

# ネットワークコンピューティング 第2回

中山 雅哉 (m.nakayama@cnl.t.u-tokyo.ac.jp)  
関谷 勇司 (sekiya@nc.u-tokyo.ac.jp)

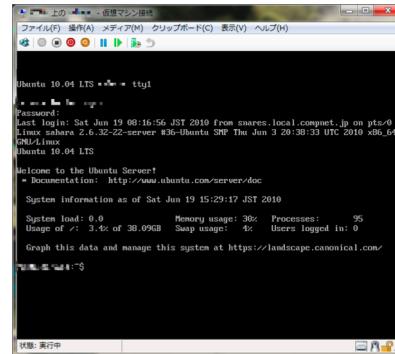
# 授業に関する情報

- 授業スライド、連絡事項、課題等に関する連絡
  - Web
    - <http://lecture.sekiya-lab.info/>
  - mail
    - lecture@sekiya-lab.info
- 実験用ホスト
  - 外部から ssh にてログインできるLinux マシンを 2台準備します
    - login1.sekiya-lab.info
    - login2.sekiya-lab.info

# Linux 端末にログインする準備

- ssh の公開鍵を lecture@sekiya-lab.info に送付してください
  - To : lecture@sekiya-lab.info
  - Subject : SSH public key
  - 本文 : 氏名, 学籍番号, 希望するログイン名
  - 添付ファイル or 本文に貼り付け : ssh 公開鍵
- 既に作成し、利用している ssh 鍵対があれば、その公開鍵を送ってもらって構いません
- **絶対に秘密鍵は送らないで下さい !!**

## 手順



ログイン可能



設定



ssh 公開鍵



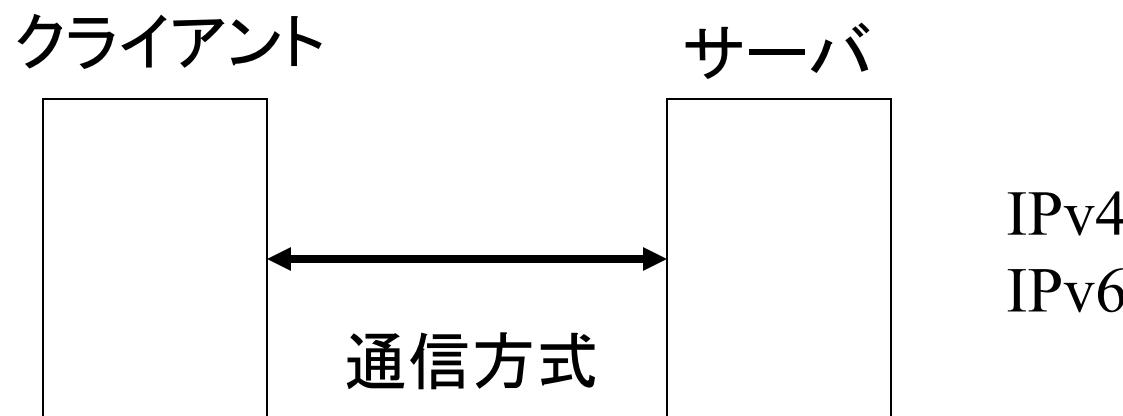
ssh 鍵対作成  
秘密鍵は手元に

# ssh 鍵対の作成方法

- UNIX, MacOS X の場合
  - ssh-keygen コマンドにて作成
  - 詳しい作成方法は検索すればたくさん見つかります
- Windows の場合
  - Cygwin の ssh-keygen コマンドを使う  
(<http://www.cygwin.com/>)
  - Linux on Windows を使う
  - PuTTY を使う (<http://www.putty.org/>)
    - puttygen.exe
  - 同じく検索すればたくさん見つかります
  - よくまとまっているもの：
    - <http://www.kuins.kyoto-u.ac.jp/news/47/putty-gen.html>
    - <http://www.cc.tsukuba.ac.jp/WinSCP/PuTTYgen/>

# サーバ・クライアントモデル

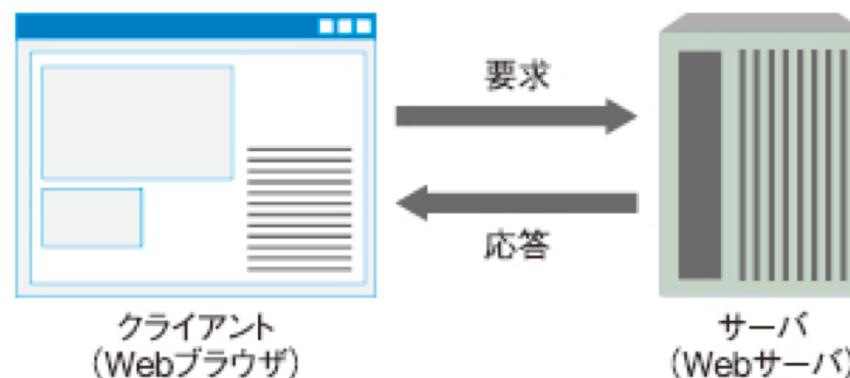
# 講義で扱う構成要素



- ・名前解決

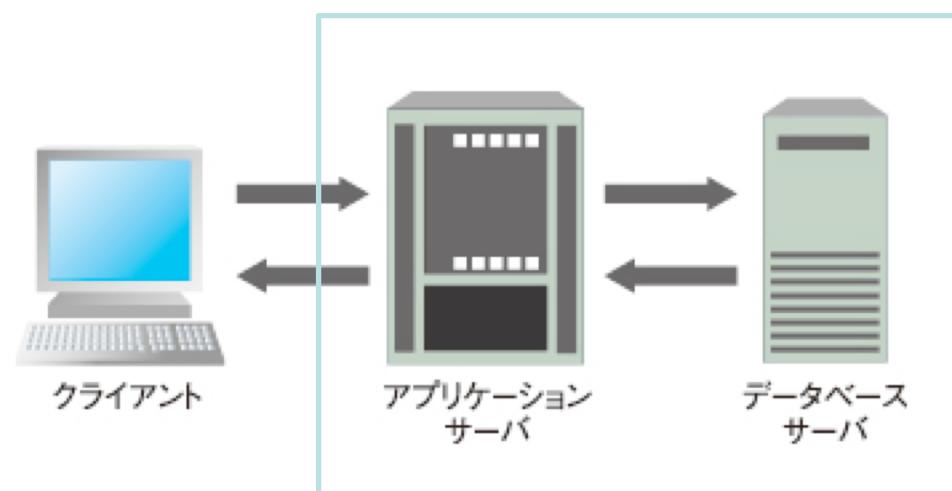
# サーバ / クライアントモデル

- 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

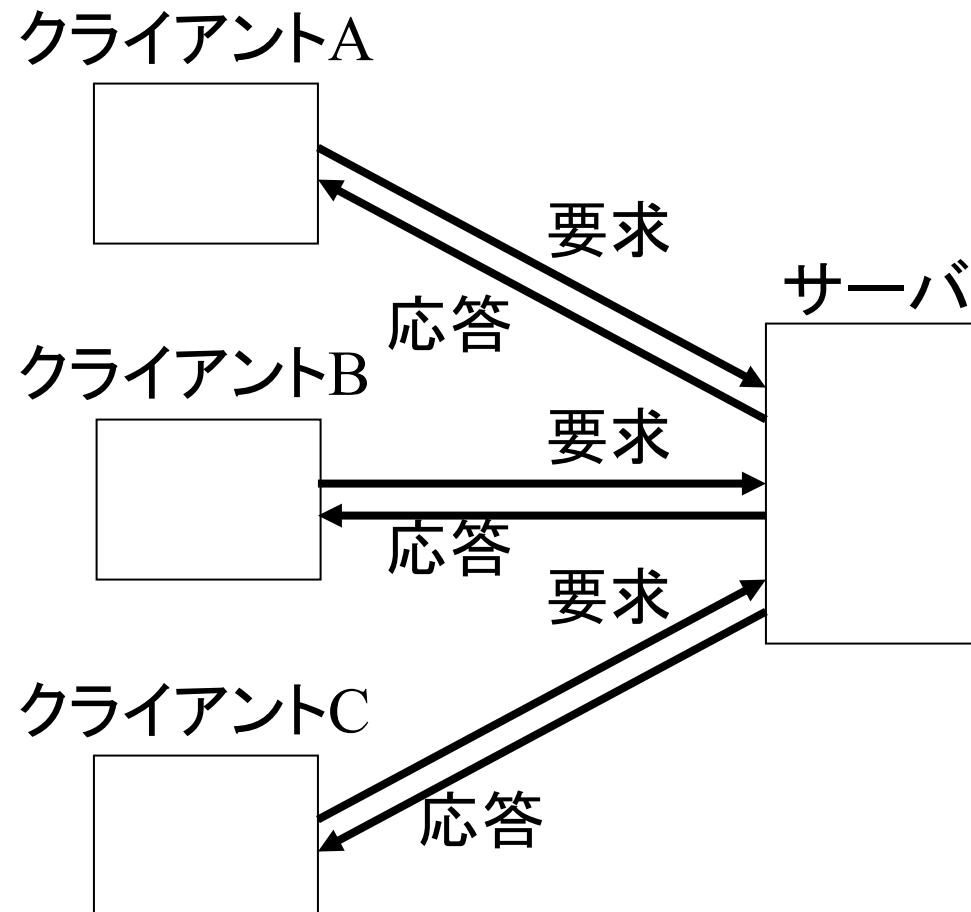


# 3層サーバ / クライアントモデル

- 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



# クライアント-サーバモデル(2)



サーバは、(同時に)複数のクライアントからの要求に対する処理をするケースもある

# アプリケーションの例

# 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(欧洲原子核研究機構) 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 の仕様

