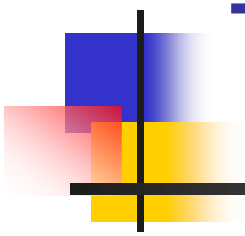
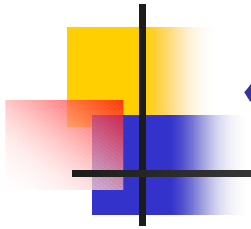


TCP/IPについてあれこれ

通信とは？

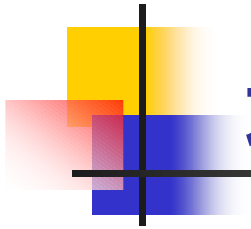
TCP/IPとは？

TCP/IPを使ってみよう



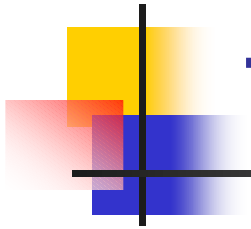
使いまわしでごめん

- 一年前に第三技術部(現Zope部+ナレッジソリューション部)でやった内容をほぼそのまま使いまわしてます。
- しかも削ってます。
- 聞いたことある人は新しいことはないから帰ってもいいよ。



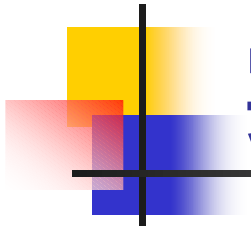
通信とは？

- オブジェクトとオブジェクトが協調動作を行なう手段。
- 情報を交換する。
- なんらかの目的を持っている。◎



プロトコルとは？

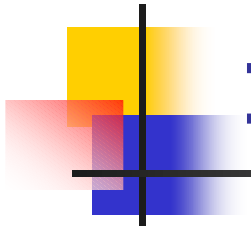
- 通信に関する約束事
- 扱う情報や手続きを規定



実社会での例(宴会プロトコル)

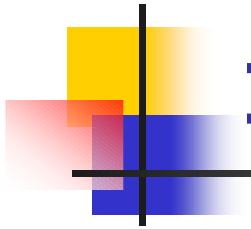
- 交換する情報:
  - 幹事、場所、面子、時間
- 目的:
  - 疲れた体と心のリフレッシュ

宴会プロトコルの下にあるプロトコル(その1)



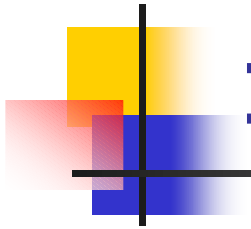
- 会話手順
- 日本人なら日本語で話すよね
- 言葉は音で伝わる
- 音は空気がないと

宴会プロトコルの下にあるプロトコル(その2)

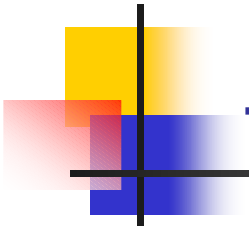


- 会話手順
- イタリア人はイタリア語を使う。
- 電話とかを使ってみる
- 電話に対しては普通にしゃべる
- 電話は電気信号を使う
- 電気信号は電気を使う
- 電気はケーブルを
- 電気っていうのはそもそも電子の流れか
- 交換機なんかも重要
- 電話機なんかも当然重要

宴会プロトコルの下にあるプロトコル(その3)

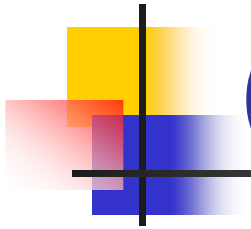


- 広報という手段もあったり
- 広報用フォーマットみたいなものは暗黙知(プロトコル)。
- 貼り紙で広報する場合は文字や地図を書いておく。
- 目で見るためには光がないと



宴会プロトコロルから考察

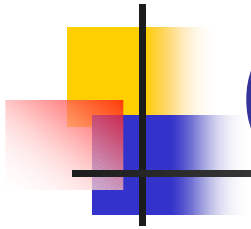
- 真面目にプロトコルを定義するのは大変。
- でも良く見るとプロトコルは普通階層構造を持っている。
- 階層構造の一つ一つぐらいなら比較的楽に定義できる。
- 階層構造を組み合わせることにより柔軟な運用が可能。



