

コンピュータネットワーク

2011/12/21

サーバの機能と仕組み

3

前回の復習

- ドメイン名
- DNS
- DNSSEC

本日の流れ

- サーバの仕組み
- P2P
- パケット解析
- ネットワークプログラミング

2

2011/12/21

サーバとは？

- ユーザに対してサービスを提供するホスト
- 普通のパソコンもサーバになる
 - Windows XP/Vista
 - MacOS X
 - UNIX (Linux/Solaris)

4

例えば。。。

- ファイル共有をする
- プリンタ共有をする
- 誰かに対して自分のリソース(持ち物)を提供しようとするれば、それはサーバとなる

5

サーバの運用

- 負荷分散
 - 多くのユーザからの接続要求
- セキュリティ監視
 - 乗っ取り攻撃
 - DoS 攻撃

7

世の中のサーバ

- Web サーバ
- メールサーバ
- VoIP サーバ
- メッセンジャサーバ
- P2P サーバ
- DNS サーバ
- すべてユーザからのリクエストに答え、なんらかのサービスを提供する

6

個人ホストでも

- サーバとなり得ます
 - 知らないうちにサーバとなっている場合も
 - 認識しておきましょう
- ソフトウェアをインストールすることにより、意図せずサーバとなることも
 - P2P 系ソフト
 - Winny
 - BitTorrent
 - Skype

8

P2P ソフトウェアについて

- P2P ソフト自体が危険なのではありません
 - そこに流れているファイルが危険
 - ウィルス入り
 - バックドア入り
- そういう意味なら Web ページ見るだけでも同じことは起こる
 - ワンクリックはもちろん
 - 画像ファイルに仕掛け

9

2011/12/21

著名なウィルス

- 山田オルタナティブ
 - 山田と名乗る人物から送られてくる
 - 勝手に自分のマシンが Web サーバとなる
 - デスクトップ画面のキャプチャが公開
 - マシン内部にあるファイルを公開
 - UPnP を使い、NAT の後ろにあるホストも全世界からアクセス可能とする
- 原田ウィルス
 - 原田と名乗る人物が画面に現れる
 - マシン内部の動画ファイルを勝手に削除
 - デスクトップ画面のキャプチャをメール送信

11

P2P の種類

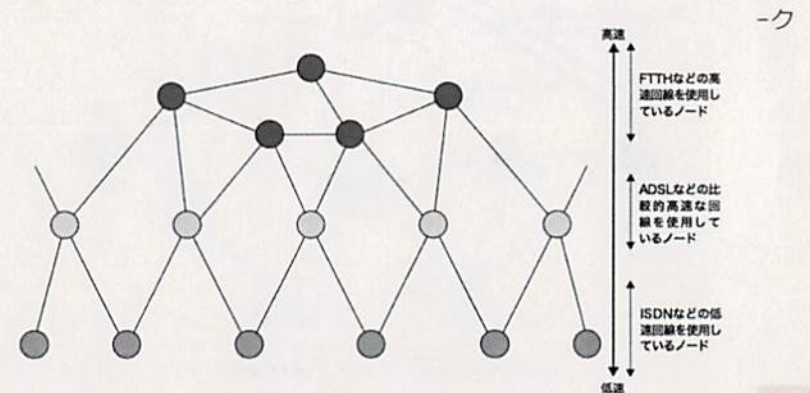
- ハイブリッド P2P
 - クラスタ単位でファイルを管理するための「スーパーノード」と呼ばれるノードが存在する
 - WinMX
 - BitTorrent
- ピュア P2P
 - 全てのノードが同様な役割を持つ
 - Winny
 - Share
 - StealthNet

10

2011/12/21

Winny の仕組み (1)

