ネットワークプログラミング第8回（ネットワークとプログラミング(3)）

２００９春学期　中村　修

授業Webページ

- SFC-SFS
  https://vu2.sfc.keio.ac.jp/sfc-sfs/

- 課題・授業資料などの情報を掲示します
  - **課題は毎回こちらに提出してください**！！
  - 今日の課題締め切り
    - 6/15(月)23:59まで！
    - 遅れて提出する方は要連絡

今期の授業スケジュール（予定）

- 第１回：イントロダクション (4/14)
- 第２回：C言語の基礎～関数・ポインタ (4/21)
- 第３回：C言語の基礎～コマンドライン引数 (4/28)
- 第４回：C言語の基礎～構造体・リスト構造 (5/12)
- 第５回：FileI/Oとシステムコール (5/19)
- 第６回：ネットワークとプログラミング（1） (5/26)
- 第７回：ネットワークとプログラミング（2） (6/2)
- **第８回：ネットワークとプログラミング（3） (6/9)**
- 第９回：応用ネットワークプログラミング（１） (6/16)
- 第１０回：応用ネットワークプログラミング（２） (6/23)
- 第１１回以降：ミニプロ (6/30, 7/7, 7/14)

第7回課題
UDPエコーサーバ

- UDPエコーサーバを作成する
- 待ち受けするポート番号(getaddrinfo時に指定するポート番号)は、学籍番号下5桁を利用
  - 例)学籍番号　80749358
    getaddrinfo(NULL, “**49358**”, &hint, &res);
    bind(sock, ai->ai_addr, ai->ai_addrlen);
- クライアントは前回作成したクライアントを使用し、送信先サーバのポートを変えデータを送信する
- 確認用URL
  - http://long.sfc.wide.ad.jp/~tatsurou/09a-npro/06-logviewer.cgi

```
#include <stdio.h>
#include <netdb.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>

int
main (int argc, char *argv[]){
   struct addrinfo udp6serv = {
   AI_PASSIVE, // (サーバ用)
   AF_INET6,
   SOCK_DGRAM,              // UDP
   0, 0, NULL, NULL, NULL  // 他の条件
はなし
   };
   struct addrinfo *ai, *res;
   struct sockaddr_storage peer;
   socklen_t addrlen_peer;
   char buf[512];
   int sd;
   int ret;
```

```
   int readlen, sendlen;

   /* open socket to receive UDP/IPv6 packet */
   getaddrinfo (NULL, “49358”, &udp6serv,
&res);
   for (ai = res; ai; ai = ai->ai_next) {
       sd = socket (ai->ai_family,
        ai->ai_socktype, ai->ai_protocol);
       if (sd == -1) {
              perror (”socket() failed”);
              continue;
       }
       ret = bind (sd, ai->ai_addr,
              ai->ai_addrlen);
       if (ret == -1){
              perror (”bind() failed”);
              close (sd);
              continue;
       }
       break;
   }
   if (res)
       freeaddrinfo (res);
```

```
  /* receive udp packet and echo to display */
  while (1)
  {
      /* receive message */
      memset (&peer, 0, sizeof (peer));
      addrlen_peer = sizeof (peer);
      readlen = recvfrom (sd, buf, sizeof (buf), 0,
          (struct sockaddr *) &peer, &addrlen_peer);
      if (readlen < 0)
             perror ("recvfrom () failed");
      write (1, buf, readlen);

      sendlen = sendto (sd, buf, readlen, 0,
          (struct sockaddr *) &peer, addrlen_peer);
      if (sendlen == -1)
             perror ("sendto () failed");
  }
 return 0;
}
```

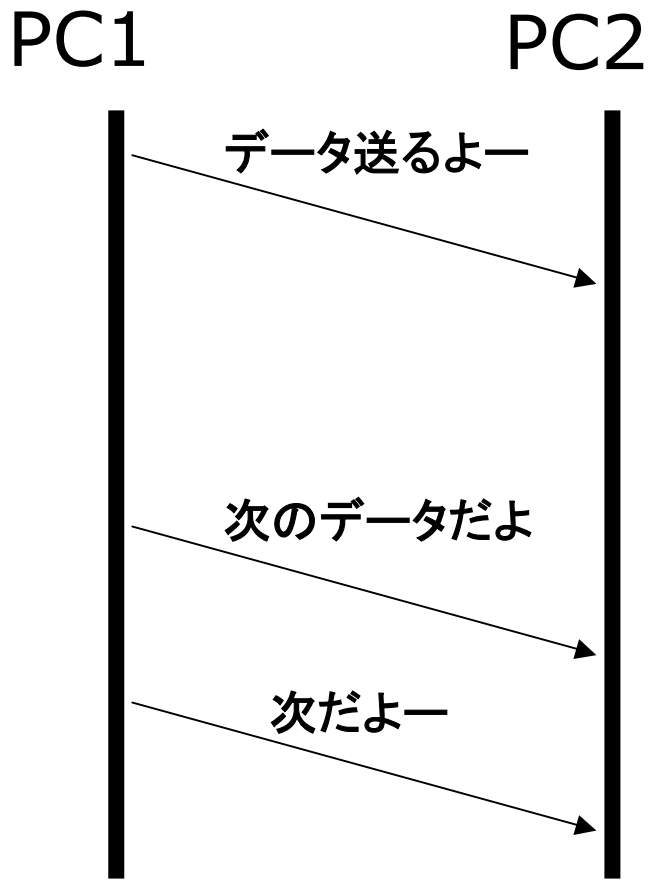
今日のお題

- 講義
  - TCPとは？
  - TCPエコークライアント
- 実習/課題：データの送受信
  - TCP echoクライアント作成
  - TCP echoサーバ作成

