ネットワークプログラミング第9回（ネットワークとプログラミング（4））

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
    - 6/22(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：ネットワークとプログラミング（4）　（6/16）**
- 第１０回：応用ネットワークプログラミング　（6/23）
- 第１１回以降：ミニプロ　（6/30, 7/7, 7/14）

ミニプロジェクト(ミニプロ)について

- グループでプログラムを作る
- OSは問わない
  - 一応、公式サポート環境はccx
  - 他の環境でもTA/SAは（できるだけ）頑張ります
- ネットワークを使うプログラム
  - プログラムの難易度
  - **独創性！**

ミニプロの提出

- 提出は、ソースコードならびにレポート

- 提出ソースコードにはコメントを付加
  - 日本語 or 英語
- レポートには以下の項目を含める
  - 何を作ったのか
  - 使用方法
  - 面白さ
  - 実行環境
  - グループ内の、各自の担当個所

ミニプロのプレゼン

- 中間発表
  - 7/7(予定)
  - 何を作るかをプレゼンする
    - A41枚程度のレジュメを作成する

- 最終発表(7/14予定)
  - 授業最終回を予定
  - できあがりをプレゼンする
  - 評価

ミニプロのメンバー

- 申請したい人は6/19(金)までにTA/SAにメール
  - 最大人数: 3人
- 申請が無い人はこちらで振り分けます
- グループが決まり次第メールで連絡します！