- 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

# 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.
```

# 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

# 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="imagetoolbar" 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">

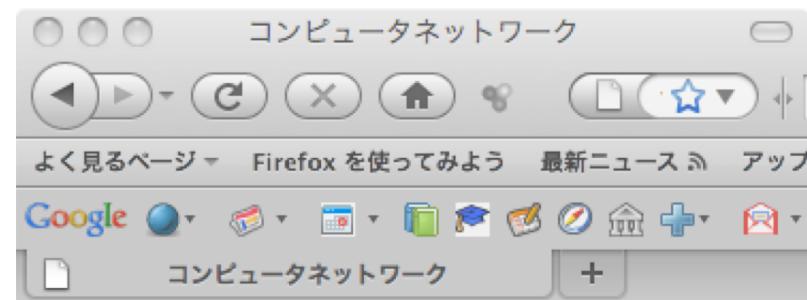
<script src="files/js/jquery-1.3.1.js" type="text/javascript"></script>
<script src="files/js/ui.core.js" type="text/javascript"></script>

<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正しく表示するには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>
```

# 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

# 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、はブラウザに任せられている
  - Internet Explorer
  - Safari
  - Firefox
  - Google Chrome

# 流れているデータ

# 流れているデータを見る

- 自分のネットワークインター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)

## 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

# 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

# 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 6ef3 8245 fb76  
0x0010: 8245 fb82  
0x0020: 4f55 5555 5555 5555 5555 5555 5555 7487
```

130.69.251.130

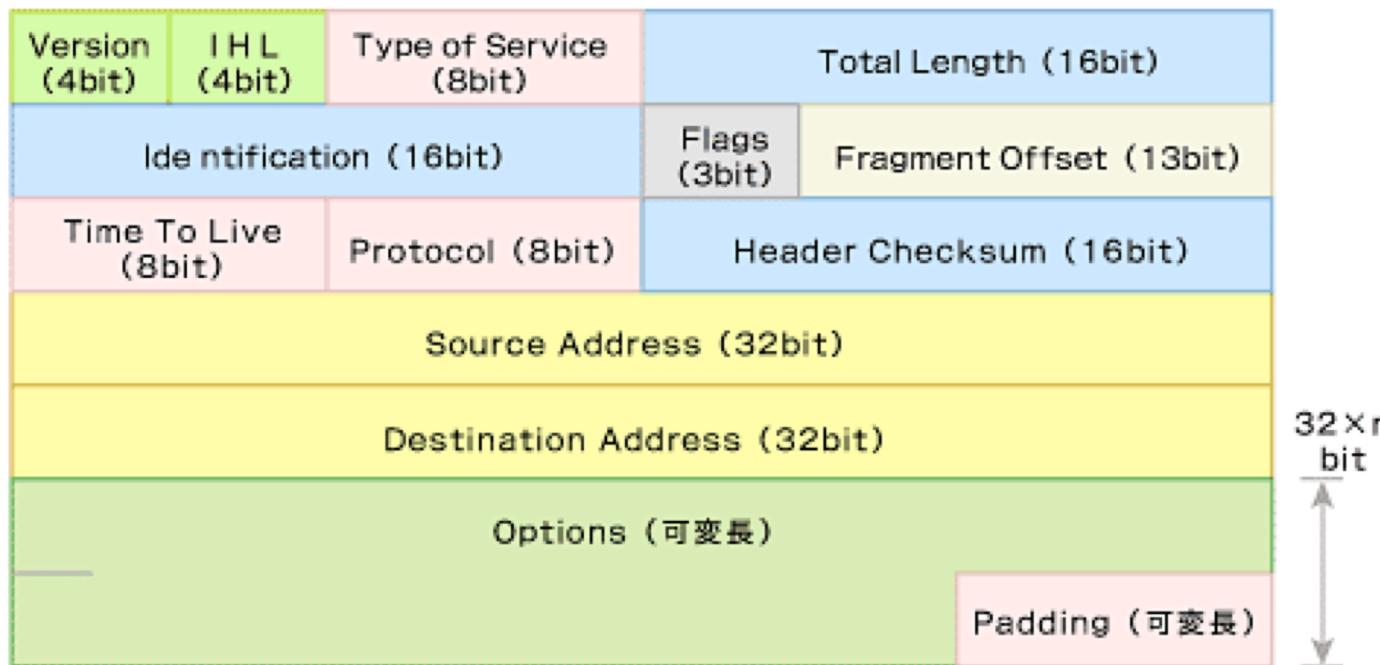
130.69.251.118



4845 4c4c 4f => HE LL O

# 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



4500 0021

1055 0000

4011 6ef3

8245 fb76

8245 fb82

# 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

$0xC009 = 49161$

$0x3039 = 12345$

# wireshark 使用例

dump - Wireshark

File Edit View Go Capture Analyze Statistics Help

Filter: Expression... Clear Apply

No.	Time	Source	Destination	Protocol	Info
1	0.000000	130.69.251.118	130.69.251.130	UDP	Source port: 49161 Destination port: italk

Frame 1 (64 bytes on wire, 64 bytes captured)

Ethernet II, Src: AppleCom\_a6:ae:c6 (00:0a:95:a6:ae:c6), Dst: AppleCom\_0e:d6:d0 (00:17:f2:0e:d6:d0)

Internet Protocol, Src: 130.69.251.118 (130.69.251.118), Dst: 130.69.251.130 (130.69.251.130)

- Version: 4
- Header length: 20 bytes
- Differentiated Services Field: 0x00 (DSCP 0x00: Default; ECN: 0x00)
- Total Length: 33
- Identification: 0x1055 (4181)
- Flags: 0x00
- Fragment offset: 0
- Time to live: 64
- Protocol: UDP (0x11)
- Header checksum: 0x6ef3 [correct]
- Source: 130.69.251.118 (130.69.251.118)
- Destination: 130.69.251.130 (130.69.251.130)

