計算機科学基礎'13 ＃4 – ネットワークの原理と構造

久野　靖*

2013.5.8

1　イントロ: ネットワークの知識とは…

今日のコンピュータシステムにおいて、ネットワークは欠かせない機能です。そのことは多くの人が経験ずみであり、今更取り上げるまでもないでしょう。しかし、その裏側にどのような技術的背景があるのかについては、あまり知られていません。今回はこれらの側面にも焦点を当てながら、コンピュータネットワークの原理や仕組みについて体系的に整理してみます。

なぜ「ネットワークの原理」を学ぶ必要があるのでしょうか? 別に原理なんか知らなくても使えればよい? 技術的なことは技術屋に相談すれば済む? しかし、技術屋と相談するには、ある程度「技術屋のことば」が分かっていないととんでもない勘違いが起きる恐れがあります。また別の例ですが、すぐそこに相談できる人がいない時に次のような疑問が湧いたらどうしますか?

> 地球の裏側 (例:ブラジル) の事業所の××管理を日本のシステムで行ってしまうことで、現地でのシステム運用をカットしてはどうかという提案があった (図 1)。となると、ブラジルとの間でデータ往復させることになるが、その所要時間はどれくらいだろう?

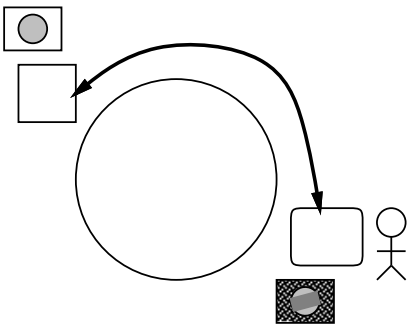

図 1: ブラジルの画面の処理を日本で?

実はこの程度のことは簡単に自分でも調べられます。具体的にはこうすればよいのです。

- Google などで「ぜったいにブラジル現地にありそうな施設」の Web ページを検索して探す。ここでは地元のホテルぽいのを探し、「Royalty Hotel Barra」を見つけました。[1]
- そのサーバ名「`royaltyhotel.com.br` を指定して `ping` コマンドを実行し、パケット往復時間を調べる。[2]

*経営システム科学専攻

[1]昨年までブラジルの博物館を計っていたのですが、どうも米国にサーバを立てたらしく急に応答時間が短くなったので、探しなおして現地ホテルにしました。

[2]GSSM の環境では、以下の実験は外界と通信可能なマシン `utogw` に `rlogin` して実行する必要があり、またコマンドは`/sbin/ping` です。皆様の個人マシンであれば `ping` だけでよいことが多いでしょう。

```
% ping royaltyhotel.com.br
PING royaltyhotel.com.br (200.219.245.77): 56 data bytes
64 bytes from 200.219.245.77: icmp_seq=0 ttl=109 time=301.177 ms
64 bytes from 200.219.245.77: icmp_seq=1 ttl=109 time=282.924 ms
64 bytes from 200.219.245.77: icmp_seq=2 ttl=109 time=282.471 ms
64 bytes from 200.219.245.77: icmp_seq=3 ttl=109 time=280.409 ms
^C  ←止めるには Ctrl-C
%
```

- 往復時間が 280msec 程度だから、片道は 140msec 程度と分かる。

種明かしをすれば簡単ですよね。そして、片道 140msec くらいなら、たとえば現地で管理画面のボタンを押すと、日本にそれが伝わって処理を行い、結果を現地に返送する、といった処理をしてもさほど問題はないことが分かるわけです。

もちろん、皆様がこういうコマンドを暗記している必要は全くありません。ただ、「素朴な疑問」が出たときに、「どのみちこれはネットワークを通っているのだから、こうなっているはずだ、だからこうすれば計れるんじゃないかな」というカンが働く程度までは知っておいて損はないと思いませんか。今回と次回を合わせて、その程度までの学習を目標に進めて行きましょう。

演習 4-1 自分でも Royalty Hotel のサイトまでのパケット往復時間と経路を確認してみなさい。確認できたら、以下の課題から 1 つ以上 (できれば全部) やってみなさい。

a. どこか「ネット的に遠い」と思う地域を選び、そこにある施設 (複数) までのパケット往復時間を計測して「遠さ」を確認してみなさい。できれば、そこまで行くのにどこを通っているのか予想してみて、その地域までの時間も計測して確認みるとなおよいでしょう。

b. 世界の複数地域までのパケットの到達時間を調べ、およその傾向をまとめなさい。できればそれらを整理して世界のネットワークのつながり方の予測できるとなおよいでしょう。[3]

c. 本来なら遠くにあると思われる施設のサーバへの往復時間が異常に速いものの例を調べて挙げてみなさい。できれば、どのようなサイトついてそのようなことがなされているかの傾向を分析できるとなおよいでしょう。

「◎」の条件: 小問 2 問以上について解答されており、いずれも「できれば」の部分までやってあって、なおかつ適切な (と担当が判断する) 整理・分析がなされていること。

2 ネットワークの基礎概念

2.1 ネットワークとその目的

まずそもそも、コンピュータネットワークとは何でしょうか? 物理的には、複数のコンピュータシステムが、互いに通信できるように接続されたものがネットワークである、と言えます。しかし例によって、コンピュータが「何をするか」はソフトウェアによってすべて変わってきますから、そのようなハードウェア (物理的な接続) をどのように使うかが重要です。ネットワークを構成する目的としては次のようなものが挙げられます。

a. 資源の共有 — 例えばあるコンピュータに入っているデータを、そのコンピュータで処理を行なう際だけでなく、別のコンピュータで処理を行なう際にも利用できるようにする、あるコンピュータの CPU 能力が不足したらデータの一部を別のコンピュータで処理する、など。

b. 信頼性 — 1 台のコンピュータであればそれが止まってしまえば処理は停止してしまうが、複数台のコンピュータをネットワークで結合したものであれば、1 台が壊れても残りで処理を進めて行くようにできる。

[3] 後から出て来る `traceroute` を使ってその正しさを確認することができます。

c. 経済性 — 大きなコンピュータ 1 台で何もかもやらせるより、複数のマシンをネットワーク結合したシステムの方がコスト的に安い。

d. 段階的成長 — 1 台のコンピュータで能力が不足したら、より大きいマシンにリプレースするしかないが、ネットワークシステムなら何台かマシンを追加する形で成長して行ける。

e. 通信媒体 — 距離的に離れたシステムどうしを接続することにより、新しいタイプの応用が可能になる。

もちろん、どの目的を主とするかによって、ネットワークの形態は大幅に異なります。たとえば、信頼性や経済性のためにネットワークシステムを構成するのなら、その各コンピュータ間の距離は比較的近く、それらの間は高速な通信方式で結ばれることになるでしょう (**LAN** — Local Area Network、局所ネットワーク)。一方、離れたところにあるデータの共有や個人間の通信が目的なら、そのネットワークは長い距離を結ぶものになるはずです (**WAN** — Wide Area Network、広域ネットワーク)。

2.2 回線交換とパケット交換; 電話網とネット

古くからあるネットワークの代表例は旧来の固定電話ネットワークです。電話はもともと、2 つの機器の間を直接繋ぐと遠隔地で通信ができるというところから始まりました。そして、個人の家から電話が引けるようになった後も、電話局で交換手が配線を繋ぎ、最終的に自分と相手との間に線がつながると通話できる、という点は同じでした。その後技術の進歩により、番号をダイアルすると自動で相手につながる等の変化はありますが、「最初に線を確保し、その後通信し、終わったら切断する」という点はずっと変わっていません。これを**回線交換** (line switching) と呼びます。[4]

回線交換の弱点は、最初に線を確保する手間が掛かり、線が確保できないと通信できない (「話し中」) ことです。逆に利点は、一度つながった線は自分たち専用に確保され、常に 100%使えるということです (話している二人が沈黙している間も線は確保されたままなので、無駄があるとも言えます)。

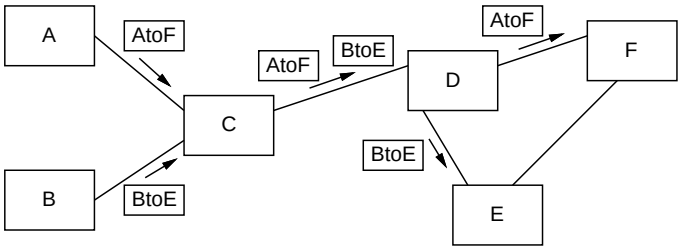

図 2: パケット交換の原理

一方、コンピュータネットワークの通信は (正確に言えば今日主流であるインターネットの方式は) これとは違っています。データは上限サイズが決まった**パケット** (packet) という単位で、ネットワークに常時送り出せます。各パケットには宛先が書いてあり、中継点ごとに宛先の方向に中継することで最終目的地まで到達します。これを**パケット交換** (packet switching) と呼びます。パケットはしばしば「葉書」ないし「小包み」に例えられます (図 2)。