ミニプロの例

- ネットワークゲーム
  - リバーシ
  - ○×ゲーム
  - 陣取りゲーム
- コミュニケーションツール
  - アドホックSkype

第8回課題
TCPechoクライアント

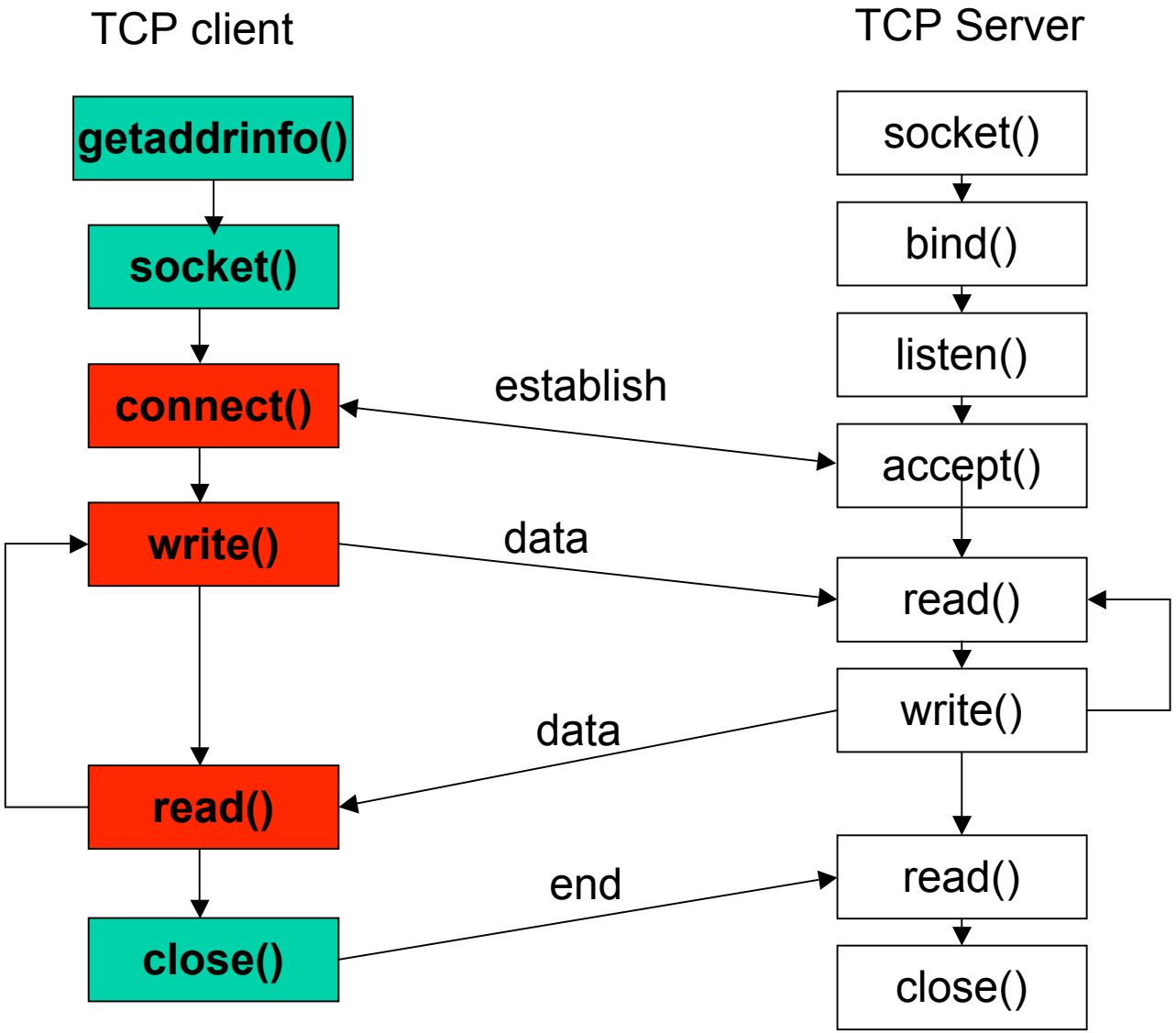
- TCPでechoクライアントを書いて見ましょう
  - 第一引数にホスト名，第2引数にポート番号を指定
  - 以下のサーバにアクセス
    - Host name: long.sfc.wide.ad.jp
    - Port number: 5525/tcp
  - 結果は以下のページにて参照可能
    http://long.sfc.wide.ad.jp/~tatsurou/09a-npro/08-logviewer.cgi

```
#include <stdio.h>
#include <netdb.h>
#include <stdlib.h>
#include <string.h>
#include <sys/types.h>
#include <sys/uio.h>
#include <unistd.h>

int main(int argc, char *argv[])  {
   int sock;
   int ret;
   char buf[256];
   struct addrinfo *ai, *res;
   struct addrinfo hints = {0,
AF_INET6, SOCK_STREAM , 0, 0, NULL,
NULL, NULL};
```

```
   ret = getaddrinfo(argv[1], argv[2], &hints,
&res);
   if (ret != 0){
       fprintf(stderr, "getaddrinfo (%s)¥n",
gai_strerror(ret));
       exit(-1);
   }

   for(ai = res; ai; ai = ai->ai_next){
       sock = socket(ai->ai_family, ai-
>ai_socktype, ai->ai_protocol);
       if (sock < 0){
              perror("socket");
              exit(-1);
       }
       ret = connect(sock, ai->ai_addr, ai-
>ai_addrlen);
       if (ret < 0){
              perror("connect");
              exit(-1);
       }
       break;
   }
```

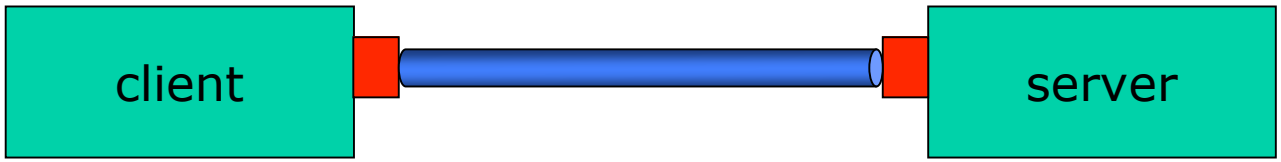
```
while (1){
    int len = sprintf(buf, "%s", "[90749358]:");
    ret = read (fileno(stdin), buf + len, sizeof(buf) - len);
    if (ret < 0){
        perror("read");
        printf("stdin is %d¥n", fileno(stdin));
        exit(-1);
    }
    ret = write(sock, buf, ret + len);
    if (ret < 0){
        perror("sendto");
        exit(-1);
    }
    memset(buf, 0, sizeof(buf));
    ret = read(sock, buf, sizeof(buf));
    if (ret < 0){
        perror("recvfrom");
        exit(-1);
    }
    printf("%s¥n", buf);
}
freeaddrinfo(res);
return 0;
```

