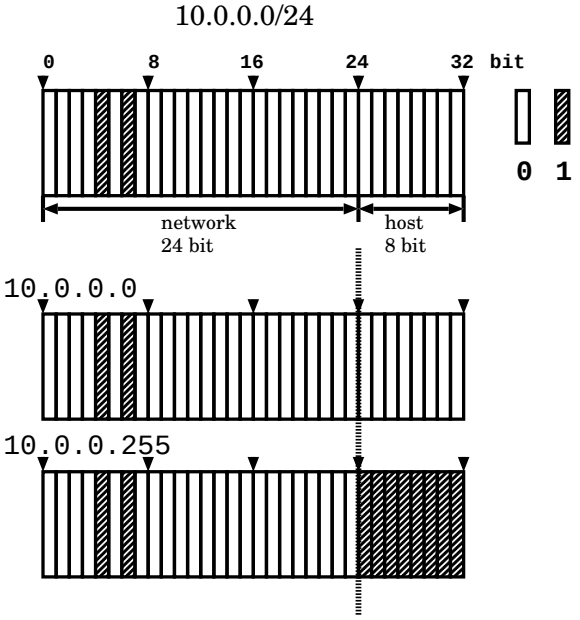

図 7.6 例: 10.0.0.0/24 セグメント。ブロードキャストも表示

7.6.8 ネットマスク

現在は 172.23.0.0/16 のような表記法が標準なので本来は不要なのですが、この表記法以前の時代にはネットマスクを使った表示方法が使われていました。

たとえば 172.23.0.0/16 は「アドレスが 172.23.0.0 で、ネットマスクが 255.255.0.0」のセグメントと呼んでいました。

10.0.0.0/24 ならネットマスクは 255.255.255.0 です。このアドレスは /24 の 24 だけ左から 1 を並べたものだということが分かるでしょうか？/16 の 255.255.0.0 も同じく左から 1 を 16 個並べたものです。

ようするに時代とともに表記法が変わっただけなのです。

ただ、ネットワーク機材の多くは過去との互換性のために、このネットマスク表記を使わないといけない場面もあります。一般人は指示にしたがって入力するだけで十分ですが、ネットワーク技術者は、瞬時にネットマスクと / 表記を切替えできるように訓練して下さい。

7.6.9 演習: 10.0.0.0/25

確認してみて下さい。

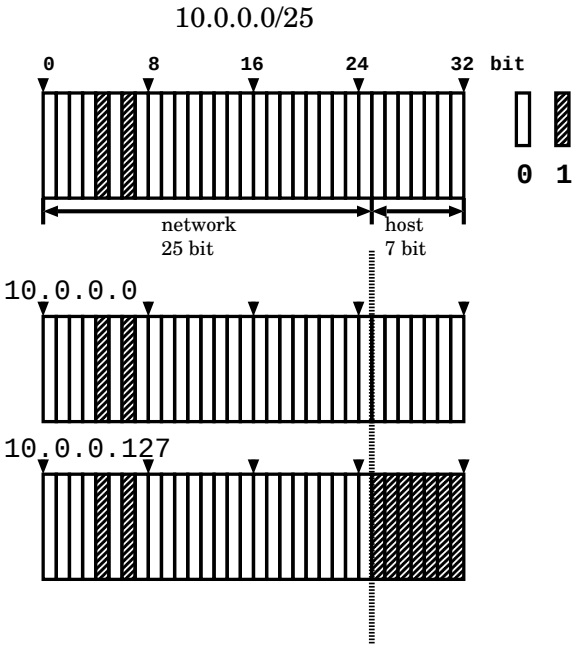

図 7.7 例: 10.0.0.0/25

- 32 - 25 = 7 ビット = 128 個。
- 利用可能な IP アドレスは 10.0.0.0 ～ 10.0.0.127 で 128 個。
- 10.0.0.0 がネットワークアドレス。
- 10.0.0.127 がブロードキャストアドレス。
- 10.0.0.1 ～ 10.0.0.126 が利用可能な IP アドレス (126 個)。

7.6.10 演習: 10.0.0.128/25

確認してみて下さい。

- 32 - 25 = 7 bit = 128 個。
- 10.0.0.128 ～ 10.0.0.255 (128 個)。
- 10.0.0.128 がネットワークアドレス。
- 10.0.0.255 がブロードキャストアドレス。
- 10.0.0.129 ～ 10.0.0.254 が利用可能な IP アドレス (126 個)。

クラス	説明
クラスA	/8 単位で組織に割り当てる
クラスB	/16 単位で組織に割り当てる
クラスC	/24 単位で組織に割り当てる
クラスD	マルチキャスト用
クラスE	予約、未使用

表 7.1 IPv4 アドレスのクラス

7.6.11 古い用語

これまた、オブジェクト指向言語の「デザインパターン」と同じで、単語を知らないと困るが、知っていてもたいして役には立たないという例です。でも実用上困るので覚えて下さい。

7.6.12 クラスA

IP アドレスの先頭 8 ビットが 00000000 ～ 01111111 までの範囲で、十進数表現では 0.0.0.0/8 ～ 127.0.0.0/8 にあたります。

これは /8 単位で割り当てる IP アドレス帯なので、大きな組織用です。/8 ということは 32 - 8 = 24 つまりホスト部分が 24 ビットあります。つまり最大 2^24 - 2 = 16777214 個の機器が接続可能です。

インターネットを昔からやっている組織には気軽に与えられていますが、この数年は早く返却しろという運動が行なわれているため、返却されたクラス A アドレスが一般のプロバイダに割り当てられるようになっています。たとえば日本でも自宅の IP アドレスが 69.1.2.3 であることは普通です。

利用している組織の例は、アメリカ軍 (.mil ドメイン) や IBM (9.0.0.0/8)、SONY (返却済み) など大規模ないしは世界にまたがる会社です。

7.6.13 クラスB

先頭の 8 ビットが 10000000 ～ 10111111 までなので、128.0.0.0/16 ～ 191.255.0.0/16 になります。

/16 単位で割り当てるので最大 $2^16 - 2 = 65534$ 個の機器が接続可能です。

中規模組織用といえます。そのため大学などによく割り当てられました。９０年代初期までにインターネットに接続した組織は、たいていクラスBを持っています。

たとえば北海道大学は 133.87.0.0/16 などを割り当てられています (正確には 2B + 1C)。

7.6.14 クラスC

先頭の 8 bit が 11000000 ～ 11011111 までなので 192.0.0.0/24 ～ 223.255.255.0/24 です。

/24 単位で割り当てます。$2^24 - 2 = 254$ のアドレスが利用可能です。

普通の組織用で一般の会社などに割り当てます。

90 年代なかば以降に接続した組織では割当単位を出来るだけ小さくしているためクラスCもしくはクラスCより小さい単位 (/24 以下) が普通です。必要に応じて /24 を複数割り当ててもらいます。

たとえば千歳科学技術大学は 210.128.51.0/24 210.128.52.0/23 です。

7.6.15 クラスD

先頭の 8 bit が 11100000 ～ 11101111 まで、224.0.0.0 ～ 239.255.255.255 です。

これはマルチキャスト用の特別な IP アドレスです。

IP アドレスの意味が異なり、住所の意味ではありません。放送局の周波数と考えて下さい。1 ch や 3 ch と同じように 224.0.0.9 というチャンネルへ送信といった具合に使います。

7.6.16 特別なユニキャストアドレス

特別なユニキャストアドレスに「ループバックアドレス」と「プライベートアドレス」があります。

ブロードキャストは動作自体が異なるのですが、これらのアドレスは普通の IP アドレスで住所の意味合いを持っていると考えてかまいません。

ただし、特別扱いされている住所になります。

ループバックアドレス

127.0.0.0/8 は歴史的に特別扱いされています。

127.0.0.1/32 という IP アドレスは自分自身を意味しループバック (loopback) アドレスと呼ばれます。自分自身で閉じる (自分から自分への) 通信で使います。

事実上 127.0.0.1/32 しか使いません。Unix では lo0 という名前のインターフェイス名です。

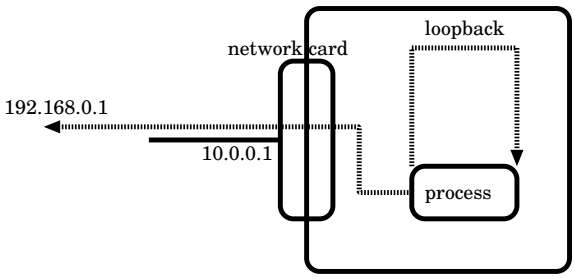

図 7.8 ループバック: ループバックはハードウエアは使わずソフトウエアで閉じる通信なので早い。

プライベートアドレス

組織が組織内で自由に使って良い IP アドレスが 3 種類あります。組織のサイズに合わせて適当に選ぶことになります。

- 10.0.0.0/8
- 172.16.0.0/12
- 192.168.0.0/16

本学では、学内のネットワークに 172.16.0.0/12 を使っています。

7.7 まとめ

学内ネットワークの構成要素

1. 図 7.1 を参照。学外 (インターネット)、学内、DMZ の三種類に分かれている。境界線にはファイアウォールというネットワーク機器を置く。
2. 図 7.2 を参照。学内側は VLAN で構成されている。違う建物にある G201、G202、B203、F106 が同じ PC 教室として利用できるのは同じ VLAN として構築してあるから。

IPv4 アドレス

1. 役割での分類
   (a) ユニキャスト (いわゆる住所としてのアドレス)
   特殊例としてループバックとプライベートアドレス。
   (b) マルチキャスト (いわば、チャンネル番号)
   (c) 予約 (利用できない)
2. 表記法での分類
   (a) クラス表記
      i. クラス A B C (それぞれ /8 /16 /24)
      ii. クラス D (マルチキャスト)
      iii. クラス E (予約、未使用)
   (b) CIDR

第 8 章

NAT〜デバッグ

前回、IP アドレスの用語について解説しました (7.6 節)。この講義は、他の講義と連続性がないので、多くの用語が新たに出てきますが、技術者として知らないと困る用語がほとんどなので、頑張って覚えて下さい。

さて、ようやく、用語の説明が終ったので、先に進めます。今回は、まずNAT (Network Address Translation) から始めましょう。

8.1 プライベートアドレスと NAT

前回、プライベートアドレス (7.6.16 節) という特別な IP アドレスがあることだけは、お話しました。

本節では、プライベートアドレスの動機、その使い方等について説明しましょう。

8.1.1 プライベートアドレス提案の動機

1980 年代〜蜜月〜

元々、IP アドレスには住所に相当する意味があります。つまり、IP アドレスがあなたの PC に割り振られたら、世界中のコンピュータの中から、あなたの PC が特定できるということです。

そもそも、本来の「通信」という観点から考えてみれば、通信相手について特定可能であるべきですから、通信する際に互いの住所がはっきり分かることは当然のことです。

さて、この「IP アドレスが住所に相当する」という考え方は 1980 年代には特に問題はありませんでした。インターネットの前身のネットワークを使っていたのは学術組織だけだったので、ユーザ数もたかがしれていました。当時、ネットワークの利用者は日本全体でも、せいぜい数万人[*1] しかいなかったでしょう。

IP アドレス (IPv4 アドレス) すべてを使えば 40 億台の PC に IP アドレスを割り振ることが出来ました。実際にはクラス A B C だけですが、それでも十数億台分は優にあったわけで、当時の利用者の数を考えれば、十分な余裕があったわけです。

1990 年代〜幼年期の終り〜

1990 年代に入るとインターネットの商用化が始まります。

すでに 90 年代前半の時点で、このままインターネットが発展しつづけたら IP アドレスが足りなくなりそうだ！ということは分かっていました。IP アドレスは、たとえクラス D や E まで転用してすべてを使ったとしても 40 億台分しかありません。地球の人口以下です。足りないことは誰にでも分かります。

ここで選択肢が分かれます。

- 次世代 IP 体系の開発と移行。
  いわゆる IPv6 です。
- 現行の IPv4 をなんとか延命させる小手先のゴマカシを考える。

[*1] なにしろ、当時、日本のネットニュースの投稿者の総数が 500 人しかいなかった時代です。

実際には、この 2 つの開発が同時並行で行なわれました。

後者が、プライベートアドレスの定義と NAT という小手先の回避技術です。

8.1.2 プライベートアドレス

そんなわけで、苦肉の策ですが、各組織が自由に使ってよいアドレス範囲を定義しました。それが 7.6.16 節で紹介した

- 10.0.0.0/8
- 172.16.0.0/12
- 192.168.0.0/16

の三種類です。

利用する際には、ネットワークの設計担当者が組織のサイズに合わせて適当なものを選択します。

コラム: プライベートアドレスは正確に覚えよう

10 は分かりやすいから間違えないと思いますが、社内ネットワークの IP アドレスに 172.33.1.100 とか 192.169.1.200 とか振らないように気をつけましょう。

これは笑い話ではなく、へっぽこ SI 業者がやっている実話です。

もっとも、重要顧客と通信できないとかいうことでもないかぎり、間違っていることがばれないというのも事実だったりします。とほほ…

8.1.3 NAT

各組織は自由にプライベートアドレスを使えます。ということは、IP アドレスが 192.168.0.1 である PC が世界中に複数あるという事態になります。

もはや IP アドレスが住所の意味をなしません。

もちろん住所が特定できなくなるため、通信も出来なくなります。

そこで、プライベートアドレスを使う場合には、各組織の出口でプライベートアドレスを住所の意味を持つ IP アドレス (グローバルアドレス) へ変換しなくてはなりません。

この IP アドレスの変換のことを NAT と呼びます。

NAT を行なうのは組織の出口にあるファイアウォールやルータです (図 7.1 も見返してみて下さい)。

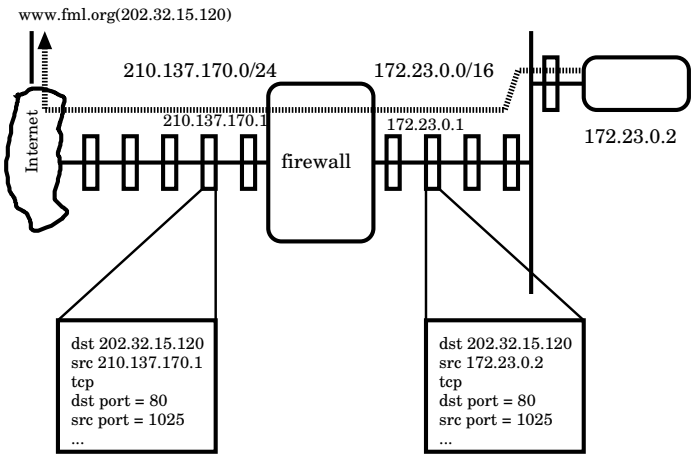

図 8.1 オリジナル NAT の例。実際のネットワーク図を簡略化してある。学内で IP アドレスが 172.23.0.2 である PC がインターネット上の http://www.fml.org/ (IP アドレスが 202.32.15.120) を見る際、学内からの出口にあるファイアウォールが NAT (172.23.0.2 を 210.137.170.1 へ変換) を行なっている。

8.1.4 ケーススタディ: NAT の動作 (1)

図 8.1 は最も単純な NAT の例です。

学内の PC から http://www.fml.org/ を見ることを考えます。

www.fml.org という WWW サーバの IP アドレスは 202.32.15.120 です。学内にあるクライアントの PC は 172.23.0.2 とします。このクライアントから 202.32.15.120 へ HTTP でアクセ

スしたいわけです。

図 8.1 は次のように読みます (図と、よく見くらべながら以下の解説を読んで下さい)。また、2 パケット分だけ説明しますが、このあとも同じように変換処理を繰り返すわけです (TCP については第 5 章を復習)。

クライアント → サーバ行きパケット

行きは 172.23.0.2:1025/tcp (IP アドレス = 172.23.0.2、ポート番号 = 1025) から 202.32.15.120:80/tcp です。

1. 送信元は src = 172.23.0.2 dst = 202.32.15.120 のパケットを送信。
2. ファイアウォールでパケットの IP ヘッダを書き換える。
   ヘッダの IP アドレス部分を書き換える。また、その変換ルールを覚えておく。
3. ファイアウォールは src = 210.137.170.1 dst = 202.32.15.120 のパケットを送信。

クライアント → サーバ行きパケット (返り)

サーバからの返りパケットは 202.32.15.120:80/tcp から 172.23.0.2:1025/tcp へ。

1. サーバは src = 202.32.15.120 dst = 210.137.170.1 のパケットを送信。
2. ファイアウォールでパケットの IP ヘッダを書き換える。
   覚えておいた変換ルールにしたがって逆の書換えを行なう。
3. ファイアウォールは src = 202.32.15.120 dst = 172.23.0.2 のパケットを送信。

もちろん、この HTTP 通信が続いている間、ファイアウォールは NAT の書き換え表を維持しつづける必要があります。

ここではファイアウォールが 172.23.0.2 は 202.32.15.120 へ書き換えるという対応を通信が続く間ずっと覚えています。

この例、図 8.1 では、ポート番号を書き換えていないことに注意して下さい。この例では「IP アドレスの書き換え」だけをしています。

NAT が使えるのは先着何台？

「IP アドレスの書き換え」だけをする場合、同時に NAT を利用できる PC の数は何台でしょう？

答えは「その組織に割り当てられている (グローバル) IP アドレス[*2] の数が上限」です。書き換える先の IP アドレスが確保できなければ NAT できませんよね？

正確には、ファイアウォールのインターネット側ネットワークインターフェイスのセグメントで利用可能な IP アドレス数です。

実際には NAT 以外にもサーバ等で使うために IP アドレスが必要なので NAT で利用可能な IP アドレス数は、割り当てられた IP 数よりもずっと少なくなります。

そのため、たとえ NAT で変換後に使う IP アドレスを 254 個用意したとしても先着 254 台の PC しか利用できないことになります。しかしながら、学内で全員が同時に PC を使ったら 1000 人を優に越えてしまうのです。

どうすれば良いでしょうか？

8.1.5 NAT の種類

NAT には、いくつか種類がありますが、大きく分けると次の 2 つになります。

- IP アドレス同士の変換
  前述の例。オリジナルの NAT 規格です。
- IP アドレスとポート番号も両方とも変換。
  IP マスカレード (IP masqurade、linux 用語)。NAPT とも呼びます。NAPT(Network Address Port Translation) はマスカレードが流行した後に考え

[*2] IP アドレスは、接続先のプロバイダを通じて割り当て申請を出し、取得します。
多くの人にとって無関係な話題なので、申請については割愛します。もし、申請を出す羽目になった場合は、プロバイダの人とよく相談して下さい。

られた用語です。

前述のように NAT では先着 254 名で終りです。しかしながら、ポート番号 (65536 個) も変換すれば 254 × 64510 台分の NAT 変換が可能になります (図 8.2)。

