Nội dung môn học

CHƯƠNG 1: GIỚI THIỆU VỀ TCP/IP

CHƯƠNG 2: THIẾT KẾ GIẢI THUẬT CHO CHƯƠNG TRÌNH CLIENT/SERVER

CHƯƠNG 3: LẬP TRÌNH MẠNG TRÊN CÁC MÔI TRƯỜNG PHỔ DỤNG

CHƯƠNG 4: LẬP TRÌNH MẠNG VỚI JAVA

Nội dung môn học(tt)

CHƯƠNG 5: LẬP TRÌNH WEB – CGI

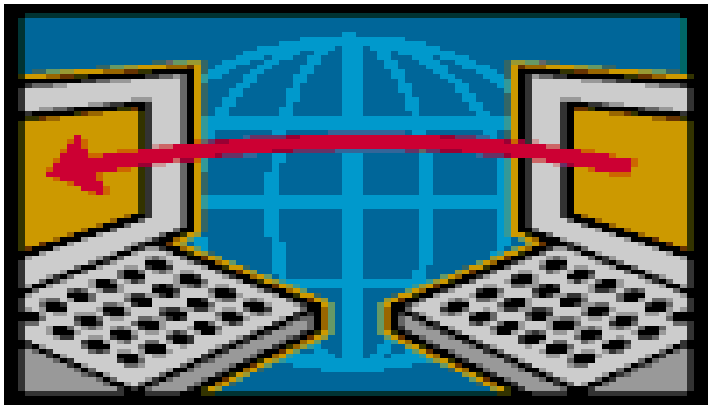
CHƯƠNG 6: LẬP TRÌNH WEB VỚI CÁC CÔNG NGHỆ PHỔ BIẾN

CHƯƠNG 7: ỨNG DỤNG XML TRONG LẬP TRÌNH MẠNG

CHƯƠNG 8: BẢO MẬT DỮ LIỆU TRUYỀN

Tài liệu tham khảo

- [1] Douglas E. Comer, *Internetworking with TCP/IP*, Prentice-Hall,1993.
- [2] W. Richard Stevens, *Unix Network Programming*, Prentice-Hall,1990.
- [3] Arthur Dumas, *Programming Winsock*, Sams Publishing,1995.
- [4] Merlin, Conrad Hughes ..., *Java Network Programming*, Manning Publications Co., 1997.
- [5] D. Travis Dewire, *Second-Generation Client/Server Computing*, Mc Graw-Hill, 1997.
- [6] John Shapley Gray, *Interprocess Comunication in UNIX*, Prentice-Hall,1997.
- [7] Deitel & Deitel. *Java How to program, 3th edition*, Prentice-Hall,1999.
- [8] Richard Anderson, ..., *Professional Active Server Pages 3.0*, Wrox Press, 1999.
- [9] Marty Hall, *Core Servlet and Java Server Pages*, Prentice-Hall PTR, 2000
- [10] MSDN.
- [11] Tập tài liệu RFC.



CHƯƠNG 1
GIỚI THIỆU VỀ TCP/IP

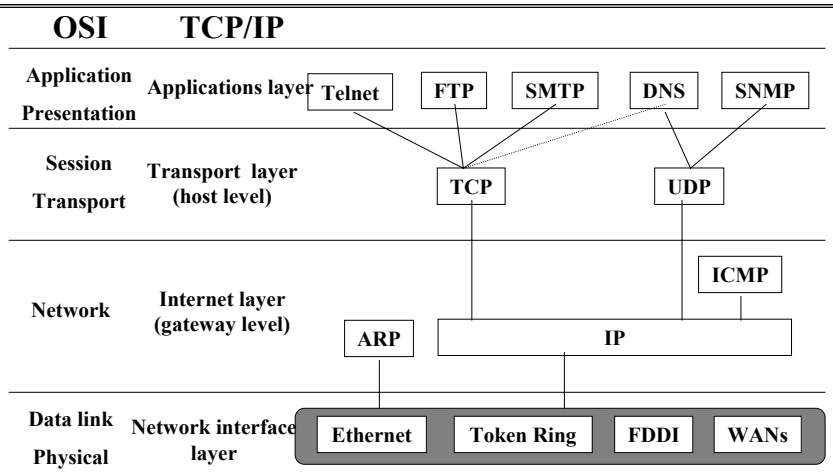
1.1 Tổng quát về TCP/IP.