UDP

(User Datagram Protocol)

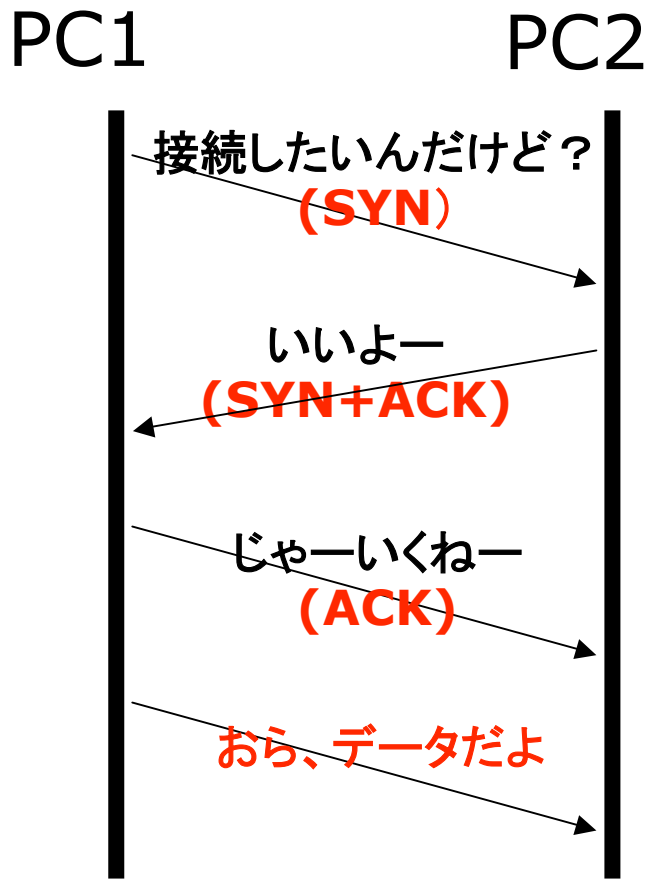
- コネクションレス型のプロトコル
- 高速なデータ通信向き
- 身近な例
  - ビデオストリーミング
  - DNS/DHCP