User Datagram Protocol, Src Port: 49161 (49161), Dst Port: italk (12345)

- Source port: 49161 (49161)
- Destination port: italk (12345)
- Length: 13
- Checksum: 0x307b [correct]

Data (5 bytes)

Data: 48454C4C4F

0000 00 17 f2 0e d6 d0 00 0a 95 a6 ae c6 08 00 45 00 ..... E.  
 0010 00 21 10 55 00 00 40 11 6e f3 82 45 fb 76 82 45 .! U. @ n. E. v. E  
 0020 fb 82 c0 09 39 00 0d 30 7b 48 45 4c 4c 4f 55 .... 0{ HELLOU  
 0030 55 55 55 55 55 55 55 55 74 87 6e da UUUUUUUU UUUUt.n.

File: "/Users/sekiya/LEC/dump" 104 B... | Packets: 1 Displayed: 1 Marked: 0 | Profile: Default

# www.yahoo.co.jp との通信

- 1~2 ARP による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
- 9~14 サーバからクライアントへ HTML (web ページ) の送信

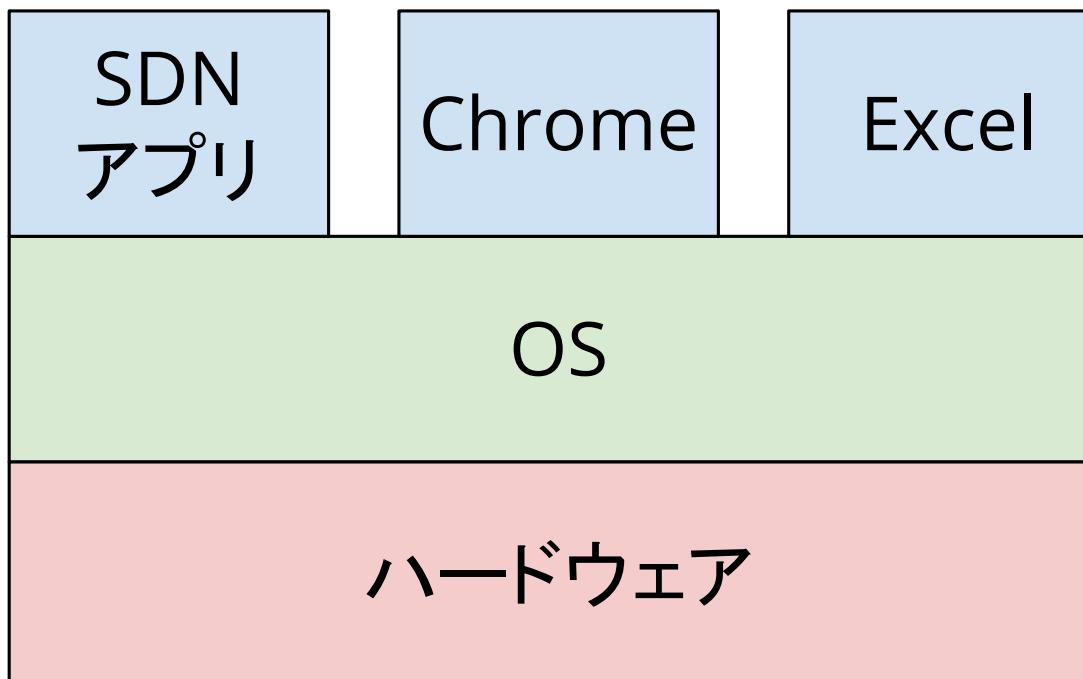
	Time	Source	Destination	Protocol	Info
1	0.000000	Matsushi-[REDACTED]	Broadcast	ARP	who has 192.168.0.1? tell 192.168.0.4
2	0.000278	directstar-[REDACTED]	Matsushi-[REDACTED]	ARP	192.168.0.1 is at 00:0d:02:[REDACTED]
3	0.000669	192.168.0.4	directstar-[REDACTED]	DNS	Standard query A www.yahoo.co.jp
4	0.034810	directstar-[REDACTED]	192.168.0.4	DNS	Standard query response A 203.216.235.154 A 203.216.235.2
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/t/networkprogramming/pkt\\_yahoo\\_02.jpg](http://blog-imgs-22-origin.fc2.com/n/e/t/networkprogramming/pkt_yahoo_02.jpg)

# OS とは

# Operating system (OS)とは？

- ハードウェアとアプリケーションの中間層
  - アプリに対してデバイスを便利で安全な操作手段を提供
  - ハードウェア特有の制御方法を隠蔽化

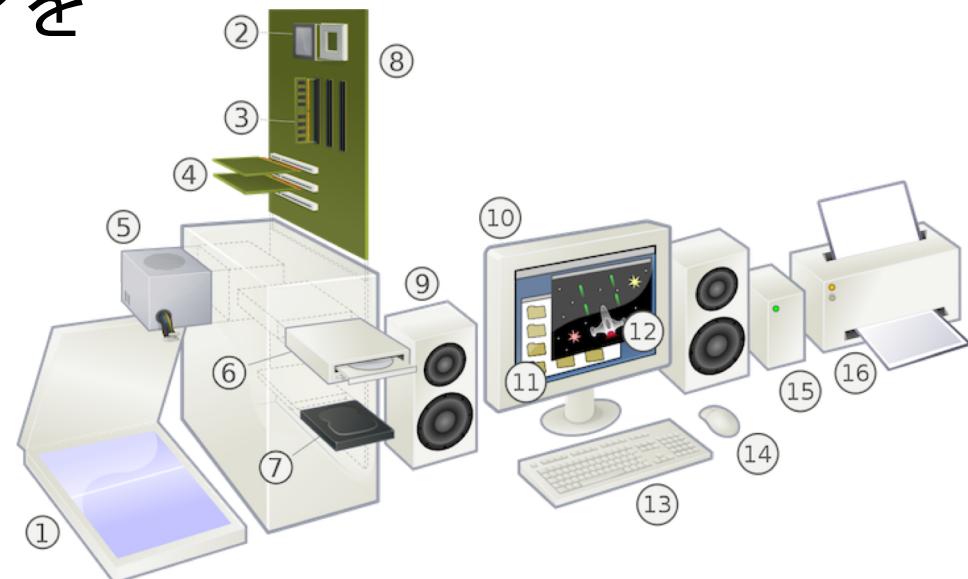


マルチタスク、  
マルチユーザ、  
HWリソース管理、  
メモリ保護....

# PCに接続される様々なハードウェア

- PCは様々なハードウェアをソフトウェアで制御可能

- スキャナ・プリンタ
- マイク・スピーカ
- モデム・NIC
- キーボード・マウス
- ディスプレイ
- CD-ROM・Blu-ray

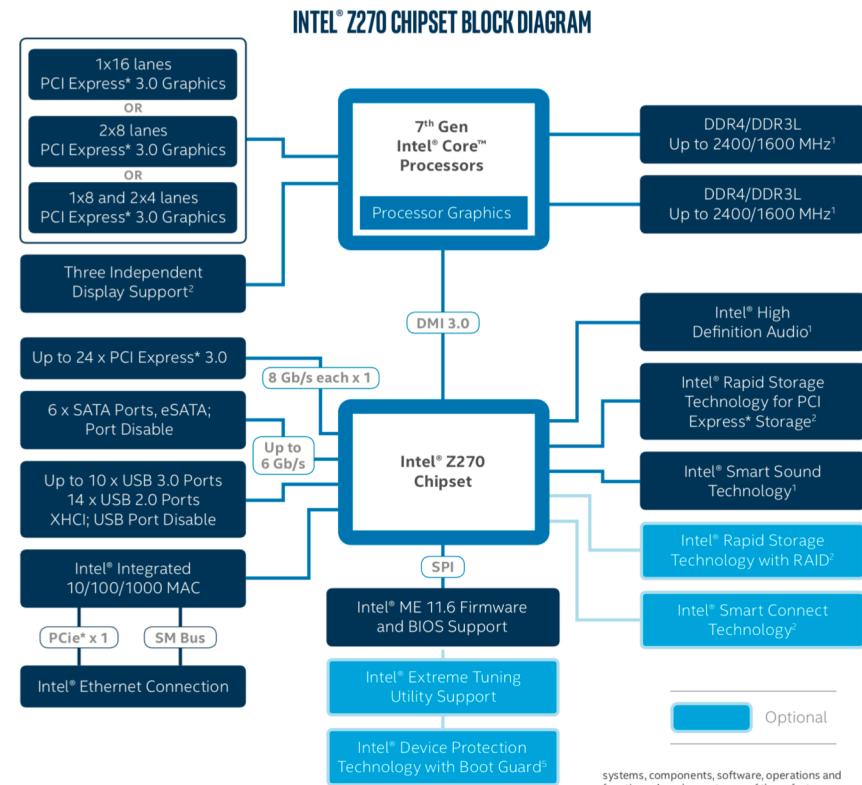


異なる特徴を持つハードウェアをソフトウェアでどのように制御するか

Wikipedia - Personal computer: [https://en.wikipedia.org/wiki/Personal\\_computer](https://en.wikipedia.org/wiki/Personal_computer)

# Intelチップセット接続図

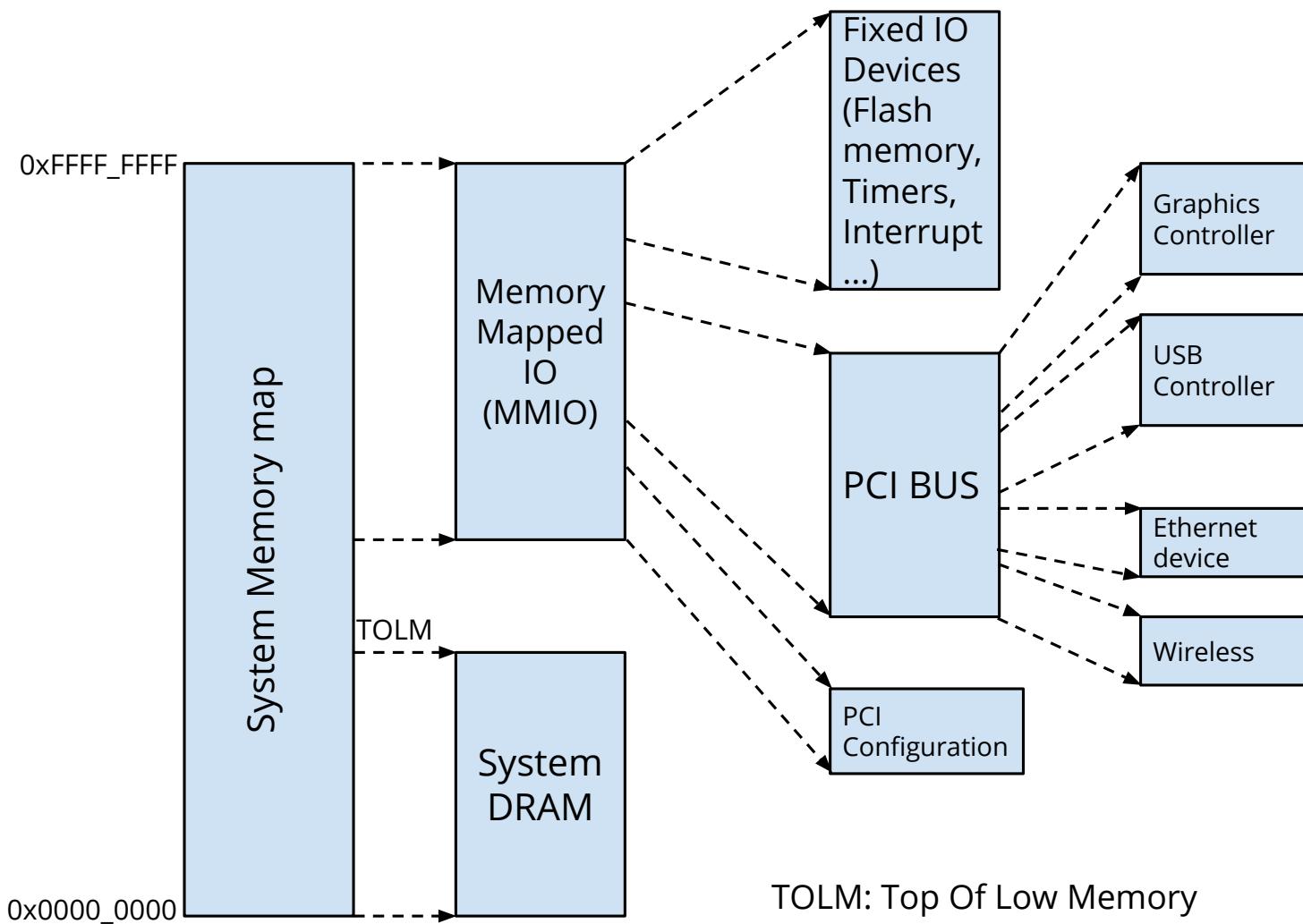
- マザーボードやケーブル、コントローラを経由して各ハードウェア、CPU、メモリが接続
  - これらの物理的な接続関係をOS(アプリケーション)でどのように操作できるか



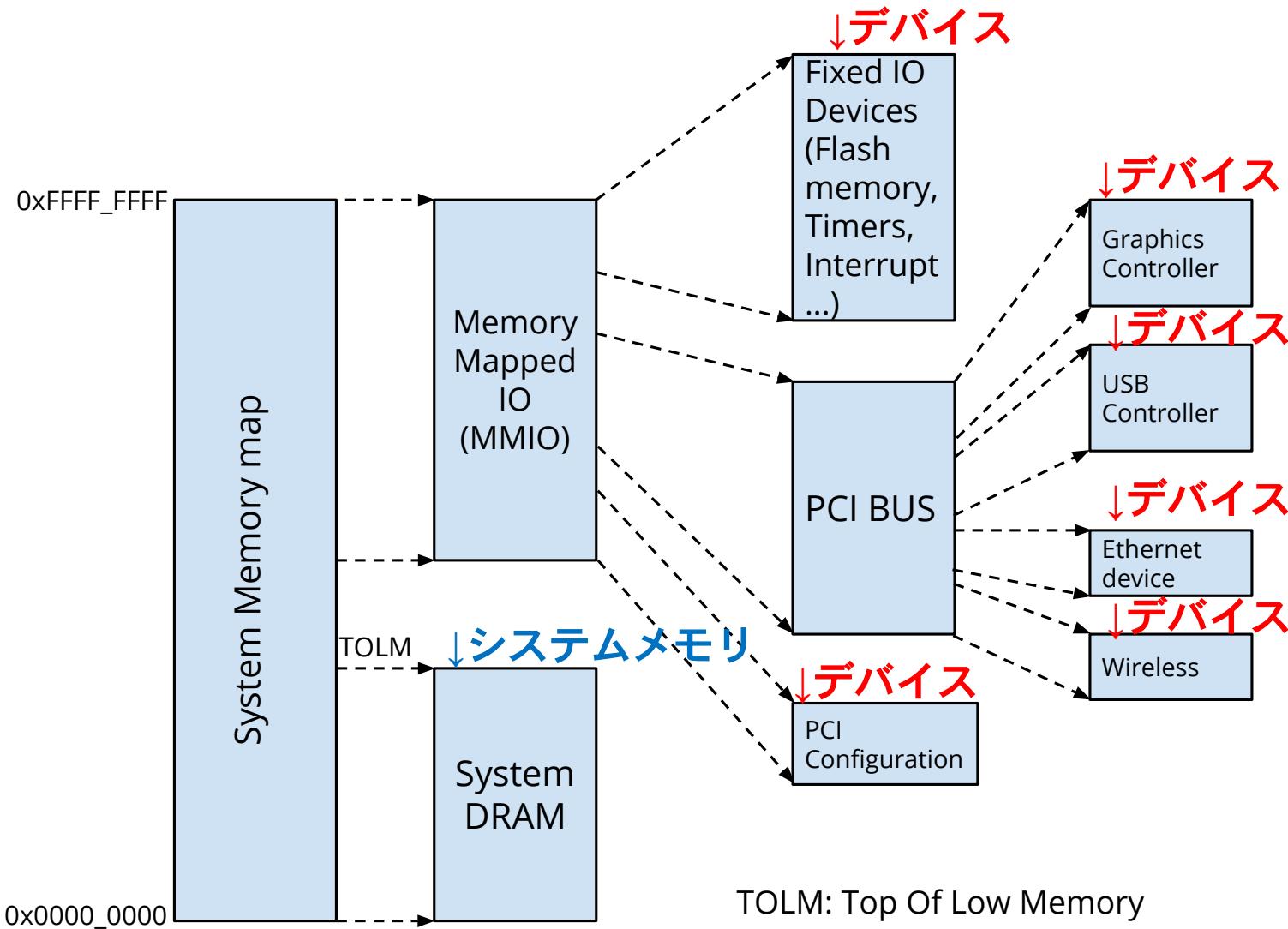
Product brief Intel Z270 Chipset

<https://www.intel.co.jp/content/dam/www/public/us/en/documents/product-briefs/z270-chipset-brief.pdf>

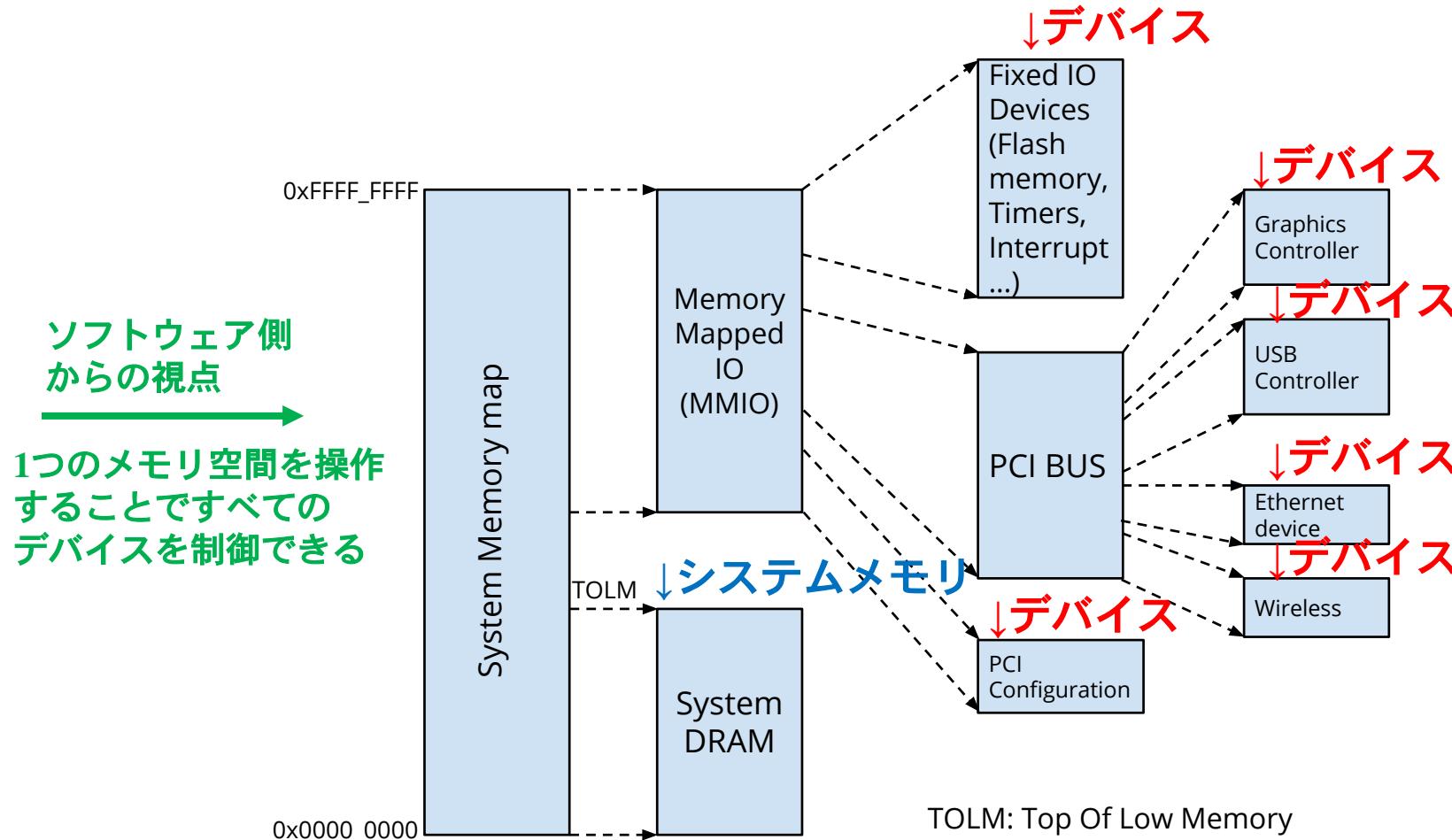
# System memory map (Intelアーキテクチャ)



# System memory map (Intelアーキテクチャ)



# System memory map (Intelアーキテクチャ)



# Linuxでのiomapの確認 (less /proc/iomem)