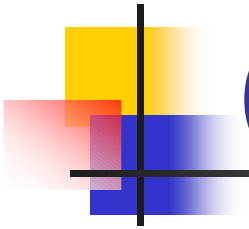
インターネットプロトコルの場合(メールを送る場合)

- SMTP
- TCP
- IP
- Ethenet, PPP
- Cat5, 電話線等

インターネットプロトコルの場合 (Web を見る場合)

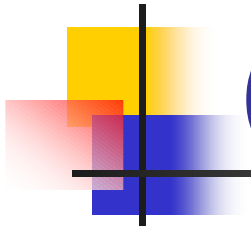


- HTTP
- TCP
- IP
- Ethenet, PPP
- Cat5, 電話線等



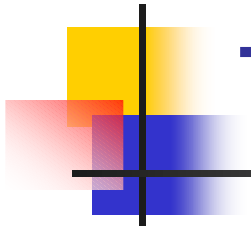
インターネットプロトコルの場合(ファイルの転送)

- ftp
- TCP
- IP



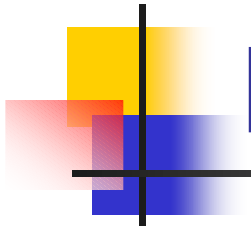
インターネットプロトコルの場合(ファイルの共有)

- SMB
- TCP
- IP



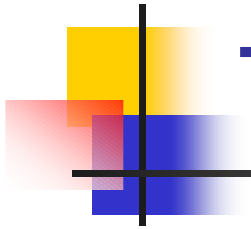
TCP/IPはなぜ流行ったか？

- 柔軟性が高い(独自プロトコルより有利)
- オープンな規格(独自プロトコルより有利)
- ちゃんと動く(OSI との違い)
- スケーラビリティがあった(IPX,NetBEUI に勝った理由)
- プログラムがそこそこ書きやすい
- いろいろなものが動く



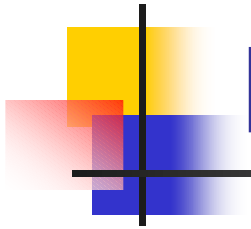
Break!

- ここまで質問あるかしら？



TCP/IPって何？

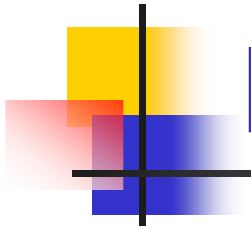
- こういう根本的な疑問を解決するには、
  - 本を読む
  - 人に聞く
  - 原典にあたる --> RFC



RFCって？

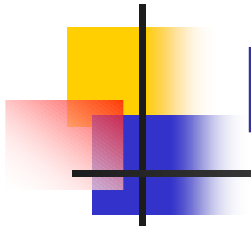
- インターネットでの通信の規約。
- 約束事にすぎない。
- 紳士協定。
- 拘束力なし。

- ftp://ftp.ietf.org/rfc/ が原典
- 普通は Ring サーバから拾う
- 会社でミラーしてあると便利 --> 誰かやらない？



RFCから情報を探す

- まずは rfc**00.txt を見る
  - Internet Official Protocol Standards」
- IP --> rfc791
- TCP --> rfc793
- UDP --> rfc768
- ICMP --> rfc792,919,922,950
- 最新のRFCの動向を知りたければMLに入ろう。



RFCの問題点

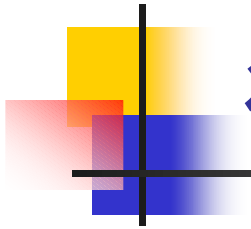
- あいまいな内容がものも多い
- 古いRFCには現状と合っていないものもあ
- RFCではない標準もある(IEEEとか)
- そもそも量が多すぎる。draft もあるし。

(余談)IETFの活動を追っていると案外面白いものがある

- ietf/noctools

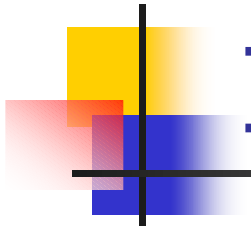
The NOC-Tools Working Group will develop a catalog to assist network managers in the selection and acquisition of diagnostic and analytic tools for TCP/IP Internets.