- Winny ネットワーク

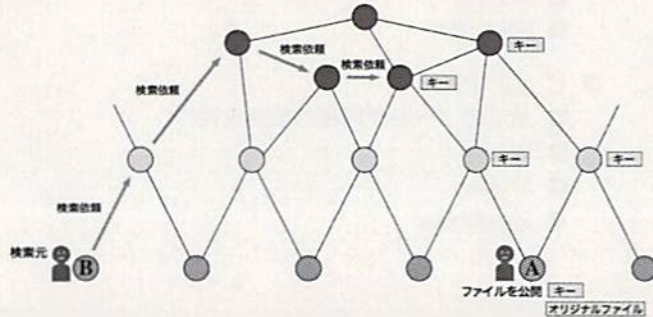


12

Winny の仕組み (2)

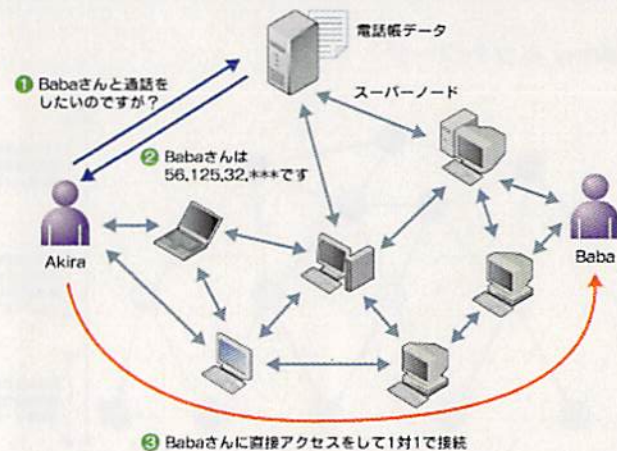
□ ファイルの検索

- ファイルに関する情報が分散して存在する
 - ファイル名、所在する IP アドレス等
- 検索キーワードが適合するまで、検索リクエストが転送され続ける



13

Skype の仕組み

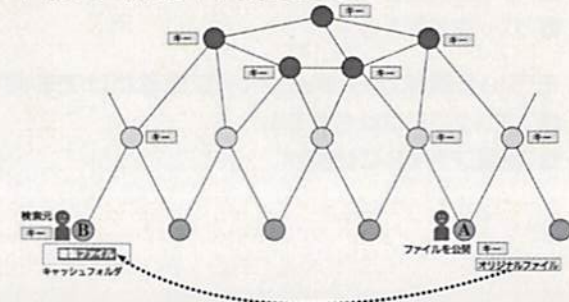


15

Winny の仕組み(3)

□ ダウンロード

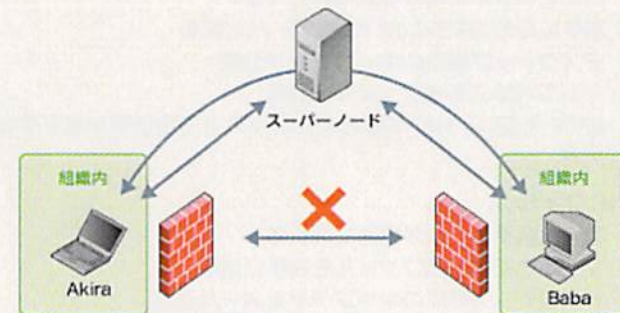
- 検索ファイルを持つ PC にダウンロード要求を送る
- その際、かならず中継ノードを選び、中継ノードを経由してダウンロードを行う
- 中継ノードはファイルをキャッシュする



14

Skype スーパーノード

- 意図せずスーパーノードになると、通信を仲介する



16

ファイヤウォールとは

- 外からの攻撃を防ぐ
- ひとことでは言え
 - IP アドレスやポートを特定し、指定した通信を遮断する仕組み
- 企業等でよくあるパターン
 - 社外→社内：全部遮断
 - 社内→社外：web の閲覧