```
00000000-0000ffff : reserved  
00010000-00057fff : System RAM  
00058000-00058fff : reserved  
00059000-0009efff : System RAM  
0009f000-0009ffff : reserved  
000a0000-000bffff : PCI Bus 0000:00  
000c0000-000cebff : Video ROM
```

...

```
dfa00000-feaffffff : PCI Bus 0000:00  
f0400000-f05fffff : PCI Bus 0000:02  
f0400000-f047ffff : 0000:02:00.1  
f0400000-f047ffff : ixgbe  
f0500000-f0503fff : 0000:02:00.1  
f0500000-f0503fff : ixgbe
```

ixgbe: Intel 10GE NICのlinux driver名

MMIO用メモリ領域 (NICにアクセス)  
DMA用メモリ領域 (NICにアクセス)

# 議論: OSによるデバイス操作手法の設計

- あらゆる接続デバイスをメモリ操作のみで制御可能
- しかし…
  - デバイスごとに制御方法は異なる。アクセスは簡単でもメモリ操作のみを用いたデバイス操作は困難
  - 誤ってメモリ操作をしてしまうとPCがクラッシュ・デバイスの誤動作の原因になる可能性がある
  - だれでもメモリアクセスできるとセキュリティ的に大問題

## 議論: OSによるデバイス操作手法の設計(続き)

- デバイスごとに制御方法は異なる。アクセスは簡単でもメモリ操作のみを用いたデバイス操作は困難
  - デバイスごとを操作するソフトウェア(デバイスドライバ)と、デバイスカテゴリごとの抽象化操作手段を提供

## 議論: OSによるデバイス操作手法の設計(続き)

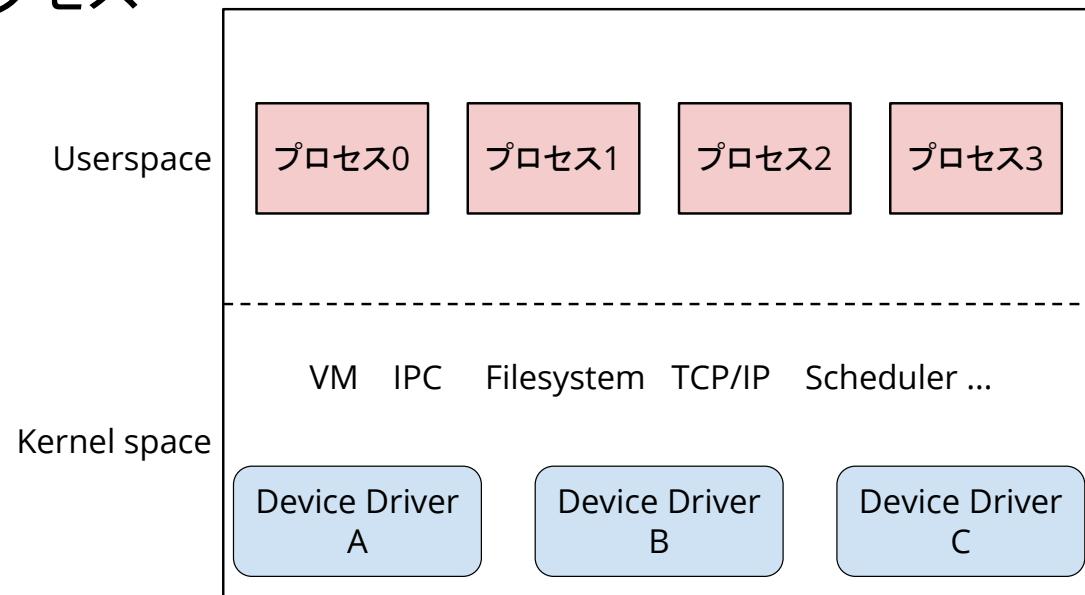
- 誤ってメモリ操作をしてしまうとPCがクラッシュ・デバイスの誤動作の原因になる可能性がある
  - 直接デバイスのメモリ操作する権限を限定し、安全な操作手段を提供
  - デバイス操作に失敗した際の原因特定・対応手段の提供

# 議論: OSによるデバイス操作手法の設計(続き)

- だれでもメモリアクセスできるとセキュリティ的に大問題
  - 安全な操作手段に加えて、総合的な監視手段の提供

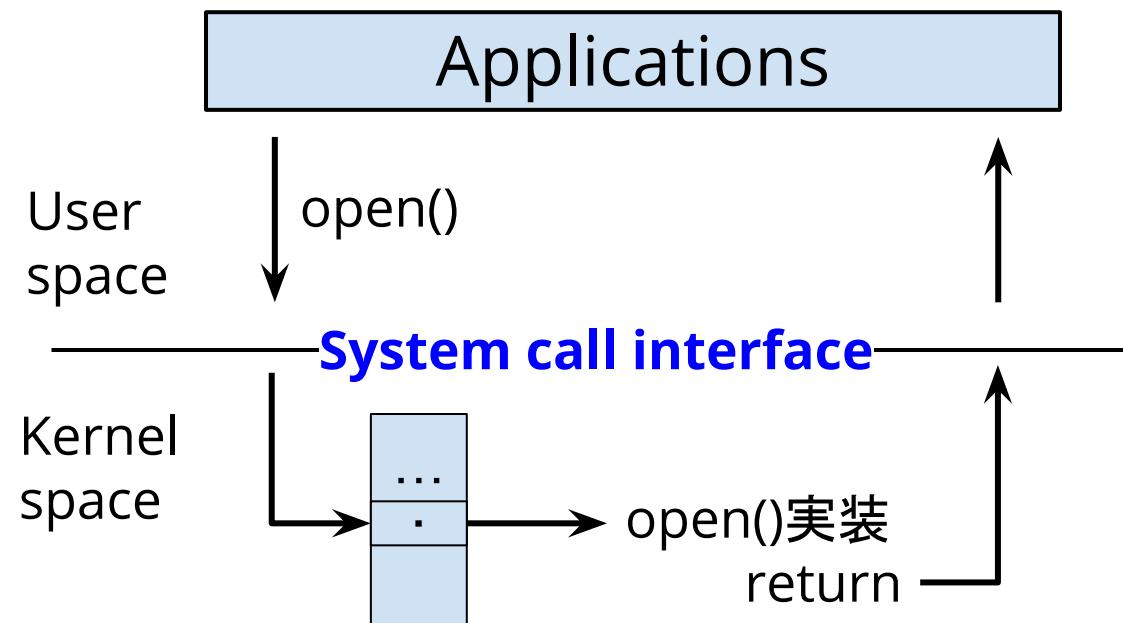
# 標準的なOS構成

- 多くのソフトウェアはユーザースペースのプロセスとして動作
- OS kernelは特権モードで動作
  - プロセスの作成と終了
  - ハードウェアへアクセス



# システムコール

- アプリケーションはシステムコール経由でカーネル機能を呼び出す (open, close, read, write, …)
- システムコールの権限を制限することでハードウェア制御方法を限定
- ライブラリ関数はシステムコールで構成される
  - printf, fgetc, scanf, etc.



## open (2)

- pathnameのファイルまたはデバイスをopenする
  - 詳しくはMacやLinuxで`man 2 open`
  - ファイルディスクリプタをreturn
  - Flagsでopenのオプションを指定