TCP/IPの位置付け



従来までの通信の考え方

- 通信を直感的に実装すると電話のような実装になる。
- 経路を確保してその上にデータを流す。
- 途中経路の一部が故障した際に通信不能になってしまう。



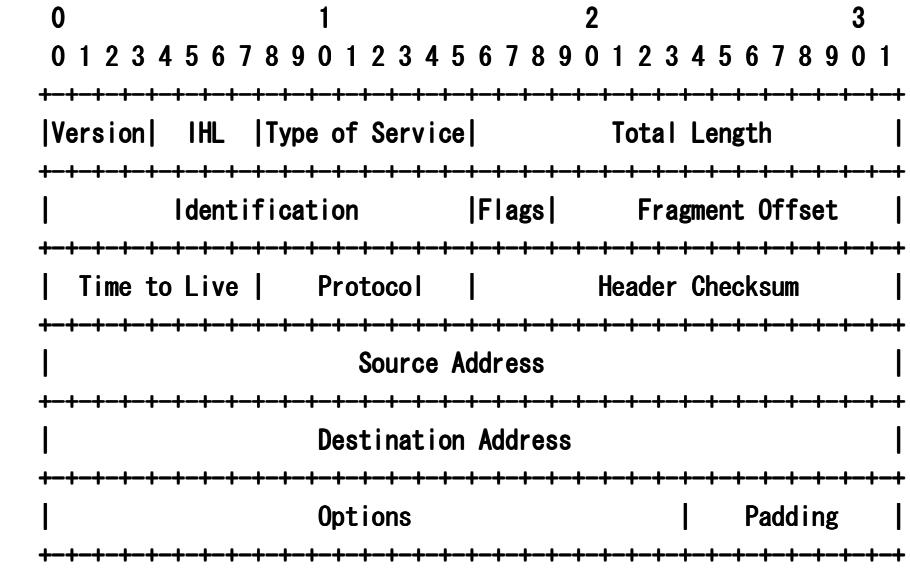
IPの考え方

従来までの通信の考え方から発想を転換

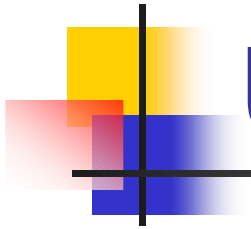
- ハガキを使っても通信は可能
- 途中はいろいろな経路を通るけど宛先が書いてあるから届く。
- 途中経路が故障しても適当に迂回して相手に届く。
- 相手に届かなかったら、再送すれば良い。
- 大きなデータは分割すれば良いし。

IPパケットの構造

A summary of the contents of the internet header follows:



Example Internet Datagram Header

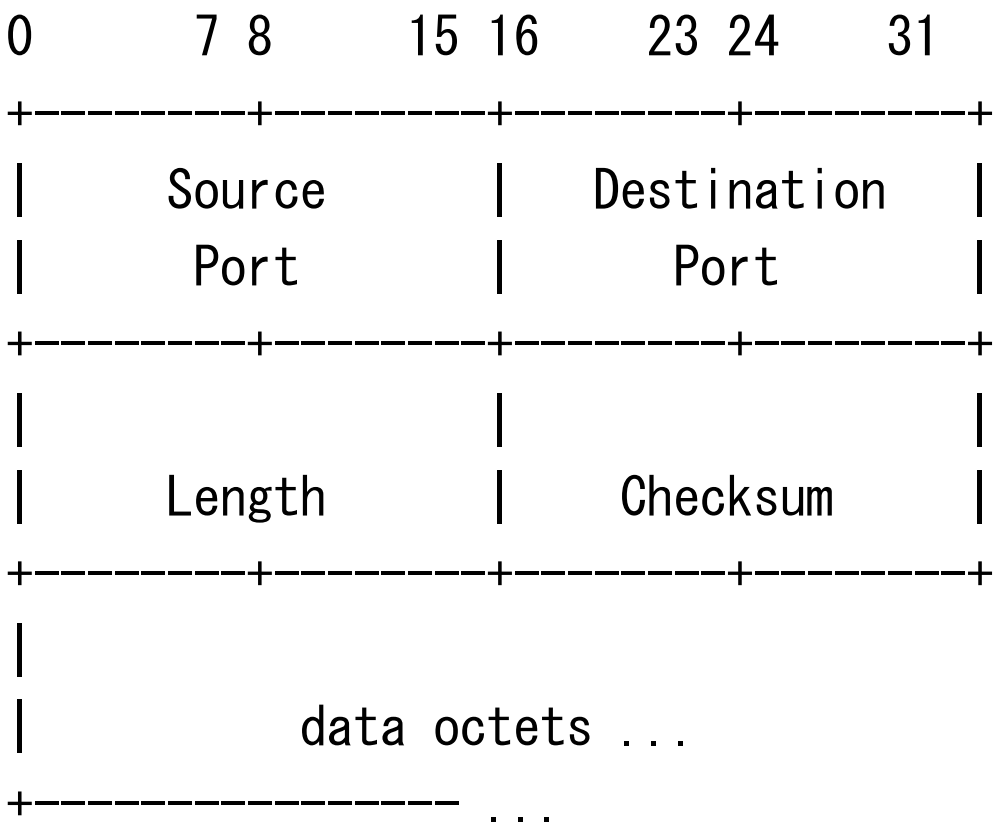


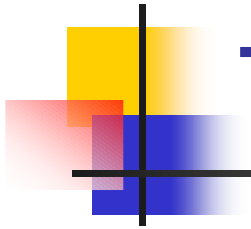
UDPとは？

- IPに、宛先ポート番号を付けてみた。
- 用途ごとに通信の種別を分けられるようになった。

- UDP --> rfc768

UDPパケットの構造

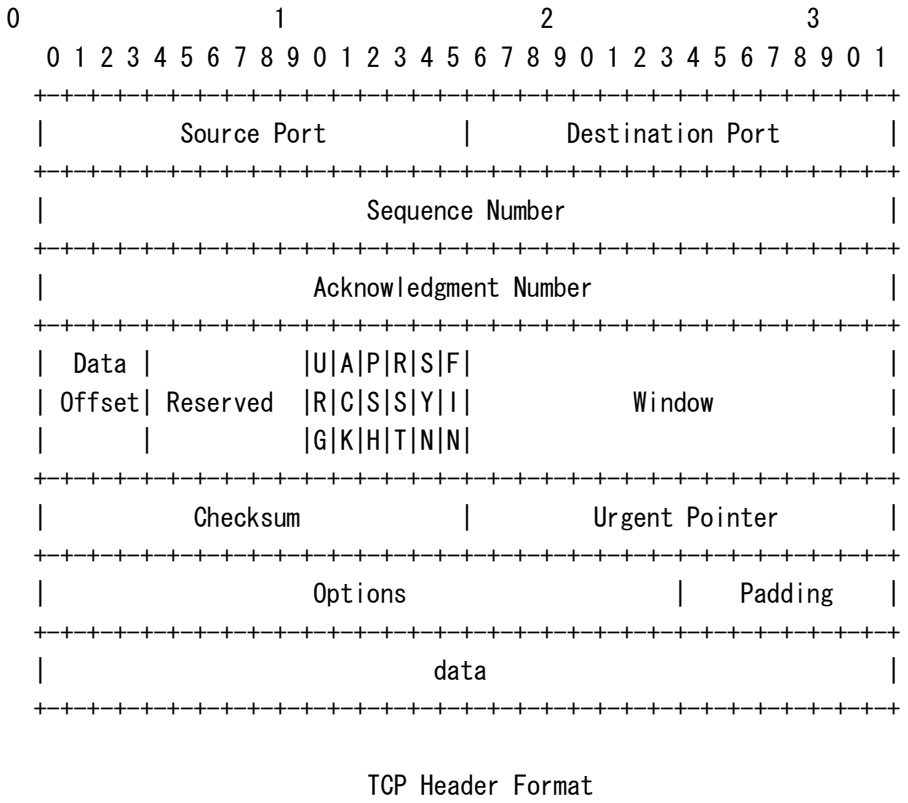

0 7 8 15 16 23 24 31
Source Port
Destination Port
Length
Checksum
data octets ...