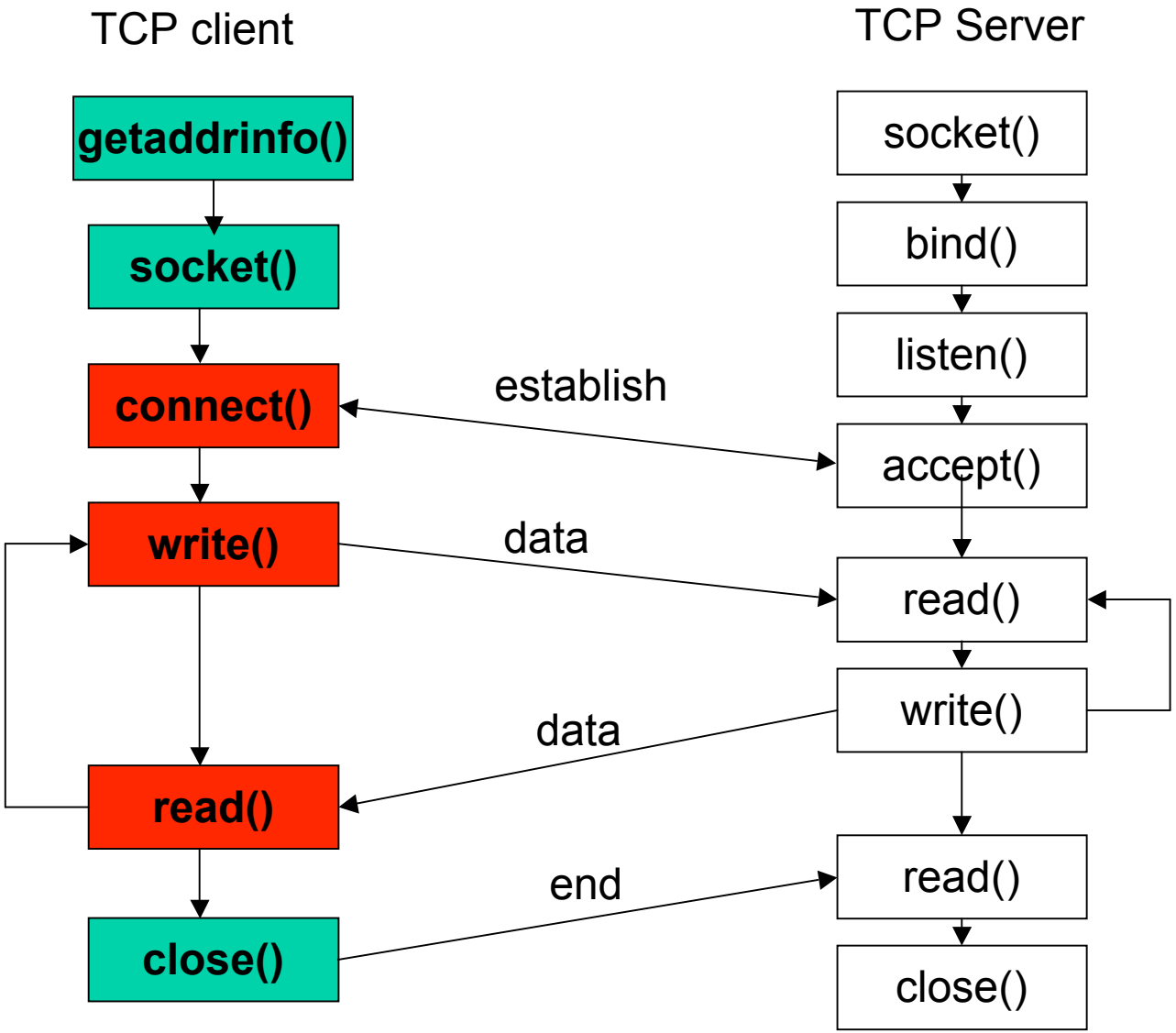
TCP

（Transmission Control Protocol）

- 信頼性のある通信を提供
- 「コネクション」によって、通信相手との接続状態を識別
- 身近な例
  - 電子メール
  - WEB
- TCPの恩恵
  - 送信エラーを検出可能
  - 輻輳制御(混雑状態に応じ、送信レートを制御)

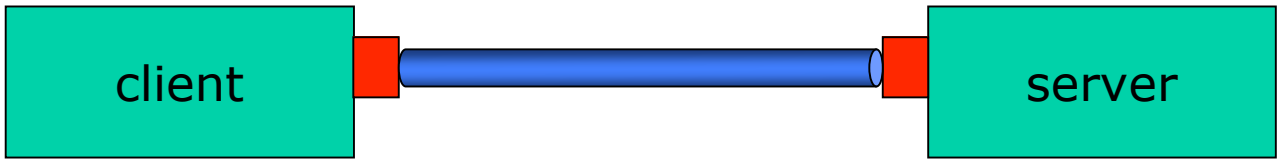


TCP 通信の流れ(client)



connect()システムコール

- **#include <sys/types.h>**
  #include <sys/socket.h>
  int connect(int sockfd, const struct sockaddr *serv_addr, socklen_t addrlen);
  - int sockfd; (ソケット記述子)
  - struct sockaddr *serv_addr; (プロトコル対応のアドレス構造体へのポインタ)
  - socklen_t addrlen; (アドレス構造体のサイズ)
- 別のソケットとの接続要求
- 返り値
  - 接続に成功するとゼロを返す．失敗すると-1を返す



(再掲)getaddrinfo

- **getaddrinfo(const char *nodename,**
  const char *servname,
  const struct addrinfo *hints,
  struct addrinfo **res);
- nodename=ホスト名, servname=ポート, hintsは情報のタイプに関するヒント, resは結果のaddrinfoのリストが格納される
- 返り値:成功すると 0 を返し、失敗すると以下の非 0 のエラーコードのいずれかを返す。
- サーバの場合は
  - nodenameはNULL
  - hintsにAI_PASSIVEフラグを設定する

addrinfo構造体

・ <netdb.h>

```
struct addrinfo {
    int      ai_flags;        /* AI_PASSIVE, AI_CANONNAME */
    int      ai_family;       /* PF_xxx */
    int      ai_socktype;     /* SOCK_xxx */
    int      ai_protocol;     /* 0 or IPPROTO_xxx for IPv4 and IPv6 */
    size_t   ai_addrlen;      /* length of ai_addr */
    char     *ai_canonname;   /* canonical name for hostname */
    struct   sockaddr *ai_addr;      /* binary address */
    struct   addrinfo *ai_next;      /* next structure in linked list */
};
```

(再掲)read()システムコール

- int read(int d, void *buf, size t nbytes)
  - 引数
    - int d: ファイルディスクリプタ
    - void *buf: 読み込んだデータを保存するための配列(バッファ)
    - size_t nbytes: 1回のread()で読める最大バイト数(バッファの大きさ)
  - 返り値
    - int: 実際に読み込んだバイト数
      - 0より大きい: 読み込み成功
      - 0: ファイルの最後(EOF)に到達
      - 0より小さい: エラー

(再掲) write()システムコール

- **int write(int d, void *buf,size t nbytes)**
  - 引数
    - int d: ファイルディスクリプタ
    - void *buf: 書き込むデータを保存している配列・ポインタ(バッファ)
    - size_t nbytes: 1回のwrite()で書き込むバイト数(書き込むデータの大きさ)
  - 返り値
    - int: 実際に書き込んだバイト数
      - 0より大きい: 書き込み成功
      - 0より小さい: エラー

