情報科学プロジェクト実験

第 6 回目 Web クライアント

コンピュータシステム研究室

成蹊大学理工学部

2016 年 10 月 27 日

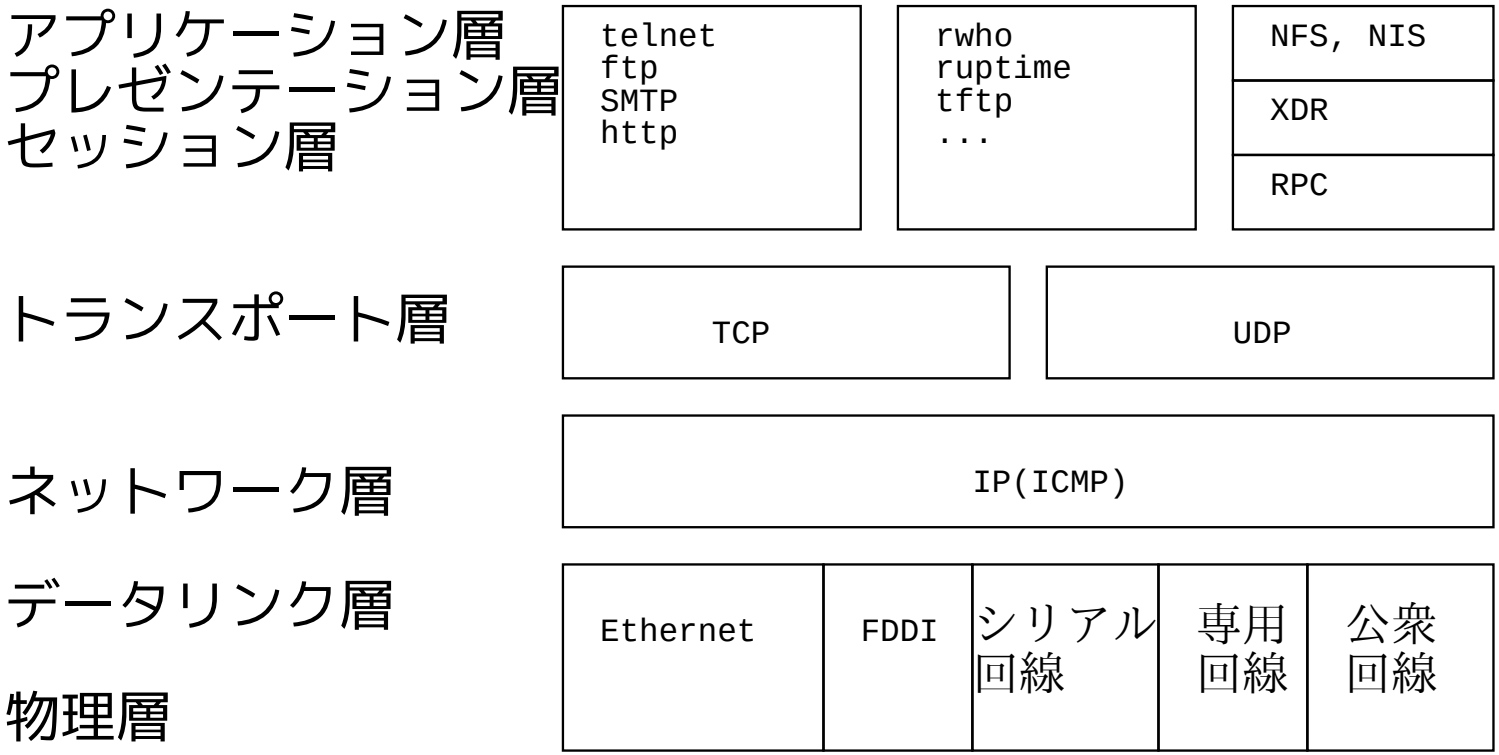
Web ページへのアクセス

- Web ページ取得プログラム作成には以下の知識が必要
  - TCP/IP を利用した通信プログラム
  - HTTP による Web サーバへの要求
- 参考文献
  - UNIX ネットワークプログラミング, オーム社
  - Web サーバ完全技術解説, 日経 BP 社

TCP/IP の概要

- The Internet の標準プロトコル
- RFC 793, RFC 1122, RFC 2001 など
  - RFC とは:
- Unix 以外の OS でも利用可能
- よく使われる TCP プロトコル
  - HTTP, SMTP, POP3, IMAP, FTP, ...