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

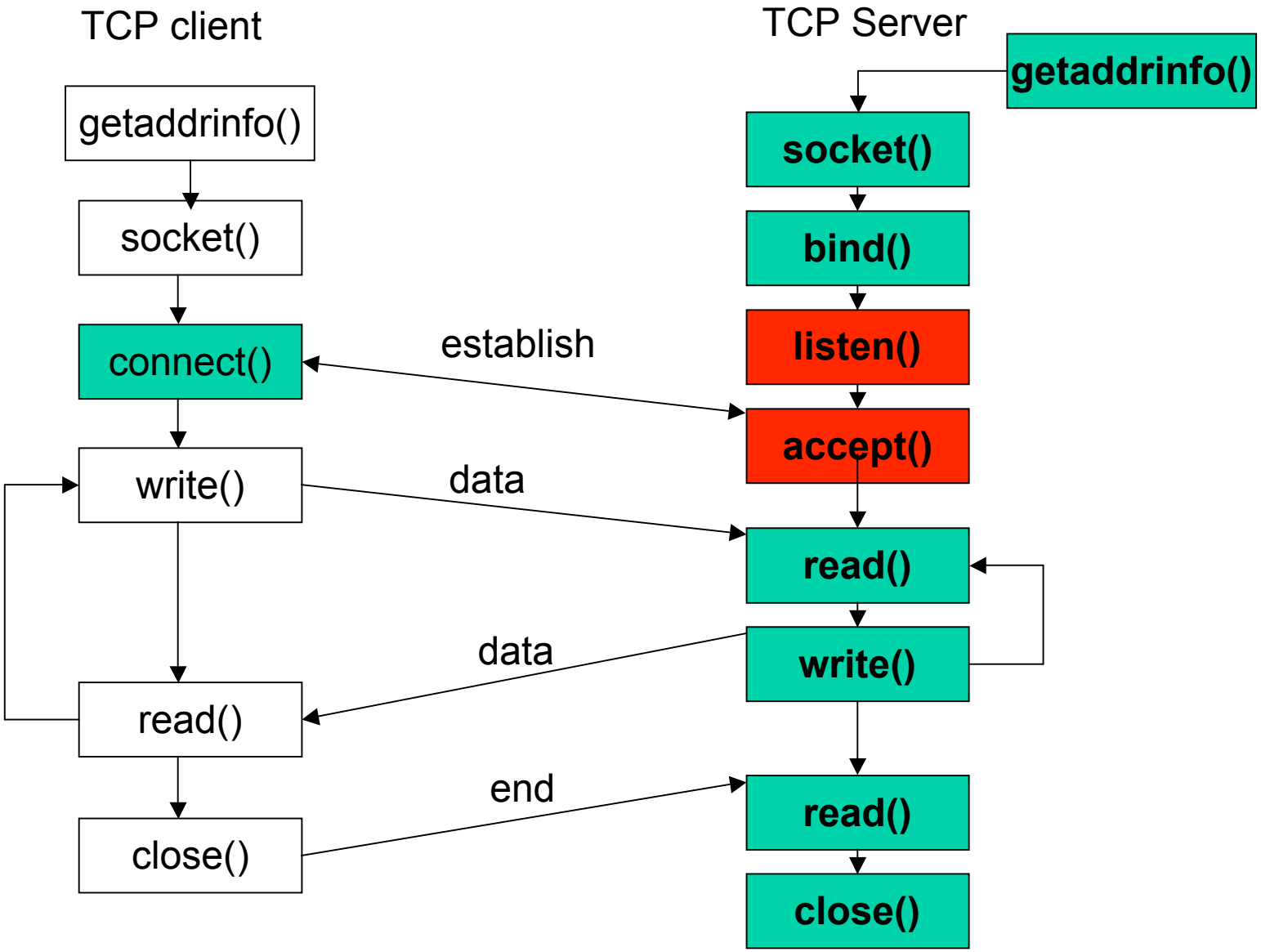