練習:TCPでechoクライアント

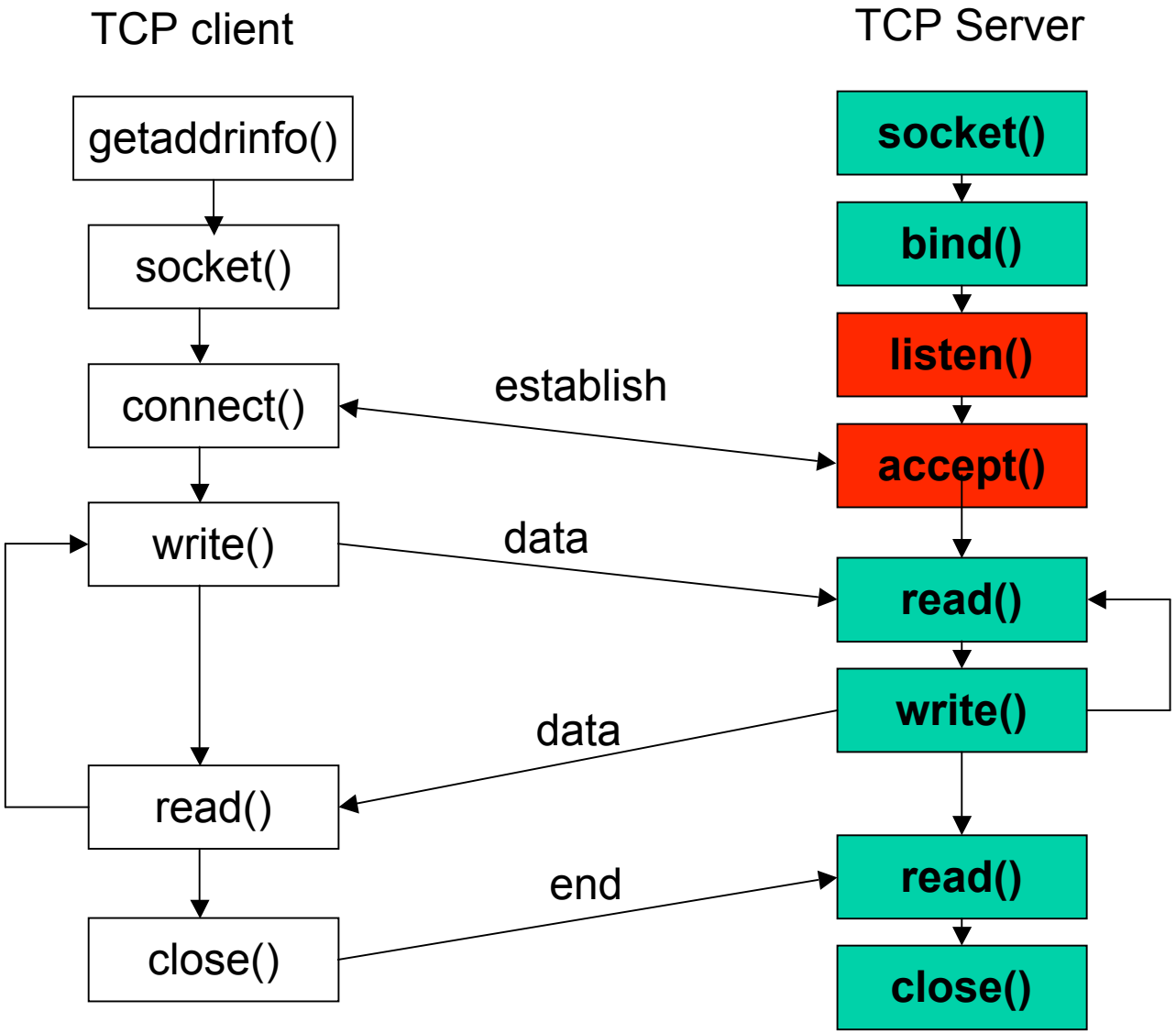
- TCPでechoクライアントを書いて見ましょう
  - 第一引数にホスト名，第2引数にポート番号を指定
  - 以下のサーバにアクセス
    - Host name: long.sfc.wide.ad.jp
    - Port number: 5525/tcp
  - 結果は以下のページにて参照可能
    http://long.sfc.wide.ad.jp/~tatsurou/09a-npro/