1.2 Các giao thức và dịch vụ trên TCP/IP.

1.3 Khái niệm về Socket.

1.4 Một số ứng dụng mạng.

1.1 Tổng quát về TCP/IP.



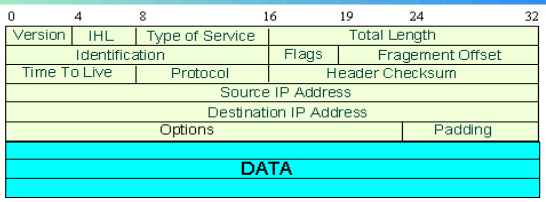
1.1 Tổng quát về TCP/IP (tt)

- Một số đặc tính :
  - Độc lập về hình thái của mạng.
  - Độc lập về phần cứng của mạng.
  - Các chuẩn giao thức mở.
  - Mô hình địa chỉ toàn cầu.
  - Nền tảng client/server mạnh mẽ.
  - Các chuẩn về giao thức ứng dụng mạnh mẽ.

1.1 Tổng quát về TCP/IP (tt)

4. IP

IP Frame



- **Flags (3): control fragmentation.**
  - Bit_0 =0. reserved Bit_1=may(0)/may-not(1) fragment . Bit_2= last (0)/more(1) fragment
- **Fragment Offset(13): pointer to fragment location**
- **Time to live: number of seconds for datagram to remain alive**
- **Protocol: identify higher level protocol user (UDP, TCP..)**
- **Header checksum: error detection bits**
- **Source/destination addresses: contain 32 bits Internet address**

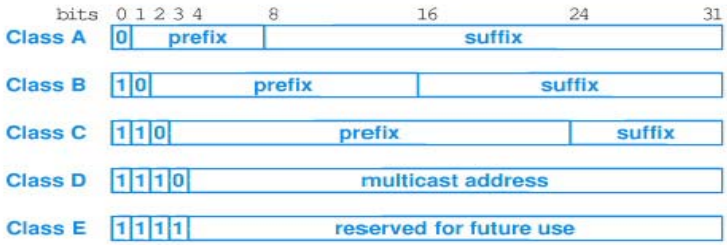
1.1 Tổng quát về TCP/IP (tt)

- Địa chỉ Internet:
  - Định vị duy nhất một máy
  - Chiều dài 32 bit
  - Cấu trúc IP (netid, hostid), các máy trên một mạng có netid giống nhau.
  - Do NIC cấp
  - Cách biểu diễn:

    10101100 00011100 00010000 00000101
    172 28 16 5
    172.28.16.5

1.1 Tổng quát về TCP/IP (tt)

- Phân lớp địa chỉ:
  - Để xác định netid (Network Identifier) và hostid (Host Identifier)



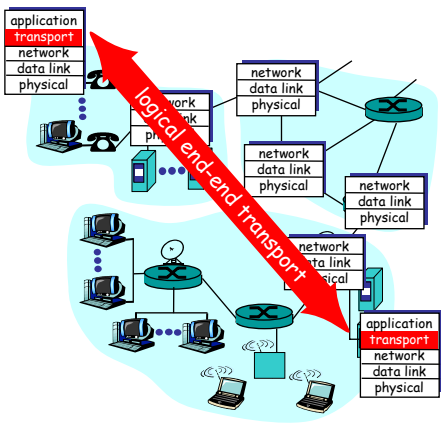
1.1 Tổng quát về TCP/IP (tt)

- Một số địa chỉ IP đặc biệt

Prefix	Suffix	Type Of Address	Purpose
all-0s	all-0s	this computer	used during bootstrap
network	all-0s	network	identifies a network
network	all-1s	directed broadcast	broadcast on specified net
all-1s	all-1s	limited broadcast	broadcast on local net
127	any	loopback	testing

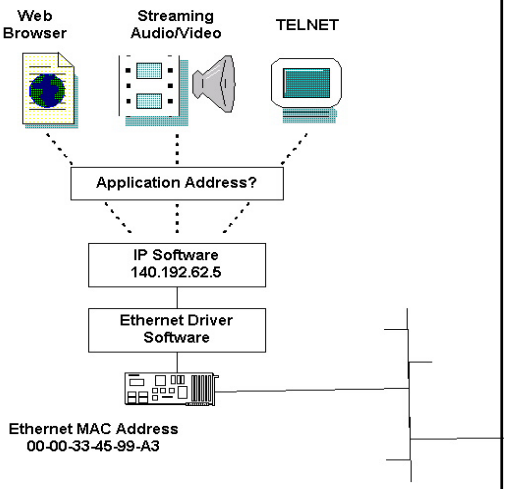
1.1 Tổng quát về TCP/IP (tt)

- Lớp Transport
  - Cung cấp giao tiếp luận lý giữa các processes trên các hosts khác nhau
  - Có hai dạng dịch vụ:
    - TCP (Transmittion Control Protocol)
    - UDP (User Datagram Protocol)



1.1 Tổng quát về TCP/IP (tt)

- Lớp Transport (tt)
  - Mở rộng cách đánh địa chỉ cho process.
  - Địa chỉ port : xác định ứng dụng mạng trên mỗi máy.
  - Địa chỉ của một ứng dụng mạng (IP,port)