パケット交換の利点/弱点は先の裏返しで、接続の手間が無く、経路の途中では多数の通信のパケットが「相乗り」するので資源が有効活用できます。ただし、通信量がまかない切れない量になると、既に通信している相手どうしでも通信しにくくなります (パケット到達時間が長くなったり、パケットが失われたりする)。また、1 パケットに入らない大きなデータを送る場合には、複数のパケットに分けて送り相手先で組み立てるなどの手間を必要とします。

電話網とインターネットについて、もう少し比較をしてみます。電話網では、端末は単機能の (バカな — dumb な) 電話機であり、音声信号の送受とダイアルパルスやフック信号を送出くらいしかで

[4]携帯電話も固定電話も今日では最下層はパケット交換に変わっています。

きません。[5]ですから、ダイアルに応じて相手に接続したり、雑音の少ない通話を提供するなど、すべての賢い機能やサービスは電話網側にあり、それを NTT などが高いお金を取って維持していたわけです。これに対しインターネットでは、端末はコンピュータですから非常に賢く何でもできます。そこではネットワーク側に必要とされるサービスは単に「相手までパケットをできる範囲で届ける」ということだけです。これを**ベストエフォート** (best effort) といい、聞こえはいいですが、要は「質の保証はしません」ということであり、その分だけネットは廉価かつ大容量にできるわけです。今日ではこのモデルが非常に成功したため、電話網も内部ではインターネットと同じ技術で音声を運ぶようになりました。そのため、電話会社の付加価値はどんどん減少しており、どこの国の電話会社も経営が大変になっています (昔の国際電話がいかに高かったか知っている人に尋ねてみてください)。

なお、同じ電話といってもスマートフォンなど (そしてタブレットも) は実質「コンピュータ」であり、通信方式もインターネットと同じです。このことは、同じ端末を 3G でも WiFi(無線 LAN) でもまったく変わりなく使えることから分かると思います。

演習 4-2 インターネットは「バカなネットワーク」であり、混雑や障害があると送ったパケットが捨てられたり遅延して到着したりします。この上で正しく通信するには「賢い端末」が、例えば失われたパケットを再送するなどの方法で「正しい内容が送られる」ように手当てする必要があります。ここでは 3〜4 人が 1 グループになり、うち 2 人が送信側と受信側になって背中合わせに座り、英数字 10 文字程度のメッセージを送信側から受信側に送るというゲームをして頂きます。残りの人はネットワーク役になってパケット (メモ紙) を送信側から受信側、ないしその逆に運ぶだけに徹しますが、時々 (2 回に 1 回くらい?)、a〜c の状況でいう「普通でないこと」を起こすようにします。1 パケットには「4 文字」までしか書いてはいけませんし、紙の手渡し以外の動作 (合図等) も禁止です。ネットワークの状況が下の a〜c のいずれか 1 つ以上 (できれば全部) の場合に、正しく送るための (送信側と受信側の間での) 約束を工夫しなさい。[6]

a. 送ったパケットは必ず 10 秒以内に到着するが、時々「×」印だけのパケットに入れ替わっていることがある。
b. 送ったパケットは到着することもしないこともある。到着する場合は 10 秒以内に到着する。
c. 送ったパケットは到着することもしないこともあり、到着する場合も遅延することがある (後から送ったパケットに追い越されたりすることもある)。

演習はグループでやりますが (役割は適宜交替して経験)、レポートは各自で書くこと。

「◎」の条件: (1)3 ケースのうち 2 つ以上について、(2) どのような約束を工夫したかが正確に書かれており、(3) その結果について適切な (と担当が考える) 考察・分析が書かれていること。

2.3 簡単なネットワークプログラム

お話ばかりではつまらないので、ここで簡単なネットワークプログラムを動かしてみましょう。このプログラムはあるホスト[7]から別のホストへパケットを使って文字を送るというもので、送る側と受け取る側に分かれています。まず受け取る側から見ましょう。[8]

```
/* recv.c -- packet receiving example. */
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
```

[5]もちろん、後の方ではもっと高機能な電話機がありましたが、単機能の電話機もずっと使われ続けており、それを前提としたシステムになっていたということです。

[6]ヒント: 「正しく受信した」ことを示す確認パケット (ACK) を返送することはどの場合も必要になります。ACK が×になったり喪失・遅延する可能性もあることに注意すること。

[7]ネットワークの世界では通信の出発点/到達点となるようなコンピュータのことを**ホスト** (host) と呼びます。

[8]今回の send/recv はおまじないのようなプログラムなので一応説明していますが全部理解することは求められません。なお、自宅でも試せるようにコンパイル済みのものを用意してあります。

```
main(int argc, char *argv[]) {
  int fd, len, fromlen; char buf[100];
  static struct sockaddr_in adr; struct sockaddr *p = (struct sockaddr*)&adr;
  adr.sin_family = AF_INET;
  adr.sin_addr.s_addr = INADDR_ANY;
  adr.sin_port = htons(atoi(argv[1]));
  if((fd = socket(PF_INET, SOCK_DGRAM, IPPROTO_UDP)) < 0 ||
      bind(fd, p, sizeof(adr)) < 0) { perror("socket"); exit(1); }
  while(1) {
    len = recvfrom(fd, buf, 100, 0, 0, &fromlen);
    if(len < 0) { perror("recv"); exit(1); }
    write(1, buf, len);
  }
}
```

まず、#include ... とあるのはネットワーク関係のデータ構造定義を取り込むための指示です。次に main の冒頭で必要な変数を用意しています。fd はソケット (ポート) のためのファイルディスクリプタ番号を入れる変数、len と fromlen はデータの長さを入れる変数、buf はパケットデータを格納するバッファです。次にネットワークアドレス型のレコード変数 adr を割り当て、そのホスト部は「任意」、ポート番号はコマンド引数で指定した文字列を整数に変換して、なおかつネットワークバイト順に変換したものを入れます。

ここまでで用意ができたので、まずソケットを作成してそのディスクリプタ番号を fd に入れ、次に bind システムコールにより先に作ったソケットのアドレスを上記の値にします (いずれかに失敗したらメッセージを出して終わり)。ここまでの所はこれ以上詳しく説明しても頭が痛いだけなので、「おまじない」だと思ってそのまま使ってください。要は OS に頼んでポート (ソケット) を準備している、ということです。

ポートが準備できたらあとは recvfrom というシステムコールを呼び、パケットの到着を待つよう OS に頼みます。recvfrom が終わった時には配列 buf にパケットデータが入り、そのバイト数が recvfrom の戻り値として帰されるので、それを write により標準出力に書き出しています。

このプログラムを動かす前に、動かすマシンの IP アドレスを調べておいてください。それには ifconfig というコマンドを使います。[9]

```
% /sbin/ifconfig
em0: flags=8843<UP,BROADCAST,RUNNING,SIMPLEX,MULTICAST> mtu 1500
       options=3<RXCSUM,TXCSUM>
       inet 10.3.25.20 netmask 0xff000000 broadcast 10.255.255.255
       ether 00:0f:ea:10:4c:71
       media: Ethernet autoselect (1000baseTX <full-duplex>)
       status: active
plip0: flags=8810<POINTOPOINT,SIMPLEX,MULTICAST> mtu 1500
lo0: flags=8049<UP,LOOPBACK,RUNNING,MULTICAST> mtu 16384
       inet 127.0.0.1 netmask 0xff000000
%
```

インタフェースが複数表示されると思いますが、その中から inet(IP アドレス) の値が表示されていて、なおかつ 127.0.0.1 ではないものを選んでください。上の例では 10.3.25.20 が IP アドレスということになります。IP アドレスが分かったら、プログラム自体は次のようにして起動します (コンパイル済みのプログラムを入手した場合はコンパイルは不要です)。

```
% gcc -o recv recv.c  ←プログラム名を指定してコンパイル
% recv ポート番号
```

[9] GSSM の環境では/sbin/ifconfig と打つ必要がありますが、MacOS X ではただの ifconfig でよいはずです。Windows では代替のコマンド ipconfig を使ってください。

(受信待ちになる)

ここで指定するポートは 0〜65535 の数値ですが、1024 より小さい値は指定できない (特別なポート番号として予約されている) ので、適宜大きな値を指定してください。では次に送り側のプログラムを示しましょう。

```
/* send.c -- packet sending example. */
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>

main(int argc, char *argv[]) {
  int fd, len; char buf[20];
  struct sockaddr_in adr; struct sockaddr *p = (struct sockaddr*)&adr;
  adr.sin_family = AF_INET;
  adr.sin_addr.s_addr = INADDR_ANY;
  if((fd = socket(PF_INET, SOCK_DGRAM, IPPROTO_UDP)) < 0 ||
      bind(fd, p, sizeof(adr)) < 0) { perror("socket"); exit(1); }
  adr.sin_port = htons(atoi(argv[1]));
  adr.sin_addr.s_addr = inet_addr(argv[2]);
  while((len = read(0, buf, 20)) > 0) {
    if(sendto(fd, buf, len, 0, p, sizeof(adr)) < 0) {perror("send");exit(1);}
  }
}
```