<http://www.dmarit.co.jp/security/special/17fivemin/fivemin00.html>

17

WWW の仕組み

Abuse Report 例

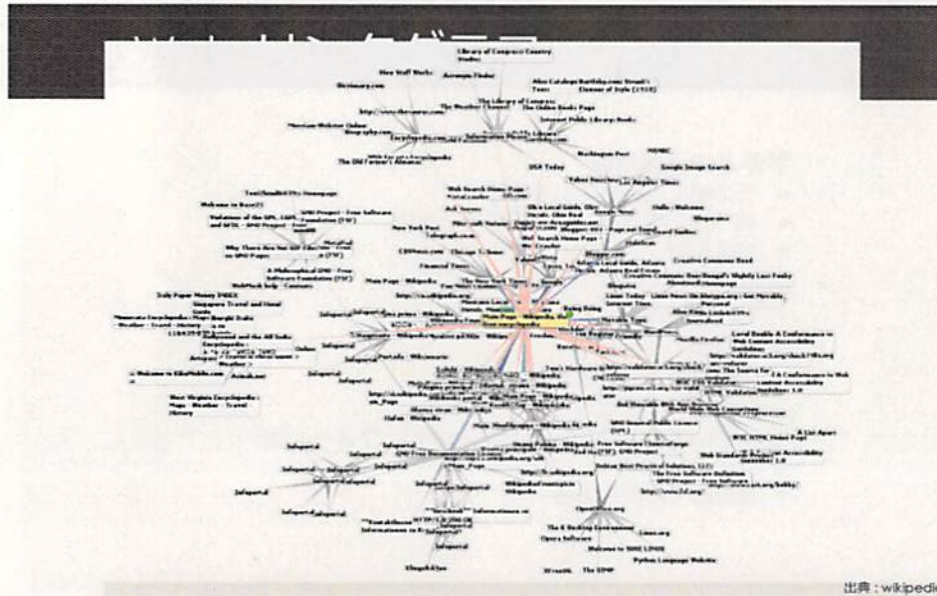
- 学外との通信遮断
 - その時に流行している攻撃
 - 明らかに学外とする必要のない通信
 - P2P 監視
- 部局間
 - ほぼ遮断無し
- 部局内部
 - 部局単位でファイヤウォール
- 使っているネットワークにファイヤウォールが存在するか？
 - 確認してみましょう

18

Web とは

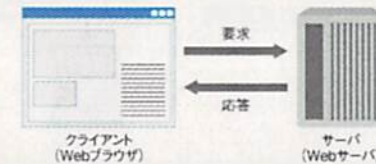
- WWW = World Wide Web
- ハイパーテキストシステム
 - 「インターネット」の代名詞
- もともとはテキスト情報を有機的に結合するための仕組み
 - ハイパーリンク = リンク
 - 情報を関連づける

20



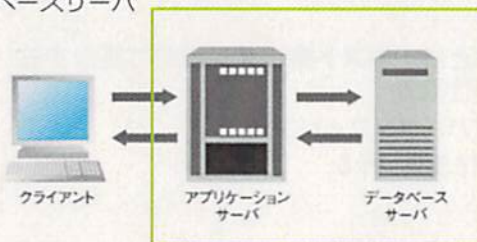
サーバ / クライアントモデル

- クライアント
 - 手元の PC
 - 「ブラウザ」というソフトウェアを用いて Web を利用
- サーバ
 - ネットワークに接続された (強めの) PC
 - Web サーバ用ソフトウェアがインストールされ、適切に設定されることで Web サーバとして動作



3層サーバ / クライアントモデル

- 近年の Web に多く見られる運用形態
- クライアントは変わらない
- サーバが 2層構造になっている
 - Web サーバ
 - データベースサーバ



実際のやりとり

- HTTP (Hyper Text Transfer Protocol) という決まりに従ってサーバ / クライアント間でデータが交換される
 - CERN(欧州原子核研究機構) にて考案される
 - HTTP 1.0 (RFC1945)
 - HTTP 1.1 (RFC2616)
- 基本的には、単なるテキスト (文章) 情報がやりとりされる
 - 画像は必須ではない

HTTP

- GET クライアントがサーバに対してデータを要求
- POST クライアントからサーバにデータを送信
- HEAD ページの情報のみを取得する
- これらのコマンドをサーバ / クライアント間で送信し合うことで情報 (Web ページ) を表示している

25

2011/12/21

HTML

- HTML (Hyper Text Markup Language)
- Web ページを書くための言語
 - 全てのWeb ページはこの言語を使って書かれている
 - Word みたいなエディタを使って作成されているのではない
 - Word で作った場合も、Word が HTML に変換したファイルを、Web サーバに置く