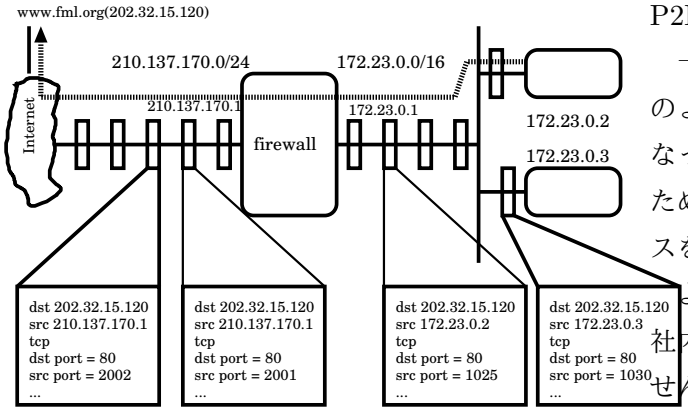

図 8.2 NAPT: IP アドレスとポート番号の変換を同時に行なうので、同時に、より多くの通信が可能となる。同時にファイアウォール越しに通信できる PC は最大「254 × NAT で使うポート番号の幅」254 ×ポート番号の幅 (1025 ～ 65535) = 254 × 64510 = 16385540 台分。

8.1.6 プライベートアドレスの問題

プライベートアドレスを使うと「よいことづくし」のようにも聞こえますが、そんなことはありません。たとえば、プライベートアドレスを使うと、通信が一方通行 (クライアント → サーバ) になります。双方向通信は特別なカラクリを考えないと困難です。

本来 TCP/IP では END TO END 通信 (今風に言えば PEER TO PEER = P2P 通信) をすることが前提です。これは「誰と誰が通信しているのか互いにはっきりわかっている」ことを意味します。

元々プライベートアドレスなどなかったわけですから、PC と PC 同士が互いに住所と名前をはっきりさせた状態で通信を行なうのが当然だったわけです。このために IP アドレスが「住所」に相当する意味を持っていると言えます。

よって、本来は、送受信、双方向通信 (どちら側から発信してもよい通信)、どんな通信形態も可能なわけです。一昔前の雑誌等では「時代の最先端は P2P」などという変な売り文句があったりしますが、それは歴史をよく勉強してないだけ。P2P は単なる先祖返りでしかありません。

一方、現代のネットワーク構成の基本は図 7.1 のように、ファイアウォール[*3] をはさんだ形になっています。そして、IP アドレスが足らないために、社内側や学内側ではプライベートアドレスを使うことが普通です。

よって、この構成では、インターネット側から社内側/学内側の PC へ向かっての通信は出来ません。プライベートアドレスでは PC の住所、つまり (住所として世界で一意な) IP アドレス、が (インターネット側から) 特定できないからです。

つまり、プライベートアドレスを使うと、通信の基本は一方通行 (クライアント → サーバ) になります。双方向通信は特別な仕組みやカラクリを考えないと困難です。

また、プライベートアドレスを使うと同じ IP アドレスを持つホストが無数にあるわけです。それでは本来の IP アドレスの「住所」という意味が消えてしまいます。それで良いのでしょうか？

さらに、NAT ではパケットの書き換えが前提です。それにも問題があります。たとえば IP アドレスやポート番号など通信の基本パラメータを含めて通信の妥当性を検証しようとしても、NAT されていると、うまく動作しません。

プライベートアドレスをやめられる？

ではプライベートアドレスの使用をやめればよいのでしょうか？

そもそもやめられますか？というと、理論上は YES です。

[*3] 「ファイアウォールの設置」と「プライベートアドレスの利用」は無関係です。それぞれ異なる理由により使っていることに注意 (本文を良く読むこと)。

元々 IPv4 アドレスが足らないためにプライベートアドレスを使っているだけから、IPv4 から IPv6 へ全面移行すれば、プライベートアドレスを使う必要はなくなります。IPv6 へ移行すれば END TO END 通信の復活です。

どのみち 2011 年には IPv4 アドレスがなくなり、2011 年以降に新規接続する組織は、いやおうなく IPv6 を使わざるを得ません。

ファイアウォールを廃止する

こちらのほうが大問題です。

理想論では、みんなで IPv6 へ移行してファイアウォールも廃止ということになっていますが、まず無理でしょう。

これだけ OS が問題だらけ、ウィルスは蔓延しているという状況で、「IPv6 へ移行し、PC を直接インターネットにさらす」のは無理というものです。

では IPv6 対応ファイアウォールがあれば問題解決だろうという気がするわけですが、実は、このあたりがさっぱり開発が追い付いていないところだったりするわけです。

コラム: IPv6 時代を目の前にして？
では、IPv6 時代を向かえたら、どうすればいいのか？
というと、まだまだ、よく分かりません。もう目の前なんですけどね…

8.2 ルータのシステムデザイン

ルータのシステムデザインは、なかなか教育的なので概観してみましょう。

なお、先祖返りをしてしまったファイアウォールのデザイン (B.5 節) と比較してみるのも興味深いです。

8.2.1 ルータメーカ

スイッチほどではありませんが、ルータを作っているメーカー (ベンダー) はアメリカを中心にいろいろと会社があります。家電メーカー各社をはじめ、謎のメーカーまでいろいろあります。

CISCO

ルータというカテゴリで最も有名な会社はシスコ[*4] システムズ (CISCO systems) です。1980 年代から今日まで代表的なルータメーカーの地位にあります。

金にものを言わせて、さまざまな企業を買収して大きくなってきたので、小さなスイッチから大きなルータまでラインナップが充実しています。そのため、さすがに家庭用としては、あまり見ませんが、業務用のシスコ製品なら至る所で見かけます。最も知られたネットワーク機器メーカーです。

しかしながら、本業はルータで、そのルータに搭載するファームウエアがIOS(Internet Operating System) です。

インターネットが商用化しはじめた時代に、この IOS の賢さ (?) のおかげで他メーカーから一歩抜きん出ることができたことが、今日の地位を築いたといえるでしょう。

Juniper

CISCO のスピンオフ (spin-off) メーカーとして Juniper があります。

ハイエンドルータしかないので、Juniper のルータを見かけることは難しいのですが、ハイエンドルータは CISCO と Juniper で一騎撃ちの感があります。

Juniper は CISCO を辞めた技術者たちが作った会社です。IOS の問題点は良く分かっているはず。

たとえば、CISCO では機種が違うとボードが使えなくなります。「いやがらせ」としか思えな

[*4] サンフランシスコのシスコです。

い仕様です。

一方 Juniper は機種やグレードを問わず同じ形のモジュールが使えるようにしてあります。CISCO の反省がいかされてますね。

さて Juniper のファームウエアですが IOS とはまったく違う構成です。核心部分は「難しいことを考える」担当の FreeBSD と「高速なパケットフォワーディング」をするための ASIC のハイブリッド構成になっているところにあります。

こういった分散アーキテクチャアは非常に現代的で理想的です。

国産

どうも電々公社関係のメーカー群の作るルータは、インターネット屋の琴線に響きません。終り。

NetBSD ベースのルータというより内輪なので、一応 SEIL も見てね。

http://www.seil.jp/

筆者は、ヤマハのルータが好きです。ヤマハというのは、発動機とかピアノで知られる、あの静岡県にあるヤマハです。ヤマハは実にいろいろなものを作っています。

日本の商用インターネットの初期を支えていたのはヤマハ RT100i でした。21 世紀になって売り出した小さな会社や事業所向けの業務用ルータ RTX1000 は 10 万台以上売れました。ルータ業界としては大ヒット作です。

一般家庭用の RT50i などもありますが、RT100i とか RTX1000 とはファームウエアが違うので我々的には興味なし。出来も悪かった。この子孫を家庭で使っている人がいるかもしれません。

バッファローなどの家庭用ルータの話題は割愛します。この手の数千円のルータは語るに値しません。よく落ちるし。

8.2.2 デザインの相違: monolithic 対 distributed

ネットワークシステムのデザインにはモノリシック (monolithic) と分散 (distributed) ないしはモデュラー (modular) システムがあると言えます。

monolithic は一つの大きなプログラムで 1980 年代的な書き方です。一方、1990 年代的な書き方は分散システムです。もちろん 1990 年代以降も理想は分散システムですが、数は多くないと思います。

Juniper はハードウエアとソフトウエアの分散システムですが、サーバソフトウエアについても同様のことが言えます。

このデザイン論は総じてプログラムの書き方の話です。

モノリシックアーキテクチャア

- 一つの巨大なプログラムにすべての機能をつめこむ。
- プログラムを書くのは楽。
- デバッグも楽。
- スパゲッティ化しやすい。
- 安全でないプログラムが出来やすい。
  小さな穴一箇所でも全滅の可能性がある。

分散アーキテクチャア

- 小さな複数のプログラムが協調して動作する。
- プログラムを書くのは大変。
- デバッグも大変。
- きちんと書ければ、より安全な安定したプログラムになる。

ケーススタディ: メールサーバの設計

ルータの場合、ソースコードが見られないので、アーキテクチャアの変遷の見本としてはメールサーバの方が現実的です。sendmail → qmail → Postfix とアーキテクチャアを比較することは非常に勉強になります。

詳しくは拙著「fml バイブル」の第 37 章を読んでみてください (大学図書館のオライリーコーナーを見よ)。

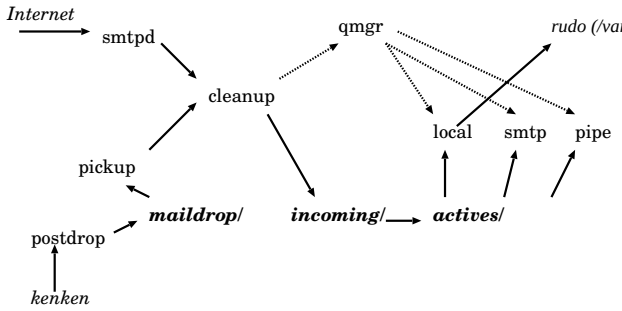

図 8.3　Postfix の内部構造: 図 3.3 再録

8.3　ICMP とデバッグ

8.3.1　おさらい

ルータやスイッチは IP パケットの転送 (IP フォワーディング) を行なう装置です。

階層モデル (2.4 節) で説明した通り、たとえば HTTP は上から順に、HTTP、TCP、IP、イーサネットと、いろいろなサブシステムの組合せで動いています。

しかしながら「ん？どの層もデータの転送をしているのでは？どこが違うの？」と混乱しそうです。いったん、まとめましょう。

- HTTP
  WWW ブラウザと WWW サーバの間での指示を出すだけ。
  データ転送やエラー訂正など高度な転送処理のすべてを TCP に頼る。
- TCP
  転送自体は IP 層に頼る。
  TCP は、ポート番号でアプリケーションを区別し、エラー訂正や転送速度調整などの高度な処理を担当。
- IP
  IP アドレスによる転送を実際に行なう層。
  住所である IP アドレスは世界で唯一のものなので、通信相手が地球の裏側でも転送は可能。
  高度な処理は無い。
- イーサネット (第 10 章)
  PC から隣の PC まで転送する役割。高度な処理は無い。

8.3.2　IP 層のエラー

IP 層は高度な処理はしないと説明しました。

ということは、IP パケットのエラーはどうなるのでしょう？たとえば IP パケットが行方不明になったら？

じつのところ IP システム単体では行方不明などのエラーは関知しません。その意味で IP 単体では信頼性のないプロトコルということになります。ただし、その分、処理が軽くなります。

意外にデータは壊れたりしないものです。

エラーが少ないなら、軽い方がよいわけです。スイッチやルータの処理も最低限でよくなれば、ファームウエアの実装も楽になるし、機器の負荷も下がり、転送能力のアップにつながります。

難しいエラー訂正などは上の層 (例えば TCP) が考えれば良いことです。

コラム: 工学的センス
いい意味でいい加減な設計方針が、半世紀近く使われ続け、世界規模で動作し続けられた大きな理由でしょう。
ガチガチなシステムを設計すると、全く動作しなかったり、とてつもなく遅かったり、ベンダーがネットワーク機器を作れなかったり、たとえ作れたとしても高価過ぎたりして使ってもらえません。

8.3.3　ICMP

必要があれば、再送などは TCP 層が命令を出してくれます。IP 層は、その指示にしたがって

いれば良いだけです。とにかくせっせと転送するのが IP 層の役目です。

この点で、IP は UDP などと同様に考えるべきで、IP もステートレスプロトコルです。

とはいえ、まったくエラー制御の仕組みがないのも困ります。

そこで、IP とは別にICMP(Internet Control Message Protocol) という仕組みが用意されています。

ICMP と IP はセットと考えて下さい。この 2 つはコンビで動作させることになっています。機器が IP を実装する場合、必ず ICMP も実装しなければなりません。

ICMP で制御情報のやりとりを行ない、IP と ICMP 双方が協調動作します。

ただ、ICMP は最も基本的な制御”だけ”と考えるべきです。IP の信頼性を向上させるほどのものではありません。

また ICMP の到達性は無保証です。

これは当然でしょう。考えてみて下さい。到達性保証までする ICMP でエラー訂正を行なおうとすると無限ループになり得ますね？ (OK ?)

転送の信頼性が欲しいなら TCP など上の層で解決するべきです。

8.3.4 ICMP の使い方

ある IP パケットのエラーを送信元へ知らせる場合、受信先の機器や、通信経路の途中にあるルータから、IP パケットの送信元へ、エラーを知らせる ICMP パケットが送られてきます。

エラー以外にも細かな通信制御情報の指示[*5]が送られてくることもあります。

具体的な内容は ICMP ヘッダの TYPE と CODE フィールドで指定します (図 8.4)。現在 TYPE は 18 種類定義されており、CODE は TYPE ごとに異なります。詳しくは /usr/include/netinet/ip_icmp.h を参照して下さい。

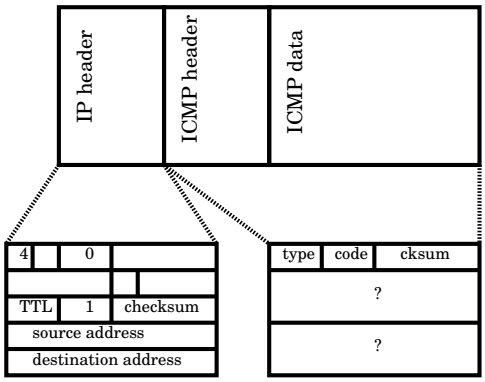

図 8.4 ICMP パケット: IP パケットの中身 (データ部分) が ICMP ヘッダ (+ ICMP データ)。IP ヘッダのプロトコル番号が 1 (ICMP)。ICMP のフォーマットは TYPE ごとにマチマチ。この図は、あくまでも一般論。

たとえば、次のようなネットワーク機器間での状態確認に用いられます。

- 機器の到達性を調べる。
- 通信経路上のルータからエラーを伝える。
- 経路を変更するべきだと知らせる。

それでは応用例を見てみましょう。

なんといっても ICMP を使う代表的なコマンドは ping と traceroute です。

例: ping コマンド

こちらから「到達したら返事を返せ」という指示を入れた ICMP パケットを送り、返事が返るかどうかで到達性を見るということができます。

この原理を利用したコマンドが ping です。ping は文字通り「潜水艦のソナーによる探索」に由来します。

例: ping コマンドで 172.23.1.1 という機器への到達性を調べる

```
% ping -n 172.23.1.1
PING 172.23.1.1 (172.23.1.1): 56 data bytes
64 bytes from 172.23.1.1: icmp_seq=0 ttl=255 time=1.195 ms
64 bytes from 172.23.1.1: icmp_seq=1 ttl=255 time=0.692 ms
64 bytes from 172.23.1.1: icmp_seq=2 ttl=255 time=0.671 ms
^C
----172.23.1.1 PING Statistics----
3 packets transmitted, 3 packets received, 0.0% packet loss
round-trip min/avg/max/stddev = 0.671/0.853/1.195/0.297 ms
```

[*5] ただ、後述するように、現代では無視することが普通です。

これは ICMP パケットを送り、その返事の ICMP パケットが 172.23.1.1 から送り返されている様子です。seq はパケットを識別するために付けられた数字 (シークエンス番号) で、time はパケットが往復するのにかかった時間 (RTT = round trip time) です。

ping は Unix 以外の OS でも利用可能です。たとえば Windows でも ping コマンドがあります。もっとも名称は一緒ですが、オプションが異なります (迷惑)。

- -n
  DNS を使わない
- -f
  flood (洪水) ping 。負荷テストをする際に使う。攻撃とみなされるので取り扱いには注意。
- -w 秒数
  タイムアウトの指定。エラーの場合、タイムアウトを短くして実行しないと、いつまでたっても終らない。
- -s サイズ
  サイズを可変させるとエラーの状態が変わることがある。レイヤー 2 以下の動作異常を切り分けるために使う。

コラム: どんなときでも -n

何の障害か分からないから、とりあえず ping を撃つ。これがエンジニアの基本です。

たんに DNS がらみのエラーかもしれません。実際よくあります。

この可能性を避けるために、つまり ping の結果が DNS のエラーに惑わされてしまわないように、-n をつけ「DNS を使わずに ping を動作させる」ようにしないと駄目です。

どんな時でも、どんなコマンドでも -n (DNS を使わないオプションが -n でないコマンドもあるので注意) と覚えて下さい。

「頭で考えなくてもワンアクションで安全装置がはずせる」くらいの勢いで、キーボードにコマンド列が打ち込めるようでないとプロとは言えません。

例: traceroute コマンド

```
prompt> traceroute www.fml.org
traceroute to www.fml.org (202.32.15.120), 30 hops max, 40 byte packets
 1  dslgw.fml.org (10.0.0.1)  2.337 ms  1.929 ms  1.909 ms
 2  hokkaido-a01.flets.2iij.net (210.130.182.4)  216.745 ms  188.281 ms  244.508 ms
 3  210.130.182.1 (210.130.182.1)  238.238 ms  237.006 ms  239.065 ms
 4  spr002lip10.IIJ.Net (202.232.14.237)  373.639 ms  405.323 ms  321.707 ms
 5  spr002bb00.IIJ.Net (210.130.232.161)  236.170 ms  192.054 ms  239.114 ms
 6  spr002gate00a.IIJ.Net (210.130.131.130)  239.415 ms  252.032 ms  239.188 ms
 7  210.138.14.141 (210.138.14.141)  239.281 ms  184.035 ms  194.359 ms
 8  www.fml.org (202.32.15.120)  238.383 ms  234.285 ms  246.141 ms
```

(注意: この例は、教科書用に分かりやすくするため、わざと -n なしで実行している)。

わざと寿命切れを起こすことで経路を推定するというトリックを効かせた秀逸なコマンドです。これは ICMP の Time Exceeded Message という命令の応用になっています。