こちらもポートの準備までは先と同様です。次に、アドレス型レコードを送り先指定にも使うため、ホストアドレスの部分を相手のアドレスに書き換えます。あとは繰り返し、入力からデータを読んでは sendto でパケットとして送ります。なお、IP アドレス、ポートはプログラムの引数として指定します。ではこれを使ってメッセージを送ってみましょう。

```
% gcc -o send send.c ←コンパイル指定は recv と同じ
% send ポート番号 IP アドレス
hello.
this is a pen.
...
```

すると、recv を動かしている側に打ち込んだものが現れるはずです (ちなみに止める機能はないので、Ctrl-C で中止させてください)。

ところで、send でリダイレクションを使えばファイルの内容を送ることができます。ちょっとやってみましょう。

```
% less t
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
bbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbb
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
ddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddd
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
bbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbb
...
% cat t t t t | send アドレス ポート ←沢山送る
```

すると、データを待っていた recv が再開されます。

```
(先の続き)
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
bbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbb
```

```
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
ddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddd
...
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
bbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbb
bbbbbbbbbbbbbbbbbbb  ←あれ?
...
```

大体よさそうですが…一部データが抜け落ちている??? これは、送り側のマシンが受け側のマシンより速いため、受け取りが間に合わなくなって取りこぼしが起きているためです (これを体験するためには速度差のあるマシンから送るなどして、しかもやや大きいファイル、たとえば 4096 文字くらいあるファイルで実験しないと再現しないのでそのつもりで)。ともかくこのように、パケットの送受信というのはベストエフォート方式なわけです。

演習 4-3 `send`/`recv` を動かせるようにして動作を確認。[10]うまく動いたら、次の事項から 1 つ以上 (できれば全部) やってみなさい。ベストエフェオートのパケットでも結構通信できることが分かるはずです。

a. 隣の人のマシンとチャットするのに使ってみる (それぞれのマシンの IP アドレスと使用ポート番号を教える必要がある)。
b. 内容の多いファイルを送ってみるか、または複数人で協力して一斉にあるマシンにデータを送ってみて、欠落が起きるか、起きるとしたらどういう条件で起きるかを調べてみる。
c. 3 人でチャットできるようにプログラムを改良してみる。[11]または、2 人で打ち合せて、それぞれが自宅など本当に別の場所にいるときにインターネット経由でチャットができるか試してみる。[12]

「◎」の条件: (1) 小課題のうち 2 つ以上に回答しており、なおかつ (2) やったことや結果に対する説明・考察が (担当者から見て) 適切であること。

3 ネットワークとプロトコル

3.1 システムと階層構造

コンピュータシステムもそうですが、多くのシステムは「階層構造」(layered architecuture) を持ちます。その基本的な考え方は、「全体を層状に構成し」「下の層の詳細は上の層に影響しないようにする」ことです。

たとえば、あなたが遠方の相手方と文書を検討するとして、その文書を秘書に送らせるとします。秘書が文書を送るのには「郵便」「クロネコメール」「FAX」など複数の手段がありますが、それはあなたは関知しなくてもよい。秘書はどの手段を使うかで作業内容が違いますが、郵便を使うとして、郵便ポストに入れたら、郵便局が集配に自動車を使うか、バイクかといったことは関知しなくてもよい (図 3)。逆に秘書はあなたの検討の内容には関知しないし、郵便局は秘書が送ったものの内容には関知しないわけです。

このように、階層構造を用いると「それぞれの層の中をうまく構成する」ことと「隣接する層とのやりとりをする」ことだけに限定して考えれば済むので、問題の複雑さが小さくなります。ネット

[10] GSSM 環境ならソースを持って来て例示のようにコンパイルする。Windows/Mac OS X 用にはバイナリの入ったアーカイブを Web に用意したので、持って来て適当な場所に展開し、そこでコマンドを実行する (ダブルクリックでは動かせないので必ずコマンドの窓を使用すること)。

[11] **ヒント:** `recv.c` は何人からでも受け取れるから変更の必要はありません。`send.c` は同時に 2 人に送る必要があるので、ホストとポートを 2 組指定するようにして、それぞれの宛先に同じものを送るようにすればよいでしょう。

[12] ネット環境によってはここで使っているプロトコルを通さない箇所があったりしてうまく動かないが、その場合はその状況を調べて報告すればよいです。

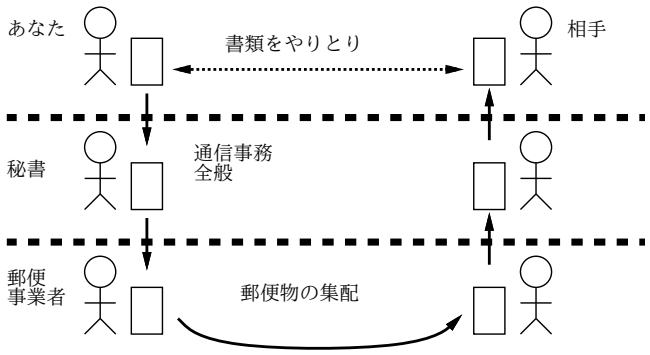

図 3: 階層構造

ワークシステム全体もそういう風に構成されているわけです (そうしないと作れないくらい込み入っている)。

たとえば、「パケットの抜けや追い越しがある伝送」という階層の上に「パケットが順番に抜けなく送られる伝送」を行う階層を作るとしたら、どうすればいいでしょうか? それには次のようにすればよいのです (図 4)。

1. 送り側は送り出すパケットに一連番号をつける。
2. 受け側は相手からパケットが来たら確認パケットを送る。
3. 送り側は一定時間待っても確認が来なかったら再送する。
4. 受け側は一連番号の順にパケットを渡す

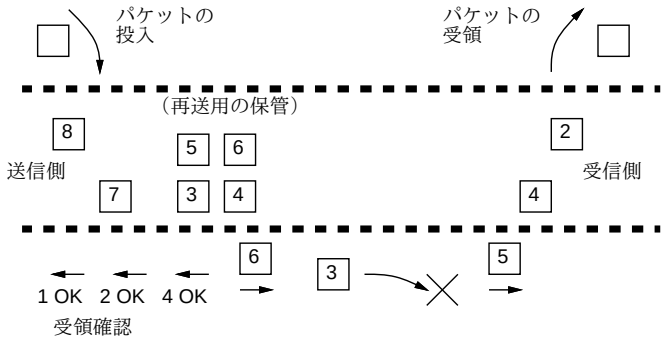

図 4: エラー制御のある伝送

このようにすると、下の階層でエラー (抜け等) があっても、上の階層にはエラーのないサービスが提供できます。ただし、下の階層でのエラーに対処するためには「再送」「確認待ち」などが必要であり、エラーが多くなればなるほど伝送は (上の階層から見て)「遅く」なるように見えます。

3.2 ネットワークソフトの階層構造

話を戻して、ネットワークソフトウェアに備わっているべき機能としてはどのようなものがあると思いますか? たとえば、あなたが遠隔地のホストから手元のホストにデータを取り寄せたいとします。そのとき、具体的にどのような機能が必要になるでしょうか? (以下にある箇条書きには同じ記号が何回も現れていますが、その理由は後で分かります。)

a. まず、あなたは相手ホストとデータの所在を指定して取り寄せるためのコマンドを起動します。
a. そのコマンドは何らかの形で相手ホストの対応する (データの取り寄せのような機能を提供する) サービスに連絡をとり、指定したファイルを送ってくれるように依頼します。