telnetコマンドとTCPサーバ

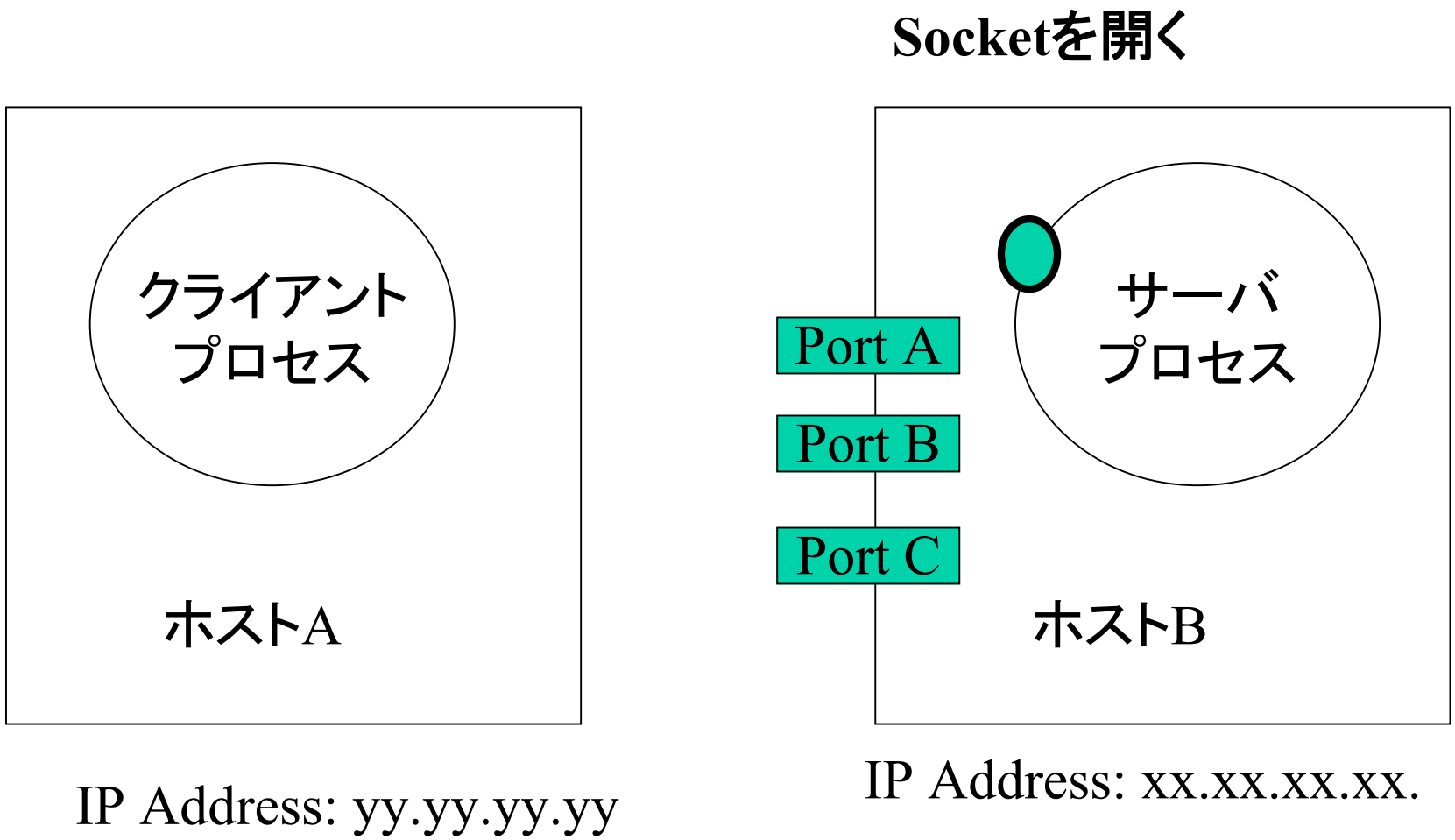
- telnetコマンド
  - ポート番号23でサービスされる、リモートコンソール
  - telnet www.sfc.keio.ac.jp
- コマンドオプションにて、接続ポート番号を変えられる
  - telnet www.sfc.keio.ac.jp 80
  - telnet long.sfc.wide.ad.jp 5525

TCPエコーサーバ

TCP 通信の流れ(server)