図 8.5 は次のように読みます。

1. TTL=1 で UDP パケットを送り出す。
   最初のルータが TTL=0 でエラーを返す。

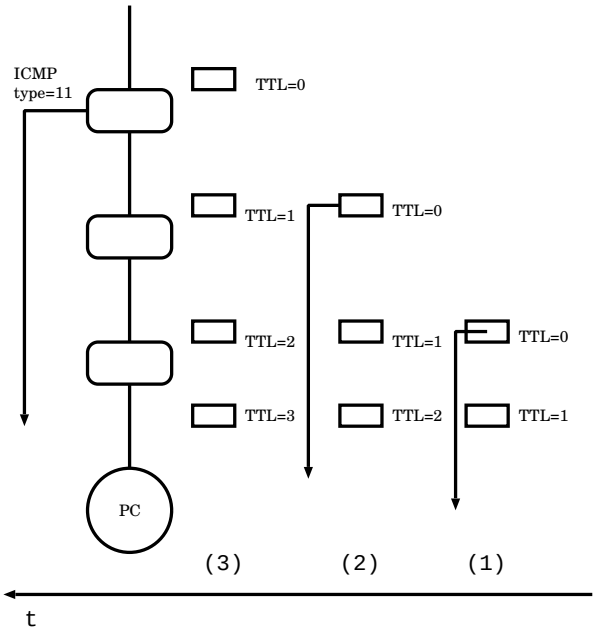

図 8.5　traceroute の原理

2. TTL=2 で UDP パケットを送り出す。
   2 つめのルータが TTL=0 でエラーを返す。
3. TTL=3 で UDP パケットを送り出す。3 つめのルータが TTL=0 でエラーを返す。

表示されるパケットの経路は、それぞれのエラーメッセージの IP アドレス (エラーの送信元) の集合となります。なお、何発もパケットを送り出して、それぞれのエラーを待つため、最終的な経路の表示をするまでに何秒もかかります。

Windows では tracert という名のコマンドで、しかもオプションが異なります。なお、20 年来ある Unix 版 traceroute と windows 版 traceroute では実装が異なる点に注意。

コラム: traceroute の結果は半分だけ信じる

このテクニックを使えば、通過しているルータの列、つまり経路が分かるわけです。ただし、現実の世界はダイナミックルーティングであり、秒単位で経路が変わっていることもあり得ます。そのため traceroute の表示は、”行き” のパケットがエラーになったルータの系列を示すスナップショットにすぎないことに注意です。

ダイナミックルーティングについての詳細は次回。

8.3.5　ICMP とセキュリティ

問題例の一つは「ファイアウォール」です。

いまどきのファイアウォールは ICMP を無視します。これはステルス化したいということもあるのですが、ICMP の悪用を回避したいという意図でもあります。

たとえば ICMP には「もっと、よい経路がありますよ」と教えてくれる命令がありますが、そんな情報を信じていいのか？というところが問題です。

うっかり信じると、嘘の情報で、パケットを覗き見する装置へ誘導されてしまうかもしれません。

こういった事情があるため、こういったメッセージをファイアウォールは無視します。

存在自体を相手に教えたくないので ping にも答えないことも普通です (つまりステルス)。

こういった機器があると ICMP の意味がありません ;-)

いやな世の中です。

8.4 まとめ

NAT

1. プライベートアドレスはインターネット上で一意に特定できないため、住所としては使えない。通信を特定するための住所として使えるのはグローバルアドレスだけ。
2. グローバルアドレスとプライベートアドレスを変換する仕組みが NAT (総称)。
   (a) NAT (オリジナルの NAT: IP アドレスと IP アドレスの変換)
   (b) NAPT (IP アドレスとポート番号の組で変換) オリジナル NAT にくらべ、数万倍変換ルールが増える。
   (c) NAPT と IP マスカレード (こちらが先) は同じ意味。
3. NAT や NAPT は、パケットのヘッダ情報を書き換える (もちろん書き換えた情報は覚えておかないといけない)。
4. 学内/社内でプライベートアドレスを使い、出口のルータやファイアウォールで NAT をもちいて、グローバルアドレスへ変換する。パケットの返事が返ってきたら、逆の変換も行う。

ICMP

1. IP は転送自体を担当し、エラー訂正は行わない。UDP と同様にステートレス。
2. エラー訂正は ICMP という別のプロトコルが行う。レイヤー 3 は必ず IP と ICMP をセットで実装する必要がある。
3. ICMP 自体のエラー訂正は行わない (エラーのエラーで無限ループになってしまう)。
4. ICMP はエラー訂正以外にもデバッグなどで用いる。有名な応用として
   (a) ping (生死確認を行うコマンド)
   (b) traceroute (相手までのルーティングを確認するコマンド)

がある。

第 9 章

ルーティング

9.1 IP パケットの転送

9.1.1 ブラックボックス

IP アドレスはインターネットにおける住所です (2.2 節)。まぁ、プライベートアドレス (7.6.16 節) とか NAT (8.1 節) という例外もありますが。

ところで、いままでは、ある住所から住所へ、つまり「ある IP アドレスからある IP アドレス」への通信を考える際に、通信の詳細はブラックボックスのまま無視してきました。

たとえば、郵便を考えると、ユーザにとっては「ポストに葉書を投函」すれば郵便による通信は成立です。あとは郵便局におまかせ。相手にどう配送されているかはユーザにとって関係ありません。この観点からいえば、郵便でも配送の詳細はブラックボックスです。

しかしながら、郵便の場合、実際には、その葉書を車や電車、飛行機で受信者へ輸送する裏方がいるはずです。

TCP/IP でも事情は同じで、裏方がいます。

たとえば WWW ブラウザのプログラムを書く際に、プログラマは「TCP で 192.168.0.1 のポート 80 に接続したい」としか書きません。そうすると、ソケット (socket) という出入口が作られ、要求を出すのも、答が返ってくるのも、そのソケットからです。プログラマがソケットの裏側を気にする必要は全くありません。

つまり TCP 層 (5 節) は IP 層に「受信者」の情報を与え、IP パケットの転送の詳細には関与しません。転送は IP 層におまかせです。

IP 層で障害がある場合、たとえばソケットからデータが出てこないようなら (TCP パケットの遅延やエラーが起きれば) TCP 層が再送要求を出そうなどの対策を取りますが、正常にソケットからデータが出てくるかぎり、TCP 層も転送の詳細については何も考えません。

TCP にとっては、データはストリームです。一方、TCP/IP は「パケット通信をしている」と説明してきたわけですから、ソケットの向う側では、ストリームではなく、無数の IP パケット群として扱われています。

実際のところ IP パケットは、どのように転送されているのでしょうか？

9.1.2 転送している実体は何か？

まずは郵便を例に考えてみましょう。

郵便

札幌から東京へ葉書を輸送することを考えます (実際には、このようにやってないのかもしれませんが、思考実験なので、そう思って下さい)。

札幌のどのポストへ葉書を投函しても、いったんはＪＲ札幌駅のそばの札幌中央郵便局へ集められます。

札幌中央郵便局では、宛先が東京方面の葉書を選別し、千歳空港へトラックで運び、東京行の飛行機へ載せます。

東京についたら、荷物は東京中央郵便局へト

ラックへ運ばれ、(以下、逆順で配送の仕方が細かくなり)、最終的に地元の郵便局の局員が各家に葉書を届けるというわけです。

では、悪天候で飛行機が飛ばなかったらどうすればよいでしょうか？

飛行機が使えないなら、ＪＲで輸送するように配送経路を切替えればよいわけです。

これが経路制御ないしは経路選択、英語でいうとルーティング (routing) です。

また、このように動的に判断を切替える動作を、動的経路制御つまりダイナミックルーティング (dynamic routing) と呼びます。逆に「いつでも東京行は千歳発の飛行機を使う」といった判断は不変なので、静的 (static) と呼びます。つまり静的経路制御、スタティックルーティング (static routing) です。

身の回りにあるたいていの PC やネットワーク機器はスタティックルーティングです。スタティックルーティングは、ネットワーク機器を導入する際に、つまり最初に一度、設定するだけです。変更の必要があれば、そのつど設定変更をします。とくに変更の必要が発生しなければ、その機器が役目を終える時までそのままです。

一方、インターネットの基幹システムはダイナミックルーティングです。

さまざまなダイナミックルーティング (dynamic routing) の手法があります。代表的なプロトコルは次のものです。

- RIP (Routing Information Protocol)
- OSPF (Open Shortest Path Fast)
- BGP (Border Gateway Protocol)
- IS-IS
  OSI の規格です。日本ではあまり使われていませんが外国では使われています。

TCP/IP

traceroute コマンド (8.3.4 節) の例で分かるとおり、インターネットには、たくさんのネットワーク機器が存在します。

```
prompt> traceroute www.fml.org
traceroute to www.fml.org (202.32.15.120), 30 hops max, 40 byte packets
 1  dslgw.fml.org (10.0.0.1)  2.337 ms  1.929 ms  1.909 ms
 2  hokkaido-a01.flets.2iij.net (210.130.182.4)  216.745 ms  188.281 ms  244.508 ms
 3  210.130.182.1 (210.130.182.1)  238.238 ms  237.006 ms  239.065 ms
 4  spr002lip10.IIJ.Net (202.232.14.237)  373.639 ms  405.323 ms  321.707 ms
 5  spr002bb00.IIJ.Net (210.130.232.161)  236.170 ms  192.054 ms  239.114 ms
 6  spr002gate00a.IIJ.Net (210.130.131.130)  239.415 ms  252.032 ms  239.188 ms
 7  210.138.14.141 (210.138.14.141)  239.281 ms  184.035 ms  194.359 ms
 8  www.fml.org (202.32.15.120)  238.383 ms  234.285 ms  246.141 ms
```

(注意: この例は、教科書用に分かりやすくするため、わざと -n なしで実行している)。

上の例では、自宅の出口のネットワーク機器 (ルータ、IP アドレスが 10.0.0.1) からインターネットのどこかにある WWW サーバ www.fml.org (IP アドレス 202.32.15.120) へ通信 (HTTP) をする際に IP パケットが通過しているネットワーク機器の一覧です。

この通信経路途上に出てくるネットワーク機器群すべてがルータです (注: 最後の 8 は WWW サーバなので異なる)。

郵便版の traceroute を疑似的に書けば、次のように次々と郵便局を経ていく様子といえるでしょう。

```
1. 札幌北三条郵便局
2. 札幌中央郵便局
3. 東京中央郵便局
4. 東京どっかの郵便局
5. 東京の受信者
```

9.2 ルーティング

9.2.1 経路選択とは？

前述のように、障害が起きた場合、通信経路を迂回させるといった決断が必要になります。

動的に決断するか、常に同じ (静的) 決断かという違いはありますが、いずれにせよ経路選択をする機械がルータです。

ルータは、選択をする際に受信者の IP アドレスだけを元にして、つまり宛先だけを見て経路選択をしています。

IP パケット (図 5.3) には、もっといろいろな情報がありますし、IP パケットの中に入っている TCP パケット (図 5.2) のヘッダを見れば、さ

らに多くの情報があります。

情報が多ければ多いほど賢い判断が出来そうですが、実際には宛先だけを見て経路選択をします。

コラム: ルータの負荷
ルータの負荷は高いです。
たとえば 1 Gbps の処理能力のあるルータなら 1 秒間に数百万 PPS (packets per second) の経路選択をしなくてはなりません。つまり、1 秒間に数百万の IP パケットの情報を検査し、それぞれのパケットについてルーティングの判断を行なわないといけないということです。
検査する情報を極力減らさないと、処理が追い付きません。
なお 1 Gbps など、今時ありふれた処理能力のルータです。B フレッツで運が良ければ 30 Mbps は出てしまいますから、理論上 40 世帯もあれば 1 Gbps の転送要求がありえるわけです。

コラム: ルータとスイッチ
通信経路上のスイッチやルータは、どちらも受けとったパケットを隣のネットワーク機器へ渡す機械です。
違いは、“置き場所” です。
セグメントを越える際に経路選択を行なうのがルータ、同じセグメント内で PC から PC へ転送するのがスイッチです。
つまりルータには「セグメントを越える」ことと「ルーティングをする」という 2 つの役割があるわけです。

9.2.2 経路選択の基準

経路選択は IP パケットの宛先だけを見て決めます。具体的には IP ヘッダの destination address フィールドだけを見て、経路一覧表 (routing table) と照らし合わせ選択します。

コラム: 他の情報は使わないの？
標準では使いませんが、パケットヘッダのポート番号や送信者情報 (source address) によって経路を変更するという動作が可能なルータも存在はします。
以前、紹介した L4 スイッチや L7 スイッチが、その代表例です。

コラム: プロバイダの仕事

郵便のたとえでも分かる通り、大きな郵便局ほど大味な処理をするだけで十分です。たとえば、札幌中央郵便局は、札幌いき、東京いき、外国いきといった単位で振分をするでしょう。

インターネットでも事情は同じです。基幹ルータ群は大味な単位で振分をし、末端のルータが細かい住所の振分を担当します。

たとえば、一つの住所 (IP アドレス 1 個) つまり/32 の経路を世界中の全ルータが知っている必要はありません。そもそも /32 の細かさで経路の一覧表を作ると何億経路にもなってしまいます。非実用的です。

細かい単位での経路制御はプロバイダの行なう仕事とし、インターネットの基幹部分では /8 とか /16 といった大きな単位の経路だけを流すようにしないと基幹ルータ群の負荷が膨大になってしまいます。

よって、基幹ルータには「日本では 10.0.0.0/8 の住所群を使う」といったレベルの大味な情報だけを教えます。外国からは「日本宛の郵便なら、とりあえず東京の郵便局へ送ってしまえ」という判断をしてもらうのと同じ具合です。

一方、プロバイダ A に加入しているユーザには プロバイダ A の責任で割り当てます。ユーザ (1) は自宅なので 10.1.1.1/32、ユーザ (2) は小さな会社なので 10.2.0.0/28 といった具合です。これが郵便の何区何丁目何番地の単位に相当する IP アドレスになります。

9.2.3 演習: ルーティングの例

PC 教室では学内でしか traceroute コマンドが使えないので図 9.1 の上だけで行ないます。

図 9.1 は学内のクライアント PC (172.23.0.2) から www.fml.org へ向かう HTTP のパケットの様子です。図は、かなり省略してあります。

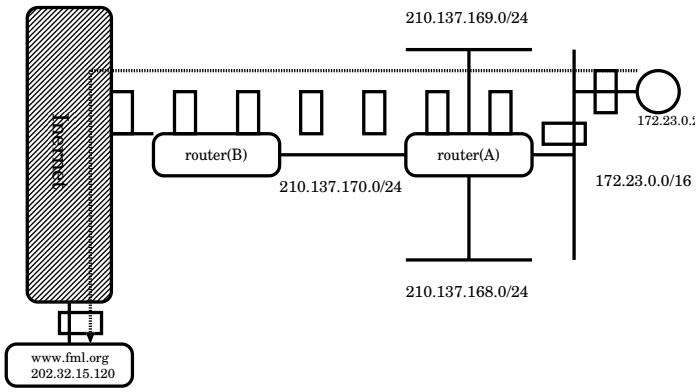

図 9.1 ルーティングの例

クライアント 172.23.0.2

クライアントはスタティックルーティングです。

172.23 ではじまる IP アドレス以外の宛先への通信は、何も考えずにルータ A めがけてパケットを送信します。このルータのことをデフォルトルート (default route) もしくはデフォルトゲートウエイ (default gateway) と呼びます。

デフォルトルートは事前に PC へ設定しておく必要があります (PC 教室の場合、納入時に SI 業者さんが設定しています)。

演習: CentOS で netstat -rn コマンドを実行してみなさい。

Windows なら route print コマンド (いや route show だっかもしれない?)。

ルータ A

送信元 クライアント PC (172.23.0.2) から送信されたパケットは、まずルータ A で判断されます。

この図の ルータ A には四つのセグメントがあります。172.23.0.0/16 210.137.168.0/24 210.137.169.0/24 210.137.170.0/24

経路選択が必要です。

宛先が、そのいづれかのセグメントであれば、そちらへ転送します。

そのどれでもない場合は？というと、やはりデフォルトルートへパケットを送ります。ルータ A のデフォルトルートはルータ B です。もちろん事前に設定しておく必要があります。

ルータ A が Unix (BSD Unix) の例 (出力例は NetBSD、fxp0 〜 fxp3 は Intel 製のネットワークカード):

```
% netstat -f inet -rn
Routing tables

Internet:
Destination        Gateway            Flags     Refs     Use    Mtu  Interface
default            210.137.170.2      UGS         14 17163217   1500  fxp3
127                127.0.0.1          UGRS         0  2688353  33228  lo0
127.0.0.1          127.0.0.1          UH           6 26168955  33228  lo0
172.23.0.0/16      link#1             UC           4        0   1500  fxp0
210.137.168/24     link#2             UC          10        0   1500  fxp1
210.137.169/24     link#3             UC          10        0   1500  fxp2
210.137.170/24     link#4             UC          10        0   1500  fxp3
```

ルータ A が CISCO や YAMAHA の例 (出力例は YAMAHA RTX1000):

```
router> show ip route
宛先ネットワーク     ゲートウェイ      インタフェース   種別   付加情報
default              -                      PP[01]    static
172.23.0.0/16          172.23.0.1             LAN1  implicit
A.B.C.184/29           A.B.C.185              LAN2  implicit
     ... 以下略 ...
```

ルータ B

ルータ B も同様です。

宛先が 202.32.15.120 の IP パケットが渡されてきた。→ 202.32.15.120 について自分は何か知っているか？→ いや、知らない。→ デフォルトルートめがけて転送。

この図には書いてありませんが、実際には上位プロバイダのルータへ転送しています。

9.2.4 用語: CIDR アドレッシング

そうはいっても、ルータの仕事量は大変なものなので、人間の側でルータの負荷を下げるように協力してあげてください。

IP アドレス体系も、ルータの処理を下げることを前提にしています。それがセグメントの / 表記の背景です。クラスをやめて / 体系へ移行した理由ともいえます。

現在、セグメントは「アドレス/ビットマスク形式」で表現しています。たとえばプライベートアドレスは

```
10.0.0.0/8
172.16.0.0/16
192.168.0.0/16
```

の三種類でしたね？

実際には、中途半端な大きさのセグメントもありえます。たとえば 10.0.0.0/22 とか。

IPv4 アドレスが足らないことは明白なので、IP アドレスを効率良く割り当てる必要があります。特に 1990 年代なかば以降は各組織に最小限の IP アドレスしか割り振られません。

