Internet Week 97 チュートリアル

インターネット・アーキテクチャ概論
―ブラウザの向こう側はどうなってるの？―

1997 年 12 月 16 日(火)

村井　純 (慶應義塾大学　環境情報学部)

目的

現在、インターネットはアーキテクチャの作り直しという問題に直面しています。これを理解していただくために、インターネット・アーキテクチャに今どんな問題があり、どんな課題を抱え、どう変わろうとしているかをお話します。次に、インターネットの動く仕組み、すなわちインターネットの中での役割分担はどうなり、どういう機能があり、どのように動いているかをお話します。

1.インターネット・アーキテクチャのコンセプトと課題

- ・アーキテクチャと抽象化
- ・ドメイン名に関する問題点
- ・自動車をつなぐ
- ・物理的空間とサイバースペース
- ・新たな要求　　―同期と集約化―
- ・アドレス空間の枯渇
- ・アーキテクチャとは　　―データ・コミュニケーション―

2.インターネット・アーキテクチャの仕組み

- ・OSI 参照 7 層モデル
- ・デジタル情報とその利点
- ・インターネット・アーキテクチャの設計原理
- ・インターネットと鉄道モデルの類似点
- ・経路表のサイズに関する2つの破滅の危機
- ・ドメイン名
- ・IP アドレス
- ・IP ヘッダ
- ・ICMP(Internet Control Message Protocol)
- ・Ping コマンド
- ・Traceroute コマンド
- ・経路制御表
- ・トランスポート層
- ・ポート番号
- ・アーキテクチャ上のバイオレーション
- ・スロー・スタート
- ・モバイル・コンピューティングにおける課題
- ・カットスルーの課題
- ・最後に

1.インターネット･アーキテクチャのコンセプトと課題

･アーキテクチャと抽象化

インターネット･アーキテクチャはどのように作られたかを考える上で、オペレーティングシステムがどのように作られたかを考えます。

unix は 1969 年にできましたが30年間変わらないまま使われています。何故 unix が30年間も生き延びたかというと、その抽象化の概念が大変強力で、かつ簡単で、わかりやすかったからです。

それは何かというと、unix では基本的にはファイルという概念で、バイトの並びがコンピュータの上で取り扱うものの全てであるというものです。ファイルを読むこと、書くこと、探すこと、めくること、そのパイプを作るためにオープンすることクローズすること、操作はこれだけです。世の中全てのものをファイルだと思って(抽象化し)使う、これが、unix を作るとき、デニスリッチ氏が最初に決めたことでした。このため、全てのプログラムは、open、read、write、close で作ればよかった、全てのものにはファイルという名前をつければよかったのです。世界中のオブジェクトを識別するところから始まっています。まずファイルという実態を作り、それでプロテクションの問題が出てきて、呼び方(名前)が決まって、内部で使う識別子ができて(インターネットであれば IP アドレス)、それを鍵に、いろいろなことを作って行きます。このどうやって作って行くかがまさにアーキテクチャです。

･ドメイン名に関する問題点

インターネット･アーキテクチャの中で、あるオブジェクトを区別し、相手を示すために良く使われるのはドメイン名です。

1994 年に IANA(Internet Assigned Number Authority：インターネット上の識別子を割り当て一意性を保証するための組織)は、「インターネットのドメイン名は商標にはならない」という声明を発表しました。これは、たとえば、コンピュータのファイル名にトヨタとつけてもトヨタ自動車の人は困らない、ということです。ドメイン名は商標との関係があるという人が出はじめたのもこの頃で、IANA は敢えて、「ドメイン名はコンピュータの中で使われている単なる識別子だ」と言明したのです。

ところが Internet Trademark Association という国際組織は、「ドメイン名はもう商標として動き始めてしまった」と発表しました。1996 年には International Ad-hoc Committee では.com が商標として非常に過激な競争になり、高価な値段がついたり、すごい使われ方をし始めたと報告があり、Internet Society、Internet Architecture Board、ITU(電話番号の管理などを行っている機関)、およびＷＩＰＯ（World Intellectual Property Organization：知的所有権の国際団体）が、トップレベルドメイン（TLD）の空間をどうするかという国際組織を作り始めました。1997 年には、商標と IP(Intellectual Property：知的所有権)は完全にインターネット上でプロテクトされなければならないということになってきました。

インターネットにアドレスを振り、識別子を付けることでインターネットが正しく動くように設計して来ましたが、インターネットそのものが社会の基盤となって役に立つときには、アーキテクチャ、技術的な要素に加えて、社会の中での意味･価値も考えて、新しいアーキテクチャを設計する必要が出て来ました。ここに大きな曲がり角と新しい挑戦が生まれて来ています。

･自動車をつなぐ

コンピュータでなくても IP アドレスがあれば、インターネットにつながります。阪神大震災のとき、停電でルータが動かなくなりネットワークが切れました。インターネット･テクノロジーはデジタル･テクノロジーで、数字を取り扱うテクノロジーです。これを行う道具としてコンピュータを使いますが、これには電気が必要です。特に災害などで電気の供給が止まったときに、電気があって、われわれを支えることのできる

空間は自動車であり、ここがインターネットにおいて一番重要な場所になるのではないかと阪神大震災の時に気が付きました。

ところが、自動車は、インターネットにつながっていません。これが全てインターネットにつながるべきであると考え、自動車をインターネットにつなげる研究をしています。現在、世界中に約6億台の自動車がありますが、世界中の自動車がインターネットにつながるといろんなことが可能になり、人類にとってものすごく大きな情報と知識の源を手に入れることになります。

たとえば、ワイパーのスイッチと地理情報がモニタできると、皆さんがワイパーのスイッチを入れるだけで、世界中の大変正確な降雨の地図が簡単に書けます。同じことを国がやろうとしたら、大変な時間と巨大な投資が必要になります。降雨量測定センターを各県に作って、なん兆円もの国家予算を使ったとしても、実現は難しいでしょう。インターネットによる情報収集の方法は分散的です。中央のコントロール無く、自律分散的に接続されていて、情報・知識の共有と交換空間を作り出しています。これがインターネット・アーキテクチャの使命で、これにはこの自動車モデルが大変よいのですが、残念ながら、全然つながっていません。

今後自動車をインターネットにつなげる研究で、たとえば6億台の自動車が全部インターネットにつながったとしたら、今のインターネットは動くのか？6億台の自動車に、IP アドレスがふれるのか？それら全てから通信が発生したら、それを処理することができるのか？といった、様々な課題に取り組んでいかなければなりません。

・物理的空間とサイバースペース

他にも考えなければならないアーキテクチャ上の問題があります。インターネットの世界は「サイバー・スペース」と呼ばれる擬似的空間です。サイバー・スペースでは、ニューヨークの人ともパリの人とも自由に話すことができます。物理的な世界では移動しなければなりませんが、距離を完全に意識しない不思議な空間がサイバー・スペースです。

一方、自動車がインターネットにつながり、ワイパーの位置や速度を見ることができれば、渋滞がわかったり、地図では表示されていなくても道があることがわかったりするという、人類が地図を作り始めて以来一度も無かったことが実現できますが、このために必要なのは、正確な経度と緯度と高さという物理的な情報です。

先ほどお話したドメイン名の話も、このサイバー・スペースと物理的な世界の結び付きによって起こる問題です。ある会社名をドメイン名に使おうとしたら、サイバー・スペースであるインターネットの上で使われている識別子は社会の中である権利を持ったものでした。この関係をどうするかという問題が発生しました。すなわち、実際の空間の識別子やアーキテクチャなどの仕組みと、サイバー・スペースのアーキテクチャや仕組みがどう結び付くのかということを考えていかなくてはいけなくなりました。

これらは、物理的空間とサイバー・スペースの強い結びつきを要求して来ています。

・新たな要求　―同期と集約化―

サテライトの利用はアーキテクチャ上大変難しい課題を含んでいますが、インターネットに大変有効な道具として研究開発に取り組んでいます。坂本龍一氏のインターネット・コンサートが平成9年1月23日に横浜で行われましたが、このとき、インターネットの向こう側でみんなが盛り上がっているのが分からず、双方向的ではないといったアーキテクチャ上の問題がありました。

このコンサートでは、坂本氏の弾くピアノの音は楽器の midi データとしてインターネットで送られ、ビデオと音声はストリーム・ワークスとかリアル・ビデオなどでエンコードして送られました。受け側は、ピアノがあるときは、それをコンピュータにつなぐとそのピアノは実際の演奏と同じように演奏され、スクリーンで映

像、オーケストラの音声は映像といっしょにスピーカーから流されました。

映像と音声は、そのまま流すと非常に大変なので「圧縮」して送られますが、圧縮には計算機で処理するため時間がかかり遅れます。一方 midi のデータは、楽器を弾くとすぐ出てきますので、両者の同期の仕組みがアプリケーションのストリームに必要になってきます。この「実時間系の技術」はインターネットの上では大変難しい技術です。

ここでもっとも困ったのは、多くの人が聞きたい上に、ビデオや音声のデータを送るので、インターネットがパンクすることです。

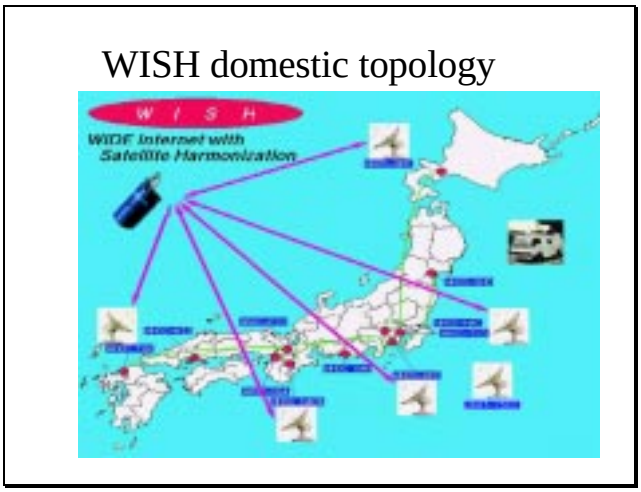
同じデータに対して多くの要求があった場合、できるだけ要求する人の近くまで１つのデータとして持ってきて、近くで分ける技術があればいいのですが、インターネットは1対1の通信を前提にしていますので、皆が要求すると皆がそれぞれデータを取って行きます。これを1本で済むようにできると、インターネット・アーキテクチャのパーフォマンスに非常に貢献できます。