Socketを開いた状態

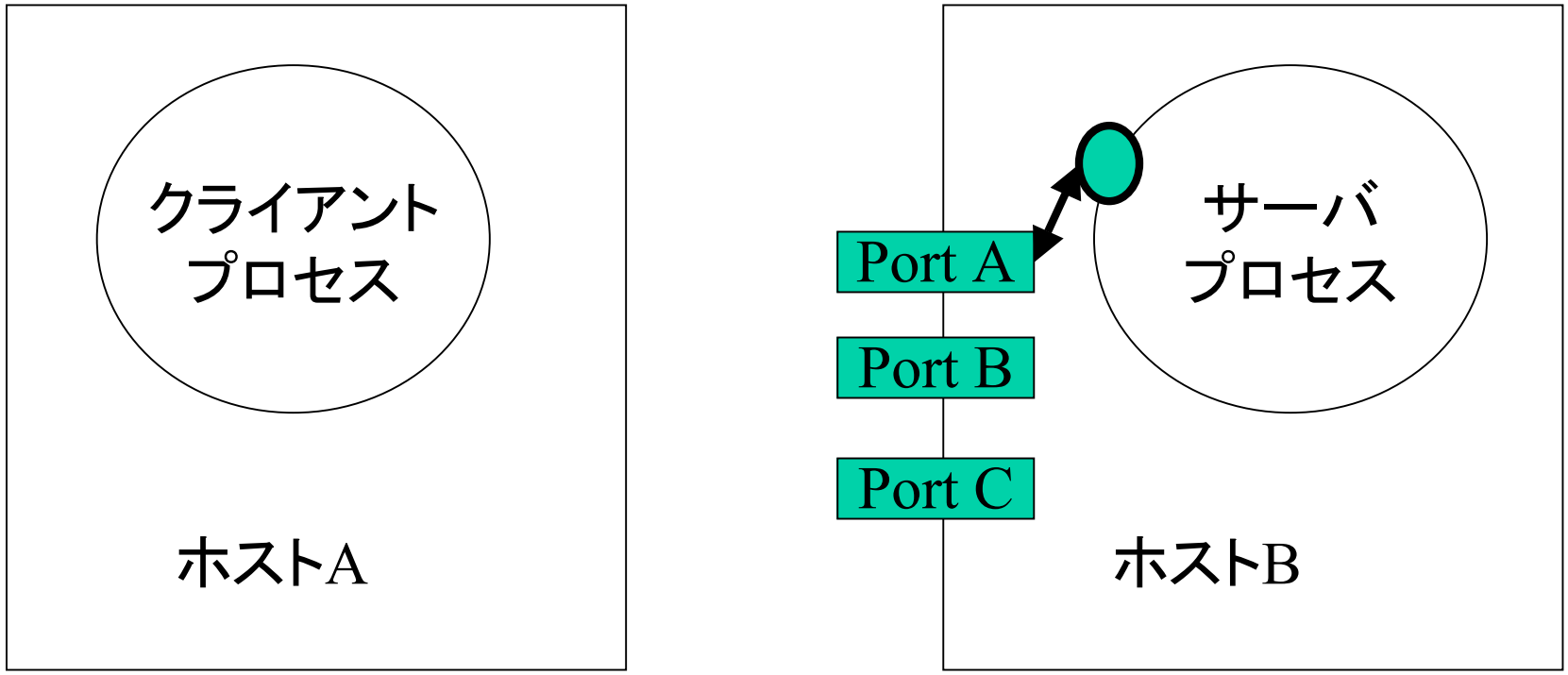


bind()システムコール

- int bind(int s, const struct sockaddr *addr, int addrlen)
- 用意したsokcetのアドレスを実際にsocketと結びつける
  - IPアドレスとTCP/UDPのポート番号の組
  - 開いたソケットのステートはclosed
- 実際のコードでは…
  - bind(listenfd, (struct sockaddr *) &servaddr, sizeof(servaddr)
- 成功なら0、エラーなら-1の返り値

bindした状態

Proto	LocalAdddress	ForeignAddress	State
TCP	*.A	*.*	Closed



IP Address: yy.yy.yy.yy

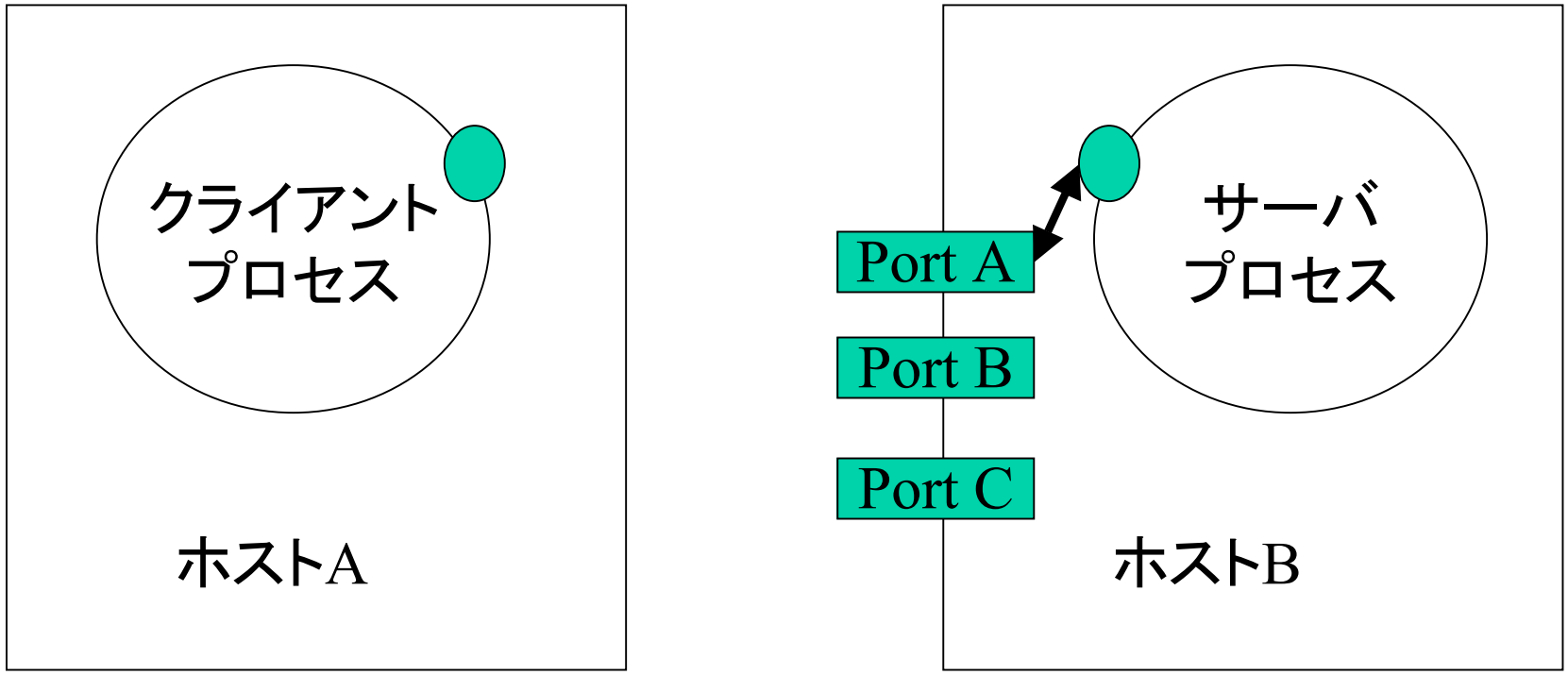
IP Address: xx.xx.xx.xx.