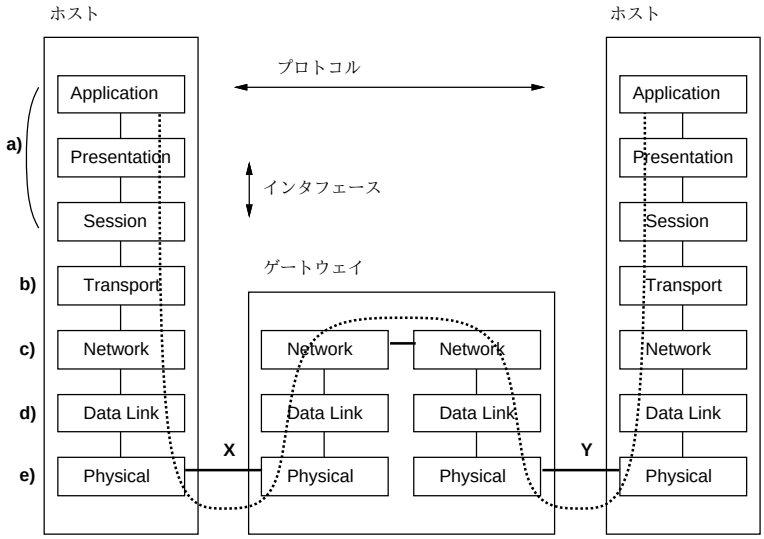

図 5: OSI 参照モデル

a. 相手ホストのサービスは依頼に応じて、データをパケットに入れて順次送り出してくれます。それを手元側で順次受け取ってファイルに格納します。

b. 既に学んだように、相手側がパケットを順次送ってくれたとしても、途中でデータが欠落したり順番が入れ替わったりすることもあります。ですから、送られた通りのものが受け取れているかチェックし、並べ直したり欠落部分を再送してもらったりすることが必要です。これを**エラー制御**といいます。

c. さらに、パケットは多くの中継機器を経由して来るので、それぞれの中継機器において正しい行き先に向けてデータを中継してもらう必要があります。これを**経路制御**といいます。

d. ホストや中継機器相互でのやりとりに当たっては、パケットを正しく伝送するための機能が必要です。とくにバス型の媒体では複数の機器が同時に送信しようとして信号が干渉するのを防ぐ制御が必要です。

e. 一番下のハードウェアレベルでは、媒体を構成する物質の上を信号が伝わって行くことで情報を運びます。

こうしてみると、ネットワークのソフトウェアというのは一番上の我々が利用したい操作 (データを取り寄せる等) から一番下のハードウェアまで、多数の**階層** (**レイヤ**) が積み重なって作られていることが分かります。

なぜ階層が重要なのかを整理しておきましょう。それは、階層に分けて考えることで機能を適切な (複雑すぎない) 大きさに分解して考えることができ、階層と階層の間の約束を決めておけばそれぞれを独立に用意し組み合わせられるからです。たとえば「パケットを送る」という機能さえ提供されれば、具体的な媒体は何であっても (光でも銅線でも赤外線でも) 同じようにネットワークを使うことができる、というのはこのような階層構造の利点です。

3.3 OSI 参照モデルとプロトコル

ISO(国際標準化機構) では、ネットワークの標準規格 (OSI — Open System Interconnect) を制定するに当たって、上に述べたような階層化の標準モデルを提案しました。これは 7 つの階層から成り、**OSI 参照モデル**ないし **7 層モデル**などと呼ばれます。その構成を図 5 に示します。

7 つの層は具体的には下から順に、次のような機能に対応しています。

1. 物理層 — 信号を運ぶ媒体に対応 (前節の e)

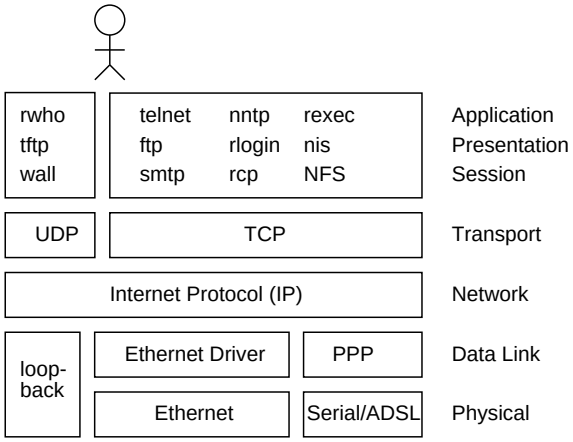

図 6: TCP/IP のプロトコル構成

2. データリンク層 — 媒体を使った信号の送受を制御 (前節の d)
3. ネットワーク層 — 経路制御によりパケットを目的地まで中継していく (前節の c)
4. トランスポート層 — エラー制御により誤りのないデータ送受を行う (前節の b)

ここから先は前の節では区別せず a と記しましたが、OSI モデルでは次のように別れています。

5. セッション層 — 通信の開始/終了/状態記憶などの機能を提供する
6. プレゼンテーション層 — 通信に適した形にデータを変換したり、それを復元する
7. アプリケーション層 — 具体的な個々のサービス (ファイル転送、メール送信、等) に対応

データを実際に送信するときは、送り側のホストのアプリケーション層の機能がそれを下へ下へと渡して、物理層を通じて送り出します。中継点ではパケットを受け取り、経路制御を行い、次の行き先に向かって送り出しますから、ネットワーク層までが稼働していることになります。受け側のホストでは、受け取ったパケットのデータは上へ上へと渡され、最後はアプリケーション層でそれを用途にあった形で処理します。

このとき、同じ層のソフトウェアどうしが共通に従う約束を**プロトコル** (protocol) と呼びます。たとえば、物理層では使用する電圧や信号の周波数などの約束が必要です。データリンク層では、パケットの開始や終了を表す信号、干渉を避けるための信号の取り決めなどが必要です。ネットワーク層では、パケットのどの場所に送り先のアドレスを格納し、アドレスの形式はどのようなものか、といった取り決めが必要です。トランスポート層では、順番をチェックするためにどのような形でパケットに一連番号を割り振るか、パケットの正しい到着や欠落をどうやって送り元に通知するか、などの取り決めが必要です。これらはすべてプロトコルの例です。そして、各層のプロトコルを合わせた全体を**プロトコル群** (protocol suite) と呼びます。その代表的なものに、インターネットで使われている**インターネットプロトコル群** (その中の代表的なプロトコルの名前を取って **TCP/IP** とも呼ばれます) があるわけです。

4 TCP/IP とインターネット

4.1 TCP/IP のプロトコル群

ここからは TCP/IP を題材としてもっと具体的に見ていくことにします。実は TCP/IP にも現在広く使われている **IPv4** と、次世代の規格として作られ実用化され始めている IPv6 とがありますが、以下では IPv4 について説明します。本書で述べる範囲では、アドレス表記やプロトコルやプログラムの名前が違う程度でして、両者に原理的な違いはありません。

プロトコル群とはネットワークの各層を構成するプロトコルの集まりだと先に書きましたが、TCP/IP の場合は図 6 のような構成になっています。ここで目につくのは、中間のネットワーク層に相当する **IP**(Internet Protocol) はすべてに共通していて、その上や下では層ごとにプロトコルが複数存在している点です。このうち、下側 (データリンク層と物理層) については、通信のための媒体が何であるかによって使い分けられます。しかし、どの媒体であっても最終的には IP のパケットをやりとりさせてくれるので、その上側では媒体が何であるかに関わらず同じように使えるわけです。つまりインターネットとは、「IP をやり取りできるすべてのネットワークの総体」ということになるわけです。

IP より上側のプロトコルは、**UDP** (User Datagram Protocol) に基づくものと **TCP** (Transmission Control Protocol) に基づくものとに大別されます。UDP は単独のパケットを送受するもので、少量のデータを迅速にやりとりしたい場合に向いています (先の実験プログラムも UDP を使用していました)。これに対し、TCP は大量のデータでも信頼性のある形でやり取りできるような機能を提供しており、多くの上位プロトコルはこれを用いて実現されています。

4.2 インターネットの構造

個別の話題に入る前に、インターネットそのものの全体像から説明しましょう。そもそも、皆様は自分の使っているマシンからどのようにしてインターネットにつながっているのか、考えたことがありますか?

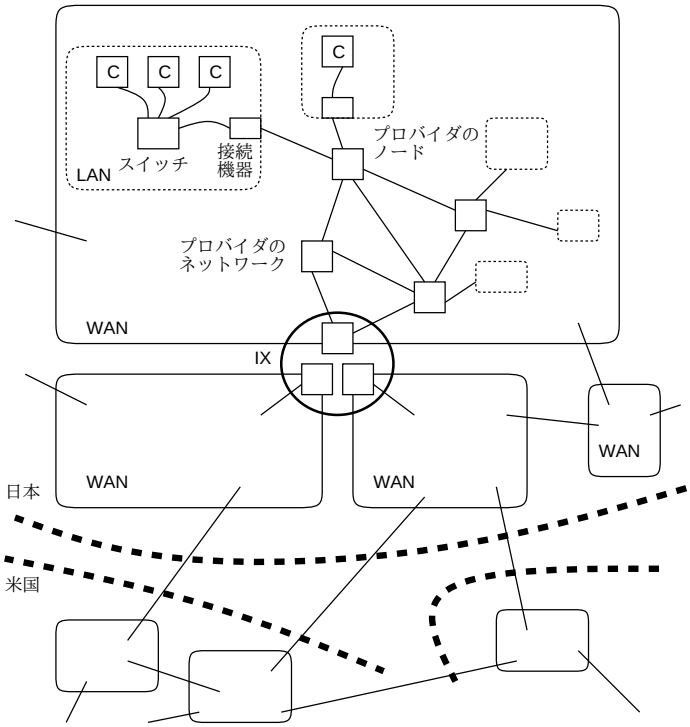

図 7: インターネットの構造

図 7 に簡単な概念図があります。左上の方にある「C」と書かれた箱のあたりが、自分のコンピュータだと思ってくださない。GSSM のように多数のホストを持っているところでは、それらのマシンは**スイッチ** (switch) ないし**ハブ** (hub) によって相互接続されていて、1 つの LAN を構成しています。個人の家のように、マシンが小数しかない場合でも、今日では無線 LAN を使うのでやはり LAN になっていることが多いでしょう。