そのため 10.0.0.0/22 とか、192.168.0.0/28 といった半端なサイズの細切れセグメントが山ほどあるわけです。

ルータのメモリ消費量

ルータの設定を考えましょう。いっけん脇道に見えますが、そうではありません。

10.0.0.0/22 はクラス C のネットワーク 4 つ分です。10.0.0.0/24 10.0.1.0/24 10.0.2.0/24 10.0.3.0/24

(演習: 確認してみてください)。

これは昔ながらの設定 (Unix) では

```
[Unix の例]

route add -net 10.0.0.0 10.0.0.1
route add -net 10.0.1.0 10.0.0.1
route add -net 10.0.2.0 10.0.0.1
route add -net 10.0.3.0 10.0.0.1
```

のように書きます (この例では 10.0.0.1 がデフォルトルートです)。

メモリ上には四つ分 (10.0.0.0/24 10.0.1.0/24 10.0.2.0/24 10.0.3.0/24) の情報を書き込む場所が必要です。各セグメントごとに、それぞれ選択先があるので、一セグメントあたり 8 バイト (32 ビット× 2 = 4 バイト× 2)、よって、この場合は最低 4 × 4 = 32 バイトは必要です (実際には、

それ以外に検索用情報分なども必要ですが、ここでは無視しています)。

上で見たように実際には、もっと細切れだったりもします。全情報をルータに教えようとしたら数千万経路はあるでしょう。ということは「8 バイト× 5000 万 = 400 M バイト」

それ以外の情報を持ったとしても 1 テラもメモリ積めば十分だろうから全然問題ないじゃないか？と思った人は、ちょっと待って下さい。

- まず、各パケットごとに数百Mバイトのデータの中を検索しなければなりません。
- その検索処理を一秒間に数百万〜数千万回行なう必要があります。
- それなら CPU の能力をあげたら問題ないのでは？というと、それは違います。
  たとえば CPU を Intel 製品にしたら、熱くなり、電力消費量も激しく上がります。
  専用のネットワーク設備を置く場所は、いろいろと高価なので、電力にかかる費用も馬鹿になりません。災害時にはバッテリーと自家発電で動作するような場所なので電力の節約は至上命題です。
  また機械を熱くすると、すぐに壊れるという問題もあり、専用機械では、古くて枯れた CPU か省電力に特化した CPU を使うことが普通です。総じて、あまり速くはありません。
- さらに、プロ用機材で使うメモリは最上級品です。ツクモで売っているバルクメモリなどとは異なります。
  mission critical な状況で使うものは最上級品でないと困るのです。壊れたら機械を止めて交換するといったわけにはいかないから。
  無数のテストに耐えた極上品は 128 M で 15 万円といったすごい値段がついています。

このような様々な理由により、ルータではメモリ消費量を抑えることに重要な意味があります。

さて、10.0.0.0/22 に戻りましょう。上の例

[Unix の例]

```
route add -net 10.0.0.0 10.0.0.1
route add -net 10.0.1.0 10.0.0.1
route add -net 10.0.2.0 10.0.0.1
route add -net 10.0.3.0 10.0.0.1
```

では 4 つ別々の設定をしていますが、これを一つに出来たらどうでしょう？

[Unix の例]

```
route add -net 10.0.0.0/22 10.0.0.1
```

もちろん /22 という情報も必要ですが、5 ビットあれば足ります。また一つのエントリになったので 1/4 になりました。5 ビット増えますが、激的に小さくなりましたね？

だから人間には扱いづらい半端な /22 や /28 といったセグメントを許すことに大きな意義があるわけです。

この表記法をCIDR (Classless Inter Domain Routing) アドレッシングと呼びます。名前の通り CIDR とは「クラス概念を無くした」体系です。つまり /8 /16 /24 以外のネットワークの大きさも OK にしたという意味になります。

なお、プロバイダアドレッシングという用語もありますが、CIDR と同義語と考えてかまいません。これはプロバイダの存在が前提のアドレス体系という意味です (後述)。

CIDR のメリット

- メモリ消費量が少なくなる → ルータの負荷が下がる。
- ユーザへの IP アドレス割当を最小限にすることができる。つまり IP アドレスの有効利用が可能。
  クラスを前提とすると /8 /16 /24 単位の割り当て区分しかなかったので、極端な話、個

人の自宅にも /24 を割り当てるしかありません。

無駄です。

CIDR では「個人には /32 を割り当てる」とか「小さい会社には /29 を割り当てる」といった割り当てを行ないます。実際、Bフレッツや ADSL などの自宅用アドレスは/32単位で割り当てています。

CIDR のデメリット

- 古い機械はクラスが前提なので CIDR 表現を理解できない。

  そのためエンジニアは、ネットマスク表現とCIDR 表記の両方とも使えないといけない。
- CIDR を使うためには特別な命令が必要になることもある。

  CISCO なら「ip classless」コマンド。ただ、最近のファームウエアではデフォルトで入っているので気にしなくても OK。

9.3 Internet Routing Architecure

いよいよインターネットの正体を説明しましょう。

9.3.1 AS

世界 (インターネット) は AS から構成されます。AS は Autonomous System (自律系) の頭文字です。

よって

```
インターネット = AS1 + AS2 + AS3 + ...
```

といえます。

AS はインターネット上の領土に相当すると考えるとよいでしょう。一つの AS は通常一つのプロバイダです。現在、世界で AS は 2 万強割り当てられていますから、そのくらいの数のプロバイダが存在します。

ただ AS と地理は無関係です。プロバイダの機材があるところは、通常すべて同じ AS に属します。たとえば、あるプロバイダの機材が、ニューヨーク、サンノゼ、東京といったぐあいにアメリカと日本にまたがっていても、もちろん一つの AS です。

世界地図の上に AS の絵を書くと、複雑に絡み合って、かつあちこちで重なっているような絵ができあがるはずです。

9.3.2 AS 番号

AS を識別するために AS 番号というものを割り当てています。AS 番号は 16 bit の数字なので最大 65536 個が割り当て可能です。

たとえば IIJ は AS 2497、WIDE は AS 2500 です。

AS 番号でも 65000 より上はプライベートということになっていて、プライベート AS と呼ばれています。各プロバイダが自由に使って良い番号です。

9.3.3 IX

インターネットは AS から構成されます。

```
インターネット = AS1 + AS2 + AS3 + ...
インターネット = プロバイダ 1 + プラバイダ 2 + プロバイダ 3 + ...
```

当然、プロバイダ 1 のどこかにある PC がプロバイダ 2 のどこかにある WWW サーバと通信したりしているわけです。ところが通信経路上のネットワーク機材は経路制御をしないといけないわけですから、経路情報を教えてもらう必要があります。

前述の例のように、学内の PC や大学の出口では転送先の選択肢がせいぜい数個しかないので、事前に設定 (スタティックルーティング) します。

一方、プロバイダのルータは世界全体の情報を知らないと転送先を決めらませんが、これは十数万経路存在します。また常に変化し続けています。とても手動で設定などできません。

そこで、プロバイダ間で、つまり AS 間で経路の一覧を自動的にやりとりする仕組みが作られて

います。

プロバイダ同士は IX (Internet eXchange) と呼ばれている場所に自分のルータを置き相互接続します。

そしてルータ同士は、自分の管轄にある経路について BGP4 プロトコルで伝えるように設定されます (図 9.2)。これは BGP で「ピアを張る」(peering) と呼ばれています。

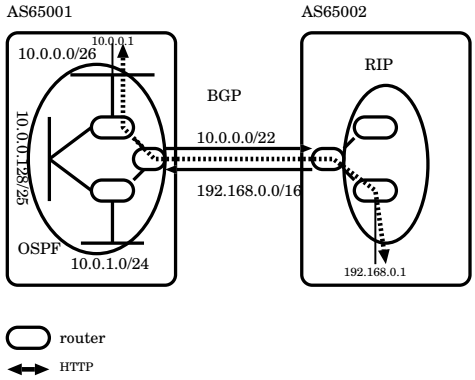

図 9.2 IX の一部分: プロバイダ同士の接続 (一箇所だけ) の例。ルータは互いに「自分の AS にある経路はこれだよ」と教えあう。BGP で渡す情報と AS 内部の経路制御とは無関係であることに注意。

9.3.4 BGP の例

図 9.2 を参照して下さい。なお、ここでは、プライベート AS を例にとっています。

また、各プロトコルについては後述します。

- AS 65001 → AS 65002
  BGP で 10.0.0.0/22 を持っていることを教える。
- AS 65002 → AS 65001
  BGP で 192.168.0.0/16 を持っていることを教える。
- AS 65001 の中は 10.0.0.0/22 を細分化して使っている
  ルーティング制御は OSPF
- AS 65002 の中は 192.168.0.0/16 を細分化して使っている
  ルーティング制御は RIP

もうすこし大域的な例を図 9.3 に挙げておきます。

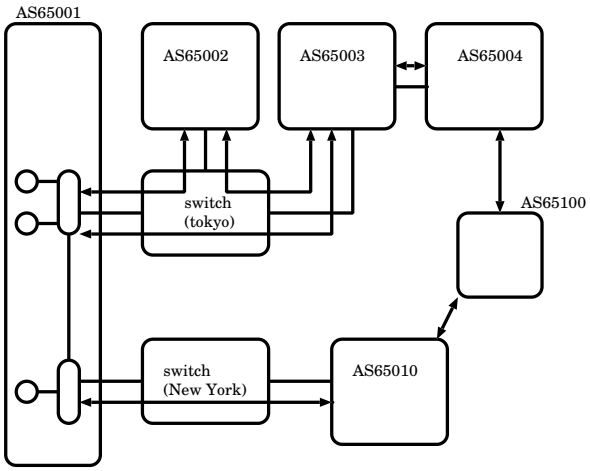

図 9.3 やや大域的な IX の様子: AS 65001 は世界規模のネットワークを構築している大手のプロバイダで東京とニューヨークの IX で相互接続し、東京 〜 ニューヨークのルータ間は専用線 (海外線) を引いている。AS 65004,65100 のように IX へ自力で接続しないプロバイダもある。

9.4 ダイナミックルーティングプロトコル各論

9.4.1 RIP

RIP は Routing Information Protocol の頭文字です。

4.2 BSD の routed というプログラムがご先祖様になります。RIP バージョン 1 (RFC 1058) は 4.3 BSD の routed デーモンを元にした規格です。

現在は RIP バージョン 2 (RFC 1723) が使われています。バージョン 1 より動作が速いですが、根本的な問題は解決されていません。

RIP の動作

RIP は自分の知っている経路情報をブロードキャストアドレスへ定期的に流しています (図 9.4)。デフォルトでは 30 秒ごとに送信しています。

ブロードキャストなので、同じセグメントにあるルータは互いに RIP 情報を受けとるわけです。

転送に使っているプロトコルは UDP で、ポート番号は 520 です。

RIP パケットにはメトリックという数字があります。これは優先順位と寿命を合わせ持った数字で、ここがポイントです。

各ルータは RIP で自分の管轄している経路情報を流します。さらに「他のルータから RIP で受けとった経路情報」も再送信しますが、その際には RIP のメトリックを +1 します。

同じ経路情報があるルータに複数の経路を伝わって届いたとしても、経路上にあるネットワーク機器の数が異なるので、たいていはメトリックが異なることになります。

「小さいメトリックの経路情報を優先する」というルールになっているので、ルータは小さいメトリックの RIP を流してきたルータへ転送するようになります。

なお、RIP を受けとる側のルータは、経路情報の更新がしばらくないと、その経路を消去します (経路情報を受けとれない場合は 30 秒ごとにメトリックを +1 していき 16 を越えたら消去しています)。

また、メトリックが大きくなり過ぎた場合 (通常 16 を越えた場合) は、無効な経路情報として、その経路情報を無効とします。逆にいうと、ルータ 16 個より遠い世界はインターネットに存在しないと仮定しているわけです。

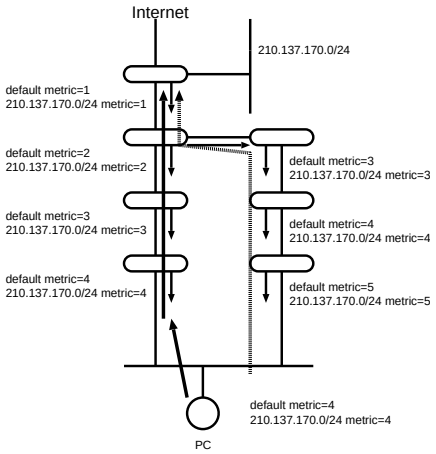

図 9.4 RIP の例:上のルータから 210.137.170.0/24 の経路情報が RIP で流れる。一番下のルータは 210.137.170.0/24 について二つの経路情報を受けとっている。正常時は左 metric=4 を使う。左系の障害時には metric=4 の情報が見えなくなる。一定時間後、左側経路のキャッシュが消えるため、metric=5 の方が選択され、右系へルーティングされる (赤い線)。

9.4.2 BGP4

BGP4 は Border Gateway Protocol version 4 の頭文字です。転送には TCP を使います。

ルータが経路選択の判断に使うというより AS 間の情報交換に使うプロトコルです。

ただ、複数のルータから同じ経路情報を受けとった場合に、ルータは「宛先までの間にある AS 数」が小さい方を選びます (図 9.3 参照)。それが基本動作です。

この「間にあるルータ数が少ない方を選択する」という観点で BGP は RIP に似ています。

9.4.3 OSPF

日本のプロバイダの多くはプロバイダ内ルーティングとして OSPF を使っている例が多いと思います。これは歴史的な成行きと考えられるので気をつけて下さい。外国では IS-IS なども多く使われています。

OSPF は Open Shortest Path First の頭文字です。

RIP のような能天気なプロトコルではありません。高速な判断をしますが、複雑なプロトコルです。

生死監視

まず、各ルータのネットワークインターフェイスは互いに生死監視をしています。

定期的に OSPF HELLO パケットを流し、40 秒間のあいだ相手の HELLO パケットが来ない場合、「相手は死んだ」とみなし、プロバイダ内の経路全体を再計算します。

このため、障害時の経路切替え判断が高速になります。オリジナルの RIPv1 に比べれば圧倒的な速さです。

コスト

OSPF にはコストという概念があります。

優先順位を示すという点で RIP のメトリックと似たような働きをしますが、RIP より細かい設定が可能です。

コストは、数字も細かく設定できますし、経路を二次元的に考えることが出来る点で RIP の大味な世界とくらべると進化しています。

コスト計算

目的地までの総コスト (コストの足し算) で最適な経路を決めています。全ルータの情報を集め、コストの合計を計算し、ルーティングを最適化するマスタールータがこの計算をしています (OSPF を使うルータ群の中で一つ選出されるようになっています)。

デメリット

- 実装がとても複雑

  bug bug ;-)
- 全体のルーティングを再計算するのは大変

  ルータどうしをつなぐから N^2 です。最悪 $O(N^2)$ で計算量が増えます。

OSPF の例

OSPF コストですが、CISCO の場合、「100,000,000 / インターフェイスの帯域 (bps)」が基礎単位です。

つまり 100 Mbps のイーサネットならコストが 1、10 Mbps のイーサネットならコスト 10、1.5 Mbps の専用線は 64 といった具合です。

これはデフォルトの例で、自動的に設定されてしまう値です。通常、これでは都合が悪いため、人間がネットワーク全体の設計を考えて、値を設定します。正常運用時だけでなく障害時のネットワークをどのようにすればよいか？などを全て考えた上でパズルを解くわけで、非常に大変です。

ただ、このあたりがルーティングマニアを誕生させる所以なのかもしれません:-)

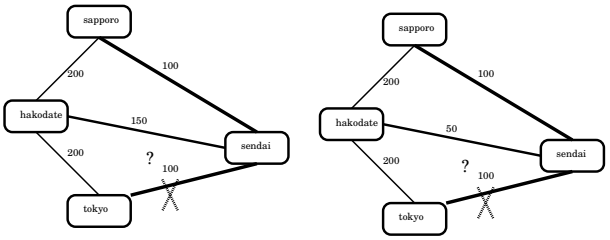

図 9.5 OSPF の例: さっぽろ → 東京間で、以下のようにコストが設定されているとする。各経路にそって足し合わせてみなさい。通常は 200 の経路が選択される。では、仙台〜東京が障害の場合はどうなるか？

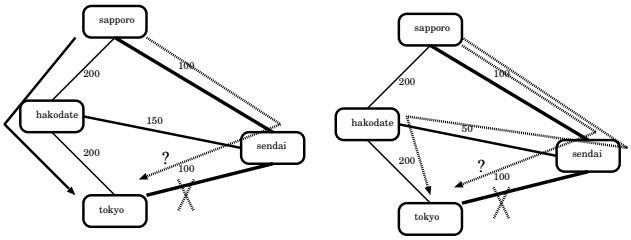

図 9.6 図 9.6 の解答

9.5 まとめ

ルーティング

1. 経路 (route) の選択・制御
2. 基本的に宛先 (destination) で制御する。その他たくさんあるヘッダ情報 (IP ヘッダや TCP ヘッダの情報) は使っていない。
3. 静的経路選択 (static routing) と動的経路選

択 (dynamic routing) がある。
（a）身近な中小規模ネットワークは static
（b）大規模なネットワークは dynamic

4. 経路を選択できないときに送る先が “デフォルトルート (default route)” で、事前に設定しておく必要がある。
5. 大手インターネットサービスプロバイダ (ISP) には最低一つの AS 番号
6. インターネット $= \sum AS =$ ISP1 + ISP2 + ISP3 + ...
7. ダイナミックルーティングプロトコル
   （a）動作による種類
      i. ディスタンスベクトル: 距離が短い (間にある機械の数が少ない) 方を選ぶ
      ii. リンクステート: 隣接機器との状態を監視、全機器が全体の状態を共有している
   （b）規模による分類
      i. 小規模
         A. RIP: ディスタンスベクトル型、機器からブロードキャストで経路情報を発信。
      ii. 中小規模
         A. IS-IS: リンクステート型、OSI の規格
         B. OSPF: リンクステート型、IETF の規格、計算量は機器の数の総当たり戦で増える
      iii. 大規模、インターネットバックボーン
         A. BGP: ディスタンスベクトル型、間にはさまる AS の数が空くない経路を選ぶ。
            実際には政治・経済・その他、大人の事情で複雑な設定をする必要が多々生じる。
            BGP4 の設定は、そんなに簡単ではない。

第 10 章

イーサネット

10.1　前回までのあらすじ

まずは階層モデルの復習から始めましょう。

- HTTP
  WWW ブラウザと WWW サーバの間での指示を出すだけ。
  データ転送やエラー訂正など高度な転送処理のすべてを TCP に頼る。
- TCP
  転送自体は IP 層に頼る。
  TCP は、ポート番号でアプリケーションを区別し、エラー訂正や転送速度調整などの高度な処理を担当。
- IP
  IP アドレスによる転送を実際に行なう層。
  住所である IP アドレスは世界で唯一のものなので、通信相手が地球の裏側でも転送は可能。
  高度な処理は無い。
- イーサネット (第 10 章)
  PC から隣の PC まで転送する役割。高度な処理は無い。