```
int open(const char *pathname, int flags, mode_t mode);
```

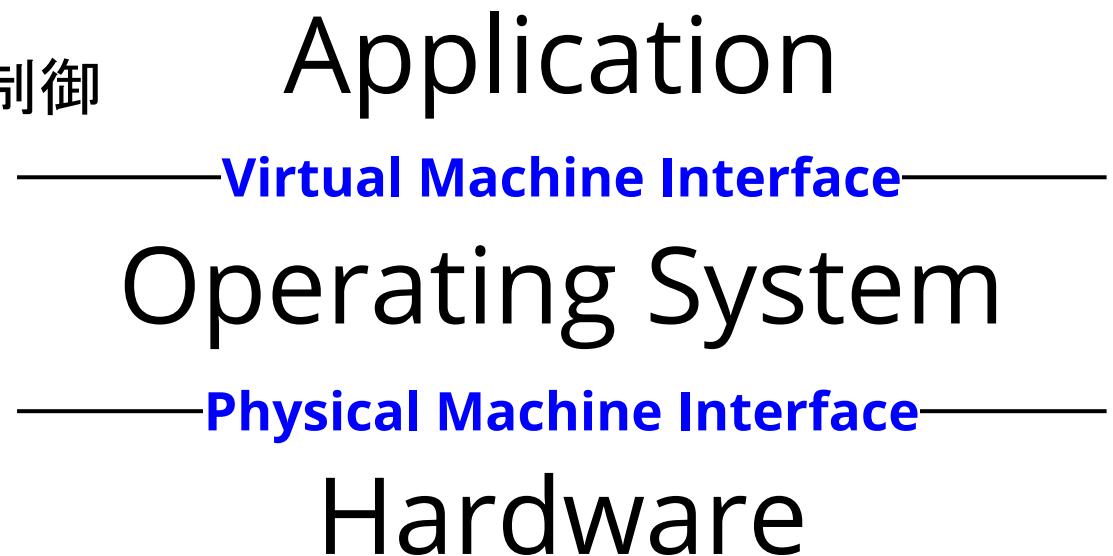
- O\_RDONLY open for reading only
- O\_WRONLY open for writing only
- O\_RDWR open for reading and writing
- O\_NONBLOCK do not block on open or for data to become available
- O\_APPEND append on each write
- O\_CREAT create file if it does not exist

# Error returns

- open(2)の場合
  - openが失敗した場合 -1 をreturn
  - 多くのシステムコールは-1を失敗としてreturn
  - 失敗理由はerrnoで確認
- #include <sys/errno.h>
  - 2=ENOENT “No such file or directory”
  - 13=EACCES “Permission Denied”
  - 16=EBUSY “Device or resource busy”

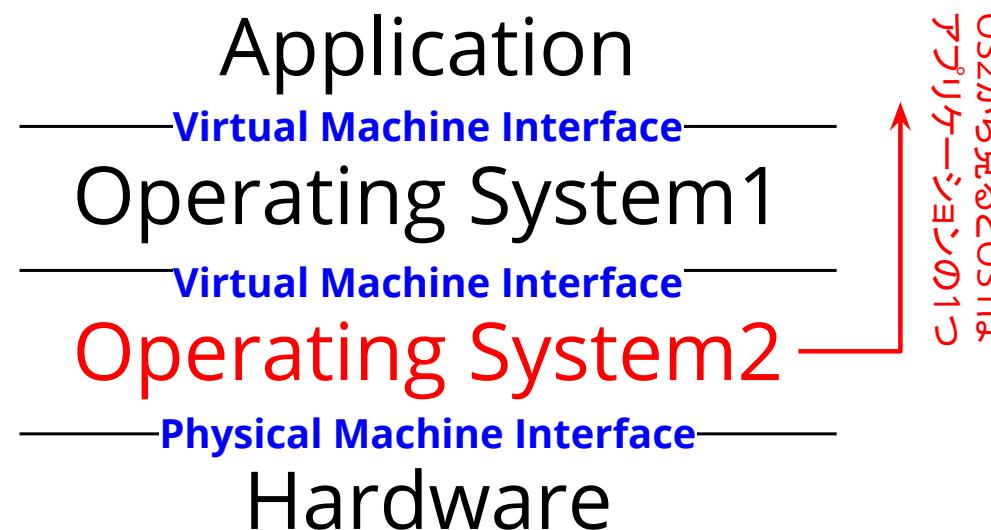
# まとめ

- OSとはハードウェアとアプリケーションの中間層
  - アプリに対してデバイスを便利で安全な操作手段を提供
  - ハードウェア特有の制御方法を隠蔽化



# OSの仮想化機能に向けて

- クラウドを支えるOSの仮想化機能
  - OS自体を1ソフトウェアとして柔軟に操作可能に



# ネットワークプログラミング

# 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
  - Python 言語
  - Go 言語

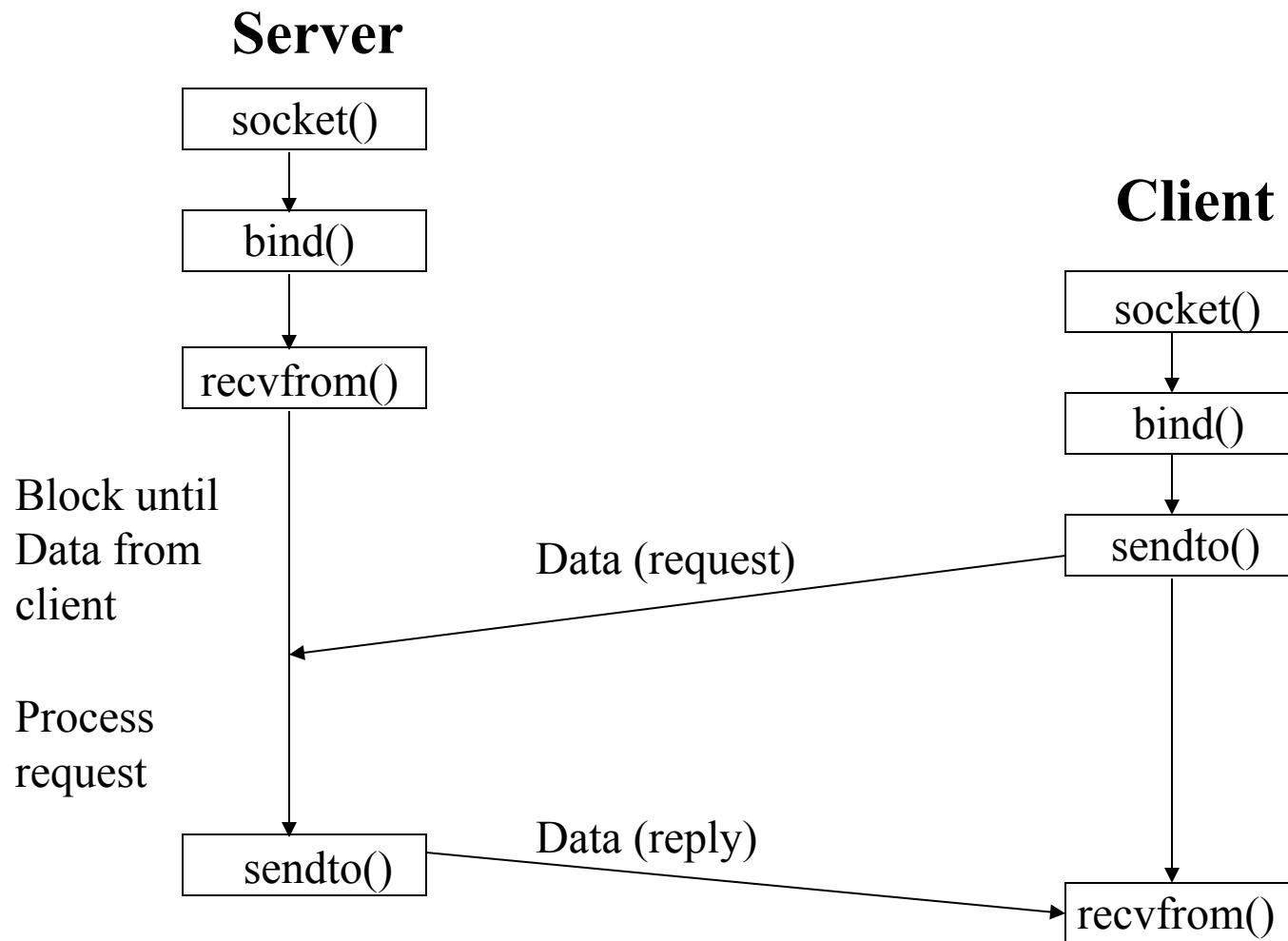
# C言語

- 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

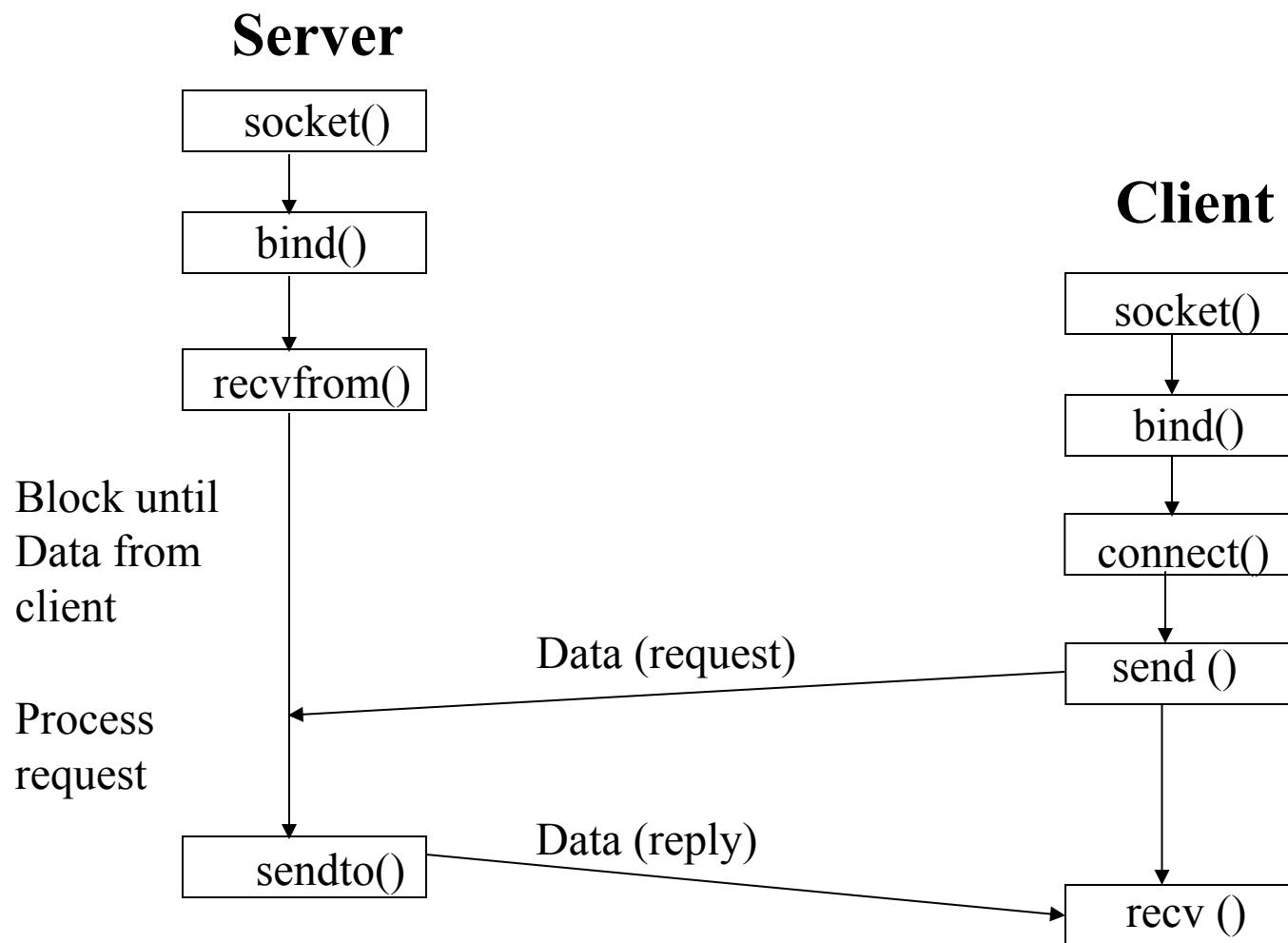
# システムコール

- 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