会社などで同じ web を見る場合に、プロキシサーバやキャッシュなどの技術があります。みんなのデータの流れを一緒にしてしまうことを、インターネットでは集約化(aggregation)といいます。できるだけ集約すると性能は上がります。

そこで、こういう共有型のビデオ・ストリームなどを考えた場合、衛星はいくつもの意味で大変重要な貢献をします。アンテナを立てればそこで受信できますので、離島などケーブルが張れないところでも利用できます。

衛星は、一度に皆に伝わる同報性があります。地上にインターネットがあり、同報型の通信が来た場合、これを衛星に打ち上げ、衛星で受け取ってそれからキャリーすれば、効率良く通信できます。しかし、衛星の通信は一方向です。一方、インターネットは双方向を前提として、経路制御など全ての仕組みが作られています。この問題をどう解決するかは重要な課題です。

衛星がうまく使えるとマルチキャスト、同報型のアプリケーションがうまく行きますが、この問題も目鼻がついてきている技術です。



このように、アーキテクチャに対する要求が変わってきた場合、根本的なアーキテクチャが持っていた設計理論も要求にしたがって変わって行かなければなりません。

・アドレス空間の枯渇

世界中の人(現在50数億人いる)がインターネットを使うと50数億個の識別子が必要になります。さらに6億の自動車が全てつながるとすると60億位の識別子が必要になります。

デニスリッチさんは、インターネットにつながるコンピュータは腕時計であると言っています。腕時計がインターネットにつながってコミュニケーションができるわけです。

そうすると、現在のインターネットは約 43 億の番号を付けられる 32 ビットのアドレス空間を持っていますが、これでは世界の人に番号をふることすらできません。32ビットでは限界があるので、次世代IPが設計され、ようやく動き始めています。これも大きなアーキテクチャの変換です。

次世代の IPv6 は 128 ビットのアドレス空間で、これは地上の1cm^2 あたりに 10^{20} 個のコンピュータがあっても対応できます。あるいは、世界中の人が 10^{28} 台のコンピュータを全部それぞれ持っていて、それがインターネットにつながっても対応できる数です。

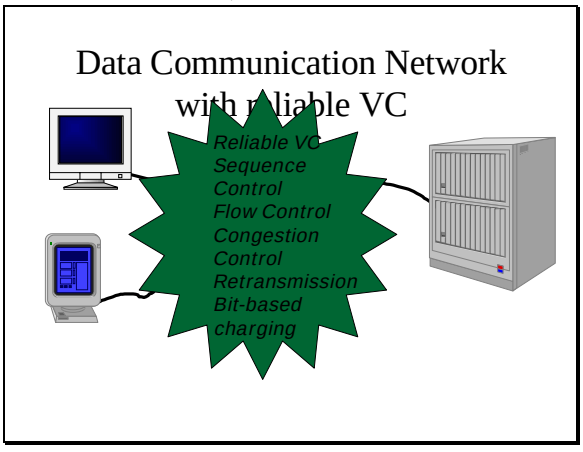
これは識別子の変更で、アーキテクチャが変わりつつあります。

・アーキテクチャとは　―データ・コミュニケーション―

名前が何で、識別子がどうなり、通信はどうなっているか、何がどういう役割分担をしているか、から始まっているのがアーキテクチャのデザインです。また、コミュニケーションの仕組みを作って行く中で、誰がどの役割を果たすのかを決めること、その役割、意味、機能を決めて、その役割分担の関係を考えて行くことです。要求が変われば、役割・機能も変わりますが、これがアーキテクチャの変更です。

インターネット・アーキテクチャの根本概念で最初にわかっていただきたいことは、コンピュータ・コミュニケーション(データ・コミュニケーション)がどういうところから出発しているかということです。

①メインフレームのデータ・コミュニケーション(インターネット以前のコミュニケーション)



メインフレームに多くの遠隔端末がつながることに、データ・コミュニケーションの概念の出発点があります。端末は何もせず、メインフレーム間にパイプ(ストリーム)があって、バーチャル・サーキットを作っていました。ここで求められたことは、信頼性のあるバーチャル・サーキットを作ることでした。信頼性のあるバーチャル・サーキットを作るには、以下のことが必要で非常に重要です。

・データが正しく順番通りに届くことが必要で、番号を付けてデータを送る。

・フローコントロール:混んだら、データを止めるなどしてペースダウンの制御を行ない、混まないようにする。

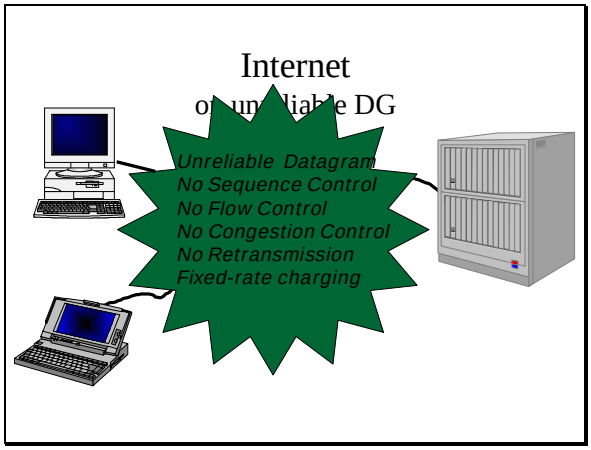
・再転送:本当に届くのを確認するまでそのデータを取っておく必要があり、届かない場合はもう一度送る必要がある。

この仕事の負荷は大きく、大量のメモリなど用意するリソースも大きく、大規模な機械が必要でした。

そのためデータ･コミュニケーションには巨大な資本が必要で、国がコントロールし、規制が存在し、それではじめて作ることができるアーキテクチャでした。

これが信頼性のあるバーチャルサーキットを作るデータ･コミュニケーションです。一方、インターネットは、「信頼性のない(unreliable)データグラムを作るネットワーク」です。

②インターネットのデータ･コミュニケーション



それぞれのコンピュータには計算する能力がありますので、間のネットワークは巨大なリソースが必要になる仕事は一切行ないません。すなわち、届くための努力はするが、うまく届かなかったらごめんねというシステムで、これを「Best Efforts（最善努力）」といいます。

こういったアーキテクチャを作ることがインターネット･アーキテクチャの一番のポイントです。なぜこれが可能かというと、エンドシステムはもはやただの端末ではなくてコンピュータであり、コンピュータネットワークであるからです。このネットワークにつながっているエレメントは、十分な計算能力をもっていると定義でき、次のことが可能となります。

･データを送り、データが届かなかった場合は、その再転送の処理は両端のコンピュータで行ない、網の中では行わない。

･混んでいる場合のペースダウンも網の中ではなく、送り手のコンピュータで行なう。

･データを1，2，3の順で送って、1，3，2と届いてもよい。

･ビデオの 20-30 個のフレームから、1フレームが落ちても影響はないので、その場合はそれを捨てる処理を受け手で行なう。

このようなことが両端のシステムでできると、機能が集中せず、「分散処理」となり役割分担が変わります。すなわち、ネットワークの仕事を全てまわりに振り分け、分担します。

網は、順番の制御、混雑の制御、再転送は行わず、Best Efforts を前提に、インターネットでは最低576 バイトのデータが届けばよい、ほとんどのときには届くようにする、と決めました。すると、インターネットの網は非常に軽くなり、簡単な設備ででき、巨大な資本は不要で、政府の介入は不要となり、規制を作る必要もなく、誰でもできるようになり、あっという間に世界中に広がりました。

網は Best Efforts ですが、アーキテクチャ全体として見た場合は信頼性があります。それは、TCP(Transmission Control Protocol)というプログラムによって可能となっています。

中央にあった仕事をばらまき、周りにうまく役割分担することで、スケールし数が増えても対応できます。

新しいことがインターネットに入ってくるときは、網ではなく、トランスポート層が変われば良いわけです。

インターネットは、web や ftp や電子メイルと思われていますが、本質的には2つの部分から成っています。1つは IP (データであるビットを端から端まで送る)と、もう一つはその上でできているアプリケーションなどです。

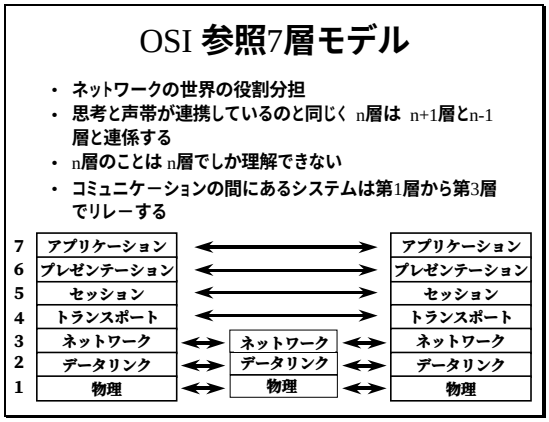
本質的なインターネットのアーキテクチャを考える場合は、そのインターネットのレイヤーの視点で見ます。ビットが端から端まで届く仕組みが分かれば、インターネットの本質的な仕組みはわかります。

2.インターネット・アーキテクチャの仕組み

これから、実際のアーキテクチャの本質の話をします。このアーキテクチャ技術の仕組みは、1970 年代に基盤ができました。

・OSI 参照 7 層モデル

アーキテクチャはコミュニケーションを行なう 2 者の役割をそれぞれ記述したもので、相手の同じ部分との役割分担があります。コミュニケーションの仕組みをコンピュータでどういうふうに作ればいいのかを役割分担の層にしたものが OSI 参照 7 層モデルです。



仕組み及び役割分担としては、両端のコンピュータにはいろいろ乗っていますが、中継には 3 階までしかありません。4 階が難しいことをやっているので、3 階は軽くて済むのです。

コミュニケーションの主体はソース(source)とデスティネーション(destination)と呼び、いろいろな役割があります。web を見ているとき、エンド・システムのコンピュータすなわち皆さんのコンピュータと web サーバは両端のビルディングにあたります。インターネットでは、コミュニケーションは中央の3階までのプロバイダをいくつも中継しながら伝わって行きます。

この役割分担が何であるかをこれから説明します。3 階が IP レイヤーで、この 3 階だけで世界中のどのコンピュータにもつながるようにしています。これが一義的なインターネットの意味です。

その上に TCP 等が乗っていますが、4 階で再転送、信頼性の確保、混雑時の対応を行っています。4 階は、両端すなわち皆さんのコンピュータにしか乗っていません。

これからの話で、皆さんが IP プログラムを書けるようにします。つまり、インターネット・プロトコルの仕組みが全部わかります。そのくらい IP プログラムは簡単です。