前回 (第 9 章) は IP ルーティングを取り上げました。

IP 層は IP パケットの転送を行ないます。

IP パケットの送信者、受信者は IP ヘッダの IP アドレスで指定可能です。IP アドレスはインターネットにおける住所に相当します。

ネットワーク機材には大きく分けてスイッチとルータがあり、セグメント内のパケット転送を行なう装置がスイッチ、セグメントをまたいだパケット転送を行なう装置がルータです。ルータはルーティングも行ないます。

10.1.1　IP アドレス

IP アドレスは PC に割り当てるものです。つまり PC をポンと置いても使えません。誰かが PC に IP アドレスを設定して、はじめてインターネットが利用可能になります。

たとえば PC 教室は、PC を納入する際に納入業者が IP アドレスを設定しています。

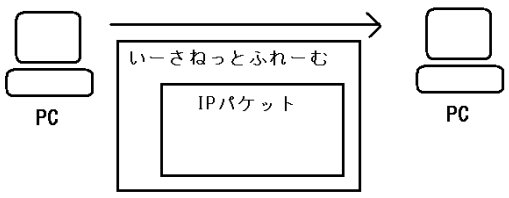

図 10.1　イーサネットフレーム: データとして IP パケットを運んでいる様子。

10.2 IP パケットを運ぶ実体は？

図 9.1 のように、IP パケットは、PC からルータ A へ、ルータ A からルータ B へと次々と隣のネットワーク機器へ転送を繰り返していきます。

われわれは、この動作を俗にバケツリレーと呼びます。火事を消す時に人が集まってバケツを右から左へ渡していきますね？あれのことです。

PC からルータ A へ IP パケットを転送する場合も、IP パケットをデータとして持つレイヤー 2 のパケットがあります。

レイヤー 2 を代表するプロトコルはイーサネット (Ethernet) です。なお、レイヤー 2 では普通パケットと呼ばずフレーム (frame) と呼びます。たとえばイーサネットフレームとなります (図 10.6)。

なおイーサネットフレームはデータとして何でも運ぶことが出来ます。つまり IP パケット以外のデータも転送可能ですが、本章では IP の例だけを考えます。

> コラム: その他のレイヤー 2 プロトコル
> イーサネット以外にも様々な規格があります。トークンリング、FDDI、HDLC、ATM、PPP など。
> これらについて調べることは読者に任せます。

10.3 LAN の構成機器

10.3.1 復習

キャンパスのネットワークといった組織内をつなぐネットワークはLAN と呼ばれます (7.2 節参照)。最近の LAN は、たいてい VLAN (7.4 節) を用いるスイッチ群から構成されています。

10.3.2 スイッチ間の接続

では、どのようにスイッチ間を接続しているか？を概観しましょう。

そもそも、キャンパスには大きな単位から順に「建物」「階」「部屋」があります。

- 建物
  複数の建物。
- フロア
  建物の中に複数のフロア。
- 部屋
  一つのフロア中に複数の部屋。
  身近な例をあげると、研究棟は各階に 20 程度の部屋があります。

建物間の接続と部屋の中での接続では異なる技術が使われます。一般に長距離接続は光ファイバー、数m単位の接続は銅線を使ったイーサネットを用います。

10.3.3 光ファイバー

詳しくは専門の授業にまかせることにして、ざっとまとめます。

光ファイバーは、大きく分けて、二種類あります。長距離用のシングルモードファイバ (single mode fiber) と、短距離用のマルチモードファイバ (multi mode fiber) です。それぞれ SM と MM と略すことがあります。別の流儀で、シングルモードを LX (long の L)、マルチモードを SX (short の S) と呼ぶ場合もあります (頭文字が同じ S でも逆のファイバになってしまうので気をつけよう)。

建物間 (つまり野外にケーブルを敷設する場合) や同一建物内でも長いケーブルが必要な場合はシングルモードを使っています。逆にフロアをまたぐ場合やフロア内でケーブルを引きまわす場合など短距離の場合はマルチモードが普通です

部屋の中での引きまわしは通常イーサネットケーブルで、光ファイバーを使うことは稀です。フロア内のケーブル引きまわしであればイーサ

年	事項
1972	Xerox で開発された
1973	Ethernet と命名された
1980	DIX (DEC, Intel, Xerox) 規格
1980	02 月、IEEE 802 プロジェクト開始
1982	DIX version 2
1983	802.3 が規格として承認
1988	10base2
1990	10baseT

表 10.1　イーサネットの歴史: なお 1980 年 2 月に IEEE のプロジェクトが始まったので 802.3 規格と呼ばれる。

ネットケーブルで十分な場合も多いでしょう。

10.4　イーサネットの種類

一口にイーサネットと呼ばれていますが、実際にはさまざまな規格があります。

10.4.1　歴史 (概要)

10 Base 5 → 10 Base 2 → 10 Base T → 100 Base → 1G (1000 Base) → 10G → 40G/100G

10 Base 2 と 10 Base T の間でバス型からスター型へと構成の大きな変更があります (後述)。

10.4.2　速度

イーサネットと言われるものには 10 Mbps ～ 100 Gbps にわたるさまざまな規格があります。

イーサネットが最初に登場した時の速度は 10 Mbps でした。

現代では 100 Gbps イーサネット規格が決まりつつある時代です。もっとも、1 Gbps 以上の高速回線は長距離回線ないしは強力なサーバ用途のもので、一般人向けではありません。

建物間の接続であれば 10 Gbps イーサネットを使う場合もあり得ますが、機材が高価なため、見る機会は多くありません。

というわけで、一般の PC や家電量販店でよく見かける、つまり身近にある回線 (安価に入手可能な回線) としては 100 Mbps もしくは 1 Gbps のイーサネットのいづれかであるといえます。

10.4.3　ネットワークの形

10 M イーサネットの初期はバス型でしたが、10 baseT と呼ばれる第三世代からはスター型になりました。

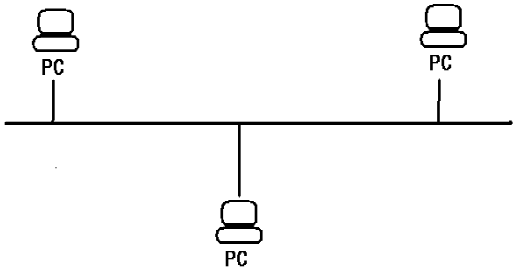

図 10.2　バス型: 10 Base 5 や 10 Base 2 は、こういった構成。

バス型

バス型 (BUS) とは、一本の回線 (同軸ケーブル) に複数の機器が接続している形です。「どこか一箇所でも切れればすべてが終り」という単純な形ともいえます。

最初のイーサネットは、この形でした。

通信媒体が一つしかないため、イーサネットフレームつまり同軸ケーブル上に電波を送信して良い PC は一台だけです。ケーブル媒体が一つしかないため、同時に複数台の PC が送信すると同軸ケーブル上で電波が混ざりあい、受けとった側で正しく解読できません。

こういった場合、どうするか？

いつものように割と単純な解決法を取りました。

適当に送信して良い PC を一台決め、残りの PC は送信を待つことになります。では、どうやって送信して良い PC をどうやって決めるのか？というと、イーサネットでは CSMA/CD という方式で制御をおこなっています。

おおざっぱにいって CSMA/CD は、あうんの呼吸で適当に早い者勝ちを決めるとでもいうような制御をしています。非常に単純な方式ですが、低速度のネットワークであれば十分動作する仕組みです。

スター型

10 base T では、自転車の車輪 (ハブ-スポーク構造、図 10.3) のようにスイッチを中心に PC をつなげる形に接続します。これをスター型もしくはハブ-スポーク型 (図 10.3) と呼んでいます。

10 Base T の普及により、もっともよく使われる形になりました。

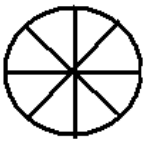

図 10.3　自転車の車輪: ハブ-スポーク型

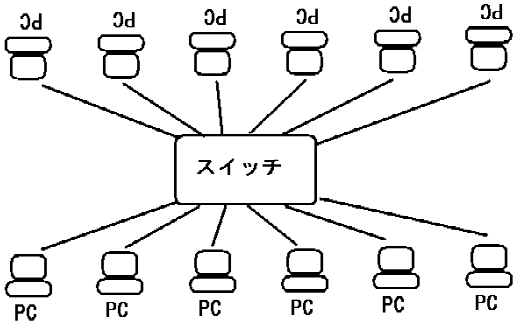

図 10.4　PC 教室の例: スイッチの周りに PC が接続されている。この形がスター状。

コラム: リング型

イーサネットでは使われていませんが、バス型、スター型と並ぶ有名なネットワークの形としてリング型というのがあります (図 10.5)。

リング型では、トークンリング (token ring) や FDDI (Fiber-Distributed Data Interface) が有名です。

FDDI は光ファイバーを使っています。最大 2 km まで通信可能です。

機器を光ファイバーでリング型に接続します。各機器 (ノード) の間はファイバーを二本接続するため、ファイバーの一本分が壊れても通信できます。ただし、その障害部分で折り返し運転です。

障害には強い作りのため、われわれのようなプロバイダの人から見ると素敵なシステムでしたがイーサネットの進化速度に対抗できず、10 Mbps より高速の規格も現れなかったため、1990 年代なかばで姿を消しました。

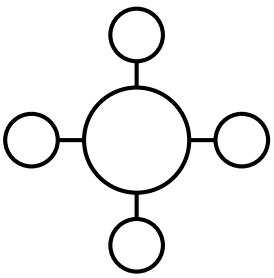

図 10.5　リング型

以下では、具体的にイーサネットの歴史を追ってみましょう。

今となっては初期のイーサネットの実用性など皆無ですが、初期のイーサネットがバス型であっ

たことは、イーサネットの動作原理を理解する上で重要です。

10.4.4 10 Base 5

元々のイーサネットとは、この 10 Base 5 規格をさします。

理論上の転送速度は 10 Mbps で、媒体は 1/2 inch = 1.27cm の同軸ケーブルでした。一つのケーブルで最大 500 m まで通信可能、ネットワークの形はバス型です。

ケーブルの外皮が黄色いものが多かったため通称「イエローケーブル」と呼びます。黄色が多かったというだけで、もちろん他の色のケーブルもあります。

太いケーブルだったので thick ethernet とか thick wire などとも呼ばれていました。

イーサネットは OSI モデルのレイヤー 2 に見えますが、正確にはレイヤー 2 の下半分の (MAC = Medium Access Control) 副層に相当します。

10.4.5 10 Base 2

10 Base 2 は 10 base 5 と同様にバス型の接続です。

速度が 10 Mbps というのも 10 Base 5 と同じですし、同軸ケーブルなのも同じです。ただケーブルが 10 Base 5 より細く (thin)、安価で、ケーブルの配線が楽なため、部屋の中の配線で用いられました。

このため thin cable とも呼ばれます。

なお、一つのケーブルで通信可能な距離は約 200 m です。

10.4.6 10 Base T

細い銅線を「よりあわせた」(twisted) ケーブルなので 10 Base T と呼ばれています。T は Twisted pair の T です。

10 Base T では、スイッチという機械を置き、スイッチと PC をイーサネットケーブルで接続します。一つのスイッチには複数の PC が接続し、スイッチは、さらに上流のスイッチへ接続しています。このため、ネットワーク構成はスター状の接続になります。

ただし、このスイッチをつないでよい段数には制限があります (規格ごとに制限が異なります)。

10.4.7 100 Base

100 Mbps 対応のイーサネット規格です。

これ以降のイーサネットでは、速度をあげるためのさまざまな工夫がされていますが、基本的なネットワーク構成や使い方は 10 Base T と同じと考えていて問題ありません。少なくとも一般ユーザは、それで OK です。

ただ、高性能のケーブルや機材を使わないと期待通りの性能が出ないので、注意しましょう。

特にケーブルの自作などは危険です。最近では市販のものでも十分安いので、無駄な苦労をさけるために購入してください (演習: 苦労したい人は実際に自作し性能テストをしてみてください。また、うまくいくまでの手間暇で時給換算してみなさい:-)。

なお、100 Base の場合、素人でもギリギリ 100 Mbps が出せるケーブルが作れることもあります。ただ、これ以上の速度のものを自作するのは不可能です。

10.4.8 1000 Base

1000 Mbps つまり 1 Gbps 対応のイーサネット規格です。

最近では PC にも 1 G のネットワークインターフェイスがつき、家電量販店で 1 G 対応のスイッチが安く売られている時代なので、すでに家庭用としても 1 G のイーサネットは十分一般化したといえるでしょう。

ケーブルの規格: カテゴリー 6 じゃないとだめ？
なお、ケーブルはカテゴリー 6 と呼ばれる容易に手に入る市販品の中で一番厳しい規格のものでないと性能が出ないので注意しましょう。
まぁ家庭用では、どのみち完全に 1Gbps を出し続けられる機材自体が珍しいので 1 Gbps イーサネットでも、カテゴリー 5 や 5e で十分です。

10.5 イーサネットの動作原理

図 9.1 のように、IP パケットは、PC からルータ A へ、ルータ A からルータ B へと次々に隣のネットワーク機器へ転送を繰り返していきます。

隣り合う機器の間では、図 10.6 のようにイーサネットフレームが IP パケットを運びます。

イーサネットフレームの転送は、どのように行なうのでしょうか？

10.5.1 MAC アドレス

ネットワークインターフェイスにはイーサネット的な住所があります。

工場出荷時に、ネットワークインターフェイス (ネットワークカード) に一つ一つ異なる数字が割り当てられています。その数字は 48 ビットの長さです。

これは MAC アドレスと呼ばれています。MAC アドレスの表示は十六進数です。48 ビットを 6 つに区切り：で区切って表記します。

普通の PC には一つのネットワークインターフェイスしかないので、各 PC には世界で唯一の MAC アドレスが割り当てられており、各 PC を区別することが出来るといえます。

演習

演習: CentOS 上で /sbin/ifconfig -a コマンドを実行して確認してみよう。

NetBSD での例: 00:02:b3:eb:50:0e の部分が MAC アドレスです。

```
% /sbin/ifconfig -a
fxp0: flags=8843<UP,BROADCAST,RUNNING,SIMPLEX,MULTICAST> mtu 1500
        capabilities=6<TCP4CSUM,UDP4CSUM>
        enabled=0
        address: 00:02:b3:eb:50:0e
        media: Ethernet autoselect (100baseTX full-duplex)
        status: active
        inet 10.255.0.1 netmask 0xffff0000 broadcast 10.255.255.255
        inet6 fe80::202:b3ff:feeb:500e%fxp0 prefixlen 64 scopeid 0x1
... 略 ...
```

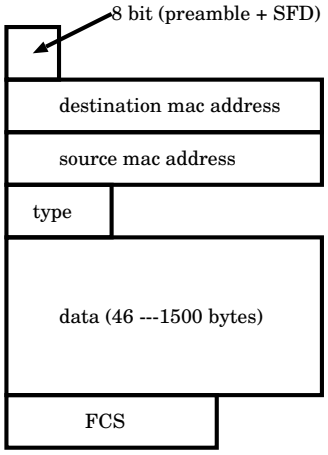

図 10.6 イーサネットフレーム

10.5.2 イーサネットフレーム

イーサネットフレームは図 10.6 のようになっています。

今まで出てきたパケットと最初の 8 ビット分が大きく異なります。また、データ部分の後にも続きがあります。

ただ、この二箇所を除けば、今までのパケットと同様です。二段目 (先頭から数えると 9 ビット目以降の 48 ビット分) は宛先 MAC アドレス、三段目は送信元 MAC アドレスです。

先頭 8 ビット

前章までと異なり、イーサネットは、かなりハードウエアに近いローレベル (low level) な話

です。

ようするに、銅線の上に電気信号を流すわけですが、電気信号には最初と最後という特別な印がありません。また、電磁波は線形方程式[*1] なので複数の信号が重なり合いもします。

そこで、イーサネットでは「送信の開始を知らせ、同期を取るため」の特別な印をフレームの先頭につけています。これは 7 ビットの preamble と 1 bit の SFD (Start Frame Delimiter) からなり、二進数では 10101011 です。

長さ/タイプ

イーサネット的な住所である 2 つの MAC アドレスに続き、16 ビットの「長さ/タイプ」フィールドがあります。

これは数字が 1500 以下なら「長さ」の意味、1536 以上の場合は「タイプ」の意味と定義されていますが、実際の実装では長さにかかわらずタイプとして使っているケースが多いようです。

たとえば 0x0800 (十進数では 2048) は IPv4 を意味します。

FCS

データ本体の後に FCS (Frame Check Sequence) があります。これはエラー検出のための CRC (Cyclic Redundancy Check) 値です。

10.5.3 イーサネットの基本動作

まず通信して良い機器どうしを選びます。これはCSMA/CD という方式で制御しています。

そして CSMA/CD で選ばれた一つの機器が、送りたい宛先の MAC アドレスに向けてイーサネットフレーム (つまり電気信号) を送信します。もちろん、そのフレームの送信元は自分の MAC アドレスです。

しかしながら、その電気信号は銅線すべてに伝わります。ということは、同じイーサネットに接続している機器すべてにその信号が見えてしまうわけです。

では、どうするのか？というと、安直な解決策を取ります。

なんにせよ、全員受けとってしまうものはしょうがありません。受けとった側で「自分宛なら受信」「そうでないなら廃棄」という取り決めとします。

ここで「全パケットを必ず調べる」ことに注意してください。イーサフレーム一つ一つ全てについて、自分宛かどうかを調べて受信するか廃棄するかを決める必要があります。これは、かなり無駄な作業です。

CSMA/CD

CSMA/CD は Carrier Sense Multiple Access with Collision Detection の頭文字です。

これは人間どうしが同時に会話している様子をモデル化したものと考えてください。

- Carrier Sense
  ネットワークを監視し、誰も使ってないようであれば送信を開始する。
- Multiple Access
  複数の機器が同時に通信を開始し得る。実際には互いに微妙に送信タイミングがずれるように工夫するのだが、衝突することがある。
- Collision Detection
  ほぼ同時に送信してしまった (衝突してしまった) 時は、あきらめ (現在の処理は破棄し)、再送する。この場合、パケット全体を破棄し全体を再送する。
  再送のタイミングも微妙にずらすなど、いろいろ工夫する (注: 途中から送るといった、より高度な再送処理が必要なら、それは上の層が行なうべき)。
  ちなみに、衝突のことを Collision (コリジョン) と呼ぶ。