OSI 参照モデルと TCP/IP



TCP と UDP の特徴

	TCP	UDP
コネクション型	○	×
非コネクション型	×	○
バイトストリーム	○	×
非バイトストリーム	×	○
全二重	○	○
誤り制御	○	×
順序づけ	○	×
フロー制御	○	×

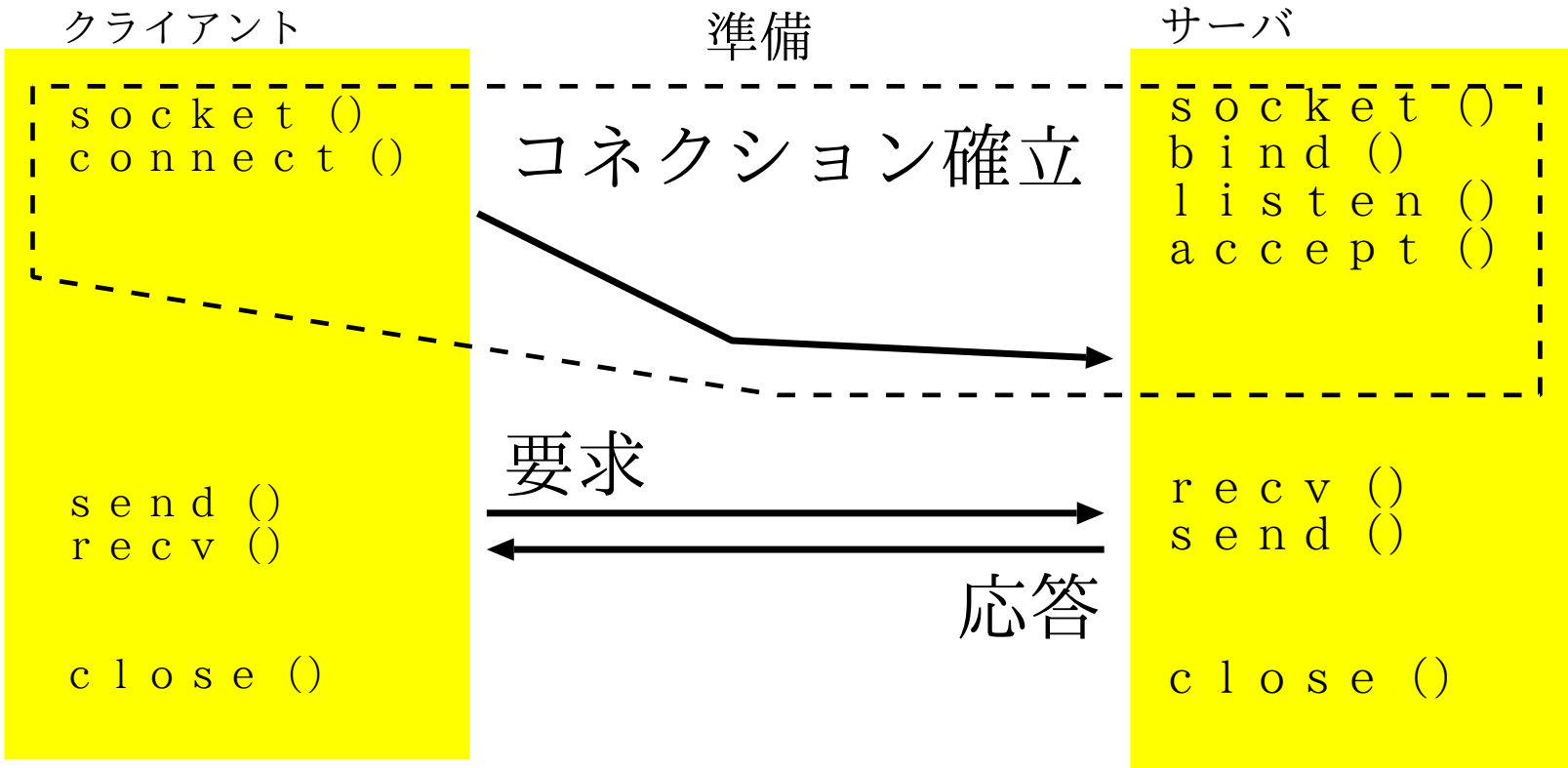
ポート番号の必要性

通信相手を指定するには相手ホストとそのホスト内のプロセスを指定しなければならない。

- プロセス番号で通信相手を指定した場合
  - 相手プログラムが実行開始するまで番号が分からない
  - 複数相手との同時通信が複雑
  - OS によってはうまくいかない
- ポート番号で通信相手を指定した場合
  - 事前に決めたポート番号を決めておくことができる
  - 複数のポート番号を用途別に割り振って利用できる

0...1023	well known port
1024...49151	registered port
49152...65535	danamic and/or private port

TCP の通信手順


クライアント
準備
サーバ
socket ()
connect ()
コネクション確立
socket ()
bind ()
listen ()
accept ()
要求
send ()
recv ()
recv ()
send ()
応答
close ()
close ()

各関数の役割

関数名	動作	使用プログラム
socket()	通信のための端点を作り ディスクリプタを返す	サーバ クライアント
connect()	ソケットの接続を行う	クライアント
bind()	ソケットにアドレスを設定	サーバ
listen()	カーネルに接続受付を指示	サーバ
accept()	接続のひとつを取り出す	サーバ
send()	データ送信	双方
recv()	データ受信	双方
close()	接続を遮断	双方

TCP クライアントの接続

```
// 接続先情報の準備
addrinfo hints, *svr;
memset(&hints, 0, sizeof(hints));
hints.ai_socktype = SOCK_STREAM;
hints.ai_family = AF_INET;
hints.ai_flags = AI_NUMERICSERV;
getaddrinfo(argv[1], argv[2], &hints, &svr);

// TCP ソケットの作成
int sockfd = socket(svr->ai_family, SOCK_STREAM, 0);

// 接続
connect(sockfd, svr->ai_addr, svr->ai_addrlen);
freeaddrinfo(svr);   // メモリの開放

//send()/recv() を使って通信
msg_proc(sockfd);

