ネットワークアプリケーション特論
(Advanced Topics on Network Applications)

担当 寺田 実 (Minoru TERADA)
居室 西2 510
メール terada.minoru@uec....
授業Web
  http://pr.cei.uec.ac.jp/~terada/lectures/protocol/2018/

授業の概要 (abstract)

Web (WWW; World Wide Web)
  Webの歴史, アーキテクチャ
  Static / Dynamic Web Pages
  HTTP

  Computer Networks (5th edition), Tanenbaum and Wetherall, Prentice Hall
    Section 7.3 THE WORLD WIDE WEB

P2P の技術的な側面を勉強する (technical aspects of P2P computing)
  動作原理の理解 (mechanism of P2P networks)
  可能性を知る (and their potentials)

  P2P の一般論
  unstructured P2P
    Gnutella
    Freenet
  structured P2P - DHT (Distributed Hash Table)
    Chord
    CAN
    Pastry
    Tapestry
    Tourist

まず一般的なTCP/IPについて

  Web, P2PアプリケーションはTCP/IPネットワーク上で動作
  その原理や特徴, 限界を知っておく必要あり

プロトコルの階層モデル (hierarchical model of protocols)
  各層が提供する相手識別, サービス
  (each layer offers services to its upper layer)
  5層モデル
    アプリケーション層 (application layer) [[具体例: HTTP]]
      web server/client etc.
    -----OSの境界----- system call
    トランスポート層 (transport layer) [[具体例: TCP, UDP]]
      信頼性のあるコネクション (reliable connection)
      1-to-1
    ネットワーク層 (network layer) [[具体例: IP]]
      IPアドレスによるパケット(datagram)配送
      (delivery of datagrams destined to an IP address)
    データリンク層 (datalink layer) [[具体例: Ethernet]]
      一本のケーブルでのパケット(frame)配送
      (transmission of frames along a single cable)
    物理層 (physical layer)
      (ex. ethernet)
      電気的信号/光学的信号 (electronic/optical signals)

アプリケーションからみたTCP/IP (viewed from application programs)
  つまり、OSが上位に提供する機能
  識別 (identification)
    IPアドレス(32ビット (IPv4)) (IP address - 32 bits)
      ホストの識別 (identification of hosts)
    ポート番号(16ビット) (port number - 16 bits)
      プログラムの識別 (identification of programs on a host)
  TCP コネクションの設立 (establishment of TCP connection)
    非対称 (asymmetric)

```
    受け側 (receiver side)
      accept()  system call
    接続要求側 (connection side)
      相手のIPアドレスとポート番号を指定して (using the peer's IP address and port number)
      connect() system call
    コネクションが成立してしまえば, 通信は対称
    (after the establishment of the connection, the communication is symmetric)
      データは単なるバイトストリーム
      (transfered data are just byte streams, without any boundary)
      境界なし
    相手が accept() してないと, connect() は失敗
      "Connection refused"

★Routingの話?

サーバクライアントモデル (server-client model)
  例: Web, email など
  サーバ側が特定のポート番号で accept() で待つ
  (the server is accepting connection on a specific port)
    ex. web -> 80
  クライアント側は connect()
  接続したら, サービス要求を送信, サービスを受信
  (after connection, the client sends service request and receives the result)

TCP/IPでは, IPアドレスをもつホスト上で動くプログラムは,
accept() で待つことで, すべて接続を受け入れることができる
(Any program can accept connection on a host)
  つまり, だれでもサーバを実行できる
  (i.e. anyone can run servers)

privileged port
  1024 未満のポートは, root (管理者)のプログラムだけが使える
  (ports with numbers less than 1024 are only used by administrators)
  一般ユーザが重要なサービスを乗っ取ることを禁止
  (to inhibit normal users from hijacking important services)
  でも, 1024以上は自由に使える
  (ports greater than 1024 are available for normal users)
    ex. http://aaa.bbb.jp:8080/...

well known port
  特定のサービス向けに予約されたポート
  (dedicated ports for specific services)
  /etc/services

システムコール
socket() - ソケット(通信用データ構造)の作成
bind()   - ソケットにポート番号を設定する
listen() - 接続待ちの準備
accept() - 接続待ち
           返り値はその相手との通信用ソケット
           副作用として、相手のアドレスとポートがわかる
connet() - 相手に接続をかける
read()   - データの受信
write()  - データの送信
close()  - 通信終了

------------------------------------------------------------
Sample programs in C and Java

common.h
     1  #include        <unistd.h>      /* for read, write, close */
     2  #include        <string.h>      /* for bzero, strlen */
     3  #include        <stdlib.h>      /* for exit */
     4  #include        <stdio.h>
     5  #include        <sys/types.h>
     6  #include        <sys/socket.h>
     7  #include        <netinet/in.h>
     8  #include        <arpa/inet.h>
```

```
 9
10  void err(char *s)
11  {
12    fprintf(stderr, "err(%s)\n", s);
13    perror("errno indicates:");
14    exit(1);
15  }
```