今日のお題

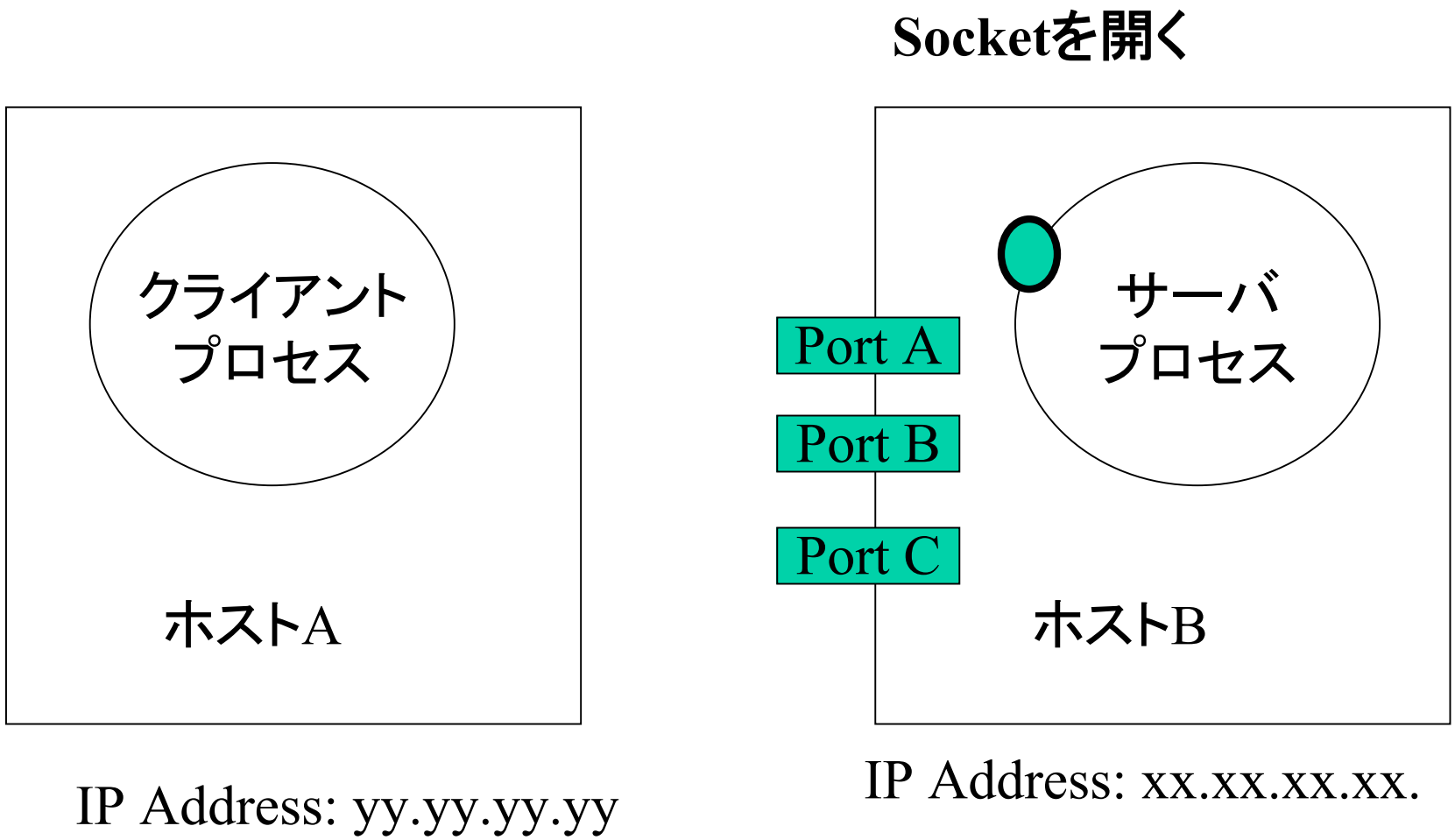
- 講義
  - TCPエコーサーバ
- 実習/課題:データの送受信
  - TCP echoサーバ作成