トピックス：ビット数の表現を直感的に分かる方法

ビットは 2 のべき乗を意味し、32 ビットは 2 の 32 乗(2^{32})です。

10 ビット(2 の 10 乗)は、10 進数で 1024 です。そこで、10 ビットがおよそ 1000 であると覚えます。10 進数表記で 1000 毎にカンマがありますが、同様に 10 ビット単位にカンマを付けます。

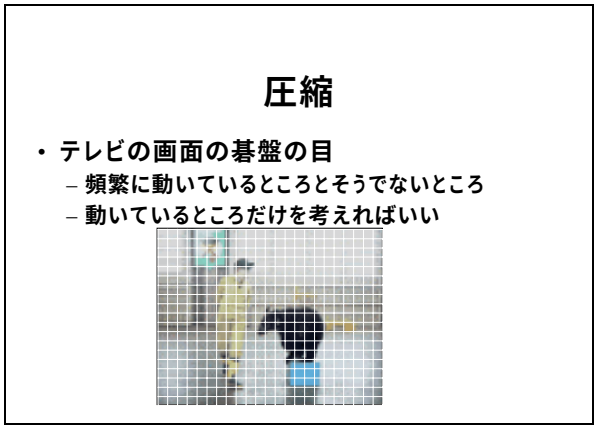
すると、32 ビットはカンマが 3 つ付きます。残りは 2 ビットですから4です。従って、32 ビットはおよそ 4,000,000,000 で約40億とわかります。

$2^{32} = 2^{10} \times 2^{10} \times 2^{10} \times 2^{2}$　　$10^{3\times3} \times 4$　　40 億

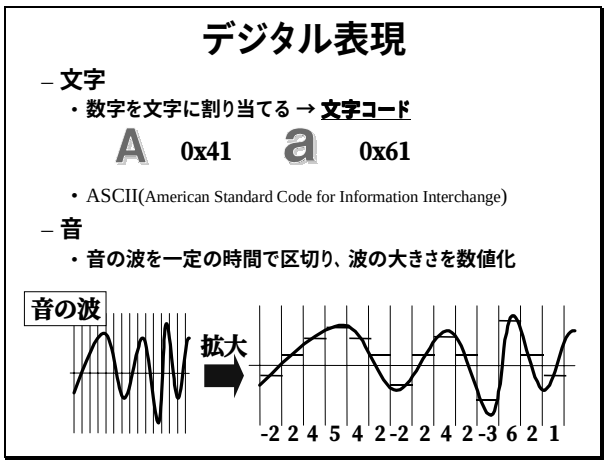
このようにすると2進数のビット数と数値が直感的に結び付くようになります。

・デジタル情報とその利点

インターネットは、デジタル情報のやり取りができる仕組みを作っています。IP でやり取りするのはデジタル情報で、デジタル情報は数字ですから数字で扱えなければなりません。先程のビデオの話で圧縮の話をしましたが、我々がアナログで画面を送るときは、テレビでは 1 秒間に 30 枚の画面を送ります。そのためには、各画面のイメージを 30 枚送る必要があります。



一方、デジタル情報は量子化を行います、すなわち情報を数値に置き換えます。画像では点情報を数値に置き換えます。そうすると、継ぎ目ができて、画像をメッシュで切り換えて考えることができるようになります。動きの無い部分は送らなくても済みます。すると1/100や1/1000に縮みます。このときの鍵は、デジタル情報には継ぎ目があることです。デジタル通信の非常に重要なテクノロジーの根拠はここにあります。



文字を数字(文字コード)で表現し、これでワープロは処理できます。
音も量子化し、数字で表現し扱っています。
このように数字で扱えれば、コンピュータ・ネットワークやインターネットで扱うことができます。

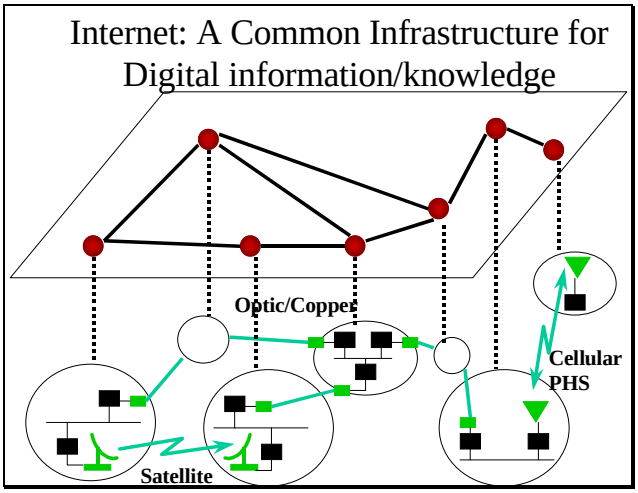
暗号化

- HAL
  - 2001年宇宙の旅に出てきたコンピュータ会社
  - 1997年1月12日設立 :-)
  - H = I - 1, A = B - 1, L = M - 1
  - IBM = {H,A,L} + 1
  - Caesar 暗号

2001 年宇宙の旅という映画の「HAL」というコンピュータ名の根拠は、IBM という名前にあります。その各文字に 1 足したものが HAL で、シーザー(Caesar)暗号と呼ばれるものです。暗号は Caesar の時代からあったものですが、これが非常に複雑な演算であった場合我々は解くことができません。これが暗号の技術で、コミュニケーションのコントロールを行います。

私はお金を払ったあの人には商品を売っていいとか、私はあの人だけにはこの秘密をばらしてもいいといったことを、オープンなインターネットの空間で行うときの暗号化の技術は、この数値の演算、すなわち数学の力を根拠にデジタル・テクノロジーによって作ることができます。

・インターネット・アーキテクチャの設計原理



情報を数値で表現すると、音も文字も映像も全部が数値の列になります。数値のストリームになると、これを取り扱う機械を作っていけばよいという、unix のパラダイムに近いことになりますが、そこにはコミュニケーション上の利点が沢山あります。

そして、誰にでも自由にこの数値の情報をやり取りできるようにするアーキテクチャを作ることこそが、インターネットのアーキテクチャの使命です。従って、先程一義的という表現をしましたが、インターネットのアーキテクチャでは、本質的に、今この絵に出ている平面、すなわち数値のデジタル情報を世界中で

やり取りできる空間を作ることが一番大切な役割です。

そうすると2つのことを考えることができます。つまり、このIPのプレーン、平面の下にサテライト、光ファイバーや無線のネットワークがありますが、このプレーンだけはちゃんとしようということが、インターネットのアーキテクチャが作っていることです。

これは今までのものとだいぶ違います。すなわち、これまでは、光ファイバーがあって、ATMがあって、こういうサービスがあって、こういうビジネスがあります、さあ皆さん使いましょうという、言わば縦のアーキテクチャの作り方でした。

一方、インターネットはこの平面ができればよく、インターネット・アーキテクチャが定義していることはこの平面から上のことだけです。従って、この下にある衛星がマルチキャストに向いていればインターネットはこれを上手に使えばよく、自動車をインターネットにつなぐとき、無線通信が向いているとわかればインターネットはこれを上手に使えばよいのです。

本質的に下は何であれ、それを十分に利用したいが、先ずは IP が通るこのプレーンを作ろう、そしてその上に必要に応じて色々アーキテクチャを作って行くことが、インターネット・アーキテクチャの設計の原理です。これは、メディア・フリー、メディア独立な環境を作ります。

・インターネットと鉄道モデルの類似点

インターネット・アーキテクチャの仕組みを、鉄道モデルで説明します。混み方も含めいろいろなところが似ています。

インターネットと鉄道網の類似点:

・客(データ)は必ず目的駅(コンピュータ)に到達する。

・駅には乗換え駅(ゲートウェイ)と普通の駅がある。

・各路線(ネットワーク)は自律運用(各鉄道会社が個別に運営)、かつ、相互協調(つながっている)。

・選択可能な経路、経路の選択は知的な作業(時間や運賃、サービス、不通など状況に応じ

・経路は選べる)。

・鉄道網の「オーナー」はいない(全体のオーナーはいない)。

OSI参照7層モデルと鉄道モデル

名前	場所	役割	仕事
Application	世界	要求-要求	わがまま
Presentation			
Session	駅付近(切符)	客-客	客-窓口
Transport	改札付近	窓口-窓口	窓口-改札
Network	ホーム上	駅-駅	改札-ホーム
Datalink	「路線」	ホーム-ホーム	ホーム-電車
Physical	線路	車両・線路	車両・線路

例えば、目白駅に子供を届けるとします。駅の改札口にホテルの案内係のような人がいるとしますと、これをセッション・レイヤーといいます。

駅に入って窓口と窓口をつなぐものをトランスポート・レイヤーといいます。コンピュータに口がいっぱいある場合、そのコンピュータの口と相手のコンピュータの口を結びつけるものをトランスポート・レイヤーといい、インターネットではTCPがそれを行います。

その上にプログラムがありますが、そのプログラムとそのコンピュータのポートを結び付ける機能をセッション・レイヤーといいます。プログラムが open、read、write などを行ったときの口とネットワークのコネクティビティであるTCPの口を結びつけるのがセッション・レイヤーの役で、TCPの口を用意しているのがトランスポート・レイヤーです。駅で言えば窓口と窓口の関係にあたります。

改札口の中に入ると、ネットワーク・レイヤーすなわち IP レイヤーのおじさんが待っています。胸には分厚い全国鉄道路線図をぶら下げて待っています。目白に行きたいというタグを付けた子供が改札を入って階段を上がったところにいる、レイヤー3(IP レイヤー)おじさんは、目白に行きたいのなら何々線に乗って何々駅で降りろと言ってホームに送り出します。

送り出されるとそこはデータリンク・レイヤー、すなわち東海道線品川行きとかに乗らされるわけです。データリンク・レイヤーは乗って降りたら着く、というネットワーク・サービスをするもので、LAN などもこれに当たります。Point-to-point ネットワークや ATM ネットワークの一部もそうです。イーサネットはその代表選手です。電車に乗ってその電車の目的地について降りるという、データリンク・レイヤーは一つの路線(たとえば東横線)です。

物理レイヤーは車両にあたりますが、インターネットのアーキテクチャでは何でもかまいません(データリンク・レイヤーも同様)。光ファイバー、同軸ケーブル、ツイステッドペアなどをどう通すか、衛星のチャネルにどう乗せるか等、これを処理するのが物理レイヤーの役割です。