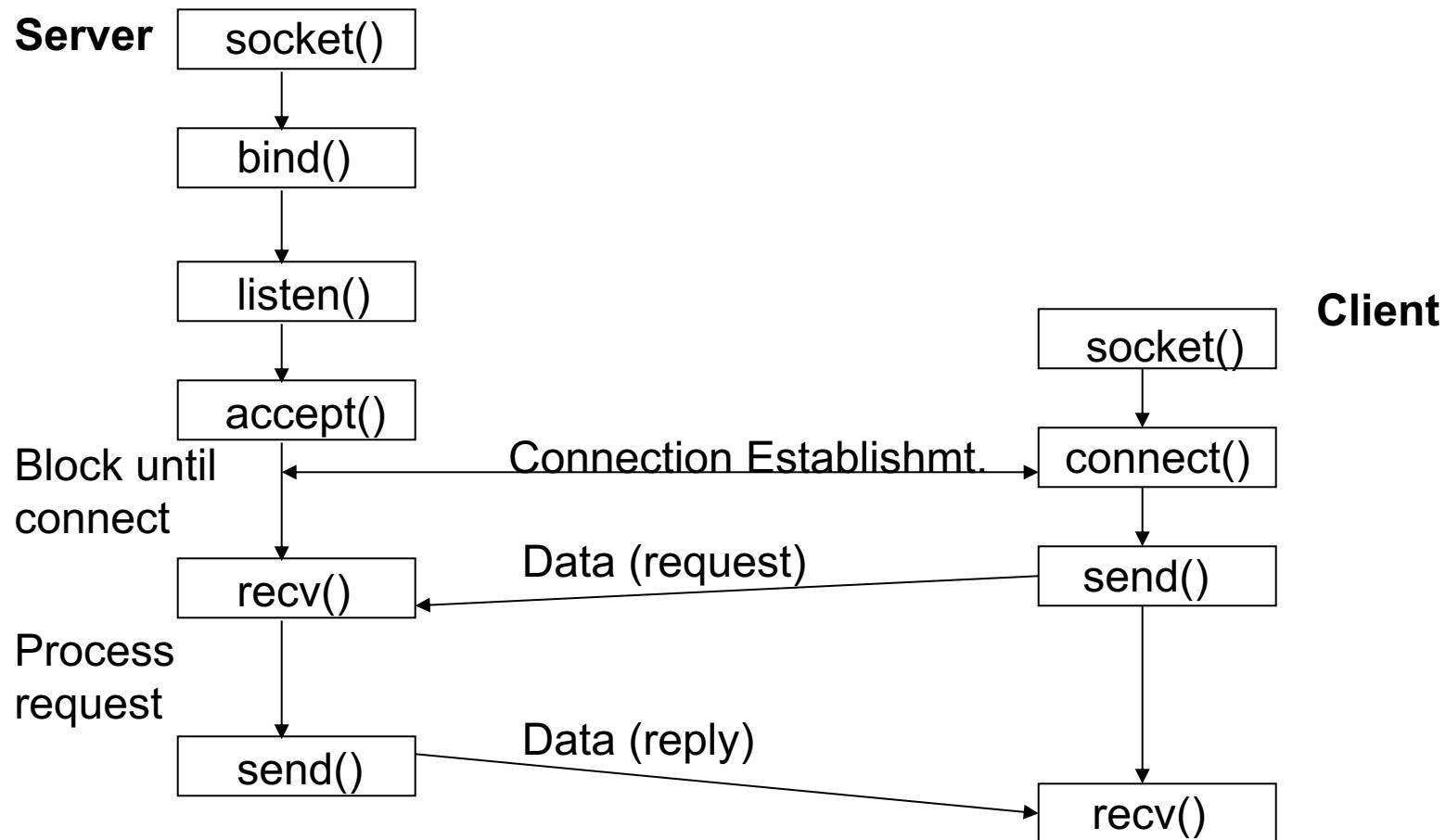
# Datagram example (UDP)



# Datagram example2 (UDP)



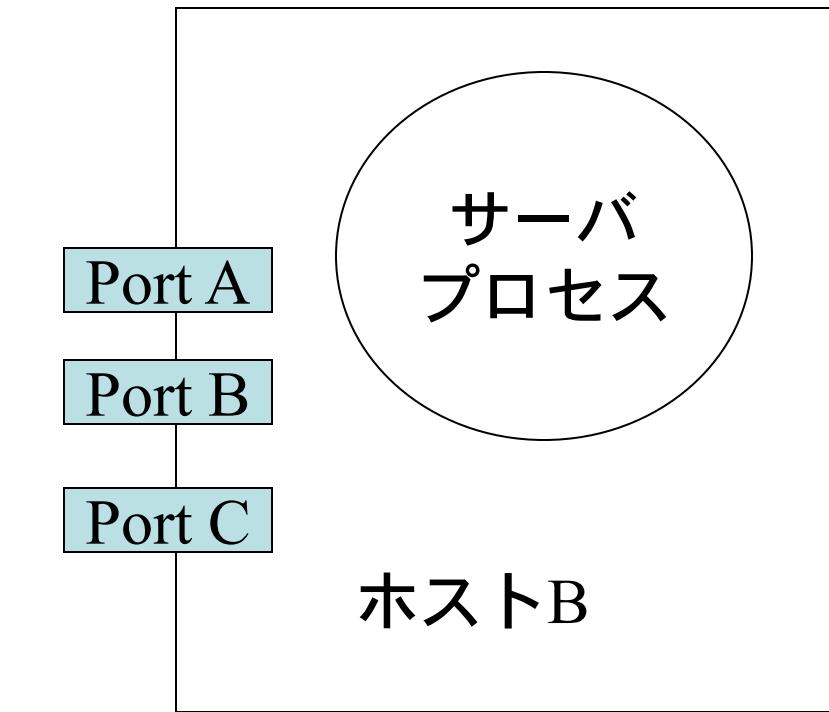
# Stream example (TCP)



# 初期状態



IP Address: xx.xx.xx.xx.

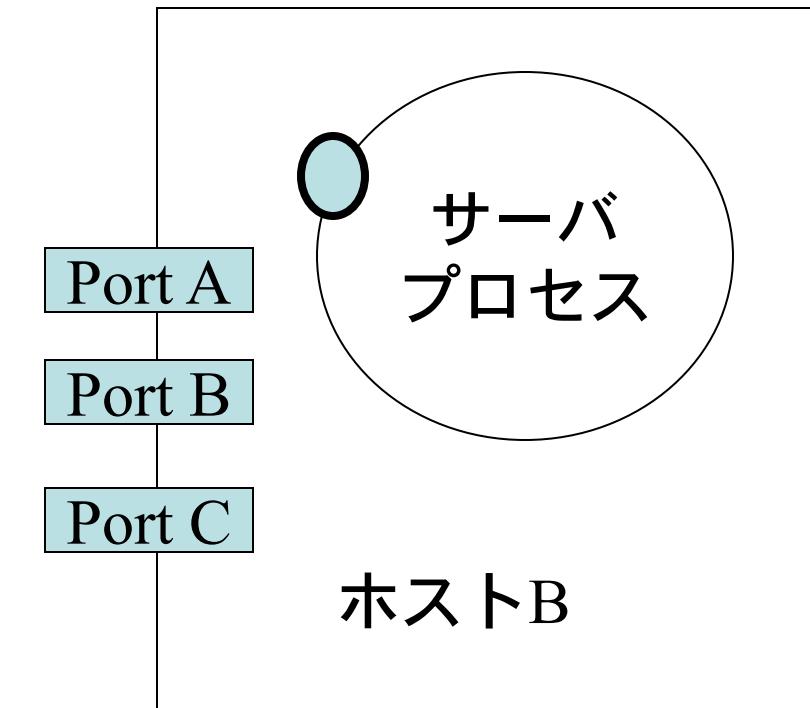


IP Address: xx.xx.xx.xx.

# Socketを開いた状態



Socketを開く



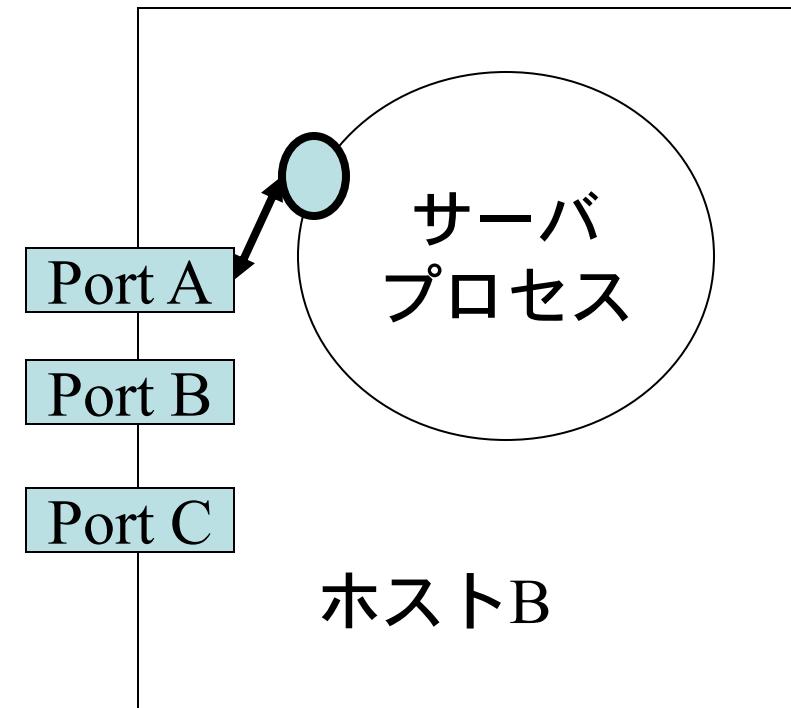
# bindした状態

Proto	LocalAddress
TCP	*.A



IP Address: xx.xx.xx.xx.

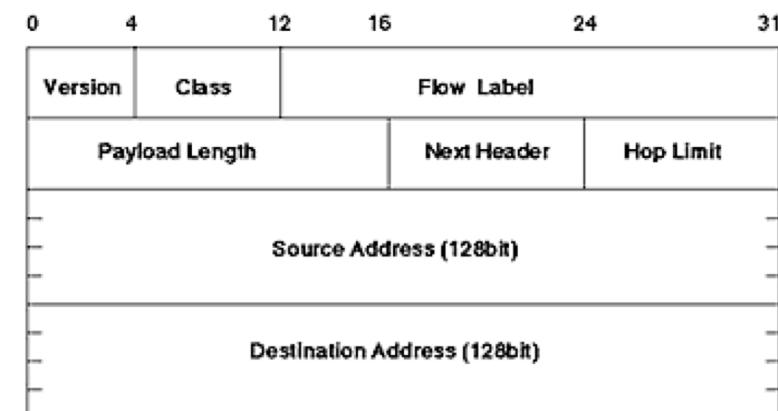
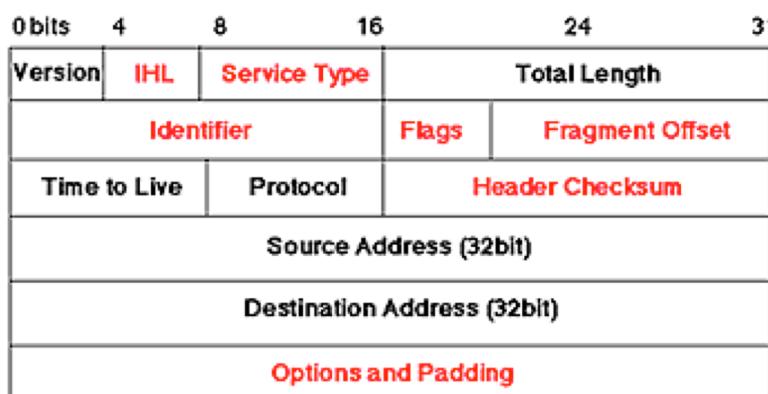
ForeignAddress	State
*.*	Closed



IP Address: xx.xx.xx.xx.

# IP Header Format

- IPv4 Header
- IPv6 Header



出展 : <http://www.ipv6style.jp/jp/tech/20030331/index.shtml>

# IPv4 Header 構造体

/usr/include/netinet/ip.h

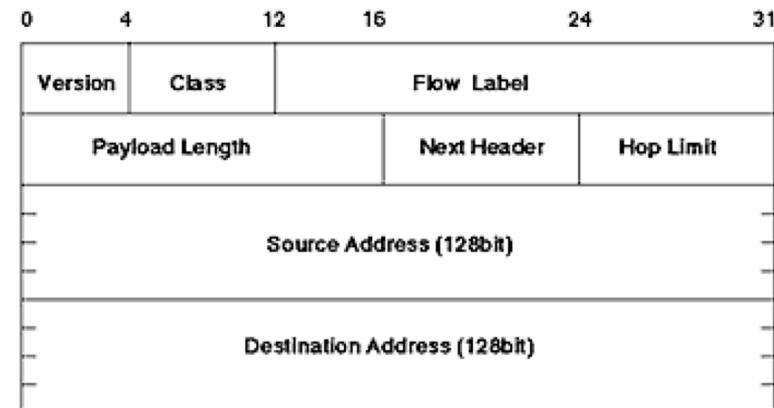
```
struct iphdr
{
#if __BYTE_ORDER == __LITTLE_ENDIAN
    unsigned int ihl:4;
    unsigned int version:4;
#elif __BYTE_ORDER == __BIG_ENDIAN
    unsigned int version:4;
    unsigned int ihl:4;
#else
#error "Please fix <bits/endian.h>"
#endif
    u_int8_t tos;
    u_int16_t tot_len;
    u_int16_t id;
    u_int16_t frag_off;
    u_int8_t ttl;