listen()システムコール

- int listen(int s, int backlog)
- 用意したsocketを待ち受け準備状態にする
  - ソケットの状態をCLOSEDからLISTENへ
- backlogはキューの長さ(backlog分の要求を保持できる)
  - accept()されるまでbacklog分のキューに保持
  - キューがあふれると、その要求は無視
- 実際のコードでは…
  - listen(listenfd, LISTENQ)
- 成功なら0、エラーなら-1の返り値

Listen()

Proto	LocalAdddress	ForeignAddress	State
TCP	*.A	*.*	Listen



IP Address: yy.yy.yy.yy

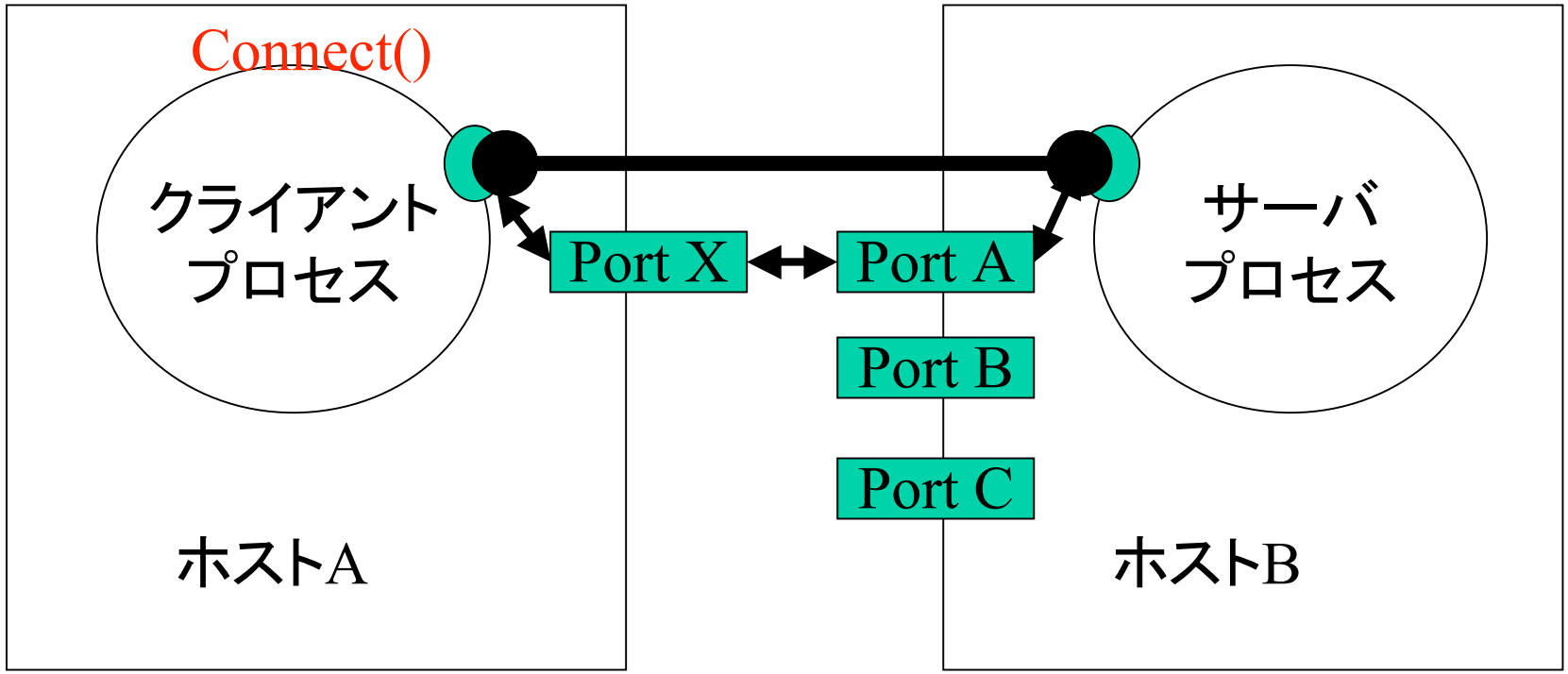
IP Address: xx.xx.xx.xx.

accept()システムコール

- **int accept(int s, strucet sockaddr *addr, int *addrlen)**
- キューで待っている接続要求を取り出して、そのクライアントと通信するためのディスクリプタを作成して、返す
- clientにはクライアントのsocketのアドレス、namelenにはclientのサイズ
- 実際のコードでは…
  - accept(listenfd, (struct sockaddr *) &cliaddr, &clilen)
- 成功なら新しいFD番号、エラーなら-1の返り値

accept()