途中、品川駅に着き、乗り換え駅である品川駅の階段を子供は上がって行くと、そこには全国鉄道路線図をぶら下げたネットワークレイヤーおじさんがいて、路線図を見てその子供を山手線のホームに送り出します。これが、乗り換え駅は3階建てであるということです。

IP の仕事は、受け取ったら胸のタグ見て、どこに行きたいかという駅を見て、そこに行くのに自分のところから一番近い乗換駅がどこかを知って、そこに向かって投げ出すこと、これさえやればよいのです。IP のやることは基本的にはこれだけです。

IP アドレスを見て、その IP アドレスがどのネットワークの先にあるかを、経路制御表を見てそのネットワークに投げるだけです。最後に着いたときは、駅で言えば改札口に出す、すなわち、最後の駅では、TCP に渡すのか UDP に渡すのかの判断をしなければなりません。

IP(レイヤー3)の仕事:

①行くときは、経路制御表を見て、IP アドレスでひいて、ゲートウェイがどこか調べて、そのネットワークに投げる。

②来たときは、下からあがってきたら、IP アドレスを見て、自分のところでなければ、行くときと同じことを行い、そうでなければ上にあげる(自分のところかどうかのチェックを行う)。途中の中継ノードの全てがこれをやっているので、インターネットは最後までデータが届きます。

インターネットを OSI 参照モデルに当てはめると、7 つのレイヤーがありますが、中継ノードでは 3 層までしか使いません。ネットワーク・レイヤーには IP があり、トランスポート・レイヤーには TCP があり、セッション・レイヤーは OS の機能です(インターネットの世界で流行っているのはソケットです)。

インターネット・アーキテクチャを考えるときに重要なことは、IP が先ずあって、上へ下へ使えるものをその時に応じて利用し、デジタル情報をやり取りするということです。

IP は中核を果たすプロトコルです。このプレーンを作るためには、駅名をきちんと識別しないといけません、すなわち、どのコンピュータがどの番号を持っているかを識別しなければいけません。そのため、現在のインターネットでは 32 ビットのアドレス空間を使って全てのコンピュータに番号を付けます。例外を除き、2つとして同じ番号を付けてはいけません。

現在、中継専門のコンピュータも出てきましたが、これをルータと呼びます(Windows９５は中継しませんが、NTは中継します)。IPの他の機能に、データの分割があり、これをフラグメンテーションと呼びます。IP がインターネットの基盤を作っていますが、信頼性のない通信を提供するという思想に基づいて作られています。ここが誤解されているところですが、信頼性を作る機能が網全体の外側にあるということで、そういう意味では、インターネットの網そのものに信頼性はありませんが、両端で補われています。

まん中はあまり何もしませんが、両側は仕事を頑張っているという話で重要な点は、まん中は 3 階のところで折り返すためタフな仕事(例えば再送)はしないということです。この役割分担によって数が増えても耐えられるのです。鉄道の例では、横浜駅などの混雑する駅で、ここを通るのにいちいち切符を見せるなどオーバーヘッドが大きいと、鉄道システム全体がものすごく混雑を起こします。という訳で、乗換駅はできるだけ早く通れるほうがよいのです。

インターネットの乗り換え駅での処理は、IP レイヤーしかないので、この仕事が少しでも軽ければ、プログラムで 1 ステップでも少なければ、1 命令少なくてメモリが 1 バイトでも少なければ、数が増えても中継ノードは耐えられます。これでインターネット全体の性能が保証されることになります。インターネット・アーキテクチャを考える上で、中継ノードはできるだけ何もするな、ということは、インターネットにつながっているノードの数がどっと増えても耐えられる大変重要な原理です。中継ノードに対する色々な要求は非常に高価につきます。そこを通るパケット数全てに効いてくるので、パフォーマンスに大きく影響します。また、中継ノードに集中する負荷は予測できません(キューイングモデルでは解けない)、すなわちオープンエンドでどれだけ混むかわかりませんので、一般論として数が増えても耐えられるように作っておくしかない、というわけで、IP の処理を少しでも軽くすることがインターネットの設計の原理です。

RFC791 (RFC:Request For Comments)に IP のスペックが定義されています。

IP レイヤーが行うことは、アドレスを決めること、フォワードする(次のノードに送る)こと、必要に応じた分割をすることです。

分割については、藤沢の授業では次のように説明しています。東海道線で藤沢に着いて江の電に乗り換えるときは、1 車両の定員において、江の電は車両が短く入りきれないので2車両に分けなければなりません。この仕事だけは藤沢駅でやらなければなりません。その代わり、途中でどんなに長い車両にもう一度乗っても 2 度と再び戻すのは止めますが、それは負荷になるからです。この分割は、中継ノードに負荷をかけ、原理と反しますが仕方ありません。イーサーネットで 1500 バイトのフレームで来て、藤沢にきたら仕方ありません。もう一度戻す仕事は、到着駅だけでやることになっています。効率は悪いですが、一度小さいパケットになったら最後までそのままで行きます。(IPv6 はもっと効率が悪く、経路上で一番小さいパケットをはじめからフラグメンテーションしないように作っています。)

・経路表のサイズに関する2つの破滅の危機

経路表のサイズを考えた場合、インターネットが破滅した2つの事件があります。1つは破滅の危機を感じ、1つは本当に破滅しました。

1つ目の危機はアドレスが無くなるということで、アドレスを配るときセグメントに区切って配っていますので無駄が出て、足りなくなりかけました。

もう1つは、経路表の溢れです。IP レイヤーのおじさんは、全国鉄道路線表を持っていますが、横浜駅の IP レイヤーだとすると目白駅に行きたいとき品川に投げ、原宿に行く場合も品川に投げ、大崎に行く場合も同じで、皆同じとなります。これを全部持っていると、世界中の駅の数だけのテーブルを全てのコンピュータは持っていなければならなくなり、これはスケールしません。

スケールするということは、階層的なアーキテクチャをアドレスに付けるということです。山手線の駅に

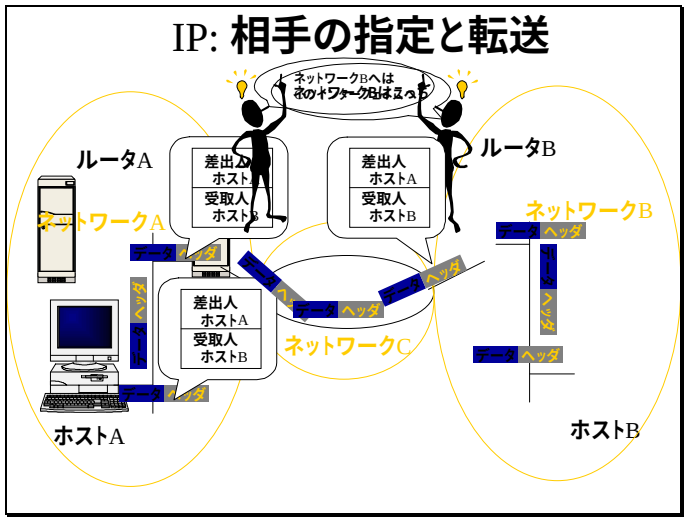
行きたければ、東海道線に乗って品川へ行けという表を作りたいのです。そのためには、アドレスを路線の名前の部分と駅の名前の部分に分けなければなりません。すなわち、ネットワークの部分とホストの部分に分けなければなりません。

そこで、32 ビットのアドレス空間を、構造化されたアドレスで作ります。これは、アドレスの意味を 2 つに分けるということで、2 つに分けた構造化アドレスとは、「ネットワーク・アドレス」と「ホスト・アドレス」で、路線名と駅名から成ります。

ネットワークアドレス部がどれだけの長さを使うのか、この敷居をきっちり区切ると無駄が出ます。自動車のナンバーで説明します。ナンバープレートでどれだけの数が表せるかというと、最初の名前の数は全国の陸運局の数になります(30 とする)、次が 2 桁の番号、次がひらがな 1 文字(50 とする)、4 桁の数字ですので、30x100x50x10000 が識別できる車の数となります。しかし、2 桁とか 4 桁とか区切ったので、隙間がいっぱいできてアドレス空間が有効に使えません。

自動車と違い、インターネットのノードは、倍倍と伸びるので、足りなくなってしまいます。というわけで、間仕切りを変えて対応するしかなくなりました。昔は、クラス A、B、C という固定した間仕切りがありましたが、もうやめました。間仕切りの長さを可変にしました。

あるネットワークのホスト A が、相手を指定して、どこかを中継して行くとき、IP は、差出人は私で、受取人はだれだれです、と示します。その情報がルータに読まれて、ネットワーク B へは C へのインターフェースだということでそのルータに届きます。



ルーティングは経路制御と呼び、それぞれのコンピュータはみんな経路制御表を持っています。Windows では、MS-DOS プロンプトで route –a と入力すると、そのコンピュータがインターネットにつながっているとき、どの経路を持っているかという表をみることができます。これがコンピュータが持っている経路制御表です。どんなコンピュータも必ず持っています。経路制御表には特別な経路があり default と呼びます。これがあるので、スケールの問題を簡単にしています。オフィスのネットワークは世界中のコンピュータのアドレスを経路表として持ちたいため、アドレスの構造でスケールして、一度小さくして持ちます。もっと小さくしたい場合は、自分のネットワークでなければ全部あのゲートウェイへと言ってしまえば、ほとんど一行で済みます。これを default といいます。

・ドメイン名

もう一つのアドレスは、電子メイルや Web の URL のアドレスと呼ばれているものですが、これは厳密にはアドレスとは呼ばず、ドメイン名と呼びます。

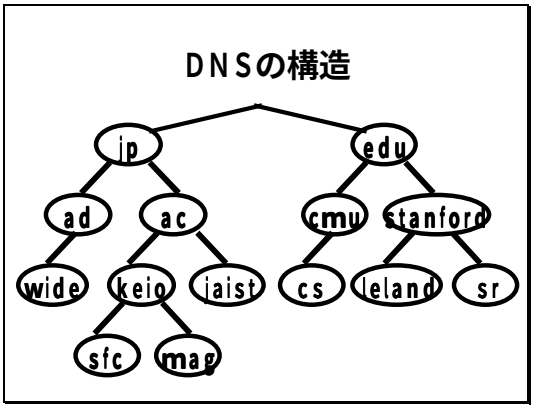
インターネット・アーキテクチャ上で本当にやりたかったことは、単に、人間として使いやすい名前を IP アドレスの数値にマップするということです。ユニークであれば管理できますから、階層的な名前をつけました。ユニークであることを保証するシステマティックなプロセスが何かと言うと、First come、First serve です。最初に登録した人がその名前を使い、他の人がその名前を使おうとしても使えないというものです。ところが、後に、その名前はうちで商標で使っているので困る、という話になってきたのが現在の議論です。