    u_int8_t protocol;
    u_int16_t check;
    u_int32_t saddr;
    u_int32_t daddr;
    /*The options start here. */
};
```

0 bits	4	8	16	24	31
Version	IHL	Service Type	Total Length		
			Identifier	Flags	Fragment Offset
			Time to Live	Protocol	Header Checksum
			Source Address (32bit)		
			Destination Address (32bit)		
			Options and Padding		

# IPv6 Header 構造体

/usr/include/netinet/ip6.h

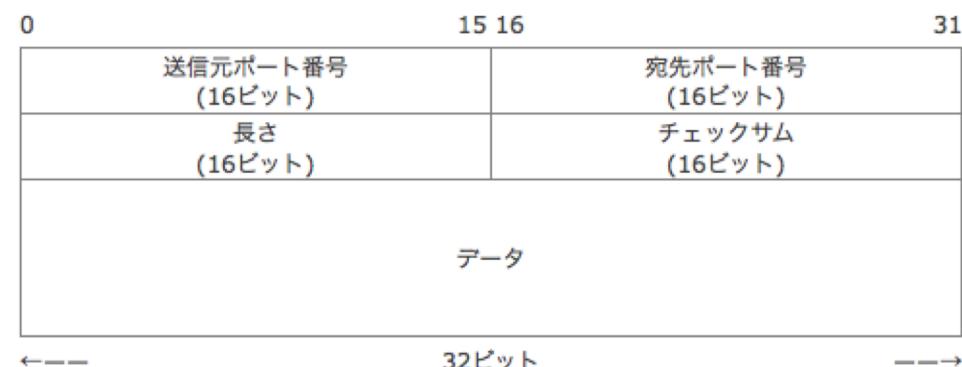
```
struct ip6_hdr
{
    union
    {
        struct ip6_hdrctl
        {
            uint32_t ip6_un1_flow; /* 4 bits version, 8 bits
TC,                                     20 bits flow-ID */
            uint16_t ip6_un1_plen; /* payload length */
            uint8_t ip6_un1_nxt; /* next header */
            uint8_t ip6_un1_hlim; /* hop limit */
        } ip6_un1;
        uint8_t ip6_un2_vfc; /* 4 bits version, top 4 bits
tclass */
    } ip6_ctlun;
    struct in6_addr ip6_src; /* source address */
    struct in6_addr ip6_dst; /* destination address */
    };
};
```



# UDP Header

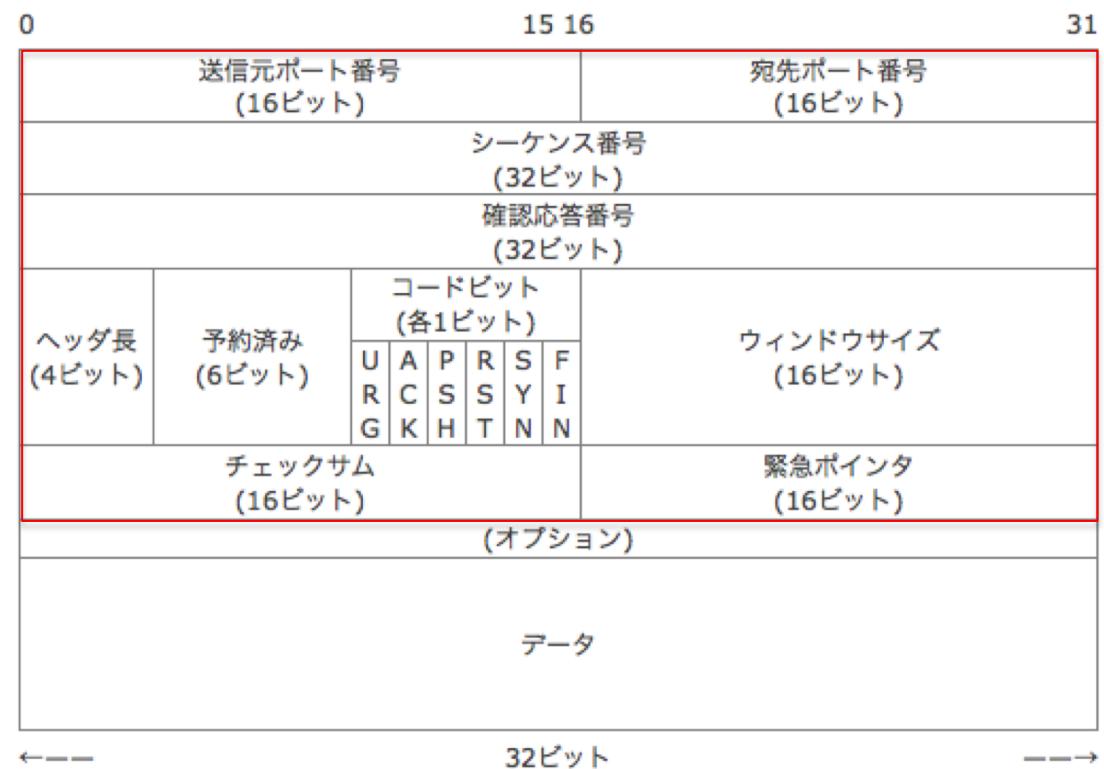
/usr/include/netinet/udp.h