[*1] ついてきてるか？マックスウエル方程式をおさらいしなさいね。

10.5.4 コリジョンの見える範囲は？

上述の通り、電気信号は銅線の上なら、どこででも見えています。

つまりコリジョン (Collision、衝突) が見える範囲は？というと、同じケーブルに接続している全てのネットワーク機器です。

コリジョンが見えるということは乱れた電気信号が見えているという意味なので、通信がうまくいっていない状態を意味します。そのため、出来るだけコリジョンが見えないことが望ましいといえます。

10.5.5 一斉通知するには？

今まで通信というと、「ある送信者」と「ある受信者」の間、つまり二者間の通信のことを指していました。

一方、「電気信号は銅線の上なら、どこででも見える」ことを前向きに考えると、「ある送信者」から全機器へ一斉に通信を送ることが出来そうです。

この一斉通知の方式はブロードキャストと呼ばれます。

ブロードキャストは、すでに聞いたことがありますね？

IP アドレスのところ (7.6 節) で、ブロードキャストアドレスというのが出てきました。それです。

今、その正体が明かされます。

10.6 ブロードキャスト

10.6.1 ブロードキャストパケット

「ある送信者」から全機器へ一斉に通信を送るためには、CSMA/CD で「ある送信者」を決めた後、特定の宛先へイーサネットフレームを送ります。

電気信号は全機器に見えているので、このフレームは必ず全機器に見えています。

全機器は「自分宛」と認識して受信します。

この特別な宛先 MAC アドレスは F-F:FF:FF:FF:FF:FF つまり 48 ビットすべてが 1 という最上位 MAC アドレスと決められています。

これが IP でいうブロードキャストアドレスの正体です。

10.6.2 ブロードキャストの例

10.0.0.0/24 のネットワークを考えます。

このネットワークにある全機器に通信を一斉に送りたい場合、10.0.0.255 宛へブロードキャスト IP パケットを送信します。これは IP 層の判断です。

OS の中では IP 層からイーサネット層に処理が渡されます。

イーサネット層は FF:FF:FF:FF:FF:FF 宛へ、この IP パケットを送信します。これがブロードキャストです。

このイーサネットフレームは、宛先が F-F:FF:FF:FF:FF:FF で、送信元は自分の MAC アドレス (たとえば 00:0d:60:cf:ad:ba) です。

なお、そのフレームに対する返事は、その MAC アドレス (00:0d:60:cf:ad:ba) 宛へ返ってくるわけです (返事のフレームの送信元は宛先の MAC アドレス、宛先は 00:0d:60:cf:ad:ba)。

コラム: セグメントの定義

物理的か論理的かという違いはありますが、LAN および VLAN というセグメントは「ブロードキャストが届く範囲のネットワーク」と再定義されます。また、この範囲をブロードキャストドメインと呼ぶことがあります。

10.6.3 ケーススタディ: ARP

ブロードキャストを使えば、隣り合う全機器に問い合わせを送ることが出来ます。同一セグメント内の全機器に問い合わせを送ることができるわ

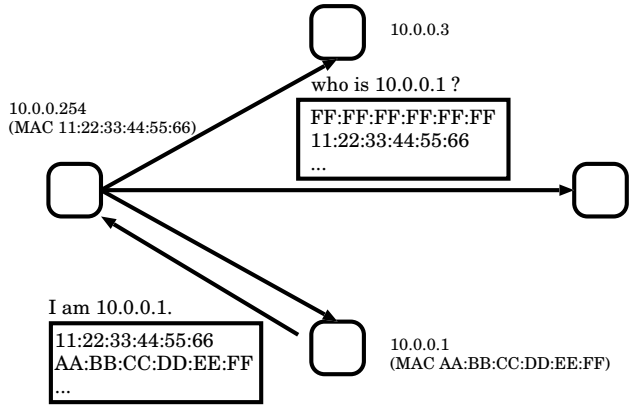

図 10.7　ARP

けです。

このテクニック「全員に聞いてみる」は大味ですが、ネットワーク機器だけで自動処理するためには必須の動作です。

ARP(Address Resolution Protocol) は、このテクニックを使う代表例です。

最初から言っている通り PC は IP アドレスで互いの住所を認識しています。一方、ある PC から隣の PC への転送はイーサネットフレームによります。この場合の住所は MAC アドレスです。

なんとかして、IP アドレスと MAC アドレスの対応関係が分からないと通信が成立しません。それも自動処理できないと嬉しくありません。そこで ARP を使います。

図 10.7 を参照してください。

1. PC (10.0.0.254) は 10.0.0.0/24 のセグメントに対して「10.0.0.1 の IP アドレスの PC は誰だ？」という問い合わせをブロードキャストで送ります。
2. このブロードキャストは全機器に届きます。
3. 10.0.0.1 の PC が自分のことだと気づき、10.0.0.254 へ返事を送ります。
4. 10.0.0.254 は、その返事のイーサネットフレームの送信 MAC アドレス AA:BB:CC:DD:EE:FF から「IP アドレスが 10.0.0.1 の PC の MAC アドレスは AA:BB:CC:DD:EE:FF」ということを知るわけです。

10.6.4　ケーススタディ: RARP

では、逆に、「MAC アドレス → IP アドレス」という変換をしたい場合もあるのか？というと、あります。

それがRARP (Reverse ARP) です。

ハードディスクのない PC の起動時などに使います。

10.6.5　ケーススタディ: DHCP

「全機器へ問い合わせを送る」の応用として「PC が自分の設定情報を教えてくれ！」という依頼を送信するということもできます。

全機器が、その依頼内容を含んだイーサネットフレームを受信してしまいますが、関係ない PC は無視します。特別なサーバが応え、設定情報を教えてくれます。

これがDHCP(Dynamic Host Configuration Protocol) です。

10.6.6　ケーススタディ: Microsoft Network

ブロードキャストを使えば「全機器へ自分の情報を送る」ことも出来るわけです。

Microsoft Windows では、ブロードキャストを使い、自分のセグメント内の全機器へ定期的に自分の情報 (ホスト名と IP アドレスなど) を送っています。各機器は、送られてきた情報をかき集め「マイネットワーク」の一覧を生成しています。

これが「同じセグメントでないとマイネットワークが使えない」「起動してすぐにはマイネットワークが見えない」という理由です。

10.7　まとめ

イーサネット

1. イーサネットではデータのかたまりをフレームと呼ぶ。TCP/IP と異なり、フレームには

ヘッダとトレイラがある。

2. 速度は 10Mbps 〜 100Gbps
3. 接続の形状による分類として
   （a）バス型 (10Base5, 10Base2)
   （b）リング型 (Token Ring, FDDI)
   （c）スター型 (10Base-T 〜)
4. 機器を区別するために 48 bit の MAC アドレスを用いる
5. CSMA/CD (Carrier Sence, Media Access / Collision Detection) を用いて、イーサネットフレームを送出する機器を一台だけ選ぶ。ちょうど井戸端会議の調停のようなものである。
6. ブロードキャストによる一斉通信を応用した例
   （a）ARP (IP アドレスから MAC アドレスを探索)
   （b）RARP (MAC アドレスから IP アドレスを探索)
   （c）DHCP (ネットワークの自動設定)
   （d）Windows のマイネットワークなど

第 11 章

セキュリティ

11.1　注意

今回の話は防衛の仕方を理解するために、さまざまな攻撃手法を取り上げています。くれぐれも自分で実際に試すなどしないように。

また、今回分については書くと問題があるかもしれないので、メモ代わりに使って下さい、程度の章です。

11.2　概要

まず「ネットワークセキュリティ」と言われているものの範囲は非常に広く、また詳細は OS の動作を抜きにして説明することが難しいのが現実です。

本章は、ユーザにとって気をつけるべきテーマを中心に取り上げます。

11.2.1　用語と分類

ネットワークセキュリティとは？というと、おおまかにいえば「ネットワークの安全を確保するための防衛策」のことになるでしょうが、ネットワーク以前の社会常識的なレベルのものも多数あります。また、広義にはネットワークを安定して使うことも含まれるでしょう。

非常に広義で曖昧な概念です。

分類例

- 様々な脅威からのシステム防衛
  - システムの弱点を防衛する
  - ウイルス対策: virus, warm, bot, ...
  - 適切なパスワード管理
  - 機密データの保護
  - 情報漏洩の監視、情報漏洩を防ぐ
- より安定した運用という面も、広義には含まれる
  - ユーザのパスワードの定期的な変更
  - 電源の安定化 (たとえば無停電電源装置の導入)
  - 重要なデータの定期的なバックアップ
  - 職種や部署によるアクセス制限
  - さまざまな保守
  - 設定の維持管理

コラム: 一昔前のネットワークセキュリティ

”最低限”のネットワークセキュリティとは「クラッカーからコンピュータを守るために、ファイアウォールを設置すること」だ。

そうか？

コラム: サイバーアタックなんて本当に行なわれているの？

11.2.2 敵

かつては愉快犯や腕だめしをしたいハッカーたちでしたが、そんな時代は、とっくに終っています。

現在、ウィルスや SPAM をばらまいたりする人たちは、お金儲けが目的です。

例

- メール
  SPAM は倍倍ゲームで増加中。
  例: SPAM メールを送ってウィルスなどに感染させる。「感染させて個人情報をぬすむ」もしくは「感染させて BOT にする」
- WWW
  ウエブページに罠を仕込む。

コラム: 無防備な Windows はどうなる？
すっぴんの Windows をインターネットにつなぐと？

11.3 ウィルス対策

金を使え！それ以外に効果のある対策はないです。

なお、会社の場合、会社がウィルス対策ソフトを支給しますので、自宅の PC の心配だけすれば十分です。

11.4 正しいパスワード管理

最も重要な 2 つのことは「正しいパスワードの付け方を教えること」そして「定期的にパスワードの変更を強制する」システムの採用です。

11.4.1 パスワードは必ず解かれる

現在のシステムでは、原理上、パスワードは必ず解かれます。

通常、パスワードに使っている文字列はキーボードにあるものだけですから、ASCII 文字列つまりせいぜい 7 ビット分の 127 文字しかないわけです (注: 実際にパスワードに使うような文字は、さらにその一部で、数十文字しかない)。

破るテクニックの基本は総あたり戦です。これはブルートフォースと呼ばれています。

辞書に出てくる単語などであれば一秒もかからずにパスワードは発覚します。これをふせぐためには、探索空間を広くすることです。

いずれにせよ、数十文字[8] くらいしかパスワードはありえないので、総辺り戦をすれば必ずいつか解かれます。問題は、それにかかる時間と経済性 (時間をかけても解くメリットがあるか否か) です。

瞬時に破られないようなパスワードであれば、敵は弱いパスワードをつけている人の方にむらがっていきます。

適当に難しいパスワードをつけ、定期的にパスワードを変更する。これが実際に運用可能なパスワード管理の妥協点でしょう。

11.4.2 正しいパスワードの付け方

- 辞書にある単語は御法度。
- アルファベットだけではいけない。
  アルファベットも大文字と小文字を混ぜ、それ以外にも、数字、特殊文字 (例 , / # $) などを混ぜる。

ただ、あまりにも難しいパスワードをつけることを強制するとセキュリティが下がります。とい

うのは、パスワードが覚えられないと、紙に書いてモニタに張る人が出現するからです;)

コラム: 掃除のおばちゃんに化ければ…

自分の好きなフレーズの頭文字などがお薦めです。

```
g'hharww = God's in his heaven All's right with the world.
```

コラム: 個人の価値は 500 円

11.5 アタック

よくネットワークのセキュリティなどと言いますが、コンピュータ以前の手口も多いです。インターネット固有の手口もあれば、そうでないものもあります。

社会常識として危ないことはコンピュータネットワークでも危ないのです。

ただ、同じ手口なら、ネットワークを利用するバージョンの方がより危ないとは言えるでしょう。これは顔をつき合わせると分かるナニカ (付加情報) が得られない分だけ「危険」という意味です。

11.5.1 例: パスワードを手に入れる

- SF
  ICE[*1] breaker (「クローム襲撃」「ニューロマンサー」)?
- 古典
  総辺り戦。
  物理攻撃。
- ソーシャルエンジニアリング
  - シャーロックホームズの「四つの署名」
    ホームズ:「オーロラ号というのは黄色に緑のラインだっけ?」
    おかみ: 「いえ、最近塗りかえたので、煙突に白い線が二本ですよ」
  - 本人や業者のフリ。
    「パスワードを忘れたので、適当なパスワードに変更して、いますぐ教えて欲しい」(注: 電話の主が本人であると言う証拠はない)。
  - オレオレ詐欺
  - 還付金詐欺
- リバースソーシャルエンジニアリング
- もう少しインターネットを使った例。
  SPAM とかウィルス、ボット、ウエブスプーフィング…

コラム: オレオレ詐欺

[*1] ギブソンの SF に出てくるネットワーク機材。現代で言うところの、(迎撃システムつきの) ファイアウォールみたいなもの。

あとがき

Unix Lecture Trilogy は第三部のオペレーティングシステムへ続きます。

本教科書 (講義) は、初心者に基本的な知識を教えることが目的です。

難易度は、会社の新人研修で利用可能なレベルを想定していますが、ネットワーク管理者やネットワーク・プログラマを目指すなら、しょせん、その第一歩でしかありません。

プログラミングのテクニックには TCP/IP 以前に、OS の内部についての知識やそれぞれのプログラミング言語固有の罠をふまない技術、そしてセンス[*2] が大事です。

センスは大人数講義形式 (いわゆる座学) で教えられるものではありませんが、(センス以外の) 知識の手がかりは付録 **??** を参照してください。

深町 賢一
三角山が今日も奇麗な形をしている琴似にて

[*2] ゴーストのささやき:-)

付録 A

用語集

あえて簡単な説明にとどめてあります。

これらの用語は講義の中で説明するので、不足している部分については自分で埋めていってください。また、用語集やインターネット検索で分かるものばかりなので、分からないところは自分で探索して埋めるように心がけて下さい。

たとえば

```
http://www.google.co.jp/
```

で用語を検索するとか、用語集のページ

```
http://e-words.jp/
```

で検索するとかが出発点です。

ただし、インターネットに限らずテレビ、新聞、雑誌もですが、与えられる情報が正しいという保証はありません (必ず書いた人の偏見が入っています)。少なくとも複数の情報源をたどり、その共通部分だけが “たぶん正しい” と考える態度を見につけるように努力してください。

そういった努力をしないと簡単にオレオレ詐欺などにひっかかる人になりますよ！

■1000 BASE-T　1 Gbps (1000 Mbps) の LAN 接続の規格。ツイストペアケーブルを使うので “-T”。

■100 BASE-T　100 Mbps の LAN 接続の規格。ツイストペアケーブルを使うので “-T”。

■10 BASE-2　10 Mbps の LAN 接続の規格。導波管を使う。

■10 BASE-5　10 Mbps の LAN 接続の規格。太いケーブル (イエローケーブル) を使う。

■10 BASE-T　10 Mbps の LAN 接続の規格。ツイストペアを使うから “-T”。

■7 ビット　ASCII のことを指すことがある。

通常 8 bit が最小単位だが文字を表現するには 7 bit (0 から 127) まであれば十分な言語もある。例えば英語は 7 bit あれば十分であるので英語の通信は元々 7 bit である。

■ADSL　“Asymmetric DSL”。電話線を利用した通信方法。上りと下りで速度が異なるので非対称 (asymmetric) と呼ばれる。NTT のフレッツ ADSL や Yahoo BB などが ADSL を使うサービスの例。

■ANSI　“American National Standards Institute”。米国規格協会もしくは米国規格協会の定めた規格の通称。

■apache　代表的な WWW サーバのソフトウエア。

http://www.apache.org/

■API　アプリケーションプログラムインターフェイスたとえば「SMTP にとって重要なのは TCP の API」。

■ARP　“Address Resolution Protocol”。

ARP は動的にインターネットアドレスとイーサネットアドレスの間の変換を行なうためのプロトコルであり、すべてのイーサネットインターフェイスドライバで使われる。

■ARPA “Advanced Research Projects Agency”。DARPA の旧名称。“DARPA” の項目を参照。

■ARPANET ARPA (のちの DARPA) の資金提供の元で作られた研究ネットワークのこと。(結果論であるが)、ARPA の資金により、後にインターネットの基礎となった研究が行なわれた。その本来の研究目的は、核兵器の攻撃に耐え得る “電話とは異なる通信方式” の研究であった。

■ASCII 7 ビットで表現される文字コードの体系。ANSI の規格。

■ASIC “Application Specific Integrated Circuit”。特定の目的のために作られた専用の集積回路 (LSI)。

■ATM “Asynchronous Transfer Mode”。セルと呼ばれる小さな単位のパケットに分割して通信する方式。

■bps “bit per second”。一秒間に伝送できるビット数。回線速度の単位。バイト単位ではないことに注意。

■BSD “Berkeley Sofotware Distribution”。

カルフォルニア大学バークレイ校が開発した Unix の方言の一つ。インターネットの起源を考える上で最も重要なオペレーティングシステム。

ARPA との契約で 4 BSD (BSD バージョン 4) に TCP/IP を実装したこと、BSD をフリーソフトエアとして当時のコミュニティに提供しようとしたことが後のインターネットの攻勢とインターネットの根底にある相互扶助の精神につながっている。

1980 年代当時 BSD は研究用のため安く手に入れることができたこと、BSD がサポートしていた DEC 社の VAX11 が 80 年代のアメリカの大学で広く使われていたことなどが、 草の根から 80 年代のインターネットが作られていく原動力となったといえる。

バークレイによる開発は 4.4 BSD Lite で終了したが、開発は NetBSD, FreeBSD, OpenBSD プロジェクトに引き継がれた。

現代ではアップル社の Mac OS X の部品となり、間接的に BSD は広く使われているといえる。また、商用への転用がライセンス的に問題がない[*1] ため、ひそかに NetBSD を使っている組み込み機器は多い。たとえばリコーのプリンター IPsio の中身は NetBSD である。

■BSD ライセンス BSD が採用しているライセンス。時代により少しずつ変わってはいるが、商用利用がしやすいラインセスである。Freedom も大事だけれど、なにより皆さんに使っていただきたいというのが BSD の理念である。

■B フレッツ 光ファイバーを使った NTT の接続サービスの名称 (商品名)。

■CIDR “Classless Inter Domain Routing”。

IP アドレスのクラス概念をなくした IP アドレスの利用体系。現代のルーテイングは、この概念を前提としている。

■CIDR アドレッシング CIDR を前提にした IP アドレスの割り当て方法。

■CSMA/CA “Carrier Sense Multiple Access with Collision Avoidance”。無線 LAN で利用されている通信方式。

■CSMA/CD “Carrier Sense Multiple Access / Collision Detection”。イーサネットの基本動作原理。

■DARPA “Defense Adfanced Research Projects Agency”。

アメリカ国防総省の研究部門の名前。ARPA の旧名称。

■DBMS “DataBase Management System”。データベース管理システム。商用の Oracle、

[*1] なお Linux など GPL 絡みのソフトウエアを使う組み込み機器を売っているような会社は法務部とかないような “いかがわしい会社” である “可能性が高い”。まともな会社なら、いつ訴えられるか分からない “法律的解釈が微妙なソフトウエア” を使うことはない。