このドメイン名をマップさせる関数を分散的に作っているシステムを「DNS(Domain Name System)」と呼びます。DNS はインターネットの上に適当に分散されて動かしてあります。DNS はすごく素晴らしいシステムで、しばしば「インターネットの奇蹟」と言われます。

分散データベースというのは人類永遠の課題です。個々がデータベースを管理し、それを矛盾なく行うことは、分散データベースの学者や研究者の中で永遠の課題ですが、DNS は新しいコンピュータを登録し数時間が経つとその名前で世界中から IP アドレスが引けます。DNS は素晴らしい分散データベースシステムですが、そのソフトウェアを見ると、何故こんな単純なコードがこんな素晴らしいことをするのかわかりません。新しい DNS の研究をやる人は、よほど勇気のある人です。これだけうまく動いていて、しかもこれがなければインターネットはほとんど動かないのです。単純なので、こうもしたい、ああもしたいと思いますが、それを入れた途端に動かなくなってしまうかもしれないことが恐く、あまりいじれない分野です。

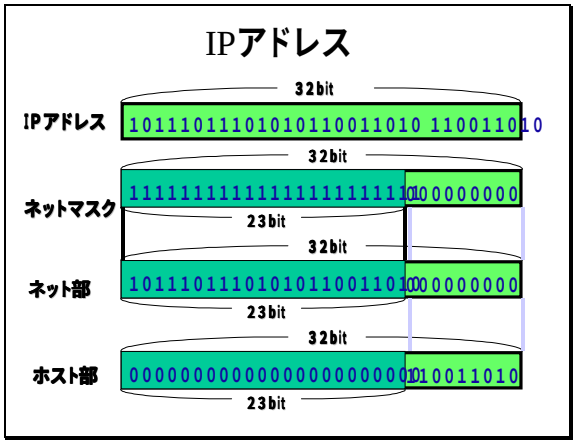
DNS は、ドメイン名を渡すと IP アドレスが返ってくる仕組みを作っています。この名前をわかり易くすればいいというわけです。

電話番号は番号でしたので、知的所有権はありませんでしたが、名前を使うと商標などの問題が発生します。構造に関しては名前も住所も同じことです。絵に描くと一つの機構図を作っているイメージを持つとよく、1つのノードの中で名前がユニークであればよいわけです。



各ノードごとに、keio という同じ名前があっても構いませんんが、tree の上部で一意性が保証されていないと困るというわけです。

・IP アドレス

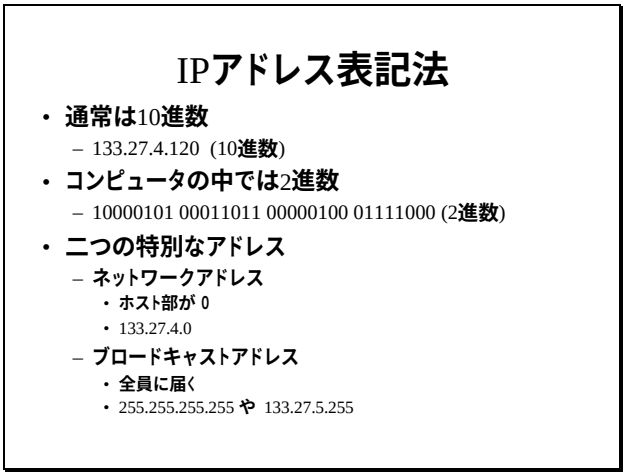


32 ビットの IP アドレスの空間を、鉄道モデルの路線の名前と駅の名前のように分けるわけですが、右半分が緑色の部分がホストの部分で、駅名を表わします。薄い色の部分がネットワーク、路線を表わす部分です。

このパターンがビット・パターンであるわけですが、32 ビットが適当に分けてあります。適当に分けてあるというのは、この敷居線はどこに動いてもいいということです。以前は、8 ビット単位で切れていて、右8ビット目で切れているのがクラス C、中央で切れているのがクラス B、左 8 ビットで切れているのをクラス A といっていましたが、これはもう止めて使用されていません。現在は、アドレスを皆さんが決めるときには、敷居の長さを一緒に表現することにしました。これで、アドレス空間を有効に使えるようになりました。これを CIDR(サイダー)と呼びます。現在はこれだけが、インターネット・アーキテクチャを決めていることです。

Windows95 などで、インターネットのアドレスの設定のところを見るとアドレスと同時にネットマスクの長さを書いて下さい、となっています。長さではなくビットパターンをバイトで区切って 10 進数で書くようになっていたり、/数値と書くものもあります。

ネットマスクを一緒に伝えることでこのアドレス構造を示しているのが今のインターネットです。



IP アドレスの中には 2 つ特別なものがあります。1つは、ホスト部分を全部 0 にしたもので、これはそのネットワークを表わすために使われます。ブロードキャスト・アドレスは、ホスト部分を全部1にしたもので、そのネットワークの機器全部を意味します。特にオールビット 1 は使ってはいけません。ですから、アドレスをコンピュータ 16 台分もらっても 15 台しか使えません。また、そのネットワークを示すアドレスであるこ

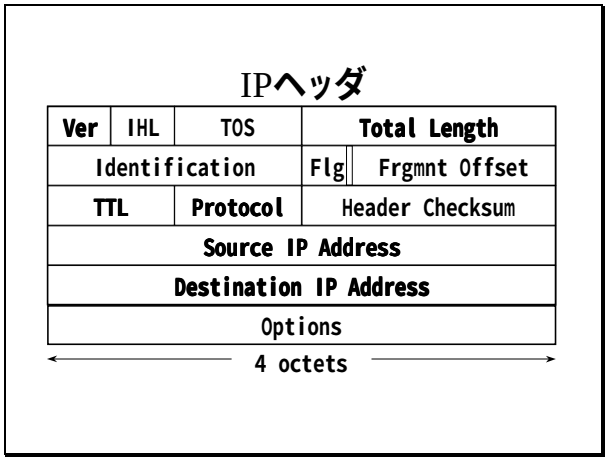
とを意識しているコンピュータを使う場合は、ゼロも使えません。

従って、サブネットで切られた場合は、割当てられたアドレスの 2 つは使わないほうが無難です。

・IP ヘッダ

具合的に皆さんがIP プログラマになっていただくためには、後は、IP ヘッダを理解していただく必要があります。

IP ヘッダは、IP プロトコルの頭に付いているのでヘッダと呼びます。通信の言葉では後ろに付くものをトレーラーといいます。中身はデジタル情報、数値情報ですが、それに付けた荷札のようなものです。この荷札は、以下の形をしています。



この荷札は、横幅は32 ビットで、Options を除くと20 バイトありますが、これだけわかっていただけると、今日から IP プログラマになれます。仕組みは簡単です。

Ver. ：バージョンフィールド。現在は 4(Version4)です。そろそろ Version6 が出回ってきましたので注意が必要ですが、まず間違いなく Version4 です。

IHL(InternetHeaderLength)：ヘッダの長さ、4 オクテットを 1 としてカウント。これも皆さんのプログラムでは考えなくて結構です。

TOS(Type of Service)：配送のタイプ。中継ノードに、中身がビデオであるとか、早く着きたいとか、お金持ちだから特別扱いして欲しいなどを伝えるところで、サービスの種類を示します。これもほとんど使われていないので見なくて結構です。

Total Length ：IP データグラム全体の大きさ、バイト長。

Identification ：もともとのデータを識別する、フラグメンテーションの際に必要。この IP がどうやって識別されているか、ほかの IP と紛らわしいときに区別するために使います。これも使わなくて結構です。

Flg(Flag) ：フラグメントされたか、フラグメントしてはならないか、Frgmnt Offset の処理を行ったときに、ここがフラグメンテーションの先頭かどうかなどを書いておきます。

Fragment Offset：元のデータのどこでフラグメントか、8 オクテット単位で計られます。先程の長い電車から短い電車に乗り換えさせるときには、書き込む必要があります。

TTL(Time To Live)：このデータグラムの寿命で、1 つのホストを経由すると 1 減る。初期値は 64。これも大変重要なインターネットの考え方の一つです。横浜から品川に行ったときに、何かの事故で、山手線が不通で、品川で横浜に行かされることがあった場合、横浜と品川でループが生じます。情報が変わって行くときに経路情報に矛盾が生じ

ることがありますが、その場合、経路が壊れ、ループが発生することがあります。永遠に回っていると、これをゾンビといいますが、インターネットはゾンビだらけになってメルトダウンしてしまいます。そこで、ここを乗り換えをするたびに 1 減らして下さい。そして、1 引いて、これが 0 になったら捨てて下さい。インターネットは Best Effort ですから、データを消して、ごめんね、と言って下さい。このときはほとんど矛盾が起こってループが生じているときです。

Protocol　：上位層は何か。これはとても大切で、IP レイヤーでデータを受け取ったときに、TCP か UDP か、どこに渡していいかを判断するために使われるもので、この中にある IP のデータが上層のどのプロトコルを使っているかを示す ID です。

Header Checksum：ヘッダ部分の検証。データが壊れていないかを調べるときに計算して比較して下さい。比較して違ったら、壊れているといって捨てて下さい。

Source Address：送信元ホストのアドレス(ソース・アドレス)。

Destination Address：送信先ホストのアドレス(デスティネーション・アドレス)。

Option　：特別な機能を提供

荷札はこれだけです。これだけでプログラムとして書けるようになります。もう一度繰り返します。下から上がってきたパケットの、先ずデスティネーション・アドレスを見て、経路制御テーブルと比較する人が横にいますから比較してもらって下さい。そうしますと、このパケットをどこに投げるかを教えてくれますので、そこに向かってただ投げて下さい。それだけです。ただし、投げるときに TTL を 1 つ減らし、0 になったらそのパケットは捨てて下さい。このデスティネーション・アドレスが自分のものかどうかをチェックして下さい。もし、自分のところに着いたものでしたら、このプロトコル番号で示された人に渡してあげて下さい。

フォワードするとき、すなわち乗り換えさせるときには TTL を 1 減らして 0 になったら捨てます。上がって来たら、自分かどうかを調べて、このプロトコルで示したところにあげます。出すときには、寿命が尽きないように TTL を十分用意して、経路制御表を引いて、示すところに投げてやります。基本的にはそれだけです。比較的短いプログラムで書けます。