27

HTTP でのやりとり

```
sekiya [-] telnet www.yahoo.co.jp 80
Trying 203.216.235.201...
Connected to www.ya.gl.yahoo.co.jp.
Escape character is '^['.
HEAD / HTTP/1.0
```

```
HTTP/1.1 200 OK
Date: Wed, 13 Jan 2010 00:06:51 GMT
P3P: policyref="http://privacy.yahoo.co.jp/w3c/p3p.xml", CP="CAO DSP COR CUR ADM
DEV TAI PSA PSD IVAI IVDI CONI TELo OTPi OUR DELi SAMi OTRI UNRI PUBi IND PHY ONL UNI
PUR FIN COM NAV INT DEM CNT STA POL HEA PRE GOV"
Cache-Control: no-cache
Cache-Control: no-store, must-revalidate
Expires: -1
Pragma: no-cache
X-XRDS-Location: http://open.login.yahoo.co.jp/openid20/www.yahoo.co.jp/xrds
Cache-Control: private
Connection: close
Content-Type: text/html; charset=utf-8
```

Connection closed by foreign host.

26

2011/12/21

HTML の例 (www.u-tokyo.ac.jp)

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
<html lang="ja">
<head>
<meta http-equiv="content-type" content="text/html; charset=Shift_JIS">
<title>東京大学</title>
<meta http-equiv="content-script-type" content="text/javascript">
<meta http-equiv="content-style-type" content="text/css">
<meta http-equiv="image toolbar" content="no">
<link rel="shortcut icon" href="http://www.u-tokyo.ac.jp/favicon.ico" type="image/vnd.microsoft.icon">

<link href="files/css/base.css" rel="stylesheet" type="text/css" media="all">
<link href="files/css/clear.css" rel="stylesheet" type="text/css" media="all">

<script src="files/js/jquery-1.3.1.js" type="text/javascript"></script>
<script src="files/js/ui.core.js" type="text/javascript"></script>

<script src="files/js/ui.tabs.js" type="text/javascript"></script>
</head>

<body>
<a id="TOP" name="TOP"></a>
<noscript>
<div id="warning">
<p style="margin-bottom: 0">東京大学ウェブサイトを表示するにはJavaScriptが必要です。<br>ブラウザの設定をオンにしてから
ページをリロードしてください。</p>
</div>
</noscript>
```

28

簡単な HTML

```
<HTML>
<HEAD>
<TITLE>コンピュータネットワーク</TITLE>
</HEAD>
<BODY>
<H1>HTML のテストです</H1>
<H2>HTML を使って Web ページを作ります</H2>
<A HREF="http://www.u-tokyo.ac.jp/">東京大学
</A>
</BODY>
</HTML>
```



HTMLのテストです

HTMLを使ってWebページを作ります

[東京大学](http://www.u-tokyo.ac.jp/)

完了

29

パケット解析

表示はブラウザ任せ

- HTTP はサーバ / クライアント間のデータ交換手順の定義
- HTML は Web ページ記述の文法を定義
- どう表示するか、はブラウザに任されている
 - Internet Explorer
 - Safari
 - Firefox
 - Google Chrome

30

流れているデータを見る

- 自分のネットワークインタフェースでどんなデータが送受信されているのか
- パケットキャプチャソフトウェア
- Linux / MacOS X
 - tcpdump
 - wireshark
- Windows
 - wireshark

エンディアン (Endian)

- Little Endian と Big Endian
 - バイト列の並び方
- 16進数で 38E2 という 2 bytes(16bit) のデータをメモリに格納する場合
 - Little Endian -> E238
 - Big Endian -> 38E2
- ネットワークで送信する時は Big Endian に統一
 - ネットワークバイトオーダー
- Little Endian (Intel)
- Big Endian (PowerPC, SPARC)

33

2011/12/21

バイトオーダーの変換

- htons / ntohs
 - Short (16bit) のデータを変換する
 - Host to Network : htons
 - Network to Host : ntohs
- htonl / ntohl
 - Long (32bit) のデータを変換する
 - Host to Network : htonl
 - Network to Host : ntohl

35

エンディアンの例

- "ABCDEF" という文字列をメモリに格納する場合
 - A B C D E F = 0x41 0x42 0x43 0x44 0x45 0x46
- Big Endian
 - 0x41 0x42 0x43 0x44 0x45 0x46
- Little Endian (16bit)
 - 0x42 0x41 0x44 0x43 0x46 0x45

34

2011/12/21

tcpdump 使用例