TCPエコーサーバ

TCP 通信の流れ(server)



Socketを開いた状態

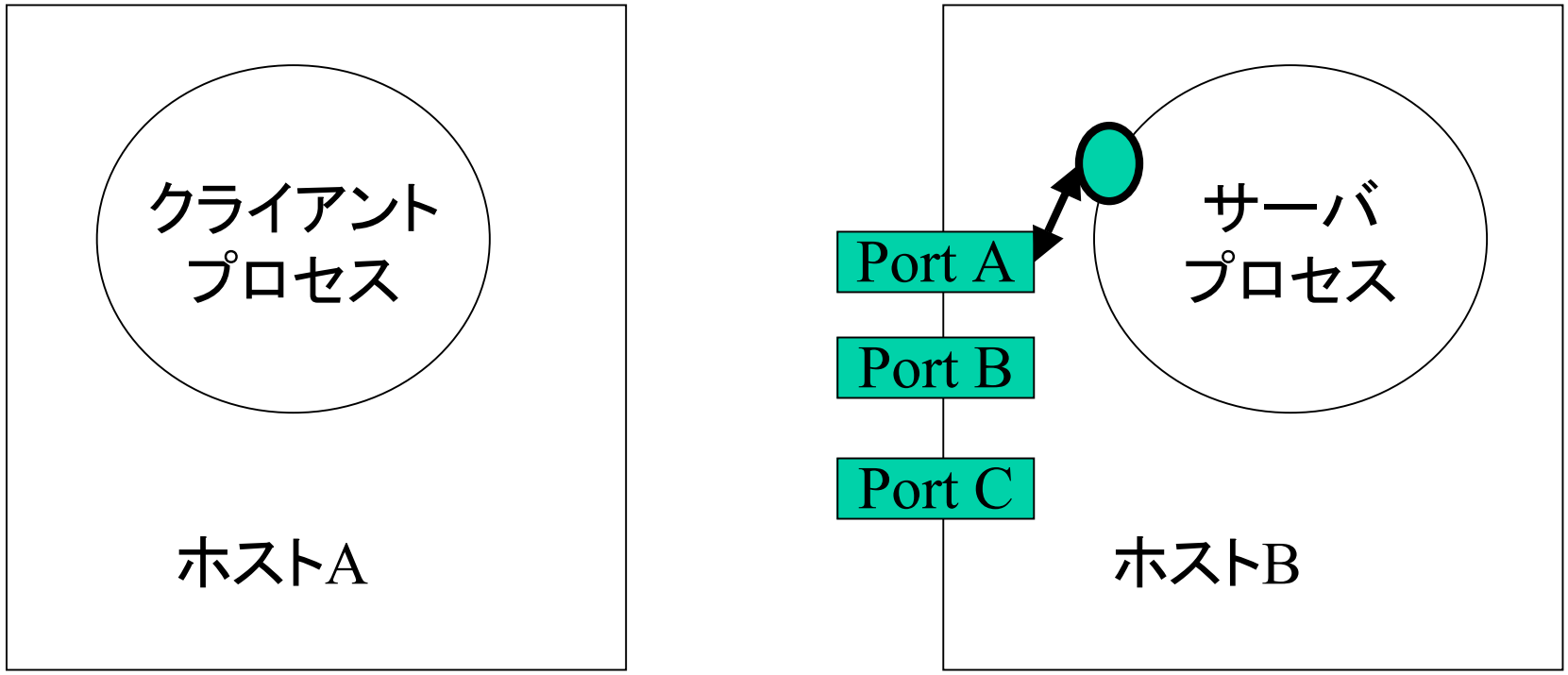


bind()システムコール