もう1つやっていただきたいことは、インターネットは Best Effort で皆さんの IP レイヤーとしては責任をもってもらう必要は全然ありません。責任はエンドシステムが取りますから、皆さんは放っておいてくださっていいんです。捨てたいものは捨てて下さい、というのが皆さんに課せられた使命ですが、1 つだけお願いしたいことがあります。皆さんのところで、諦めたり、障害があったら捨ててかまいませんが、送信者に対して「レポート」を送ってくださるとありがたい。無理ならいいですが、できるだけ送って下さい。

・ICMP(Internet Control Message Protocol)

エラーはどういうことで起こるかといいますと、チェックサム(checksum)が違ったり、経路制御表を引いても行先が無いとき発生します。そのときは捨てて下さい。また、TTl が 0 になったら捨てて下さい。その代わり送信者ソース・アドレスに対してデータの報告をして下さい。その報告は IP データグラムで送りますが、別の名前がついています。ICMP(Internet Control Message Protocol) と呼びます。

IP とエラー

エラー：データグラムの紛失、チェックサムの不一致、フラグメントしたデータの一部欠落。

エラー訂正：IP はエラーの訂正は行わない。

エラー報告：送信者にエラーの報告を行う、送信者はアプリケーションにエラーを報告する。

Windows では、MS-DOS プロンプトで netstat –s と入力して下さい。そうすると皆さんのコンピュータが受信した ICMP エラー報告のリストが出てきます。Windows すなわちエンドシステムはほとんど何もこれを使っていません。見るのは皆さんだけです。見たければ見て下さい。自分の出したパケットがどこでどんな障害にあって戻って来たかがわかります。統計情報が残っていますから見て下さい。送信先に着かなかった場合に、この ICMP が返すことになります。これを利用しているのが Ping です。

ICMP(Internet Control Message Protocol)

・IP が送信者にエラーを知らせる機能：送信先への到達不能、送信元へのフロー制御、経路変更、TTL＝0 の時 パケットを破棄。

・送信先への到達可能テストとその状態：ping コマンド

・Ping コマンド

ping の例

```
[9:27am]shonan~(@_@)ping altavista.digital.com
PING altavista.digital.com (204.123.2.66): 56 data bytes
64 bytes from 204.123.2.66    icmp_seq=0 time=224 ms
64 bytes from 204.123.2.66    icmp_seq=1 time=223 ms
64 bytes from 204.123.2.66    icmp_seq=2 time=225 ms
64 bytes from 204.123.2.66    icmp_seq=3 time=229 ms
^C

----altavista.digital.com PING Statistics----
4 packets transmitted, 4 packets received, 0% packet loss
round-trip (ms)  min/avg/max = 223/225/229
[9:27am]shonan~(-_-)
```

ping の動作

- ICMP Echo Request と Echo Reply
  - Echo Requestを受けたホストはEcho Replyを返す
- パケットを出すと同時にタイマをスタート
- TTLは255（最大）に設定
- それぞれのメッセージにIDが振る
- 同IDのパケットを受け取ったらタイマをストップ
- 時間とTTLを表示

この Pingコマンドが行うことは、基本的にはあるアドレスにデータを投げますがそのときストップウォッチを押します、Echo Reply という機能が ICMP の中にあり、IP に行って ICMP が戻ってきますから、それをタイマーで計ります。投げたデータの sequence 番号を増やしていき、そのときの時間を調べます。CTRL-C で割り込んで止めます。

この例は、altavista に4パケット投げて、4パケットちゃんと戻ってきて、パケットが戻って来なかったのは 0%で、最小/平均/最大時間(ms)が 223/225/229 であることがわかります。これも Windows95 にあり、MS-DOS プロンプトで動きます。他にフリー・ソフトも沢山あります。

これで、ネットワークの距離を測ることができます。光の速度は、だいたい地球を百何ミリ秒で回ります

が、藤沢から Altavista(DEC)のあるニューハンプシャーまで行って返ってくるのに、片道 100ms 強で行っていますから、ほとんどロスがありません。

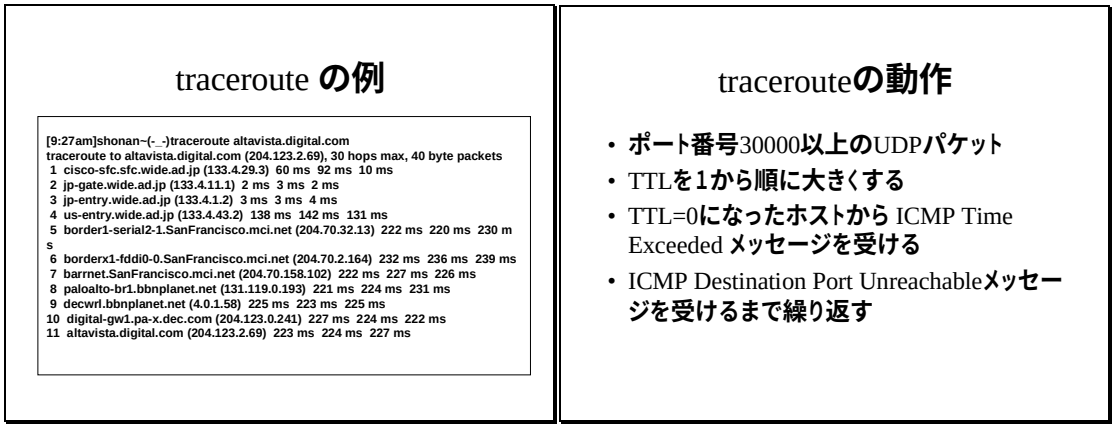
Ping は、ICMP の Echo request と Echo reply を出して行います。

・Traceroute コマンド

もう1つこの ICMP を使ったおもしろい悪戯をお話します。

インターネットは世界中のどこへ行くにしても、全部ホップ・バイ・ホップ、すなわち乗換駅毎に経路を計算されます。これをダイナミック・ルーティングと呼びます。そうすると毎回どこを通るかはわかりません。インターネットを使っていて、どの経路を使って、どの ISP を使って、どのノードを通って、目的地に到達しているかを知る方法は基本的にはないんです。しかし、ICMP にこんな仕組みがあるんなら、到達のその経路を知ることができるかもしれないぞ！と思った人がいます。Van Jacobson という素晴らしい人なんですが、これで一度、ガタガタガタとインターネットが壊れたことがあります。ICMP をちゃんと作っていなかったのが原因です。

これが何をやっているかといいますと、ここに書いていることをやるんです、簡単です。



まず、普通は TTL は 100 とか 200 とか 255 で出します、そうすると目的地に着きます。乗換えるごとに 1 減らし、0 になったら捨てます。0 になったら捨てて、そして送信先に、あなたのパケットはここで寿命が尽きて死んだというレポートを ICMP で返します。

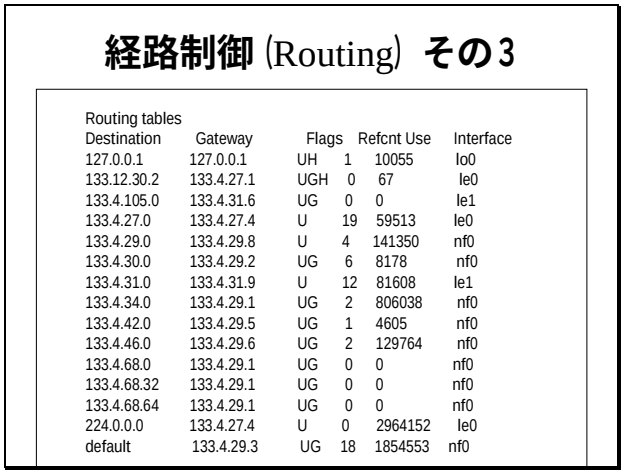
遠くの目的地に対して TTL を 1 で出します。すると 1 つ目のノードで寿命が尽きたとレポートが戻ってきます。次に TTL を 2 にすると、2 個目でレポートが戻ってきます。次に 3 にします...これを到達するまで繰り返してやり、この ICMP のレポートを集めると、到達経路がだいたいわかります。TTL はこのためにあったのではなく、ループを防ぐために、ゾンビを防ぐために作ったのですが、これを利用して、永遠の謎であったダイナミック・ルーティングをあからさまに見せるプログラムを作ってしまいました、これが Traceroute です。

これは、どこを通って来るか、どこで時間をくっているか、などがわかります。上の例では、慶応の藤沢を出て、国際線に乗ります。3 番の日本のノードから 4 番のアメリカのノードで太平洋を渡りますが、ここで 100ms かかっていることがわかります。最後に 200ms かかりますが、アメリカ国内で 100ms かかっていることがわかります。ここも何とかして欲しいですね。

ということが分析でわかってしまいます。このコマンドはこの仕組みでできていることがわかっていただければ結構です。

このコマンドは、unix では traceroute ですが、Windows では tracert でやはり MS-DOS プロンプトの中にあります。

・経路制御表



これは、unix のルーティングテーブルの例です。Windows では rout –a と入力すれば見れます。

Destination と Gateway のアドレスがありますが、この目的地に行くのならこの乗換駅に行きなさい、というわけです。そして、Interface すなわちこのホームに降りなさい(例えば、イーサネットの1番だとか、FDDI のポートの0番に落ちなさい)とかが書いてあるわけです。

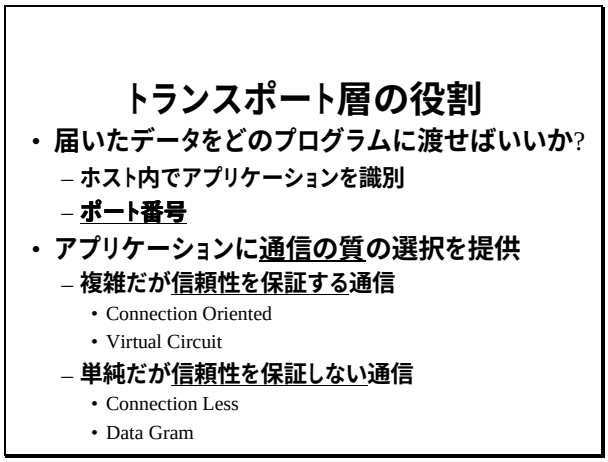
default は、0.0.0.0 という IP アドレスになっています。これでどんなパケットも行先が決まったことになります。

これがルーティングテーブルの正体で、普通はそう大きくなくて済みます。バックボーンのちゃんとしたところは、世界中のいろいろなアドレスを持つ必要がありますが、そこはまた別の仕組みでルーティング情報を交換できるようになっており、結果としてインターネットは世界中のルーティングの交換をしているわけです。