これらの LAN が直接接続する相手先は **ISP** (Internet Service Provider) ないし「プロバイダ」と呼ばれる事業者のネットワークです。ISP は契約者から料金を取ってインターネットへの接続を提供するのが商売であり、そのために大規模な WAN を自社の設備によって (または他社の設備を論理的に借りるなどして) 構築しています。

しかしこれだけでは、私がプロバイダ A と契約していて、知合いがプロバイダ B と契約しているときに、相互に通信できないことになります。そこで、プロバイダはそれぞれの WAN を相互に接続しており、そこを通じてパケットを相互に流通させます。[13] A と B が直接繋がっていなくても、間接的につながってさえいれば互いに通信ができるのです。このようにして、多数のネットワークが相互につながってできた集合体がインターネットです。そういう意味で、インターネットは「草の根」的な構造をしており、どこかに「本部」があるわけではないのです。

もちろん、海外との通信のためには海底ケーブルなどをによる国際リンクを通る必要があります。これもまた、大手の ISP(NTT や KDD など) によって提供されており、これらを通って我々のパケットは海外と行き来できるわけです。しかし自分は某 ISP と契約しているけど、NTT や KDD にはお金を払っていないのに、と思うかも知れません。インターネットは「どんぶり勘定」の世界であり、あなたが行う個々の通信の経路を調べて課金するようなことは (手間も掛かるし) 省略していますが、皆がネットのために費すお金がまわって間接的にこれらのコストも支えているわけです。[14]

ときに冒頭では `ping` を使って到達時間を計りましたが、`traceroute` というコマンドを使うと相手先までの経路を (できる範囲で) 調べることができます。ブラジルのホテルでやってみましょう。[15]

```
% traceroute royaltyhotel.com.br
traceroute to royaltyhotel.com.br (200.219.245.77), 64 hops max, 40 byte packets
 1  plalagw (210.154.96.161)  2.427 ms  0.517 ms  0.523 ms
 2  218.44.77.41 (218.44.77.41)  10.041 ms  10.060 ms  9.847 ms
 3  218.44.77.62 (218.44.77.62)  10.782 ms  10.675 ms  10.802 ms
 4  118.21.174.117 (118.21.174.117)  11.615 ms  10.806 ms  11.433 ms
 5  i118-21-197-77.s99.a049.ap.plala.or.jp (118.21.197.77)  11.475 ms  11.221 ms  11.586 ms
 6  i118-21-179-4.s99.a049.ap.plala.or.jp (118.21.179.4)  55.065 ms
    i118-21-178-4.s99.a049.ap.plala.or.jp (118.21.178.4)  19.120 ms  11.191 ms
 7  218.43.251.133 (218.43.251.133)  18.661 ms  11.012 ms  10.833 ms
 8  118.23.146.45 (118.23.146.45)  19.257 ms  11.365 ms
    118.23.146.53 (118.23.146.53)  19.293 ms
 9  60.37.27.89 (60.37.27.89)  11.991 ms  11.695 ms  11.350 ms
10  ae-5.r21.tokyjp01.jp.bb.gin.ntt.net (129.250.11.53)  11.864 ms  12.141 ms  11.675 ms
11  as-2.r21.snjsca04.us.bb.gin.ntt.net (129.250.4.44)  135.900 ms  114.748 ms  119.323 ms
12  ae-2.r06.snjsca04.us.bb.gin.ntt.net (129.250.5.55)  130.286 ms  129.767 ms  128.190 ms
13  te3-3-10G.ar3.SJC2.gblx.net (64.215.195.137)  115.858 ms  116.621 ms
    te6-4-10G.ar4.LAX1.gblx.net (64.212.107.1)  124.299 ms
14  64.214.59.99 (64.214.59.99)  286.350 ms  293.328 ms  291.043 ms
15  static.200.143.173.154.datacenter1.com.br (200.143.173.154)  286.456 ms
    static.200.143.173.110.datacenter1.com.br (200.143.173.110)  304.868 ms
    static.200.143.173.114.datacenter1.com.br (200.143.173.114)  297.920 ms
16  static.200.219.215.59.datacenter1.com.br (200.219.215.59)  296.797 ms  287.204 ms  293.739 ms
17  static.200.219.245.77.datacenter1.com.br (200.219.245.77)  292.347 ms  289.656 ms  288.660 ms
%
```

このコマンドは指定した宛先までの通信が通っている中継点とそこまでの往復時間を順次リストアップします (場所によっては中継点が複数マシンから構成されていてタイミングでどのマシンかが代わるものがあり、このような時は中継点が複数表示されます。また、セキュリティのため情報を返さない中継点もあり、その場合は「`* * *`」のような表示になります)。これを見ると、おおよそ次のようなことが分かります。

- GSSM のマシンから plala という会社 (ISP です) のネットに入り、そこの内部で数段の中継がある。そのあたりまでは 10 ミリ秒程度。
- plala から NTT に渡され、NTT の所有している線で米国に渡る (時間が 120 ミリ秒くらいに)。

[13] これを効率よく行うため、多数の ISP が相互に接続を取るための地点を設けることもあります。これを **IX** (Internet eXchange) と呼びます。

[14] しかし、動画など大きなデータを大量に送受する利用者は、メールや普通の Web などしか使わないユーザよりも圧倒的に多くのネットワーク帯域を使います。このため、定額料金制をやめて、多く使うユーザはお金を多く払うようにしよう、という方向の動きもあります。

[15] GSSM のシステムでは `traceroute` コマンドは一定期間だけ使用可能にします。皆様の個人マシンではそのような制約はないです。あと、Windows ではコマンド名は `tracert` になっています)。

- 米国内で Gbl X という通信会社に渡され、そこでブラジルに渡る (時間が 300 ミリ秒くらいに)。
- そのホテルのサービスはブラジルのデータセンターに収容されているらしい。

演習 4-4 演習 4-1 でパケット到達時間を調べたホストまでの経路を `traceroute` で確認してみなさい。それができたら、次の事項から 1 つ以上 (できれば全部) やってみなさい。[16]

a. 日本国内のさまざまな箇所までの経路を調べ、自分のいるところから国内がどのようにつながっているかを整理してまとめなさい。

a. 世界の複数の地域までの経路を調べ、日本から世界各地にどのようにつながっているかを整理してまとめなさい。

b. GSSM から各地までの経路と、自宅 (ないし職場など) から各地までの経路を比較し、どこまでが違ってどこからは共通かなどを整理しなさい。

「◎」の条件: (1) 小課題のうち 2 つ以上に回答しており、なおかつ (2) やったことや結果に対する説明・考察が (担当者から見て) 適切であること。

4.3 IP アドレス

ここまでにも既に何回も「アドレス」という言葉が出て来ました。通信をするということは、相手を指定する必要があるわけで、その相手を指定する「もの」が「アドレス (番地)」です。ここでは IP が使うアドレス、すなわち **IP アドレス** (IP address) についても少し詳しく見ておきます。

TCP/IP が今の形になったのは 1980 年ごろですが、その当時はネットワークの規模はごく小さく、通信帯域は小さく、コンピュータも非力でした。パケット方式ではすべてのパケットに宛先つまりアドレスを入れなければならないので、その形がコンパクトでコンピュータによって効率よく扱えることは重要でした。そこで IPv4 ではアドレスとして「32 ビットの値」を使うことにしたわけです。我々が普段目にしている「192.168.1.1」のような書き方は、32 ビットを 8 ビットずつに区切って、各バイトの値を 0〜255 の十進数として表記したものです。

グローバル IP アドレス (global IP address) すなわち「正式な」IP アドレスは、世界中 (インターネット中) で重複がないように管理されていて、ある特定の IP アドレスを持つホストは世界中でただ 1 つしか存在しないようになっています (実際にはネットワーク番号を重複しないように各組織に割り当て、ホスト番号はその組織のネットワーク管理者が割り当てます)。これによって、世界中のどこからでも「あのコンピュータ」という指定をすればそのコンピュータに向かってパケットが転送されて来るようになるわけです。

もちろん、重複が無いように割り当てるには、管理が必要です。IP アドレスの割り当ては、国際非営利組織 **ICANN**(Internet Corporation for Assigned Names and Numbers、`http://www.icann.org/`) が管理を行っています。また、各国には下請けとしてその国の範囲での割り当て調整組織がありますが、日本の場合は **JPNIC**(JaPan Network Information Center、`http://www.nic.ad.jp/`) がこれに相当します。実際にアドレスを必要とするプロバイダ (接続業者) 等は、これらの管理組織から自社のネットワークおよび自社の顧客のためのネットワーク番号を割り当ててもらって使用する、という仕組みになっています。