```
% tcpdump -n -x -s 1500 -i en0 udp port 12345
```

18:20:03.025982 IP 130.69.251.118.49161 >
130.69.251.130.12345: UDP, length 5

0x0000: 4500 0021 1055 0000 4011 6e13 8245 fb76
0x0010: 8245 fb82 0009 3039 000d 307b 4845 4c4c
0x0020: 4f55 5555 5555 5555 5555 5555 5555 7487

130.69.251.118
130.69.251.130

4845 4c4c 4f => H E L L O

36

IPv4 ヘッダー

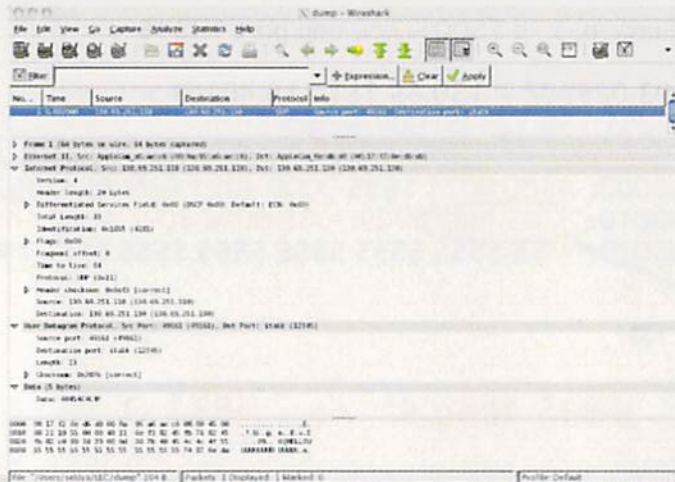
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 ビット

Version (4bit)	IHL (4bit)	Type of Service (8bit)	Total Length (16bit)		4500 0021
Identification (16bit)		Flags (3bit)	Fragment Offset (13bit)		1055 0000
Time To Live (8bit)	Protocol (8bit)	Header Checksum (16bit)			4011 6ef3
Source Address (32bit)					8245 fb76
Destination Address (32bit)					8245 fb82
Options (可変長)					32×n bit
Padding (可変長)					

37

2011/12/21

wireshark 使用例



39

UDP ヘッダー

送信元 ポート番号	宛先 ポート番号
長さ	Checksum

c009 3039

000d 307b

0xC009 = 49161
0x3039 = 12345

38

2011/12/21

www.yahoo.co.jp との通信

- 1~2 デフォルトルータの発見
- 3~4 DNS による www.yahoo.co.jp の IP アドレス解決
- 5~7 TCP による 3way handshake
- 8 HTTP によるクライアントからサーバへの GET メッセージ

Time	Source	Destination	Protocol	Info
1.0.000000	Matsushita, [REDACTED]	Broadcast	ARP	who has 192.168.0.1? tell 192.168.0.4