Sybase、Microsoft SQL サーバ、フリーソフトウエアの MySQL、PostgreSQL などが有名なソフトウエア。

この単語は “データベース” より “管理システム” というところが重要であることに注意。

■DHCP “Dynamic Host Configuration Procotol”。動的にコンピュータの設定を行なう仕組みのこと。DHCP クライアントは設定を DHCP サーバから教えてもらう。

■DMZ ファイアウォールなどで守られた「インターネット公開サーバを置く」セグメントのこと。

業界用語で “出島” などとも呼ぶ。

■DNS “Domain Name System” の略。ドメイン名と IP アドレスの変換を行なう分散データベースシステム。

■DNS サーバ DNS の問い合わせに答えるプログラム。もしくはそのプログラムが走っているコンピュータのこと。

■DSL “Digital Subscriber Line”。

■DSU “Digital Service Unit”。ISDN や専用線、B フレッツなどの終端装置として置かれるネットワーク機器。

■Ethernet “イーサネット” を参照。

■FQDN “Fully Qualified Domain Name”。完全修飾されたドメイン名は、インターネットにおける正式なコンピュータの名称に相当する。例: mail.chitose.ac.jp

■Free Software Free of Charge (無料) な Software ではなく Free(dom) of Software。Open Source と Free Software は (利用者にとっては同じようなものだが) 似ていても違う概念。

もっとも “違う” と思っている派閥と “同じ” と思っている派閥がいるわけなのだが…

■Free Software Foundation GNU プロジェクトを推進する団体。Richard. M. Stallman が創設。

初期は MIT 人工知能研究所の一室にあった。旧職員 (R. M. Stallman) の謎の活動に理解を示し、部屋を貸して上げた MIT 人工知能研究所の太っ腹なところも讃えたい。

■FSF “Free Software Foundation” を参照。

■FTP “File Transfer Protocol”。ファイルを転送するプロトコル。

■Gigabit Ethernet 1 Gbps (1000 Mbps) の LAN 接続の規格。

■GNU “GNU is Not Unix”。

1980 年代に Richard Stallman が始めた Free Software で Unix 互換のものを一式作り上げるプロジェクト。GCC (GNU C Compiler)、GNU Emacs を始め有名なソフトウエアが他数ある。

なお、1990 年代なかば以降の GNU プロジェクトは 1980 年代の GNU とは別物であるように見える。

■GNU Public License GNU の名前を冠するソフトウエアが採用しなければならない GNU プロジェクトのライセンス。コピーレフト (Copyleft) という概念と密接に関連がある。

GNU ライセンスのソフトウエアを商用利用することには注意が必要である。だが、世の中の製品にはこのライセンスの意義を無視しているものも多い。

なお、1990 年代なかば以降の GNU プロジェクトは 1980 年代の GNU とは別物であるように見える。

■GPL “GNU Public License” を参照。

■HDLC “High level Data Link Control”。レイヤー 2 のプロトコル。専用線サービスなどで使われている。

■HTML “HyperText Markup language”。ウエブページを記述する言語。

W3C が定義を提唱している。

■HTML メール 元々、電子メールはテキストで記述することが期待されている。よって HTML で書かれている、もしくは HTML ファイルが添付されている電子メールは嫌われる。

ただメールソフトによっては自分が HTML メールを出していることに気づかないこともあり、知らないうちに人に嫌われていることがあるので注意するべきである。

■HTTP “HyperText Transfer Protocol”。

WWW ブラウザと WWW サーバ間の間のデータ転送プロトコル。

■HTTPS 安全なデータ転送が出来るよう SSL の上に HTTP を流すプロトコル。つまり “HTTP over SSL” のこと。

■HUB ハブを参照。

■IAB “Internet Architecture Board”。インターネット技術の標準化を検討・決定する機関。1983 年創設。下部組織に IETF や IRTF がある。

■IANA “Intenet Assigned Numbers Authority”。

インターネット上の資源の標準化や割当 (IP アドレス、プロトコル番号など) を行なう組織。ICANN 創設に際し IANA の機能は ICANN へ移された。

■ICANN “Internet Corporation for Assigned Nmaes and Numbers”。ドメインや IP アドレスを管理する非営利組織。IANA の後継。

こういった組織が国家の支配下にないことがインターネットの特徴の一つと言える。

■ICMP “Internet Control Message Protocol”。

レイヤー 3 の IP プロトコルとセットで通信を制御するプロトコル。いろいろなメッセージやエラーメッセージなどを伝えるためのもの。

■IEEE “Institute of Electrical and Electronics Engineers Inc.”。

電器電子部品や通信方式の研究開発、標準化を行なうアメリカの学会。

■IEEE 802.11 無線 LAN の標準規格群。802.11b (2.4 GHz 帯で約 11 Mbps)、802.11a (5.2 GHz 帯で約 54 Mbps)、802.11g (2.4 GHz 帯で約 54 Mbps) など。

最近では 802.11n というのが出てきている。

■IEEE 802.3 イーサネットの標準化規格のこと。

IEEE における規格化が 1980 年 2 月に始められたため 802 という数字がついている。

■IETF “The Internet Engineering Task Force”。RFC を発行し、意見を求め、標準化を推進する組織。

http://www.ietf.org/

■IIJ “Internet Initiative Japan, Inc.”。株式会社インターネットイニシアティブ。

日本で最初[*2] の商用インターネットサービスプロバイダ。

http://www.iij.ad.jp/

■IMAP4 メールサーバからメールを取り出すプロトコルの一つ。RFC 1730 他を参照。

POP3 と異なり、クライアントで行なう検索などの機能をサーバ側に持っていることが特徴。移動体通信システムで便利なプロトコルだが、ISP がサポートしていないのでいまいち流行している気配はない。

■IP “Internet Protocol”。レイヤー 3 のプロトコル。“到達性を保証しない” という意味で UDP 的な性質を持つ。

■IPv4 IP プロトコル、バージョン 4。現行の IP バージョン。

■IPv6 IP プロトコル、バージョン 6。次世代の IP 。

■IP アドレス インターネットにおける住所に相当するもの。地理的な住所とは無関係であることに注意。

現在、特に指定しなければ IP アドレスは IPv4 アドレスのことを指している。

[*2] 諸般の事情により、役所に書類を出せたのが二番目なので最初ではないという突っ込みをする人もいるが…

■ISP “Internet Service Provider”。インターネットサービスプロバイダ略してプロバイダ。

インターネット接続を売る業者のこと。

■IX “Internet eXchange”。

ISP 同士の接続点。ISP 同士の物理的な接続方法は様々 (イーサネットや ATM など) だが、ISP 間は BGP4 により情報交換を行なう。

■LAN “Local Area Network”。

一つの部屋などにある近くのコンピュータ群から構成されたネットワークのこと。イーサネットでコンピュータとハブをつなぐスター型の構成が代表例。

■LDA MDA と同義語。

■LDAP “Leightweight Directory Access Protocol”。ディレクトリサーバと通信するための規格。

■Mach Mach は Richard Rashid による「部品から OS を組み上げていけるか？」を研究したプロジェクト。マイクロカーネルにもとづく OS の代表例の一つである。

分散オペレーティングシステムではあるが、どちらかといえば、ネットワークよりマルチプロセッサ向きの作りである。

プロジェクトが始まったころ、すでに BSD Unix の人気が絶大だったため、BSD のプログラムが使えるように BSD Unix エミュレータを備えている。

Windows NT 計画のアーキテクト David Catler は Windows NT を設計する際に Mach を手本にした。また、Apple の Mac OS X の先祖は Mach である。その意味で現代の OS の重要な先祖である。

ちなみに Mach は “マーク” と発音する。

■MAC アドレス イーサネットネットワークカード一つ一つについているアドレス。工場で一意に設定されているので、カードを一意に識別できる。

■MDA “Mail Delivery Agent”。ローカル配送を行なう役割のプログラムのこと。たとえば sendmail における mail.local (/usr/libexec/mail.local) のこと。

■modular architecture モジュールからなるモノの作り方。反対語は monolithic 。たとえば Unix は monolithic で、Mach は modular であるし、sendmail は monolithic で、postfix や qmail は modular である。

■monolithic architecture 一つの大きなプログラムの中に何でも入っているモノの作り方。反対語は modular 。たとえば Unix は monolithic で、Mach は modular であるし、sendmail は monolithic で、postfix や qmail は modular である。

■MSA “Mail Submission Agent”。SMTP でメールを受信という意味では MTA と同じだが、特に「クライアントからサーバに対してメールを転送する」際に、メールを受信する役割の何か (プロセス) のことを指す概念。

かつては MTA ともいっていたが、のちにメールサーバという概念を MTA MSA MDA(LDA) という 3 つの概念として奇麗に再構築したさいに提唱された用語。

■MTA “Mail Transfer Agent” SMTP でメールを転送する機能の何か (プロセス)。

特にメールサーバ間でメールを転送する機能のことを指す。

■OSI (1) “Open System Interconnection”。ISO が定めた規格の名称。“OSI 7 階層モデル” が有名。

(2) “Open Source Initiative” の略。

■OSI 7 階層モデル OSI が提唱したネットワークの概念モデル。現実のネットワークが常に 7 階層ということではなく、ネットワークの説明の際に参照されるモデルであることに注意。

■OSPF “Open Shortest Path First”。

ダイナミックルーティングプロトコルの一つ。

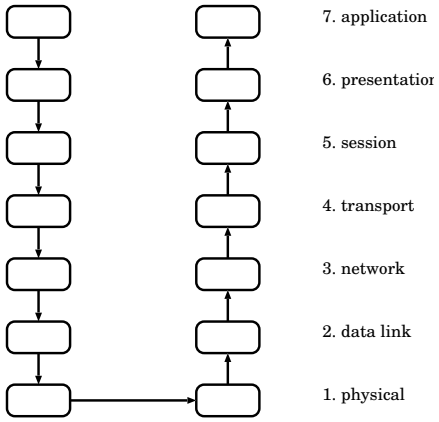

図 A.1 OSI 7 階層モデル

■POP3 “Post Office Protocol” バージョン 3。メールサーバからメールを取り出すプロトコル。

■Postfix tcp_wrapper や satan で有名な Wieste Venema 作の MTA。ちなみに Venema は物理学出身である。数学者の作った qmail と比べてみると大変興味深い。

sendmail 互換でいながら qmail の分散アーキテクチャの良いところを受け継いだ優れたアーキテクチャアを持つ。

http://www.postfix.org/

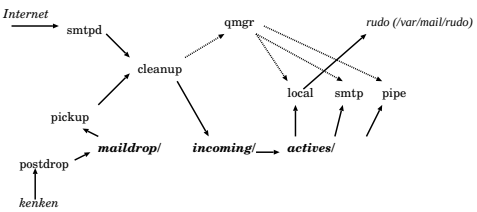

図 A.2 Postfix のアーキテクチャア

■PPP “Point to Point Protocol”。

２つの機器を接続する際に使うプロトコル。

一般には、電話回線でプロバイダにダイアルアップ接続する際に使うプロトコルとして知られているが、２つの機器を接続する構成なら別の物理構成でも利用可能である。たとえば PC と PC を PPP で接続することが出来る。

■PPPoE “PPP over Ethernet”。PPP の機能をイーサネット上で利用するためのプロトコル。RFC 2516 を参照。

現代の日本では ADSL や B フレッツなどの安価な接続サービスが利用している。

■qmail 1990 年代初頭くらいに登場した MTA。その中身は分散アーキテクチャアであり、sendmail の対極といえる。作者は数学者 Daniel. J. Bernstein 。

http://www.qmail.org/

■QoS “Quolity of Service”。一定の通信速度を保証する技術。

■RDBMS “Relational DataBase Management System”。

■RFC “Requests For Comments” という I-ETF が発行する文書。

http://www.ietf.org/rfc/

RFC とは直訳すれば「御意見募集」くらいの意味だが、実際にはインターネットの規格文書である。なお、規格以外にも、エイプリルフールに発行される冗談企画などが混ざっているので真に受けてはいけない文書もある。

インターネットの規格は何回かの RFC 発行を経て正式な標準規格となるプロセスを踏む。

http://www.ietf.org/rfc/std-index.txt

■RFC821 もしくは 821 メール転送規格 SMTP のこと。RFC の番号に基づき 821 と呼ぶ。RFC 821 および 2821 を参照。

■RFC822 もしくは 822 メールの基本フォーマットのこと。たくさんあるが、初代の RFC の番号に基づき通称 822 と呼ぶ。RFC 822 および 2822 を参照。

■RIP “Routing Information Protocol”。

ベクトルディスタンス型のダイナミックルーティングプロトコル。

■sendmail Eric Allman 作のインターネットの初期からある代表的な MTA (正確には MTA + MSA + LDA)。現在では商用版とフリーソフト

ウエア版がある。

http://www.sendmail.org/

■SMTP “Simple Mail Transfer Protocol”。メール転送プロトコル。RFC821 および RFC2821 を参照。

■SPAM アメリカの豚肉の缶詰。コーンビーフの豚版のようなものだと思えば良い。

SPAM をパンにのせたり、(沖縄では) 味噌汁に入れたりする。塩辛いので SPAM チャーハンなどしてもおいしい。

いづれにせよ、高級食材とはいい難いところがポイントだ。

イギリスの有名なコメディ番組 Monty Python's Flying Circus の第 25 話「SPAM」が SPAM メールの語源と言われている。

■SPAM メール 勝手に送られてくる迷惑メール (内容は宣伝や詐欺など) の通称。UCE や UBE といった言い方をすることが推奨された時期もあるが、結局、通称の SPAM が最も広く使われているようである。

■SSL “Secure Socket Layer”。TCP の上に位置する安全を確保する層。ネットスケープ社が提唱した。

のちに IETF は SSL 3.1 を元に TLS という規格を策定した。

■TCP “Transmission Control Protocol”。「信頼性」のある「安全な論理回線」を提供する「コネクション指向」のプロトコル。

■TCP/IP インターネットを代表する基幹プロトコルである TCP と IP のことを指す。転じてインターネット技術の総称として TCP/IP という言い回しをすることが多い。

■TLS SSL 3.1 を元に IETF で正式に定めた規格。TCP の上に位置する安全を確保する層。

■TTL “Time To Live”。パケットの生存時間の上限を決めるパラメータ。

IPv4 規格では「秒」の意味でもよいとなっていたので Time To Live なのだが、そういう実装をしているルータが一つもないため、IPv6 規格では Hop Limit という名前に変わった。

■UDP “User Datagram Protocol”。

TCP と異なり信頼性のない通信方式だが、そのぶん動作が軽い。

なおレイヤー 3 には IP を使うが、TCP/IP とは違い UDP/IP という言い方はしない。

■Unix 1969 年に AT & T ベル研究所で開発されたオペレーティングシステム (OS)。

法律上、AT & T は OS を売ることができなかったので、低価格のライセンス料で、サポートなしでかつソースコードごと提供された。そのため、世界各地の Unix コミュニティにより拡張やデバッグが進められ、それがまた次のバージョンに反映されることとなった。

AT & T 版から spin off したバージョンである BSD は特に重要である。インターネットの基礎が作られる過程で BSD は重要な意義を持った。

現在では、商用の Unix およびフリーソフトウエアの NetBSD、FreeBSD、OpenBSD などが使われている。

現在 Unix という登録商標自体は紆余曲折を経て The Open Group (旧 X/Open) が所有しており、正式なライセンス供与をしている。ここの正式なライセンスを受けていないオペレーティングシステムは Unix Clone とでも言うべきである。

■URI “Uniform Resource Identifier”。

“識別可能なものすべて” を xxx:yyy 型で表現することが目的である。よって URL は URI の部分集合である。

たとえば、http://www.google.co.jp/ や ftp://ftp.fml.org/ といった URL 以外に「TEL:電話番号」や「ISBN:本の識別番号」といった表現がありうる。

■URL “Uniform Resource Locator”。

インターネット上の場所を指定する。「プロトコル://ドメイン名/ファイル名」などという記述スタイル。たとえば

http://www.chitose.ac.jp/

■VLAN “Virtual LAN”。

論理セグメント。物理的な形状と無関係に作られたセグメントのこと。

■VoIP “Voice Over IP”。音声をデジタルデータ化し IP パケットとして流す。これによりインターネットと電話が統合される (はず)。

■VPN “Virtual Private Network”。

物理的に離れた場所にあるネットワークを仮想的に一つにつなぐこと。たとえば東京オフィスと札幌オフィスを VPN でつなぎ社内ネットワークを仮想的に作るといった具合に使われる。

■W3C “World Wide Web Consortium”。

WWW で利用される技術の標準化をすすめる団体。

■WAN “Wide Area Network”。

離れた場所にある LAN をつなぐような大きな単位のネットワークのこと。

■Web WWW を参照。

■WWW “World Wide Web”。

HTTP で WWW サーバから情報を取りだし、WWW ブラウザで、そのデータを整形して表示する。URI により、HTTP だけでなく様々なサービスが統合されている。

インターネットが一般に広く知られるようになった原動力の一つ。

■うえ (上) 「よりユーザ側に近いレイヤー」を上と呼ぶ。業界の言い回し。例: レイヤー 7 はレイヤー 4 より上側である。

■こうかいかぎあんごう (公開鍵暗号) 公開鍵暗号は公開鍵と秘密鍵の 2 つで一組からなる。公開鍵は一般に公開され、秘密鍵は持ち主だけが知っている鍵。

公開鍵で暗号化されたデータはペアの秘密鍵で (のみ) 復号することができる。逆に秘密鍵で暗号化されたデータはペアの公開鍵で (のみ) 復号することができる。

■した (下) 「より地面側に近いレイヤー」を下と呼ぶ。業界の言い回し。例: レイヤー 2 はレイヤー 3 より下側。

■じょうたいせんい (状態遷移) 状態間を移り変わること。“transition”。

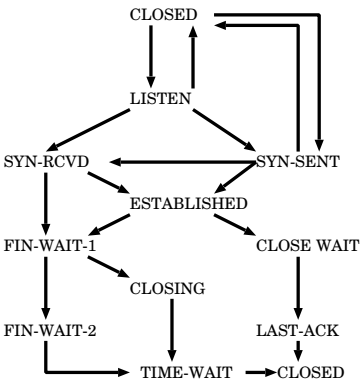

図 A.3 TCP の状態遷移

■たんまつ (端末) コンピュータのこと。

■つうしんきやく (通信規約) “プロトコル” を参照。

■でんしめーる (電子メール) “メール” を参照。

インターネット版の郵便。実体は、特定のフォーマットのテキストファイルである。

■でんしめーるあどれす (電子メールアドレス) “メールアドレス” 参照。

■なまえかいけつ (名前解決) ドメイン名から該当する IP アドレスを見つけ出すこと、および逆の動作。

■ひみつかぎあんごう (秘密鍵暗号) 暗号化と復号化を同じ鍵で行なう方式。

■アプライアンス製品 特定の機能に特化した製品のこと。

ASIC を搭載しているものも多い。

■イーサネット CSMA/CD 方式を使う LAN の規格。IEEE 802.3 が標準規格の名称。

初期は 10 base-5 で速度が 10 Mbps だが、徐々に拡張され、いまや 1000 base (1000 Mbps つまり 1 Gbps) が一般家庭でも利用可能である。