TCPとは？

- 通信を直感的に実装すると電話のような実装になる。やっぱりこういう直感的な実装も欲しい。
- まずコネクションを張って、その上でデータのやりとりをする。
- 大きな長文の分割や再構成や再送を TCP が行なう。
- TCP --> rfc793
- src address,port が一緒でも dist address,port が異なれば、異なる接続路。

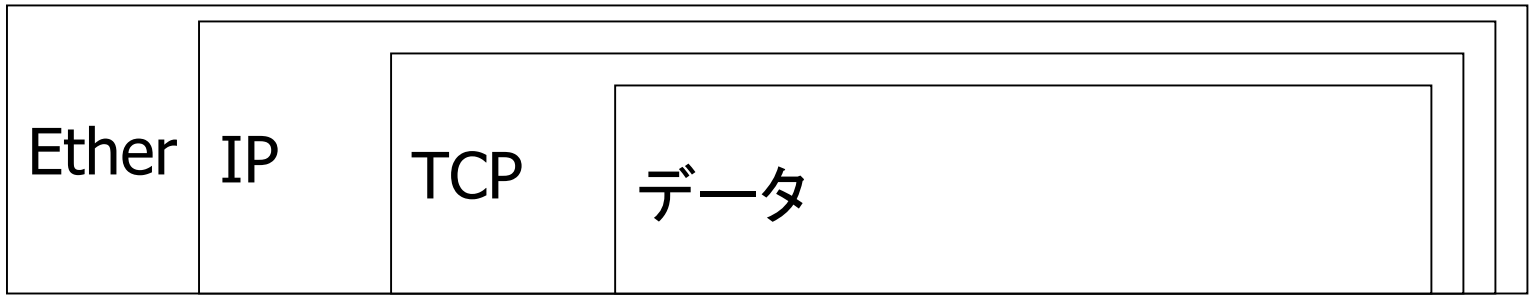
TCPパケットの構造

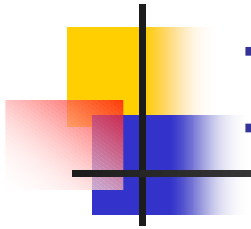


TCP Header Format

データのカプセル化

- イーサネットを流れるデータは以下のようにカプセル化される。

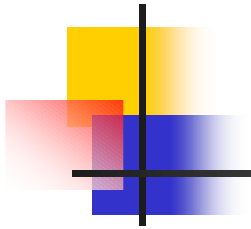




ICMPは何？

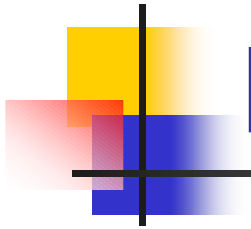
- TCP/IPの制御用プロトコル
- IPと組み合わせて用いられる

- ICMP --> rfc792,919,922,950



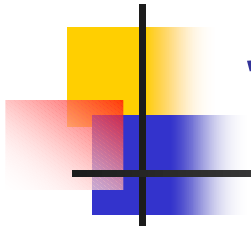
良くわからない...

- まずは使ってみましょう。
- いろいろ試してみましょう。



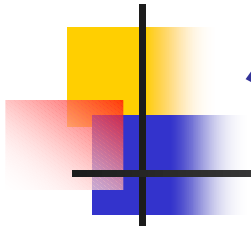
Break!