2.0.000278	directstar- [REDACTED]	Matsushita, [REDACTED]	ARP	192.168.0.1 is at 00:0d:02: [REDACTED]
3.0.000669	192.168.0.4	directstar- [REDACTED]	DNS	Standard query A 203.216.235.154 A 203.216.235.1
4.0.034810	directstar- [REDACTED]	192.168.0.4	DNS	Standard query response A 203.216.235.154 A 203.216.235.1
5.0.090233	192.168.0.4	www.yahoo.co.jp	TCP	worldfusion2 > http [SYN] seq=0 win=65535 Len=0 MSS=1260
6.0.120338	www.yahoo.co.jp	192.168.0.4	TCP	http > worldfusion2 [SYN, ACK] seq=0 ack=1 win=65535 Len=0
7.0.120507	192.168.0.4	www.yahoo.co.jp	TCP	worldfusion2 > http [ACK] seq=1 ack=1 win=65535 Len=0
8.0.128988	192.168.0.4	www.yahoo.co.jp	HTTP	GET / HTTP/1.1
9.0.197873	www.yahoo.co.jp	192.168.0.4	TCP	[TCP segment of a reassembled PDU]
10.0.209680	www.yahoo.co.jp	192.168.0.4	TCP	[TCP segment of a reassembled PDU]
11.0.209764	192.168.0.4	www.yahoo.co.jp	TCP	worldfusion2 > http [ACK] seq=416 ack=2521 win=65535 Len=0
12.0.222010	www.yahoo.co.jp	192.168.0.4	TCP	[TCP segment of a reassembled PDU]
13.0.222098	192.168.0.4	www.yahoo.co.jp	TCP	worldfusion2 > http [ACK] seq=416 ack=3781 win=65535 Len=0
14.0.252483	www.yahoo.co.jp	192.168.0.4	TCP	[TCP segment of a reassembled PDU]