逆にいえば、自分で勝手に IP アドレスを設定してインターネットに接続することは厳禁なわけです。しかし、外部に接続しない孤立したネットワークを構成する場合にもいちいちアドレスの割り当てを受けるのは合理的でありませんから、そのような孤立したネットワークでは自由に使っていいネットワーク番号がいくつか用意されています。これらのネットワーク番号に属する IP アドレスのことを**ローカルアドレス** (local address) と呼びます。たとえば GSSM 内場合、そのほとんどのマ

[16] a と b は GSSM でやってもよいですし、自分の私有 PC で調べてもいいです。

シンはローカルアドレス「10.*.*.*」を用いて相互に通信をしており、外部とやり取りをするマシン utogw のみがグローバル IP アドレスを持つようになっています。[17]

ところで、インターネットの急激な成長のため、当初 32 ビットあれば十分だろうと考えられていた IP アドレスが多数必要となり、既に枯渇状態になっています。この教訓から、新しい世代の IP(IPv6) では IP アドレスを一挙に 128 ビットに増やし、アドレス不足がまず起こり得ないようにしました。ただ、プロトコルの移行は簡単ではないので、現状ではまだ IPv4 を使い続けている組織が多数派です。そのような組織の多くは、GSSM のように、グローバル IP アドレスを少数だけ用い、多数のマシンはローカルアドレスを用いて運用しています。また、個人宅で LAN を組んでいる場合も、ブロードバンドルータは外側だけにグローバル IP アドレスを用い、内側の (家庭内の) ホストにはローカルアドレスを割り当てるようになっています。

ただし、ローカルアドレスのマシンは直接インターネットと通信できないという不便さがあります。GSSM のサイトでは、最もよく使う Web が見られればあとはあまり直接外と通信する必要はないという方針で、Web サービスを utogw で中継し、あとは (今回演習でやっているように) 必要に応じて utogw にログインして対外接続の必要な作業を行ってもらうようにしています。

一般家庭の場合はユーザにそいういう技術的な使い分けを強制できないので、代わりにブロードバンドルータに **NAT** (Network Address Translation) という機能を搭載しています。これは、LAN 側から外に向かって通信を行う時に、パケットに埋め込まれているアドレスを系統的に書き換えることで、インターネット側の接続相手にとってはグローバルアドレスを持った中継ノードと接続しているように見せるものです。ただし、NAT は「見せかけ」により外部の相手をだましているので、自由に外部からアクセスできるようなサービスの提供は難しいという弱点があります (セキュリティ上はその方が安心という面もありますが)。そういうわけで全体としてては、徐々に IPv6 へ移行することによって IP アドレス枯渇問題を克服することが望まれています。

4.4 ドメイン名と DNS

ここまでに述べて来たように、IP での通信はあくまでも IP アドレスを用いて行われますが、人間が見て理解するにはもっと「普通の」(意味のある) 名前を使うことが望まれます。このため、インターネット上では**ドメイン名** (domain name) と呼ばれる文字列の名前サポートし、ドメイン名から IP アドレスを検索するためのシステムとして **DNS** (Domain Name System) と呼ばれるサービスが運用されています。

ドメイン名は簡単にいえば複数の名前を「.」でつなげたもので、その複数の名前は右側ほど広い範囲に対応する階層構造になっています (日本の住所の表記方法とは反対ですが、英語では住所、街、州、国の順で書くのでそれに合わせたわけです)。たとえば筆者の所属する組織のサーバのドメインアドレス「utogw.gssm.otsuka.tsukuba.ac.jp」は次のような階層に対応しています。

```
utogw.gssm.otsuka.tsukuba.ac.jp
                                ↑日本
                              ↑教育組織
                          ↑筑波大学
                      ↑大塚地区
                  ↑専攻名
              ↑ホスト名
```

そして、DNS はこの階層構造を利用して、たとえば次のような形で検索を実現しています。

- インターネット全体について、「ルート」と呼ばれる数台の DNS の「元締め」マシンがある。

[17] 正確に言えば utogw は内部ネットワーク向けのインタフェースの他に外部に接続されたインタフェースも持っていて外部への通信では後者を通じてパケットを送受するわけです。なお、GSSM には学生はログインできない外接続用サーバもあります。

- そこに最右側の名前 (**TLD**、Top-Level Domain) を渡して問い合わせると、その TLD ならどこに聞いたらいいかを教えてくれる。
- .jp を管轄するサーバは「.ac.jp」「.co.jp」などそれぞれについて、どこに聞いたらいいかを教えてくれる。
- .ac.jp を管轄するサーバに.tsukuba.ac.jp を聞くと、筑波大の DNS サーバを教えてくれる。
- 筑波大のサーバに.otsuka.tsukuba.ac.jp を聞くと、大塚の DNS サーバを教えてくれる。

このように階層を降りていくと、いつかはそのドメインアドレスに対応するホストの IP アドレスを直接知っている (またはそのようなドメインアドレスのホストはないことを知っている) サーバに到達しますから、そこから結果を返してもらえます。

DNS サーバに対する検索は多くの場合、ドメインアドレスを受け取る各プログラムが (対応する IP アドレスを調べるために) 発行しますが、コマンド **nslookup** を使ってユーザが直接 DNS サーバに検索を依頼することもできます。

- nslookup ドメインアドレス — ドメインアドレスに対応する IP アドレスを検索表示させる

これを使うようすを次に示しておきます。

```
% nslookup www.yahoo.co.jp.
Server:  utogw.gssm.otsuka.tsukuba.ac.jp  ←検索の入口となるサーバ
Address:  192.50.17.2

Non-authoritative answer:          ←結果は「ヒント」であることを示す
Name:    www.yahoo.co.jp
Addresses:  202.229.198.216, 203.141.35.113, 210.81.150.5
%
```

上の例では IP アドレスが 3 つ返されていますが、これは大量のアクセスをさばく必要がある WWW サーバなどでは複数のマシンにアクセスを分散させるようにしているため、IP アドレスも複数持たせているためです。

検索については分かりましたが、そもそもドメイン名はどのようにして割り当てられているのでしょう? 勝手に好きな名前をつけるわけには行かないのは明らかですね。ドメイン名も IP アドレス同様、元締めは ICANN ですが、その管理の構造は少し違います。まず、TLD には.jp(日本)、.uk(英国) など各国に対応した国別 TLD と、.com(企業等)、.edu(教育機関) など国を特定せず組織種別等で分類した **gTLD**(Generic TLD、汎用 TLD) があり、それぞれについて ICANN が管理組織を定めています (たとえば.jp は JPNIC が母体となって発足させた事業者 JPRS が管理を引き受けています。

そして IP アドレスと違ってドメイン名は文字列であり非常に自由度が高いため、個人でも団体でも「まだ取られていない名前であればお金を払って登録する」形で取得できます。ICANN その他の管理組織はとてもそのような個別の事務はできないので、ICANN から許諾を得て一般に名前を販売する**レジストラ** (registerer) と呼ばれる業者が多数存在し、そこから購入するようになっています。レジストラはたいてい、DNS への登録まで含めた年間維持料金を取るようになっているので、あなたも自分の名前の入ったドメイン名を簡単に取得できます。[18]

演習 4-5 nslookup を用いていくつかの (よく知られている) サーバの IP アドレスを検索してみなさい。その後、以下の 3 つの課題から 1 つ以上 (できれば全部) やってみなさい。

a. ホスト名と IP アドレスの間にどのような関係があるか、また地域と IP アドレスの場合はどうか、検討してみなさい。

[18] メール転送や Web ページ提供などのサービスも用意しているところがあり、それらを利用すれば自分でマシンを用意しなくてもある程度取得したドメインを役立てられます。

b. 自分の名前の入ったドメイン名が存在するかどうか、試してみなさい。存在するなら、それはどこに割り当てられているかも検討してみなさい。[19]または、適当なドメイン名販売サイトを探して来て、自分の名前や著名な組織の名前でドメイン名を取ろうとするといくら掛かるのか調べてみなさい。

c. 「IPv4 アドレスはほとんど枯渇した」とか「多様な gTLD が使われるようになっている」などのことがらを、JPNIC のサイトの情報などを用いて検討しなさい。必ず、どのサイトを参照したか記すこと。

「◎」の条件: (1) 小課題 2 つ以上に回答しており、(2) 調べたことがきちんと説明されていて、(3) 検討内容が (担当者から見て) 適切であること。

4.5 物理層とデータリンク層 (第 1・2 層)

アドレスまでが分かったところで、今度は物理層から順に TCP/IP のプロトコル構造を見て行きましょう。先に説明したように、物理層は実際に信号を運ぶ媒体 (銅線とか光ファイバーとか電波とか) を意味し、データリンク層はその媒体の上でパケットをやり取りするための制御を行う機能を意味します。媒体の種類ごとに制御のしかたも変わって来ますから、物理層とデータリンク層は密接に関連しています。