- ここまでで質問あるかしら？



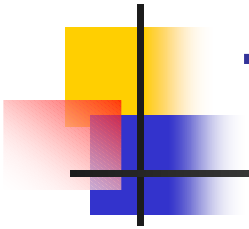
今回はTCP/IPというお題目なので…

- UDPは無視
- 面倒なのでキャラクタ型の通信のみ扱う



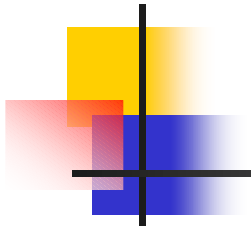
シェル(bash)でTCP/IPを使う

bash$ echo hoge > /dev/tcp/127.0.0.1/**



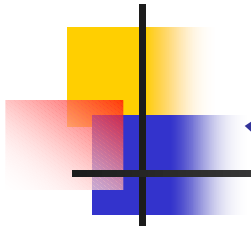
プログラムからTCP/IPで通信

```
#!/usr/local/bin/ruby
require "socket"
s = TCPSocket.open("localhost", 13)
print(s.gets)
s.close
```



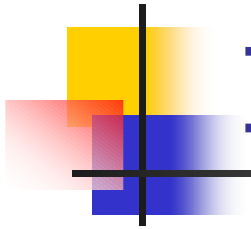
受信できるように見張る

```
require "socket"

gs = TCPServer.open(11111)

while true
  Thread.start(gs.accept) do |s|
    while s.gets
      s.write($_)
    end
    s.close
  end
end
```



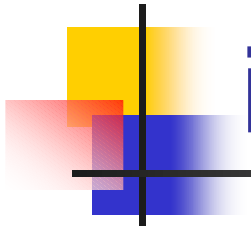
スーパーデーモン「inetd」

- 簡単にデーモンを作れる
- 標準入出力がそのままソケットの入出力になる



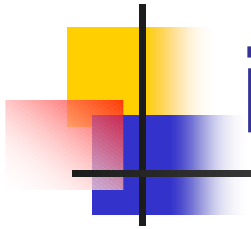
Inetdの使い方

- man inetd
- /etc/services にサービス名を書く
- /etc/inetd.conf に起動するプログラムを書く
- inetd に SIGHUP を送る



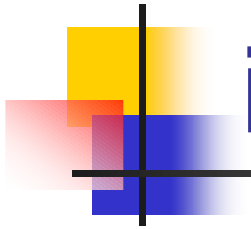
inetdの利用例 その1

```
#!/bin/sh
echo hoge
date
```



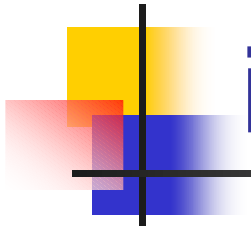
inetdの利用例 その2

```
#!/usr/bin/perl
open(F,">/home/pochi/20020628/tmpfile");
while(<>){
   print F $_;
}
```



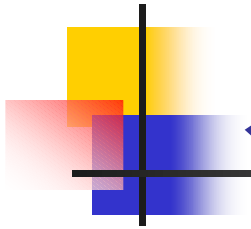
inetdの利用例 その3

```
while(<>){
    system “/usr/local/sbin/apachectl start”
        if (/start/);
    system “/usr/local/sbin/apachectl stop”
        if (/stop/);
    system “/usr/local/sbin/apachectl restart”
        if (/restart/);
    last;
}
```



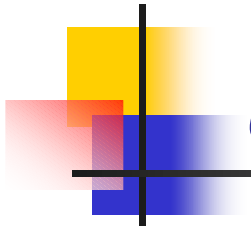
inetdの限界

- 単純なデーモンしか書けない。
- 複数のコネクションを開くことはできない。
- UDPでも使用した場合に同時に複数の接続を使用できない。
- DoS攻撃に弱い



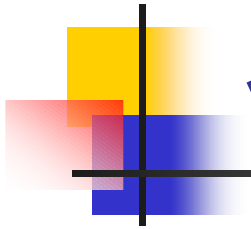
スーパーデーモンの別種

- Xinetd
- tcpserver



おしまい

- ご清聴？ありがとうございました。



今後の予定

- DNS
- TCP/IPルーティング、フィルタリング編
- SMTP、POP
- HTTP
- HTML
- プロジェクト管理手法
- セキュリティの基本
- 見積もりの作り方
- FreeBSDのインストール、管理手法
- Debianのインストール、管理手法
- Zope
- Java の基本
- DBの基本
- Apache
- スクリプトの書き方