- 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

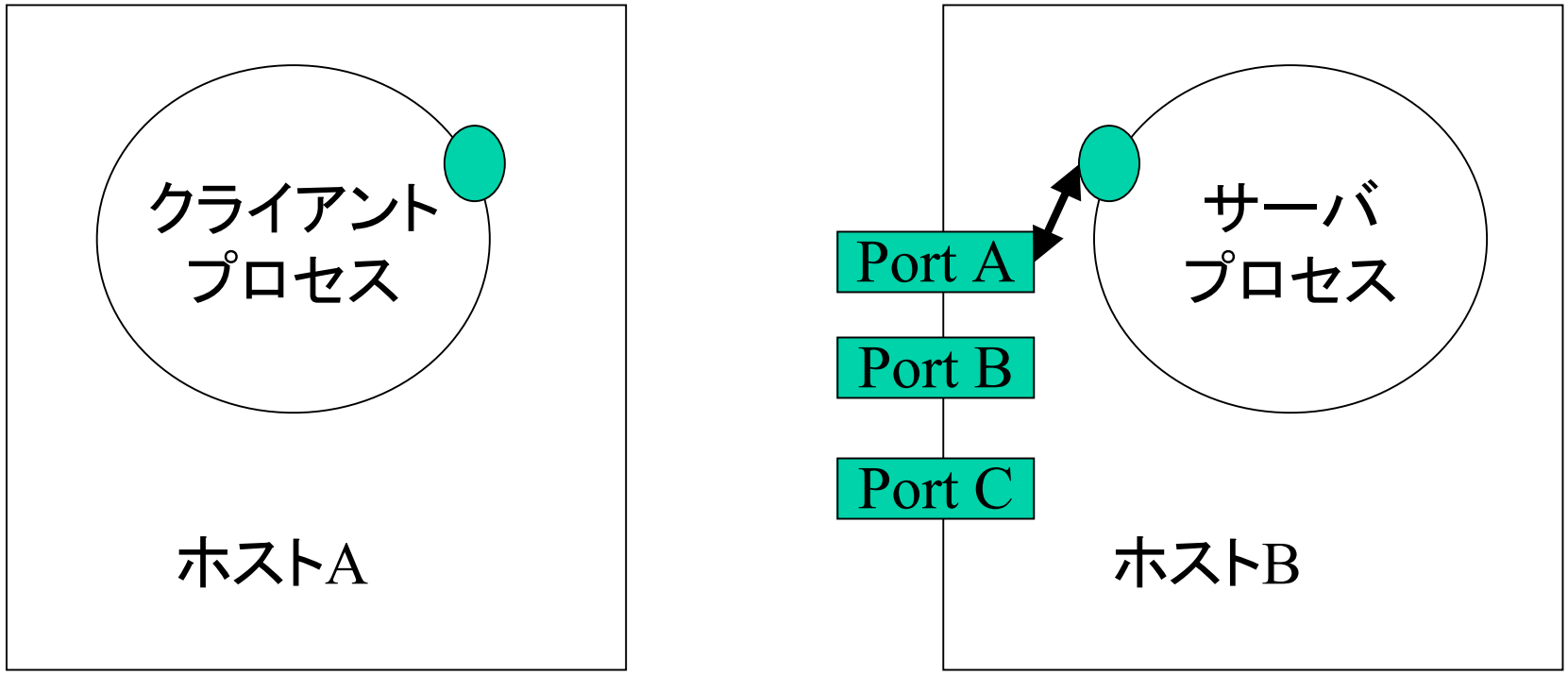
IP Address: xx.xx.xx.xx.