server.c

```
 1  #include "common.h"
 2
 3  /* 単純なサーバのプログラム
 4     クライアントから接続を待ち, 送られてきた10進数文字列を
 5     16進数表記文字列に変換して返す */
 6  int main(int argc, char **argv)
 7  {
 8    int sockfd, newsockfd;
 9    socklen_t clilen;             /* じつは unsigned int */
10    struct sockaddr_in cli_addr,serv_addr;
11    char buf[256];
12    int n, x;
13
14    /* socket システムコール - 返り値はソケットのファイルディスクリプタ */
15    if ((sockfd = socket(AF_INET,SOCK_STREAM, 0)) < 0)
16      err("server: socket()");
17
18    /* serv_addr: アドレス/ポート番号を格納するための構造体 */
19    /* まず 0 でクリア */
20    bzero((char *) &serv_addr, sizeof(serv_addr));
21    serv_addr.sin_family = AF_INET;
22    /* 自分のIPアドレスは自動的に設定される */
23    serv_addr.sin_addr.s_addr = htonl(INADDR_ANY);
24    /* ポート番号(8888)は自分で設定する */
25    serv_addr.sin_port  = htons(8888);
26
27    /* bind システムコール - ソケットにポート番号をつける */
28    if (bind(sockfd,(struct sockaddr *)&serv_addr,sizeof(serv_addr))<0)
29      err("server: bind()");
30
31    /* listen システムコール - 接続待ちの準備 */
32    listen(sockfd, 5);
33
34    while(1){                     /* サーバは無限ループ */
35      /* accept システムコール - 接続待ち
36         返り値は通信用のソケット
37         第二引数の cli_addr に, 接続してきた相手のアドレスが入る */
38      clilen = sizeof(cli_addr);
39      newsockfd = accept(sockfd,(struct sockaddr *) &cli_addr,&clilen);
40      if (newsockfd < 0) err("server: accept()");
41
42      /* 通信用ソケットからデータを読み出す */
43      n = read(newsockfd, buf, sizeof buf);
44      if(n <= 0) err("server: read()");
45      buf[n] = '\0';              /* 文字列の終端 */
46
47      /* データを10進数で解釈して */
48      sscanf(buf, "%d", &x);
49      /* 16進数として相手に送り返す */
50      sprintf(buf, "%x", x);
51      n = write(newsockfd, buf, strlen(buf));
52      if(n <= 0) err("server: write()");
53
54      /* 通信用ソケットをクローズする */
55      close(newsockfd);
56    }
57    /* サーバ終了. 接続待ち用ソケットもクローズする */
58    close(sockfd);
59    exit(0);
```

```
    60  }
```

client.c

```
     1  #include "common.h"
     2
     3  /* 単純なクライアントのプログラム
     4     第一引数で指定するサーバに接続して第二引数で指定した文字列を送信し,
     5     受信した文字列を標準出力に表示する */
     6  int main(int argc, char **argv)
     7  {
     8    int sockfd;
     9    struct sockaddr_in serv_addr;
    10    char  buf[256];
    11    int i, n;
    12
    13    /* サーバのアドレスを指定するための構造体の用意 */
    14    bzero((char *) &serv_addr, sizeof(serv_addr));
    15    serv_addr.sin_family = AF_INET;
    16    /* サーバのIPアドレスは, 第一引数で 10.0.0.1 のように指定する */
    17    serv_addr.sin_addr.s_addr = inet_addr(argv[1]);
    18    /* サーバのポート番号は 8888 で固定 */
    19    serv_addr.sin_port  = htons(8888);
    20
    21    /* socket システムコール - ソケットの生成 */
    22    if ((sockfd = socket(AF_INET,SOCK_STREAM,0)) < 0)
    23      err("client: socket()");
    24
    25    /* connect システムコール - 接続要求 */
    26    if (connect(sockfd,(struct sockaddr *)&serv_addr,sizeof serv_addr)<0)
    27      err("client: connect()");
    28
    29    n = strlen(argv[2]);          /* 第二引数 */
    30    /* 文字列を送信する */
    31    i = write(sockfd, argv[2], n);
    32    if(i != n ) err("client: write()");
    33
    34    /* サーバからの返事を受信 */
    35    n = read(sockfd, buf, sizeof buf);
    36    if(n <= 0) err("client: read()");
    37    /* 標準出力に表示 */
    38    write(1, buf, n);
    39    write(1, "\n", 1);            /* 改行 */
    40
    41    close(sockfd);
    42    exit(0);
    43  }
```