// サーバ側に切断を伝える
close(sockfd);
```

TCP クライアントの送受信

```
// サーバに要求を送る
string s = "....";
if (send(sockfd, s.c_str(), s.size(), 0) == -1)
   error("send");

// サーバから結果を受け取る
const int bufsz = 128;  // 通信内容によって適当なサイズを指定する
char buf[bufsz+1];
int n = recv(sockfd, buf, bufsz, 0);
if (n == -1) error("recv");
if (n > 0 ) {        // 接続が切れた場合に 0 となる
   buf[n] = '\0';
   std::cout << buf;
}
```

recv() 関数が返す値

TCP/IP の通信はバイトストリームであるために、相手が連続でデータを送信する場合には切れ目が OS には分からない。そのため、recv() は引数で指定したサイズ以下の受信でも OS の都合で戻る場合がある。

- 基本は読み出したデータのバイト数
- 何らかのエラーが起きたならば -1
- 届いているデータを全て読み取った状態で、もし、相手が正常な処理として接続を遮断していた場合には 0

ネットワークバイトオーダー

インターネット越しに通信を行う場合には相手のコンピュータのエンディアンが異なるかもしれない。そのため、整数はすべて「ネットワークバイトオーダー」にして送るのがマナー（決まりと言っても良い）。「ネットワークバイトオーダー」に対して、ホスト側のエンディアンを「ホストバイトオーダー」と呼ぶ。
変換には以下のような関数を用いる。

```
#include <arpa/inet.h>
uint32_t htonl(uint32_t hostlong);
uint16_t htons(uint16_t hostshort);
uint32_t ntohl(uint32_t netlong);
uint16_t ntohs(uint16_t netshort);
```

ホスト名-IP アドレス, ポート番号とサービス

- IP アドレスまたはホスト名を文字列で指定する
  - IP アドレスはドット形式（例:192.168.1.1）
  - ホスト名は FQDN（例：www.seikei.ac.jp）
  - プログラム中では 32 ビットの整数（IPv4 の場合）
- ポート番号はサービス名と対応している
  - 80 $\rightarrow$ http
  - 25 $\rightarrow$ smtp
  - プログラム中では 16 ビットの整数（IPv4 の場合）
- 変換は `getaddrinfo()` 関数で行う
  - 第 1 引数にホスト名やドット形式の IP アドレス
  - 第 2 引数にサービス名やポート番号の文字列