出典: http://blog-imgs-22-origin.fc2.com/n/e/1/networkprogramming/pkt_yahoo_02.jpg

40

ネットワークプログラミング

C言語

- どの OS でも利用可能
 - OS 自体が C 言語で書かれている場合が多い
- Linux
 - gcc というパッケージをインストール
- MacOS X
 - Xcode をインストール
- Windows
 - Cygwin をインストール / Visual Studio を購入

ネットワークプログラミング

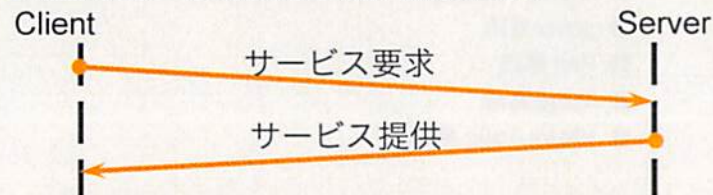
- ネットワークを使ったプログラム
 - Socket プログラミングと呼ばれる
- OS / プログラミング言語が API (Application Programming Interface) を提供
 - C / C++ 言語
 - Java 言語
 - Perl 言語
 - Ruby 言語
 - Visual Basic 言語

システムコール

- OS がユーザ(プログラマ) に提供している機能群
 - ネットワーク通信も、複数のシステムコールを呼び出して実現されている
- 各種言語でネットワークプログラミングが可能
 - C言語が一番システムコールをそのまま呼び出しているイメージ
 - C++/Java はクラスを用いて抽象化されている
 - スクリプト言語 (Perl / Ruby / Python) はより簡単にネットワークプログラミングが可能となっている

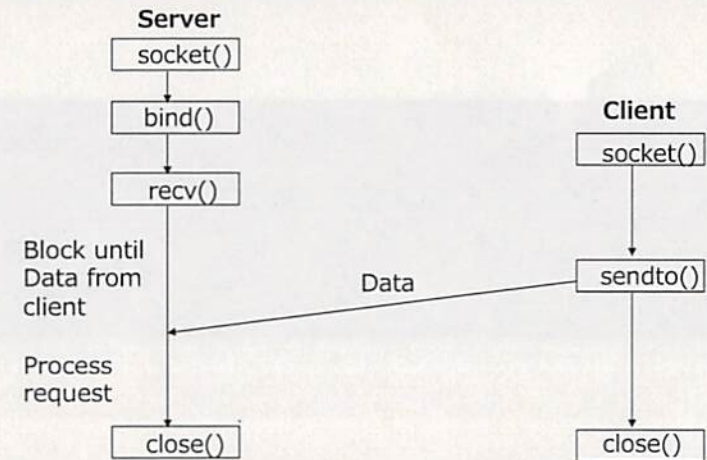
クライアント・サーバモデル

- ネットワークを介したサービスにおける通信モデル
- サーバ
 - 受動的にサービス提供する側、待っててくれる
- クライアント
 - 能動的にサービス提供を促す側、接続しに行く



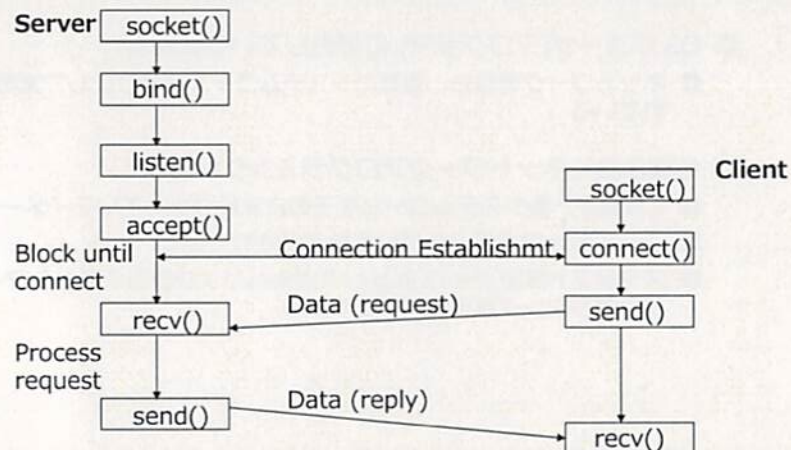
45

Datagram example (UDP)



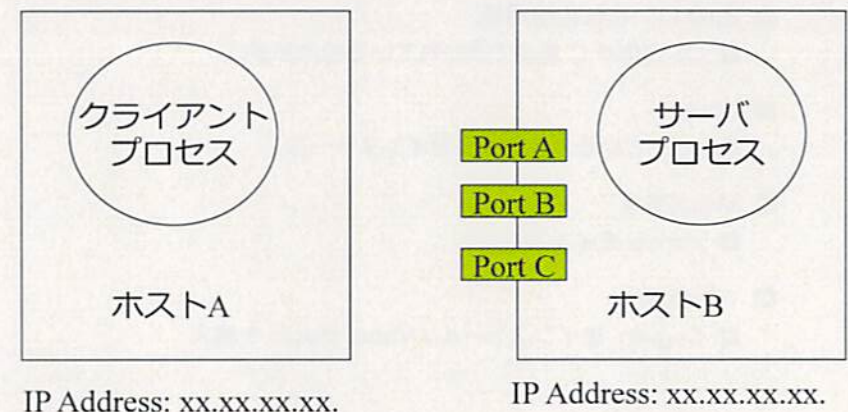
46

Stream example (TCP)



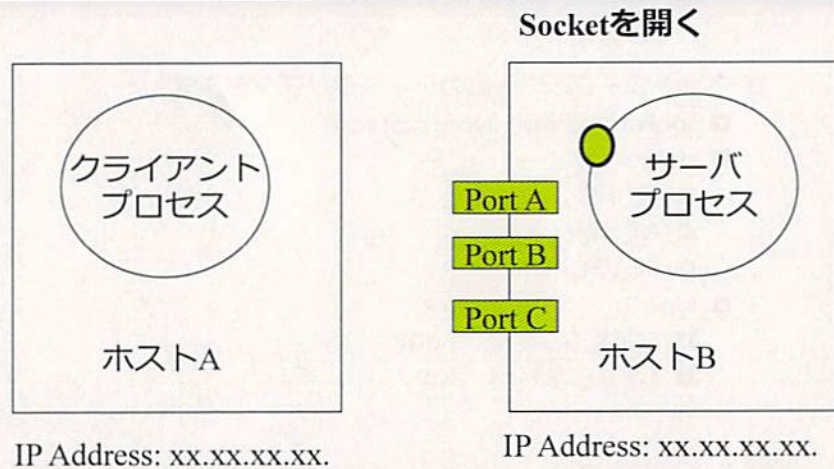
47

初期状態



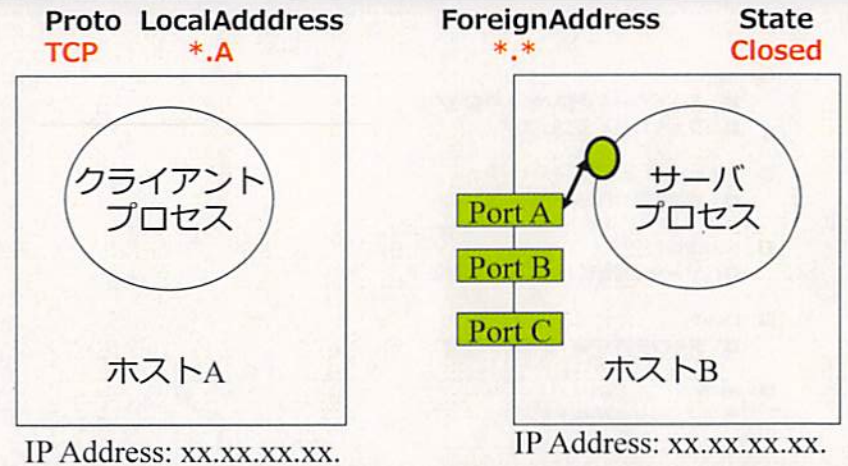
48

Socketを開いた状態



49

bindした状態



50

IPv4 UDP 送信プログラム