```
struct udphdr
{
    u_int16_t source;
    u_int16_t dest;
    u_int16_t len;
    u_int16_t check;
};
```



# TCP Header

```
/usr/include/netinet/tcp.h
struct tcphdr
{
    u_int16_t source;
    u_int16_t dest;
    u_int32_t seq;
    u_int32_t ack_seq;
    #if __BYTE_ORDER == __LITTLE_ENDIAN
    u_int16_t res1:4;
    u_int16_t doff:4;
    u_int16_t fin:1;
    u_int16_t syn:1;
    u_int16_t rst:1;
    u_int16_t psh:1;
    u_int16_t ack:1;
    u_int16_t urg:1;
    u_int16_t res2:2;
    #elif __BYTE_ORDER == __BIG_ENDIAN
    u_int16_t doff:4;
    u_int16_t res1:4;
    u_int16_t res2:2;
    u_int16_t urg:1;
    u_int16_t ack:1;
    u_int16_t psh:1;
    u_int16_t rst:1;
    u_int16_t syn:1;
    u_int16_t fin:1;
    #endif
    u_int16_t window;
    u_int16_t check;
    u_int16_t urg_ptr;
};
```



# TCP を使ったクライアント作成

- 必要な関数
  - socket()
  - connect();
  - bind(); <- 使わなくても可能
  - memset();
  - htons();
  - inet\_ntop();
  - perror();
- 必要な構造体
  - sockaddr\_in
  - sockaddr\_in6

# コマンドライン引数:argc,argv

- コマンドの引数を用いるには?
  - cat -n file
  - cat file1 file2 file3

# コマンドライン引数の利用法

- main( int argc, char \*\*argv)
- main( int argc, char \*argv[])
  
- argc には引数の数
- argv[0] にはコマンド名
- argv[1] には1番目の引数
- argv[2] には2番目の引数

# コマンドライン引数 サンプルプログラム

```
#include<stdio.h>

int main(int argc, char *argv[]){
    int i;
    for(i = 0; i < argc; i++){
        printf("arg[%d]: %s\n", i, argv[i]);
    }
}
```

```
% gcc -o arg_example arg_example.c
% ./arg_example ABC DEF 123
arg[0]: ./arg_example
arg[1]: ABC
arg[2]: DEF
arg[3]: 123
```