HTTP プロトコル

- HTTP: HyperText Transfer Protocol
  - RFC 7230 (HTTP/1.1)
  - RFC 2616 (HTTP/1.1)
- World Wide Web **で使われる情報をやりとりするための プロトコル**
- TCP/IP **接続を通してどのようにお互いが通信するかを 定めたルール群**
- **ポート番号には慣習として** 80 **番を使う**
- **サービス名は** http

HTTP リクエスト

クライアントからサーバへの要求

1. **メソッド**: 1 **個の処理要求の指定**
2. **オブジェクト：リクエストされる対象の名前** (URI)
3. **バージョン：プロトコルバージョン**
4. **オプション：修正または補足情報**

メソッド	アクション
GET	オブジェクト返す
HEAD	作成日時などオブジェクトの情報を返す
POST	サーバに保存する情報を送る
PUT	サーバ上の情報を別の情報で更新する
DELETE	オブジェクトを削除

HTTP リクエストの例

```
GET /index.html HTTP/1.0
User-Agent: foo-bar for X Windows
Accept: text/plain
Accept: text/html
Accept: image/gif
```

フィールド	情報
User-Agent	リクエストしているブラウザの種類
If-Modified-Since	指定日よりも新しいオブジェクトのみ返す
Accept	ブラウザが受取可能な形式 (MIME 形式)
Authorization	ユーザパスワードまたは他の証明

HTTP レスポンス

サーバからクライアントへ

1. **ステータス行：成功か？失敗か**?
2. **レスポンス行の情報の説明。メタ情報**
3. **改行** (CRLF, C++**プログラムでは** `\r\n`)
4. **リクエストされた情報**

コード	説明	理由
200	Document follows	リクエストは成功し情報が続く
301	Moved Permanently	ドキュメントは移動した
302	Moved Temporarily	ドキュメントは一時的に移動した
304	Not Modified	ドキュメントは変更されていない
401	Unauthorized	情報は保護されている
402	Payment Required	料金の支払が必要
403	Forbidden	アクセスは禁止されている
404	Not Found	見つからない
500	Server Error	サーバがエラーを起こした

HTTP レスポンスの例

```
$ telnet www.ci.seikei.ac.jp 80
Trying 133.220.149.120...
Connected to www.ci.seikei.ac.jp.
Escape character is '^]'.
GET /index.html HTTP/1.1
Host: www.ci.seikei.ac.jp    <<<----- ここまでが入力
                             <<<----- 入力の終わりを示すために 2 個の改行
HTTP/1.1 200 OK              <<<----- まずメタ情報が返る
Date: Wed, 15 Oct 2014 08:58:46 GMT
Server: Apache/2.2.15 (Red Hat)
Last-Modified: Wed, 03 Apr 2013 10:26:01 GMT
ETag: "1abe0003-1c7-4d97247cda040"
Accept-Ranges: bytes
Content-Length: 455
Connection: close
Content-Type: text/html; charset=UTF-8

<html>                            <<<------- 2 個の改行の後にコンテンツ
<head>...
```

- **改行コードには** CRLF(\r\n) **が使われる**

課題:Web ページの取得

- Web ページを取得するプログラムを作る。ただし, 対象となる Web ページは, その URL をコマンド引数として指定する。
- ヒント:取得する Web ページのサイズはあらかじめ分からないので、サーバが接続を切るまで、recv() で情報を読みつづける必要がある。

レポートに最低限書くべき事項

- レポートの表題, 名前, 学生番号
- 課題の内容
- 課題で求められている点 (箇条書でよい)
- 文章による作成したプログラムの説明
- 実行時テストの目的, 方法, 結果, 考察.

  目的: 何を調べるためのテストか?
  方法: 目的を達成するため何をしたか?
  結果: 整理して書く. 出力をそのまま書かない.
  考察: プログラムが正しく動作しているかの検証. 期待した性能, 計算の精度, 計測の精度が得られているか?
- 感想 (一番重要! , 最低でも 5 行は書くこと)
- プログラムリスト (コメントつき)