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

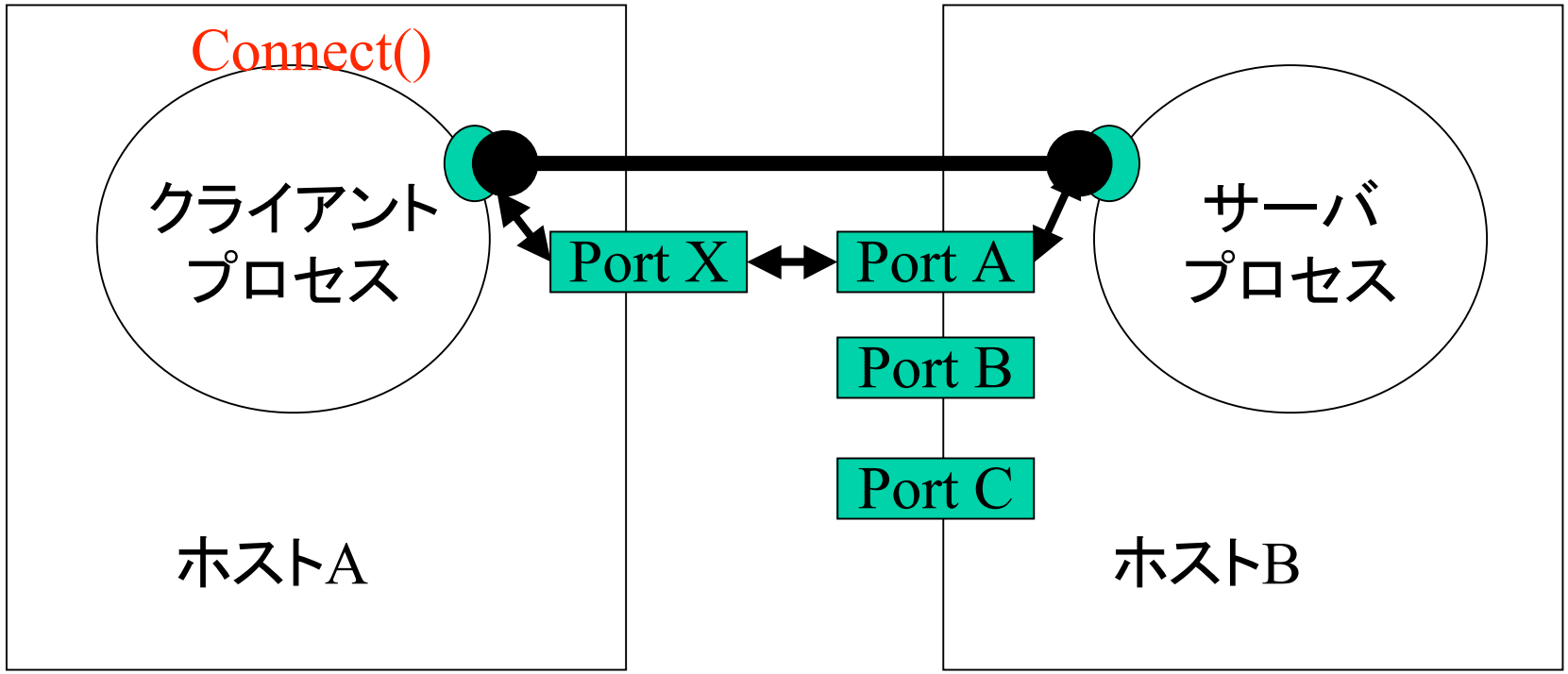
IP Address: xx.xx.xx.xx.