Server.java

```
     1  import java.io.BufferedReader;
     2  import java.io.InputStreamReader;
     3  import java.io.PrintStream;
     4  import java.net.ServerSocket;
     5  import java.net.Socket;
     6
     7  public class Server {
     8
     9    public static void main(String[] args) throws Exception {
    10      int port = Integer.parseInt(args[0]);
    11      ServerSocket ss = new ServerSocket(port);
    12
    13      while(true){
    14        Socket s = ss.accept();
    15
    16        BufferedReader r =
    17          new BufferedReader(new InputStreamReader(s.getInputStream()));
    18        String msg = r.readLine();
    19        System.out.format("server read: %s\n", msg);
    20
```

```
    21        PrintStream p = new PrintStream(s.getOutputStream());
    22        p.format("Goodby\n");
    23
    24        s.close();
    25        if(msg.equals("quit")) break;
    26      }
    27      ss.close();
    28    }
    29
    30  }
```

Client.java

```
     1  import java.io.BufferedReader;
     2  import java.io.InputStreamReader;
     3  import java.io.PrintStream;
     4  import java.net.InetAddress;
     5  import java.net.Socket;
     6
     7
     8  public class Client {
     9
    10    public static void main(String[] args) throws Exception {
    11      String server = args[0];
    12      int port = Integer.parseInt(args[1]);
    13      String to_send = args[2];
    14      Socket s = new Socket(InetAddress.getByName(server), port);
    15
    16      PrintStream p = new PrintStream(s.getOutputStream());
    17      p.format("%s\n", to_send);
    18
    19      BufferedReader r =
    20        new BufferedReader(new InputStreamReader(s.getInputStream()));
    21      String msg = r.readLine();
    22      System.out.format("client read: %s\n", msg);
    23
    24      s.close();
    25    }
    26  }
```

NAT (Network Address Translation)

IP(v4)アドレスの枯渇対策
(running out of IP addresses)

組織のネットワークを閉じたものにしてしまう
(isolate the internal network of a organization)
  内部相互は普通に使える
  (hosts can communicate within the organization as usual)
  private address を利用 (using private addresses)
    IP address の一部, 自由に使ってよい(が, 外部には出せない)
    10.xx.xx.xx
    192.168.1.xx
外部との通信のために
  外部との接続点に NAT Box (broadband router)
  (connect to the outside using a special router, NAT Box)
  NAT box の外側には, 正しい(not private) IP address をつける
  (the outside interface of NAT Box has proper (not private) IP address)

内部→外部の connection (outgoing connection)
  発信元IP (source IP address)
    IP datagram のヘッダに存在 (contained in the IP header)
  発信元Port (source port number)
    TCP datagram のヘッダに存在 (contained in the TCP header)

  (内部IP, 内部port) → (外部IP, 外部port) と変換
  (translate IP address and port number at the NAT Box)

TCP接続開始時にこの対応を記憶しておく
(and remember the translation in a table)
外部から届いた返信パケットは, この対応表を利用して逆変換
(incoming packets are translated back using the table)

外部→内部の connection (incoming connection)
あらかじめ設定してなければ, 破棄
(discarded without any configuration in advance)
設定しておけば, (内部IP, 内部port) に転送
(configure the translation of IP address and port number make the connection to be forwarded to a host in the internal network)
内部にサーバを設置できる
(i.e. servers can be located in the internal network)

効果としては (the benefit of NAT)
ひとつの(外部)アドレスを使って
(only one global IP address)
内部のホストは自由に外部に接続可能
(any number of internal hosts can make outgoing connections)
設定によってはフィルタも可能
(filtering outgoing connection is also possible)

外部からは, NAT Box で static に設定したポートにだけ接続可能
(incomming connection are permitted only if configured explicitly)
一般ユーザは接続を受けることはできなくなる
(normal hosts will not receive incoming connection requests)
安全性向上 (increased security)

P2Pの観点からは (from the standpoint of P2P)
外部から接続ができない (incoming connections are prohibited)
対称性が崩れる (hosts are not symmetric any more)