・トランスポート層

インターネットはまず、この IP が動けば、インターネットの世界はできあがります。そして、ここにアーキテクチャ上の特徴があります。インターネットが動くための残りは、インターネット層以上のところで、誰が何をすればいいのかということです。

今までのところで、デジタル情報は全てのコンピュータの間をやり取りできる、というところまでお話ししました。行った先で、これからどうすればよいかということが、トランスポート層です。



トランスポート層を考える上で、IP のアーキテクチャを考えたとき、先ず、識別子である IP アドレスから

はじまったことを思い出して下さい。アーキテクチャを設計するときには、相手をちゃんと指示できる仕組みを決めます。次に、その人と話す仕組みで、この 2 段階で決めます。

そうすると、トランスポート層では、コンピュータの上の複数の口が、それぞれのパートナーと結び付くようにします。つまり、渋谷駅の 25 番ポートとか横浜駅の 18 番ポートと、この 2 つを結びつけます、これを行うのがトランスポート層の役割です。

トランスポート層が、先ずやらなければいけないのは、ポート番号を決めることです。1つのコンピュータの中で番号を決めます。

そして、そのポートとポートがつながり、その通信をサポートするときに忘れてはならないのは、IP レイヤーではやめた信頼性の確保です。再転送、エラーチェック、混雑時のコントロール、ペースダウンとかコンディションコントロール、フローコントロールなどをやってくれないと困ります。

ポートとポートの間で通信をするときに信頼性は要らないという通信があることも忘れてはいけません。ビデオ・フレームなどがそうで、どこかで 1 フレームが落ちても構いません。これには IP がそのまま使えますが相手先がコンピュータでなく、ポート番号であるものも必要です。

これがトランスポート層の 2 つの役割です。1つが TCP で、これが信頼性のあるストリームを作ります。もう1つは IP そのままでいいですというもので、これを UDP(User Datagram Protocol)と呼びます。

UDP はポート番号を相手にしますが、中身は IP そのままです。IP は IP アドレスを使いましたが、UDP はポート番号と IP アドレスの組で相手を識別します。

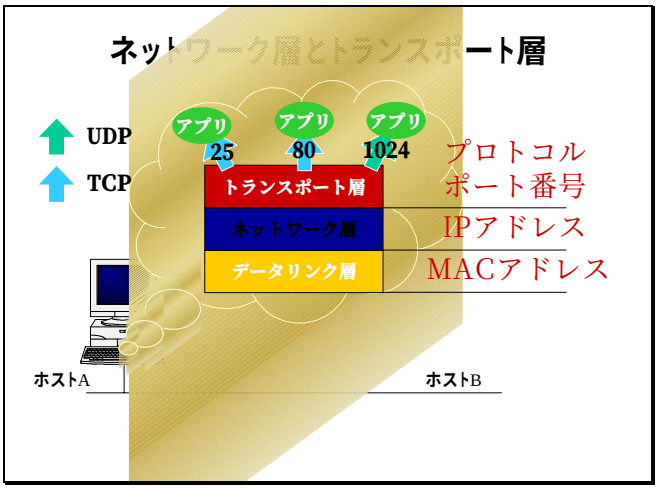
TCP は、ポート番号と IP アドレスの組でパイプの先を指定しますが、そのパイプは、信頼性があり、再転送などの面倒をみるパイプのストリームを作ります。これで、はじめて、アーキテクチャの最初のところで話した、信頼性のある、あたかも物理的につながっているような口であるバーチャル・サーキットを TCP は作ります。しかし、エンドシステムの両端だけで作っており、インターネットの網全体は何もやっていない、というわけです。

・ポート番号

ポート番号は、プロトコルごとに決まっています。何をつけても構わないわけですが、約束が1つ決めてあります。即ち、いくつかのポート番号は世界中でこのために使うということが決まっています。インターネット上のアプリケーションでは、決まったポート番号があります。例えば、電子メールのプロトコルをしゃべるプログラムは 25 番のポートで待っていて下さい、www のサーバーのプログラムは 80 番のポートで待っていて下さい、という約束が決めてあります。

従って、アプリケーション・プログラムを作る場合は、例えばブラウザを作るとします。www.a.xxx というURL を見ますと、URL をユーザから与えられますからそれからドメイン名がわかります、それから IP アドレスを引きます。http:と書いてあると 80 番の口で待っているという約束になっていることが表を見ればわかり、80 番と IP アドレスの組がわかりますから、そこで TCP のパイプを張ります。張ったところで get と叫べば、どっとホームページが流れてきます。そうしたら、そのホームページのルールに従ってユーザのために画面に絵を描いてあげて下さい。これでブラウザを作れます。

これが、トランスポート層で番号をつけて行っていることです。もう、アプリケーションのプログラムの説明まできました。



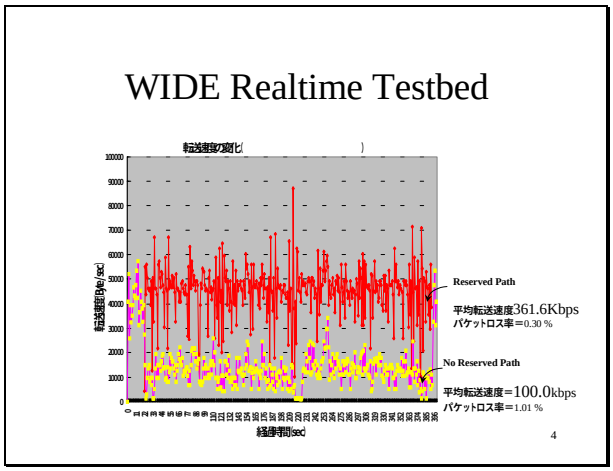
アドレスのことをまとめますと、データリンク層のイーサーネットとかにあるアドレスの MAC アドレス、ネットワーク層の IP アドレス、トランスポート層のプロトコルの種類とポート番号の空間を作っています。ただし、トランスポート層で相手を指定するときには、これに加えて IP アドレスも利用している、というところもアーキテクチャ上の大きな特徴です。

いろいろなアプリケーションは、その決められた番号を使って通信をするということがインターネット全体で行われているわけです。

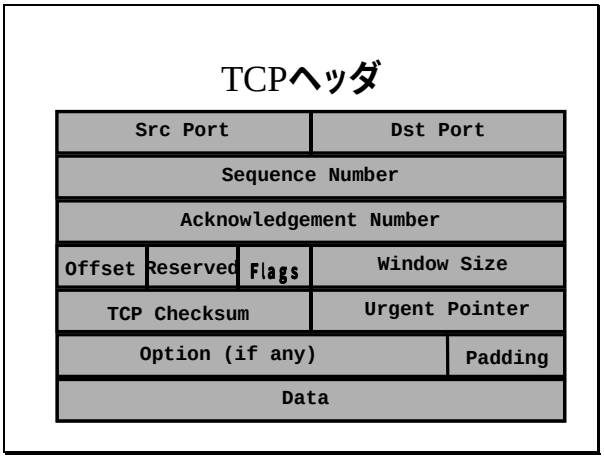
アーキテクチャ上のバイオレーション(violation)

そうしますと、この 2 つの関係の切り分け方法がいくつかあると思いますが、まず、切り分けとしては先ほど申し上げたようなことが原則です。しかし、このレイヤーの役割を犯しているメカニズムで、インターネットの中で有効に使われていることがいくつかあります。

その代表的なことで、おかしいぞ、矛盾が起こっているぞ、と思われることの1番目はファイアウォールです。NAT もそうです。ファイアウォールの目的は、信用できるネットワークと信用できないネットワークを切り分けることです。ここで何をしているかといいますと、通信を切っているわけですが、そのときに何を見るかといいますと、基本的には IP アドレスで切るのはルータで、レイヤー3 ですから IP アドレスしか見きれません。従って、IP アドレスをフィルタリングします。すなわち、あそこから来るパケットは通さないぞ、そういうことはできますが、それ以上のことはここではできません。しかし、普通のルータともう既に違う仕事をやっています。最近のファイアウォール(これは商品ですが)でやっていることは、web だけは通した方がいい、電子メイルだけは通した方がいい、ということをやりたいがために、ここでは盗み見をしています。ルーターにも関わらず IP だけではなく IP の中身を覗いて、TCP のポート番号を見て、それによってコントロールしているケースもあります。ここのところが、アーキテクチャから見たときのバイオレーションです。



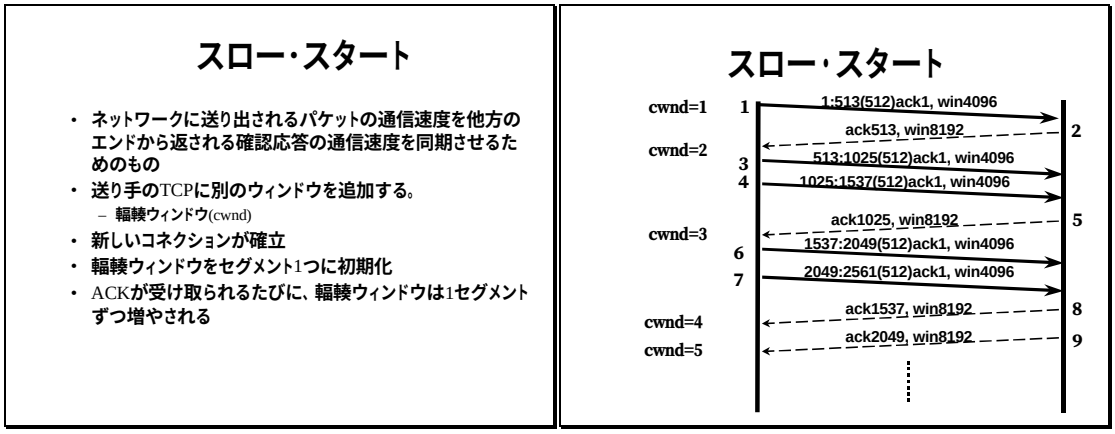
今、見ていただいているのは、RSVP というリアル・タイム通信をするための新しいアーキテクチャに対する試みの実験をしたときのデータです。見ていただきたいのは、ここがそのリザベーションをしているときの通信で、そうでない場合はものすごく他の通信に邪魔されて、通信ができていないというのが黄色い部分です。何をやっているかというと、これはまだ実験段階で世の中のインターネットには入っていませんが、中継ノードに対して通る資源の予約をしてしまうんです。言い換えれば品川駅は混むので横浜駅を出て行くときに、品川駅に、「おい！今から大事なビデオ・ストリームが通るので場所を取っておけ」ということをいいますと、品川駅では慌てて場所を通路を確保します。すると混んでいてもビデオ・ストリームは遅延なく送れるようになります。これがリソース・リザベーションというテクノロジーです。これをやりますと、やはりこのレイヤーのアーキテクチャを超えまして、ルーターが特別な処理をすることになりますので、先ほど言っていたことと矛盾した行動をルーターに強いることになりますが、インターネットでそれができなければ実時間処理をするのが大変難しくなる場面があります。そのTCPがそのコントロールをするためには、あるヘッダがありますが、ここは見ていただける様に、いろいろな番号の中でソース・ポートとデスティネーション・ポートというポート番号を持ったヘッダが付いています。