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

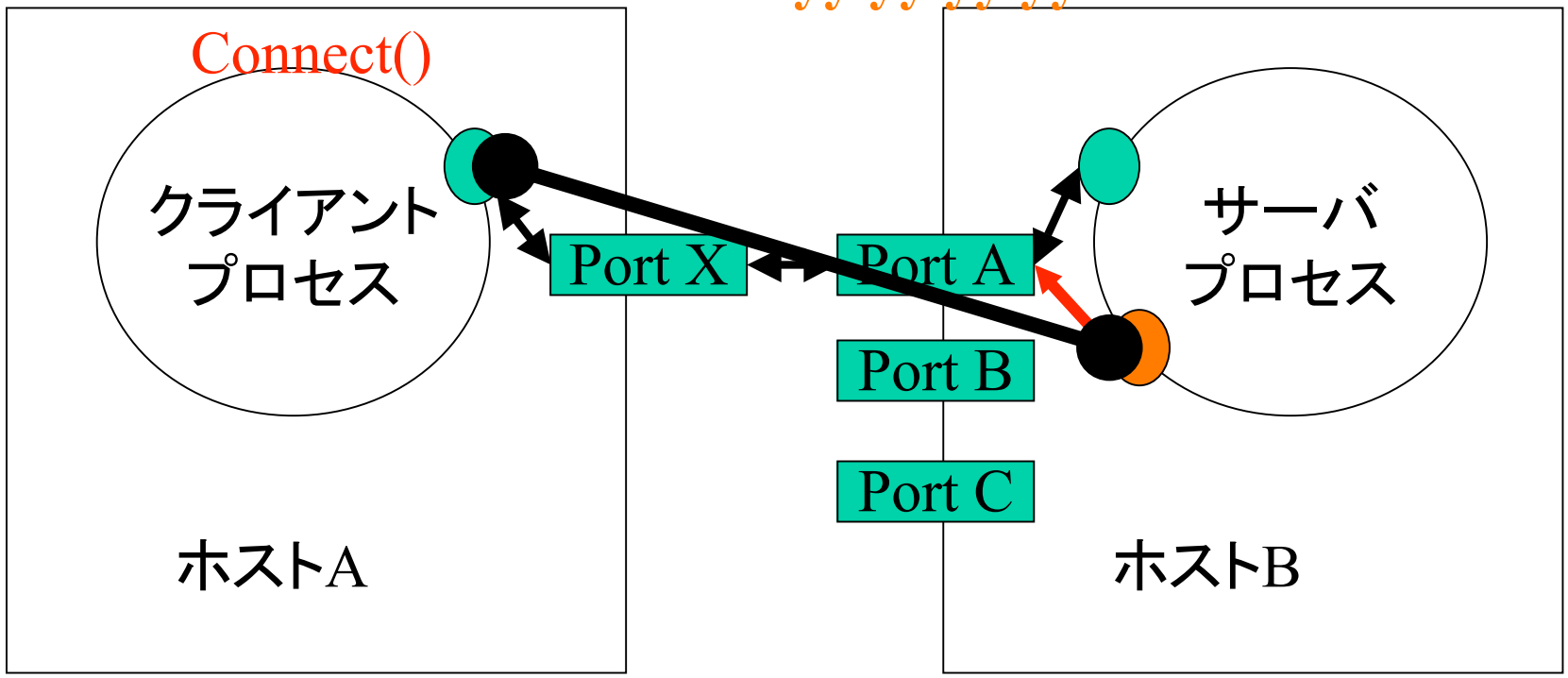


IP Address: yy.yy.yy.yy

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)
- 提出はSFC-SFSにて

実習

- サーバプログラムができたら、近くの人と、自分のプログラムが動いている計算機のIPアドレスと、ポート番号を教えあい、他人のクライアントからのエコーサーバを実行
- （それができたら）チャットプログラムを作ってみよう
  - メッセージを受信したら、自分からメッセージをおくる
  - クライアントプログラムと、サーバプログラムの混合