■**イエローケーブル**　10 base 5 の太いケーブル。黄色い外層のケーブルが多かったのでイエローケーブルと呼ばれる。

■**インターネットイニシアティブ**　“IIJ” の項目を参照。

■**インターフェイス**　二つの間で情報のやりとりの仲介をするもの。「界面」と訳すこともあるが、普通は「インターフェイス」で通じる。たとえば「ユーザインターフェイス」「API」「このブラウザのユーザインターフェイスは良い」のように使う。

■**イントラネット**　企業内の LAN のこと。

■**ウエブ**　WWW を参照。

■**クライアント (client)**　サービスの提供をうける側のプログラムのこと。もしくはそのプログラムを実行しているコンピュータのこと。たとえば、メールソフトやメールリーダ、ウエブブラウザ、それらを実行しているコンピュータのこと。

■**クラス**　(1) IPv4 の CIDR 以前の時代にあった大まかな分け方。クラス A から E までである。

(2) オブジェクト指向言語の用語。

■**グローバルアドレス**　組織に一意に割り当てられた (インターネット上の住所を表す) IP アドレスのこと。

■**コマンド**　命令。“command”。

■**サーバ**　“server”。サービスを提供するプログラムのこと。もしくはサービスを提供するプログラムを実行しているコンピュータのこと。たとえば、メールサーバ、DNS サーバ、ウエブサーバ。

■**サブネット**　一つのネットワークを複数の小さなネットワークに分割すること。もしくは分割されたネットワークのこと。

たとえば「192.168.1.0/24 は 192.168.1.0/25 と 192.168.1.128/25 の 2 つのサブネットに分けることが出来る」などといったぐあい。

■**スイッチ**　ハブの形態をもつネットワーク機器。CSMA/CD の無駄な振舞いの部分について賢い振舞いをするハブとでもいうべきもの。通常 ASIC ベース。

■**ステータス**　状態。”status”。

■**ステータスコード**　状態を示す数字。

■**ステート**　状態。

■**ダークファイバ**　敷設されているが使われていない光ファイバのこと。

正式なカタログ等には載っていないかも知れないが、実際には NTT などのキャリア (第一種通信事業者) から安価に購入することが出来る。

ただ、それはサポートがないために安価なだけであるため、安いかわりに自己責任で利用する必要があるところがポイントである。

■**ダイナミックルーティング**　状況に応じて動的に変化する経路制御 (ルーティング)。

■**ツイストペアケーブル**　銅線をねじり (twist) 合わせたケーブル。10 base-T を筆頭とするイーサネットケーブルで広く使われている。

■**テキスト**　文字だけで表現されたもの。

■**テキストファイル**　文字だけで表現されたファイル。

■**データセンター**　ハウジングと同義語の場合もあるが、ハウジング＋運用まで含まれた概念がデータセンター。

プロバイダによってデータンセンターが単なるハウジングのこともあれば、きちんと運用されたサービスであることもある。

■**データベース**　系統的なデータの固まり。

■**データベース管理システム**　“DBMS” 参照。

■**デーモン**　(裏で) 走り続けているプロセス。さまざまなサービスを提供し、OS を影で支えている。

■**ディレクトリ**　ファイルを格納する入れもの。Unix では “ディレクトリ” だが、Windows や Mac OS では “フォルダ” と呼ぶ。

■**ディレクトリサービス**　リソースや社員情報を参照するしくみのこと。おおざっぱには電話帳の拡張された概念と思えばよいだろう。

■デファクトスタンダード　“defact standard”。“事実上の業界標準” という意味。

インターネットは国家主導で成長してきたわけではないので政治的に決められた標準規格よりデファクトスタンダードのほうが重要な場合も多い。この点、とても『インターネットらしい』ものだといってよいだろう。

■デフォルト　(1) “あらかじめ決められている” 値などのこと。

(2) デフォルト値のこと。設定可能な変数に、あらかじめ決められている値。

(3) デフォルトゲートウエイのこと。指定されたルーティングが特にない場合に使う行き先。

インターネット用語では良く使われるが、世間では通じない (インターネットを知らない人には、金融用語のデフォルトと間違われるだろう) ので注意。

■デフォルトルート　指定されたルーティングが特にない場合に使う行き先。デフォルトゲートウエイとも言う。

■トップレベルドメイン　“jp” や “com” といったもっとも大きな単位のドメインのこと。たいていは国を意味する jp や uk などであるが、近年 tv などサービスを意味する特別なドメインが創設されてきている。

■トレイラ　パケットの終りにつく制御情報部分。トレイラがないプロトコルもある。なお、フッタ (footer) でもいいような気がするが、そういう言い回しの例はないかもしれない。

■ドメイン名　例: chitose.ac.jp

■ネームサーバ　DNS サーバのこと。

■ネットマスク　サブネットの大きさを指定する形式。たとえば 255.255.255.0 。

■ハウジング　ネットワーク機材を置くために場所を貸すサービス。もしくは、そのサービスを購入してサーバなどを置くこと。

■ハブ　スター型ネットワークでイーサネットケーブルを集線する装置。

LAN ケーブルの集線装置。たとえば 10 base-T ケーブルをハブに指しスター状のネットワークを作る。

■パケット　小さなデータの固まりのこと。

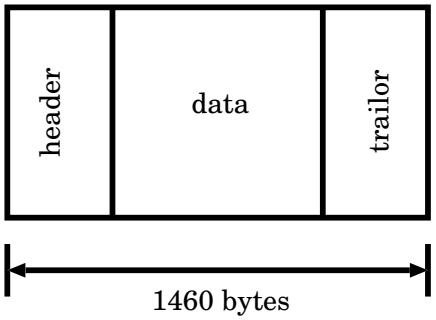

図 A.4　パケットの構造: ヘッダ、データ、フッタ

■フォルダ　“ディレクトリ” を参照。

■フッタ　ヘッダの反対。最後の部分。

■プライベートアドレス　各組織が自組織内で自由に使って良い IPv4 アドレスの範囲のこと。つまり世界中に同じ IP アドレスが重複している。よって住所としての IP アドレスになりえないので、組織外部に直接晒してはいけない。

通常、ファイアウォールやルータなどが IP アドレスの変換 (NAT) を行なうことで、組織外部に晒すことはしていない。

■プロセス　コンピュータ上での処理の実体。プログラムが動いている状態 (厳密な定義は難しい)。例: エクスプローラ、Windows Media Player

■プロトコル　通信方法の取り決めのこと。

■プロバイダ　ISP を参照。

■プロバイダアドレッシング　CIDR アドレッシングと同義。CIDR を前提にした IP アドレスの割り当て方法。

■ヘッダ　パケットの先頭につく制御情報部分。制御情報の例は、住所 (IP アドレス)、ポート番号など。

■ポート番号　サービスを特定する数字。通信を特定するためには、プロトコルと送信元と受信先双方の「IP アドレスとポート番号」の組が必要。例: (10.0.0.1 1103 192.168.0.1 25)

■マルチキャスト　“multicast”。一対多の通信。

■メール　ネットワーク版の手紙。“電子メール”のこと。

■メールのフォーマット　メールのフォーマットを定義したもの (図参照)。RFC822 および RFC2822 他多数

■メールアドレス　“user@domain” 形の文字列。電子メールを送受信するのに必要でインターネット上での「住所＋名前」に相当するものといえる。

■メールサーバ　MTA + MSA + MDA(LDA) 例: sendmail, qmail, postfix, exim, zmailer, ... (図参照)。

■ユニキャスト　“unicast”。一対一の通信のこと。

　なお、IP アドレスといえば、普通ユニキャストで使う IP アドレスをさす。ただし「ループバックアドレス」と「プライベートアドレス」はユニキャストの中で特別な意味のある IP アドレス。

■ルータ　セグメント境界に置いてパケットを転送する装置。

■ルート　“root” と “route” がある。前者は木構造の根 (root) にあたる部分のこと、後者は経路の意味。

■ループバックアドレス　自分自身へ通信する時に使う特別な IP アドレス。IPv4 では 127.0.0.1、IPv6 では ::1 を使う。

■レイヤー　“layer” もしくは「層」。OSI 7 階層モデルの関連で使う業界用語。

■レイヤー 8、9、10　半分は洒落。技術を越えた層について言及する際に使う。例: 政治層

■出島　“DMZ” を参照。

■通信路　“circuit”。物理的な通信路と論理的な通信路がある。

付録 B

参考書

B.1　安価

- ネットワーク資格本の類。
  たとえば「ネットワークスペシャリスト重点教本」のような本。BOOK OFF の 100 円コーナーあたりで叩き売られている。授業の内容は基本だけなので、古いのでもヨシ。
- 高校の「情報C」の教科書
  もし、高校の「情報C」の教科書を持っている人は、それを引っ張りだしてきてください。本講義の用語集の補足としては最適です。
  ただ、この授業の本命は「動作の説明」なので、「情報C」は、そういったレベルでは使えません。

B.2　普通

- 「ネットワークの教科書」増補改訂版— 4 つのステップで完全理解!あなたの知識をさらに強化する!! (IDG ムックシリーズ)
  ちょっと現場 (しかも SI 業者) 向けに内容が偏っている気がするけど、レベルが、ちょうどいいかな？と思っています。
  - ムック: 225 ページ
  - 出版社: アイ・ディ・ジー・ジャパン (2007/3/9)
  - ISBN-10: 4872802675
  - ISBN-13: 978-4872802672
  - 発売日： 2007/03/09。

B.3　専門家向け

- TCP/IP Illustrated.
- Unix Network Programing.
- Internet Routing Architecture (CISCO).

B.4　機材: ファイアウォール

B.4.1　ファイアウォールの意義

“ファイアウォール”(firewall) とは、組織とインターネットとの境界に置き、セキュリティポリシーを押しつけるための装置です。

身近な例をあげれば、大学からインターネットへの出口においています。

セキュリティポリシーとは、たとえば「学内からインターネットへの HTTP は許可するが、学外から学内への HTTP は許可しない」といった通信に対するルールのことです。もっと細かなルール、たとえば「学内からインターネットへの HTTP は許可するが、2ch.net は不許可」といった細かなポリシーもあり得ます。

ファイアウォールには、インターネット側から学内への (幼稚な手法による) 侵入を防ぐ効果もあります。

ネットワーク構成の詳細は第 7 章で説明します。

ファイアウォールの定義

もう少し硬い定義を書けば、次のようになるでしょう。

ファイアウォールとは、組織とインターネットとの境界に置き、セキュリティポリシーを制御する装置。

ファイアウォールに期待される機能には次のようなものがあります。

- 通信の流れを監視する。
- 外部からの不正侵入を防ぐ。
- 内部からの通信へのアクセス制御。
- 外部からの通信へのアクセス制御。

B.4.2 ファイアウォール装置

実にさまざまな製品が販売されています。

Unix でフィルタをかける (自作) ものから高価な製品までいろいろあります。値段は自作で 0 円から数千万円の製品まで、機能や速度などにより値段は様々です。

すべてを理解している人ならば、ファイアウォールを自作してもかまいません。ただ、毎日、山のように発生するセキュリティ情報に目をとおして、それらを理解し、自作のソースコードを修正していくと言った作業が必要です。つまり自作は事実上不可能といえます。

B.4.3 ファイアウォールサービス

インターネットサービスプロバイダなどでは、その道の「セキュリティのプロ」が 24 時間 365 日体制でファイアウォールの面倒を見てくれるサービスを販売しています。

毎日、山のように発生するセキュリティ情報に目をとおし、それらを理解し、必要なら設定変更、場合によってはファームウエアの更新を行ないます。

設定やファームウエア変更前には、さまざまな検証も必要です。また、ファームウエアの更新には多額の費用が発生する場合もありえます。

これらのことを考えれば、セキュリティサービスを購入することは決して高くなく、むしろ非常に安いことが分かります。

ファイアウォールの種類

ファイアウォールサービスのラインナップから選ぶことが基本です。そのため、技術的な相違など一般ユーザが知っている必要は、ほとんどありません。

しかしながら、授業なので:)、一応、説明します。

大きく分けて、ファイアウォールの種類には次のようなものがあります。

- パケットフィルタ
- 単なる NAT 箱
- アプリケーションレベルゲートウエイ
- ステートフルインスペクション

どのタイプの製品でも、通信に対してアクセス制限を押しつけることが可能であるという点で、最低限の機能は満たしますが、付加価値の部分は千差万別です。

B.5 ファイアウォールのデザイン

ファイアウォールの歴史を研究すると、おおまかには次のようなことがいえます。

1. ソースコードは公開から非公開へ。
2. ソフトウエアからハードウエア (ASIC) へ。
3. より安く、より家電製品のように。

このデザインの流れはファイアウォールにかぎりません。どんな機器でも同様の流れを見ることが可能です。

ただ、ファイアウォールの場合、例外的に「ソフトウエア → ハードウエア (ASIC) → ソフトウエア」という逆転現象がおきています。これは i386 系 PC の歴史上異常なまでの革新速度によるものでしょう。

以下、歴史的に重要な製品について概観してみます。どのような設計思想に基づいているかに着目してください。

B.5.1 Gauntlet

TIS (Trusted Information System) 社製 Gauntlet (ガーントレット) は筆者の一番好きなファイアウォールです。

大学の初代ファイアウォールでもあります。

Gauntlet は「クリスタルボックス」という思想でも有名です。

「クリスタルボックス」とは「ソースコードを公開し、みんなに分析してもらうことで危険が少なくなる」という「フリーソフトウエア / オープンソース」の思想の流れの延長にあるもので、Unix 文化の自然な帰結です。

ただ、すべてを公開していたため、ノウハウが流出してしまい、結果として、この製品の理念をつがない「まがいもの」を生み出す原動力となった、とも言えそうです。

Gauntlet はアプリケーションレベルゲートウエイ (application level gateway) です。

アプリケーションレベルゲートウエイでは、アプリケーションごとに中継するプログラム (proxy) が個別に動作しています。

各プログラムはアプリケーションプロトコルを理解します。そのため、HTTP や SMTP のプロトコルの細部にまで踏み込んだ非常に細やかな制御が可能です。その反面、動作は遅くなります。

そのため、インターネットの高速化、大容量化が進んだ 1990 年代後半には処理速度が追い付かなくなり、ファイアウォールの非主流派になっていきました。

技術者も抜け、Unix 版もなくなり、どうでもよくなりましたが、いまでも TIS はあるし、Windows 版の販売もしているはずです。ま、どうでもいいですけど。

コラム: 動作原理
アプリケーションレベルゲートウエイの動作原理は
http://www.fml.org/software/nfgw/ja/
を見てください。
この説明はバージョン 1.0 用ですが 2.0 でも基本的に同じです。
2.0 のソースコードは…なぜか非公開のままだな。
とりあえず、うちでは、ファイアウォールつくってもらうところから研修だよん。

B.5.2 Checkpoint Firewall-1

イスラエル製のファイアウォールのソフトウエアです。「ファイアウォールのソフトウエア」なので、OS が別途必要です。

名前のとおりで、すべてソフトウエアで動作しています。

当初は SUN の Solaris や Windows NT の上で動かしていましたが、いまや携帯電話で有名な NOKIA が「IBM PC の FreeBSD 版 Firewall-1 セット」(PC とのセットもの) を売り出して以降、ファイアウォール市場で圧倒的な優位性をもつようになりました。

現在、ファイアウォールといえば、Firewall-1 か Netscreen (後述) のどちらかしか選択肢がないといって良いくらいです。

「ステートフルインスペクション」(Stateful Inspection) という概念を提唱したことでも知られています。

これは、アプリケーションレベルの中継まではしないが、それなりに通信の状態を監視することで、それなりに高速な動作を行なえるという方法です。

アプリケーションレベルゲートウエイの動作が遅いのを解決するというのが建前ですが、実際の現場では、細かい制御を必要とする通信に対

して Firewall-1 をアプリケーションレベルゲートウエイとして設定します。たとえば SMTP は専用中継プログラムを使い、それ以外は Stateful Inspection といった設定です。

まぁ、ようするに Gauntlet と Firewall-1 のいいところをハイブリッドにしているのが現実の運用方式です。

B.5.3 Watchguard Firebox

組み込み機器化したファイアウォールの第一世代です。

実体はソフトウエア (Linux + NetFilter + なんか) ですが､ハードディスクなどを持たないハードウエアです。そのため唐突に電源 off/on をしても壊れません (反対に Gauntlet や Firewall-1 は普通の PC なのでハードディスクが壊れるかもしれません)。

これは家電製品の一種で、いわゆる組み込み機器です。

Linux ベースのため、GNU ライセンスなのにソースコードを公開しないが、いいのか？などと疑問がいろいろあるのですが、誰もつっこみません。

いずれにせよ、現代では遅くて話にならないので、前時代の産物です。

B.5.4 Netscreen

組み込み機器化したファイアウォールの第二世代です。

ここからはハードウエア制御、つまり専用 LSI (ASIC) の時代になります。

Netscreen 社は、完全な組み込み機器化をし、ソフトウエアからハードウエアへ、つまり ASIC 化を推し進めたことが歴史的に重要です。

さらに Netscreen を抜けた人たちが Fortigate を作りました。Fortigate はウィルスチェックも ASIC 化した (と自称しています)。

Netscreen は Juniper に買収され、2007 年の新製品からは PC 上のソフトウエアでの動作、つまり「力業 (ちからわざ)」へ戻りました。

Intel CPU があまりにも高速で安価なので、ASIC を開発するより、力業に戻った方がコストパフォーマンスがよいということなのでしょう。

B.5.5 Sonicwall

路線的には Netscreen と同様だが、さらに安い製品。

安い分、性能も…

2004/06 ハードウエアが PC のものを出してきました。この製品以降 ASIC から PC で力業路線が始まります。

歴史を語る上では、この力業路線回帰の引金を引いたところが重要かと思いますが、それ以外は、どうでもいい会社です。

B.5.6 まとめ:技術の歴史総論

ソフトウエア（プロトタイプ）→ハードウエア化 (ASIC) →→ PC が安くなり過ぎて、ソフトウエア型 (力業) へ戻る。