TCP のそれぞれのメカニズム、信頼性をどうやって作るか、という話をしていると時間がなくなるので飛ばします。

TCP で理解していただきたい仕組みの中で、皆さんがインターネットを使っていて大変重要なことがありますのでこれだけを次に話します。

・スロー・スタート



インターネットがアーキテクチャ上で混まないようにコントロールするこの仕組みはとても大切で、TCPの役割です。

混んでいたら皆ペースダウンして欲しい、そうでないと何が起こるかといいますと、人間は遅いなといって何度も繰り返します、そうするとどんどん混んでしまいます、そうすると２度と再びリカバーできなくなり、本当のメルトダウンとなります。従って、これを避ける仕組みをインターネットは持っている必要があり、これを TCP に入れてあります。

簡単な原理はこうです。まず原則を決めます、インターネットは混んでいる、あるいは混みやすい、オープンエンドで中継ノードを追加するのですから当たり前です。それで、もし混んでいることがわかったら、すぐデータを出すのを止めるだけではなく、5 歩ぐらい下がってうな垂れて、もう一度出直してきて欲しいくらいの気持ちがあるわけです。これをスロー・スタートといいます。

もし、混雑を検知したら、ものすごくナイーブに、小さなデータから出直して下さい。大きなデータを送ることを、ウィンドウサイズといい、ウィンドウ・コントロールというメカニズムで TCP はできています。大きいデータをどかんと送ると困りますが、小さいデータを少しずつ送るとリラックスできます。従って、大きいデータを送ってうまくいかなかったら、すぐ小さいデータに切り替えて下さいよ、ということがこの引き下がって下さいということのポイントです。

１番目のポイントは混雑の検知で、何をもって混雑だと思うかです。皆さんがTCPで送ったら今度は返事を反対側のシステムからもらいます。その返事が遅れて来たら混んだと思って下さい。返事を ACK といいますが、送ったデータの確認が遅れて来たら、混んでると思って、送るデータの量を少なくし、ペースダウンして下さい。これをスロー・スタートといいます。

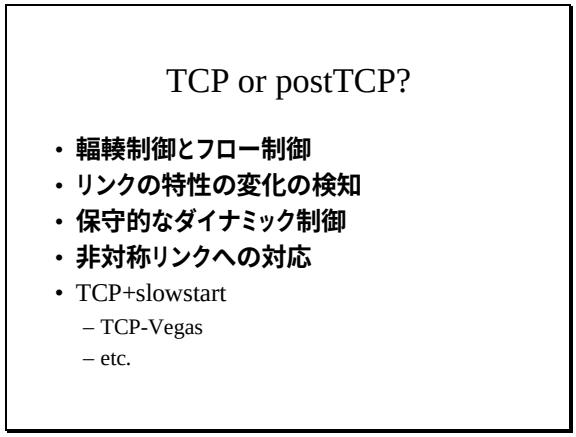
これがあるので、インターネットはうまく行きます。インターネットが混み始めると、インターネット中のエンドノードが混んだといいながら、出す量を止めるわけです。そうすると、だんだん混雑が緩和して、また動くようになります。この次に出すときは、スロー・スタートという位ですから大変なんです。ちょっと出して OK なら、もうちょっと大きくして OK なら、それよりもうちょっと大きくして、下からずるずるずるずるとゆっくり上がって来てもらいます。これがスロー・スタートのアルゴリズムというものです。これができたので、インターネットはこれだけネットワークが増えても保たれている訳です。インターネットがずーっと増えてきたエッジのところでこのアルゴリズムが紹介され、それで使われるようになりました。

実は、これは、今は少し難しい局面も持っています。携帯電話でインターネットを使っていて、トンネルに入るとします。トンネルを出てまたつながりますが、切れるので返事が遅れます。すると、ものすごく小さいデータを送らないといけないと思って、トンネルの前と同じ通信路を持っているのに、小さなデータしか送らないんです。ゆっくり伸びて行きますから、無駄が出ます。

衛星もそうです。宇宙まで行って降りてきますから、遅延がすごく大きくなります。太い線の衛星を持っ

ても、そこで TCP の通信をしますと、返事が返ってくるとき衛星回りで返ってくるので、遅延が大になります。遅れがあると思ってそこの経路の TCP は、小さなデータしか流しませんので、太い衛星の通信を持っていても、TCP の通信はこの太いパイプを埋めることができません。

スロー・スタートのアルゴリズムはインターネットを本当に助けた、アーキテクチャにとって非常に重要なアルゴリズムですが、今、ATM が日米間に通り、衛星で太い線が使え、ワイヤレスで時々通信が切れて、すぐ元の太さに戻るような通信ができた現在、このTCPはもっといい方法があるのではないか、ということで、今、アーキテクチャで大変大きな課題になっています。



これが、その問題です。輻輳制御、フロー制御、リンクの特性がどうやって感じられるのか。

これをどう考えて行くのかというのがアーキテクチャ上の重要な問題になっています。

・モバイル・コンピューティングにおける問題点

もう1つ、重要な問題があります。今、アーキテクチャを見てきたときに、識別子という問題で3つの話をしました。1 つはドメイン名のようなアプリケーション、我々がインターネットを利用しているときに相手が何であるかを識別する識別子です。次に TCP、IP アドレスとポート番号の組で相手のコンピュータのあるプログラムの口を識別しようということです。それから、IP アドレス、これはコンピュータを識別するために付けた番号です。これらがインターネットのアーキテクチャです。このインターネット・アーキテクチャには大きな問題があります。

ドメイン名は対象を識別するために使われています。IP アドレスも対象(コンピュータ)を識別するために使われています。ポート番号も対象を識別するために使われています(例:80 番はweb ポート、25 番は電子メイルのポート)。相手を識別するために使うのか、送るために使うのか、では目的が違います。これがアーキテクチャ上で、モバイル・コンピューティングに非常に難しい問題を生みはじめています。モバイル・コンピューティングにおいて「A さんのコンピュータ」という識別は、A さんがネットワーク上のアメリカのネットワークにつながろうと、他の会社のネットワークにつながろうと、「A さんのコンピュータ」として識別したいのです。というのは、識別にはセキュリティと認証がからんでいるからです。

「Aさんのコンピュータ」に課金をしたければ、Aさんと一緒に動くコンピュータに関してこれを行いたい。しかし、A さんのコンピュータは動いていますから、どうやってそこへデータを運ぶかという経路制御は別の概念です。ところが、ここで同じ経路制御を使っている訳ですが、これが今、モバイル・コンピューティングの上で考えたときの新しいアーキテクチャに対する課題になります。

・カットスルーの課題

TCP の話は先ほどしましたが、もう1つあります。中継ノードをどうするかという、アーキテクチャ上の全く新しい課題があります。ATM の網が沢山世の中にあったとしますと、中継ノードでは、通信は本来は中継ノードに上がって、レイヤー3 ですから、そこでルーティングをやって下りて来て、そしてまた ATM で行きます。ところが、インターネットの上ではこういう中継をすることになっていますが、いつもこうやって回って行くのは面倒くさいので、このように通るものはいつも ATM でひょんと取ってしまえ、と IP レイヤーが下の ATM のレイヤーに言うことができるわけです。これをカットスルーといいます。

これは、今、イプシロンの IP スイッチやシスコのタグ・スイッチで使われている技術ですが、これはもう商品です。商品ですが、まだ、このアーキテクチャを一般的にどうすればいいかということは決まっていません。

例えば、ここで TTL を 1 引くべきですが、カットスルーしたとき、どうするか？ATM だからぴゅーっと下に行きますが、インターネットの上ではここを通ったことになります。イプシロンではこの先の着いたところで2引きます。ところが、東芝の CSR という商品ではこのノードでは引きません。商品はありますが、まだアーキテクチャ上の決定ができていません。しかし、重要な問題です。これができると高速になります。

・最後に

インターネットにとって重要なことは、IP のレイヤーで賢くなったら、それはできるだけ使いたいということです。しかし、アーキテクチャ上の根本的な新しいアイデアが出てきます。インターネットというのは、まだ1つのインターネット・プロトコルです(これは今、v4 から v6 になりますが)。そして、多数のプラットフォーム、web のプラットフォーム、色んなアプリケーションがあり、色んな交換技術や無限の媒体があります。ここには色んな競争があって、新しく良くなればいいんじゃないでしょうか。ただし、デジタル情報だけは、とにかく、安く、いつでも、普遍的に、自由に地球上で必ず作っていきましょう。これが、インターネットのアーキテクチャの大変重要な部分だと思います。

Dave Clark というインターネットのアーキテクチャを作っている大変重要な学者がいます。この人が INET 神戸で言った重要な言葉があります。「標準化は投票で決めたり、誰か一人で決めたりしますが、そうではなく、だいたいラフなコンセンサスと実際に動いているものでこの技術を作っていけばよい」と言っています。

インターネットのアーキテクチャは、今、申し上げたように、まだ決まっていない ATM の取り扱いや、まだ決まっていないモバイル・ノードが実際にもう動いて、それ自身がインターネット・アーキテクチャの一部なのです。アドレスのアーキテクチャを説明するときクラス A、クラス B、クラス C と説明していたのはつい昨年までです。しかし、今はネット・マスクの問題だけだという説明の仕方をしました。このように、基本的にはインターネット・アーキテクチャは大変いろいろなところが変わってきます。

今日、説明できなかったアプリケーションのアーキテクチャに感心のある方がいらっしゃると思いますが、これもそれぞれの意味で毎日毎日変わって行くものです。それを評価して、ランニング・コード、本当に貢献したもので評価をして新しいものを作っていくというところがインターネット・アーキテクチャの大変重要な使命ではないかと思います。

そして、その自律分散的な方法と、なぜインターネットがこれだけのスケールで動いているかということをわかっていただければと思ってお話をしました。

以上