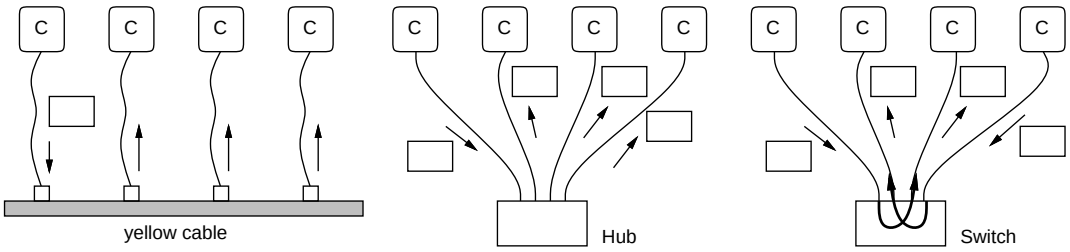

図 8: Ethernet の原理

最も代表的な物理層は**イーサネット** (Ethernet) と呼ばれる LAN 技術で、1970 年代初期に Xerox 社の研究所で開発されたものです (しかし Xerox はこれを商品化はせず、発明者がスピンオフして起こした会社 3COM が広く普及させました)。イーサネットの最初のものは、(なぜか黄色い色の被覆のものが主に使われたため)「イエローケーブル」と呼ばれる太い同軸ケーブルを床に一筆描きのように這わせ、そこにコンピュータを繋ぎ込んでいました (図 8 左)。あるコンピュータからパケットを送信すると、その信号はケーブルを伝ってすべてのコンピュータに届きますが、パケットには宛先が書いてあるのでその宛先のマシン以外は無視します。

複数のマシンが同時に送信しようとすると、パケットの**衝突** (collision) が起きますが、各マシンとも衝突したことが分かったら送信をやめてランダム時間だけ待ってから再度送信を試みます。この方式を CSMA/CD (Carrier-Sense Multiple Access with Collision Detection、つまり共有媒体を複数端末が監視しながら使用し、衝突の検知を行うという意味) と呼びます。混雑がひどいとランダム時間待った後でも再度衝突するかも知れませんが、連続して衝突が起きるごとに待ち時間を倍々にして行くことで、最終的には衝突なしに送れる、というアイデア (exponential backoff) が使われています。

技術の進歩とともにイエローケーブルの代わりに安価なシールド無しより線対ケーブル (UTP、Unshielded Twisted-Pair cable) でも十分な速度が出るようになり、全てのマシンに信号を中継するためには中央に**ハブ** (hub) と呼ばれる装置を置くようになりました (図 8 右)。今日では、パケットの中身を見て中継の必要なところだけに信号を送る**スイッチングハブ** (switching hub) が主流になっています。より線対を使ったイーサネットでは通信速度は 100Mbps〜10Gbps で距離は 100m 程度ですが、光ケーブル等を使ってより高速/長距離の通信を行える機器もあります。

[19] `kuno.jp` とか `kuno.com` というのはもう取られているようです。

無線 LAN はイーサネットの通信線の代わりに電波を使うもので、いわば「空間」がハブに相当することになります。当然、衝突によるスループットの低下もありますが、他の機器等電波との干渉などの問題も起きやすいという弱点もあります。また、電波だと盗聴しやすいので、WPA(Wi-Fi Protected Access)、WPA2(Wi-Fi Protected Access 2)、WPS(Wi-Fi Protected Setup) などの暗号機能を併用することが多く行われます (無線 LAN 機器に組み込まれている)。[20]

Unix ではデータリンク層の機能は**ネットワークインタフェース**に付随した形で提供されています。先に出てきた `ifconfig` コマンドを使うと、自分が使っているマシンにどのようなインタフェースが備わっているかが分かりますが、併せてそれぞれのインタフェースがどのようなデータリンク機能を提供しているか、その状態がどうなっているかも見ることができます。物理的な (ケーブルのつながった) インタフェース以外に、仮想ネットワークに対応するインタフェースも、`ifconfig` コマンドでそのようすを調べることができます。また、ネットワークに関する各種情報を表示させるコマンド `netstat` にはさまざまなオプションが用意されていますが、その中の「`-I` インタフェース名」というオプションを指定し、さらに秒数を指定することで、あるインタフェースを通過したパケット数を定期的に (指定した秒間隔で) 表示させて調べることができます。

```
% netstat -I fxp0 1
            input        (fxp0)         output
   packets  errs   bytes    packets  errs   bytes colls
         0     0       0          0     0       0     0
         1     0     160          0     0     226     0
         2     0     675          3     0    2056     0
         3     0      60          2     0      66     0
         2     0       0          3     0       0     0
         1     0     941          0     0     785     0 ←データ
         3     0   41961          4     0   27492     0  取り寄せ
        36     0   19145         32     0   17968     0
        56     0   67193         50     0   60948     0
        33     0   38244         30     0   57064     0 ←完了
       118     0       0        111     0       0     0
         2     0       0          2     0       0     0
         0     0       0          0     0       0     0
^C  ← Control-C により中止
%
```

4.6 ネットワーク層 (第 3 層)・IP と経路制御

ネットワーク層と IP の最大の「魔法」は、行き先を指定するだけで、それが世界中どこであっても、パケットをその行き先に届けてくれることです。この機能を**経路制御** (routing)、経路制御機能を持つ中継機器を**ルータ** (router) と呼びます。[21]

IP の経路制御がなぜそんなに偉いのでしょう? 郵便物の場合を考えて見ましょう。ある郵便局に集まってきた葉書に「東京都目黒区駒場 1-1-1」という宛先が書いてあったとすると、そこが東京都以外の郵便局であれば、東京に「目黒区」という区があるかどうか知らなかったとしても、とにかく東京中央郵便に送れば済みます。東京中央郵便局では、東京都のどの区や市はどの局の受け持ちか知っていますから、「駒場」という地名を知っていてもいなくてもとにかく目黒郵便局に送れば済みます。これが可能なのは、住所が「都道府県→区市町村→地名→番地」という**階層構造**になっているからです。アドレスが階層構造になっていれば、各中継地点では「自分の受け持ち範囲でないアドレスはとにかく上位の中継地点に送る」「受け持ち範囲のアドレスはより小さい受け持ち範囲の中継地点か個々の宛先に送る」という方法で経路制御が行えます。

しかし IP ではアドレスは 32 ビットの数値であり、上で述べたような階層構造にはなっていません。32 ビットのうち上位何ビットかがネットワークアドレス、残りがホストアドレスという形で 2 つに分けられていて、1 つのネットワーク (LAN のセグメント) につながるホストは同じネットワークアド

[20] 無線 LAN 普及初期の暗号化機能である WEP(Wireless Encription Privacy) は、解読方法が確立されてしまっているため、今日では使用すべきではない。

[21] また、データリンク層の機器に属するスイッチも高度化してルータのような機能を持つようになったため、このような機器をとくに**レイヤ 3 スイッチ** (layer 3 switch) と呼ぶこともあります。

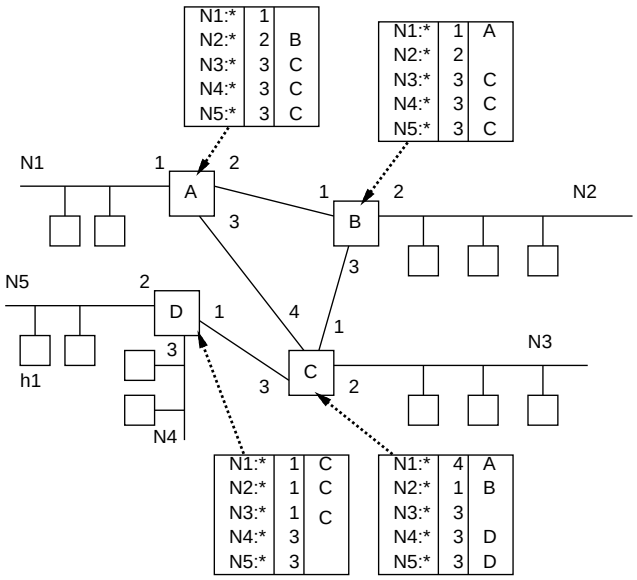

図 9: 経路表

レスということは決っていますが、ネットワークアドレスのつけ方は任意です。[22]つまり「駒場」「八雲」「鷹番」などの名前だけが与えられ、それを見て送り先を決める必要があります (鷹番 N 丁目、というのが近くにまとまっていることは保証されますが)。つまり、すべての主要な郵便局には全国のあらゆる地名の一覧表があり、その一覧表に「この地名はこちらの方に送る」と書かれていて、それに従って中継するわけです。このデータのことを**経路表** (routing table) と呼びます。