Proto	LocalAdddress	ForeignAddress	State
TCP	xx.xx.xx.xx.A	yy.yy.yy.yy.X	Establish

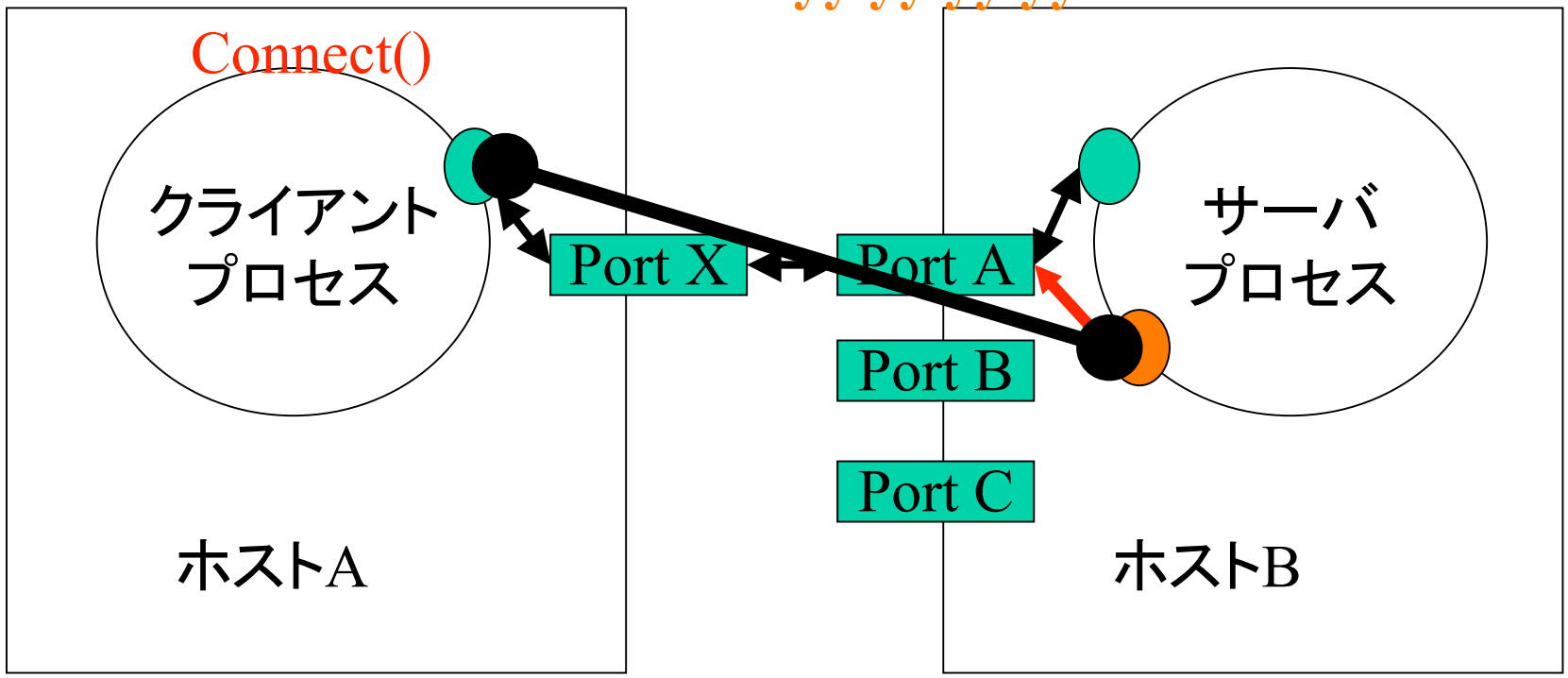


IP Address: yy.yy.yy.yy

IP Address: xx.xx.xx.xx.

accept()

Proto	LocalAdddress	ForeignAddress	State
TCP	*.A	*.*	Listen
TCP	xx.xx.xx.xx.A	yy.yy.yy.yy.X	Establish



IP Address: yy.yy.yy.yy

IP Address: xx.xx.xx.xx.

使い方例

```
int socket_fd, accept_fd;
int client_addrlen;
int readlen;
struct sockaddr_in server, client;

getaddrinfo(NULL, argv[1], &hints, &res);
socket_fd = socket(ai->ai_family, ai->ai_socktype, ai->ai_protocol);
bind(socket_fd,   ai->ai_addr,   ai->ai_addrlen);
listen(socket_fd, 5);
memset((void *)&client, 0, sizeof(client));
client_addrlen = sizeof(client)
accept_fd = accept(socket_fd,(struct sockaddr *)&client, (int *)&client_addrlen);
      ---(read/writeなどの処理)----
close(accept_fd);
close(socket_fd);
}
```

実習・課題
TCPエコーサーバの作成

- TCPを用いたechoサーバを作ろう。
  - 第一引数にポート番号を指定しよう
    %./a.out 49538
  - **ポート番号は、学籍番号下5桁利用**
- できたら、先に作成したTCPエコークライアントを用いて確認
- または、telnet コマンドにても確認可能