```
int main()
{
    int sock;
    struct sockaddr_in addr;

    sock = socket(PF_INET, SOCK_DGRAM, 0);

    addr.sin_family = AF_INET;
    addr.sin_port = htons(12345);
    addr.sin_addr.s_addr = inet_addr("130.69.251.130");

    sendto(sock, "HELLO", 5, 0, (struct sockaddr *)&addr, sizeof(addr));

    close(sock);
}
```

51

IPv4 UDP 受信プログラム

```
int main()
{
    int sock;
    struct sockaddr_in addr;
    char buf[2048];

    memset(buf, 0, sizeof(buf));

    sock = socket(PF_INET, SOCK_DGRAM, 0);

    addr.sin_family = AF_INET;
    addr.sin_port = htons(12345);
    addr.sin_addr.s_addr = INADDR_ANY;

    bind(sock, (struct sockaddr *)&addr, sizeof(addr));
    recv(sock, buf, sizeof(buf), 0);

    printf("%s\n", buf);
    close(sock);
}
```

52

使われた主な API (関数)

- socket
 - ネットワーク通信の出入り口を作成
 - ファイルディスクリプタ
- sockaddr_in 構造体
 - 通信相手や自分の情報を入れる
- sendto
 - データを宛先に対して投げつける
- bind
 - データを待ち受けるための準備
- recv
 - データを待ち受ける

53

sockaddr_in 構造体

- MacOS X の場合

```
struct sockaddr_in {
    __uint8_t      sin_len;
    sa_family_t    sin_family;
    in_port_t      sin_port;
    struct in_addr sin_addr;
    char           sin_zero[8]
};
```

- sin_family
 - AF_INET
 - AF_INET6

55

socket

- ネットワークへの入出力ディスクリプタを作成
 - socket (domain, type, protocol)
 - domain
 - PF_INET
 - PF_INET6
 - PF_LOCAL
 - type
 - SOCK_DGRAM : UDP
 - SOCK_STREAM : TCP

54

sendto

- sendto(int s, const void *buf, size_t len, int flags, const struct sockaddr *to, socklen_t tolen);
- メッセージを送信する
- 主に UDP の投げっぱなしアプリケーションで利用される

56

bind / recv

- bind : データを待ち受けるための準備
 - 手元のどのインタフェースの
 - どのポート番号で

待ち受けるかを指定

- recv : データを受信
 - 受信したデータをメモリバッファに格納
 - ブロッキング受信

Blocking / Non-Blocking

- Blocking
 - データを受信する際、データが到着するまで処理を中断して待機する
- Non Blocking
 - 処理を中断することなくデータを待機し、受信したデータを後ほど取得する方式
 - ポーリング方式とも呼ばれる
 - select という API を用いて実現