たとえば図 9 のように 4 つのルータでネットワーク N1〜N5 がつながっているとします。それぞれのルータは、自分が直接つながっているネットワークについてはそこにつながるインタフェースにパケットを送ればいいと分かりますが、それ以外については経路表を参照してパケットの転送先を決めます。たとえば、ルータ A に「N5:h1」あてのパケットが来たとすると、表を参照して「インタフェース 3 番を通じて C に送ればいい」と分かります。C にそのパケットが来ると、C も同様にして「インタフェース 3 番を通じて D に送る」と分かり、D に到達するとそこから直接 N5 に送るわけです。

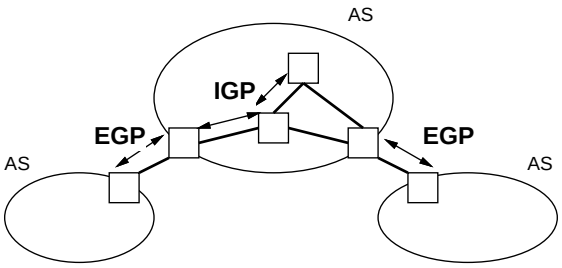

図 10: 自律システムと IGP/EGP

このような経路表を作るために、経路情報を交換するプロトコルが用いられています。実際には、インターネット全体をまとめて管理するのは無理なので、プロバイダの WAN などまとまって管理されるネットワーク群を **AS** (Autonomous System、自律システム) という単位とし、AS 内部では **IGP** (Interior Gateway Protokol、内部ゲートウェイプロトコル) を用いて内部の経路情報を制御します。一方、AS と AS の境目部分ではこれとは別の **EGP** (Exterior Gateway Protocol) を使い。「うちにパケットが来た場合、これこれのネットワークに到達させられますよ」という情報を相互に交換

[22]厳密にはそれだと経路表が大きくなって大変なので、できるだけ隣接したネットワークには隣接した番号をつけ、外部からはあたかも 1 つのネットワークに見えるようにします。これを経路の**集約** (aggregation) と呼びます。

しています。[23]

4.7 トランスポート層 (第 4 層) と UDP・TCP

TCP/IP のトランスポート層プロトコルには TCP と UDP の 2 つがあることは既に述べました。**UDP** は IP の機能をほぼそのまま利用者に提供するものであり、パケット単位で送り先を指定してデータを送受信する機能を持ちます。UDP はベストエフォート型であり信頼性の機能は提供しませんが、その代わりに UDP ではオーバヘッドの小さな通信が行えます。信頼性が無いのでは役に立たないと思われるかも知れませんが、UDP を使うアプリケーション側の用途によってはそれでも問題ないこともありますし (たとえば音声通話などの場合は時々パケットロスがあっても雑音が入る程度で実用上問題ないかも知れません)、アプリケーション側をパケットロスに対処するように作ることも考えられます。

一方、**TCP** はここまでに説明してきたような方法でエラー制御を行い、信頼性のある通信を提供します。ただしそのためには、まず通信相手との間で接続を確立し、準備を行ったあと、連続したデータ送受を行い、最後に切断する、という手順になります。これは回線交換とまったく同じなので、**仮想回線** (virtual curcuit) と呼びます (仮想というのは、内部はパケット通信だけれど外から見ると回線交換みたいだという意味です)。TCP はエラー制御に加えて、ネットワークの速度に併せて送信側の進度を制御するなど多くの機能を提供しています。

ところで、IP アドレスはあくまでもホスト (正確にはホストについている各ネットワークインタフェース) を識別するものです。実際には各ホストでは多数のプロセスが動いていますから、ネットワーク接続では「どこと接続するか」を指定する必要があります。たとえば同じホストに対してでも、「メールを送信したい」というのと「ファイルを転送したい」というのでは使用する (つまり接続相手となる) プログラムが違うわけです。

TCP と UDP では、この「どの相手」を指定するのに**ポート番号** (port number) と呼ばれる 16 ビットの数値を使用します。本章の冒頭で出てきた例題プログラムでは、実験用に適当なポート番号を選んでそのポート番号で接続を行っていました。普段実用に使っているネットワークソフトウェアでも、何らかの方法でポート番号を決める必要があります。このためにポート番号 0〜1023 の範囲は**公知ポート番号** (well-konwn ports) として予約されており、TCP と UDP それぞれ個別に、どのサービスはポート何番を使うかが決っています (この割り当てもやはり IANA が管理しています)。Unix システムではポート番号とサービス名称の対応表が
`/etc/services` というファイルに記述されていて、これに基づいてポート番号と名前の変換を行っています。

プロセスが使用している TCP および UDP ポートの一覧は `netstat` コマンドに「`-a`」というオプションを指定することで表示させられます。

```
% netstat -a
Active Internet connections
Proto Recv-Q Send-Q  Local Address    Foreign Address  (state)
tcp4       0      0  sma.nfsd         smp.798          ESTABLISHED
tcp4       0      0  sma.nfsd         smri06.1019      ESTABLISHED
tcp4       0      0  sma.49244        smr04.x11        ESTABLISHED
tcp4       0      0  sma.canna        smm.36645        ESTABLISHED
tcp4       0      0  sma.nfsd         smr04.1017       ESTABLISHED
tcp4       0      0  sma.canna        smri21.50085     ESTABLISHED
tcp4       0      0  sma.nfsd         smri21.999       ESTABLISHED
tcp4       0      0  sma.nfsd         smri17.1011      ESTABLISHED
tcp4       0      0  sma.nfsd         smr05.987        ESTABLISHED
tcp4       0      0  sma.nfsd         utogw.788        ESTABLISHED
tcp4       0      0  localhost.smtp   *.*              LISTEN
udp4       0      0  localhost.ntp    *.*
udp4       0      0  sma.ntp          *.*
```

[23] 一方、個別組織の LAN などでは規模が小さく経路が単純なため、管理者が手動設定で経路を設定したり、ごく単純な経路制御プロトコルで運用したりします。

```
udp4       0      0  localhost.1019   localhost.1022
%
```

tcp4、udp4 はそれぞれ IPv4 の TCP と UDP を意味します。アドレスの「.」より前はホスト名、後はポート番号ですが、/etc/services にサービス名記述されているポートについてはサービス名で表示されています。TCP については接続状態 (接続中、接続待ち) が表示されています。

演習 4-6 「netstat -a」や「netstat -I インタフェース 秒数」でネットワークの接続状況やパケット通過状況が観察できることを確認しなさい。うまく行ったら、次の課題から 1 つ以上 (できれば全部) 選んでやってみなさい。

a. 「他のマシンに rlogin で接続する」「ブラウザでページを開く」「他のマシンの窓を開く」「日本語入力する」などの操作をやってみて、それに対する接続が増えるかどうか、増えるとしたらその接続はどうやって見分けるかを検討しなさい。

b. send/recv プログラムを動かした状態でネットワーク接続の確認をおこない、UDP ソケットは TCP ソケットと比べてどのような違いがあるのか検討しなさい。

c. 枚秒のパケット通過状況を表示させた状態で「ファイルサーバ上に大きなファイルをコピーする」「ブラウザで大きなファイルを取り寄せる」などの操作をおこない、大きなデータを移動する場合にパケット 1 つあたり何バイトくらい入っていると思われるか検討しなさい。

「◎」の条件: (1) 小課題 2 つ以上に回答しており、(2) 調べたことがきちんと説明されていて、(3) 考察が (担当者から見て) 適切であること。

4.8 セッション層以降の上位層 (第 5〜7 層)

OSI の 7 層モデルでは、伝達層より上にセッション層、プレゼンテーション層、アプリケーション層の 3 つを置いていますが、TCP/IP ではこれらの層を明確に区別せず、ネットワークを利用する (ひいてはネットワークサービスの機能を提供する) 各種アプリケーションがこれらの機能を必要に応じて組み合わせて提供しています。これは次のような理由によります。

- TCP/IP の設計時点ではネットワークの各種機能についてあまりよく知られていなかったので、これらの機能を分けなかった。
- セッション層やプレゼンテーション層の機能が必要かどうかは、アプリケーションの種類によって違って来るので、各アプリケーションに任せてしまうのが簡単だった。

このため、以下では各アプリケーションに対応するプロトコルを (セッション層、プレゼンテーション層、アプリケーション層を併せて) 単に「アプリケーションプロトコル」と呼ぶことにします。整理すると、TCP/IP の伝達層より上では、各種のネットワークサービスごとにそれを取り扱うためのアプリケーションプロトコルが用意されている、と考えればよいでしょう。

ネットワークを通じてデータを送受するという機能は伝達層以下がきちんと提供してくれていますから、アプリケーションプロトコルのレベルでは、「ネットワークサービスのために必要な情報をこの順序でやり取りする」という形での約束が中心になります。具体的なネットワークサービスとそのためのプロトコルについては、次回に取り上げていくことにします。

5 まとめ

今回は、コンピュータネットワークの基本的な概念や原理について学びました。普段何気なく利用しているネットワーク機能は膨大な約束ごとやソフトウェア群によって支えられていることがお分かり頂けたかと思います。