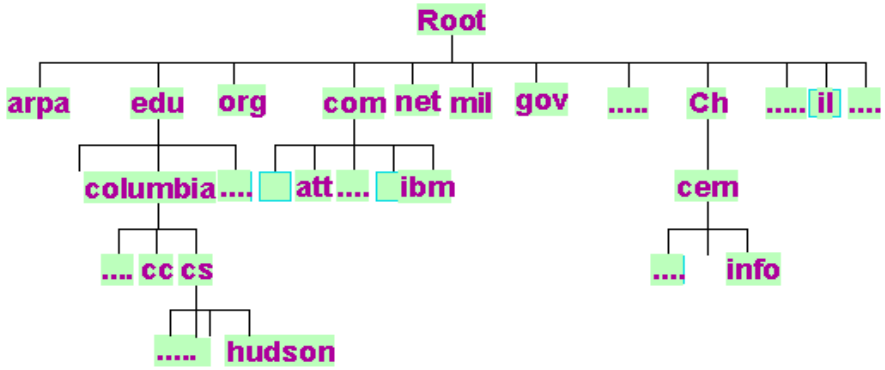


1.2 Các giao thức và dịch vụ

- Hệ thống tên miền DNS (Domain Name System)
  - Dùng chuỗi ký tự để đánh địa chỉ, không phân biệt chữ hoa, thường, mỗi thành phần có thể 63 ký tự và tên đầy đủ không dài quá 255, dưới đây gọi là tên.
  - Tên được đặt theo cây phân cấp
  - Địa chỉ tài nguyên biểu diễn dạng tên được hình thành từ nó cho đến root

1.2 Các giao thức và dịch vụ (tt)

- Hệ thống tên miền DNS (tt)

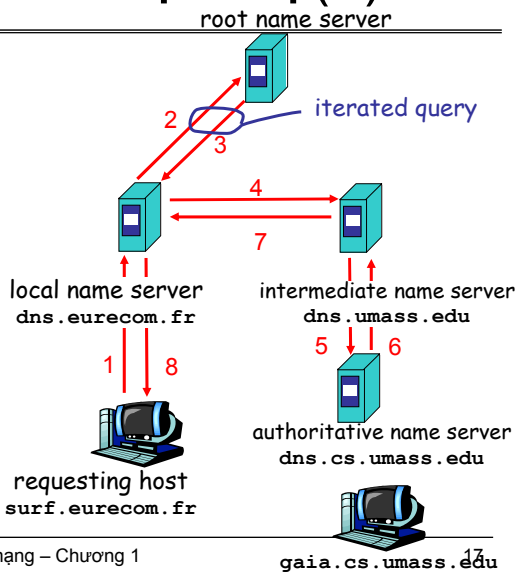


1.2 Các giao thức và dịch vụ(tt)

- Hệ thống tên miền DNS (tt)
  - Network chỉ hiểu địa chỉ IP (binary) => ánh xạ giữa địa chỉ IP và tên.
  - Hệ thống tên miền được hiện thực theo *distributed database*, quản lý theo dạng phân cấp với *name servers*
  - Network chỉ hiểu địa chỉ IP (binary) => ánh xạ giữa địa chỉ IP và tên.
  - Mỗi ứng dụng mạng phải chuyển tên sang địa chỉ IP

1.2 Các giao thức và dịch vụ(tt)

- DNS (tt)
  - Ứng dụng giao tiếp với local name server để hỏi địa chỉ ánh xạ.
  - Local name server sẽ trả lời hoặc request tiếp…



1.2 Các giao thức và dịch vụ(tt)

- Giao thức ở lớp ứng dụng
  - Ứng dụng mạng : trao đổi thông tin giữa các processes trên mạng.
  - Các ứng dụng phải định nghĩa protocol để giao tiếp với nhau.
  - Protocol qui định thứ tự các thông điệp trao đổi, hành động khi nhận mỗi loại thông điệp.
  - Ứng dụng cũng phải hiện thực phần giao tiếp với người dùng.

1.2 Các giao thức và dịch vụ(tt)

- Giao thức ở lớp ứng dụng(tt)
  - User agent là giao tiếp giữa người sử dụng và ứng dụng mạng.
    - Web:browser
    - E-mail: mail reader
    - streaming audio/video: media player

1.2 Các giao thức và dịch vụ(tt)

- Mô hình mạng client/server
  - Server : là phần tử thụ động
    - Chờ yêu cầu từ client, xử lý và trả kết quả cho client
  - Client : là phần tử chủ động
    - Kết nối đến server để gởi yêu cầu.
    - Chờ nhận kết quả trả về và xử lý kết quả.

1.2 Các giao thức và dịch vụ(tt)

- State và Stateless
  - State : lưu giữ trạng thái giữa các lần kết nối (request/response).
  - Stateless : Mỗi lần request/response thì cầu nối hủy bỏ. Không giữ trạng trái trước đó.

1.3 Khái niệm về Socket.

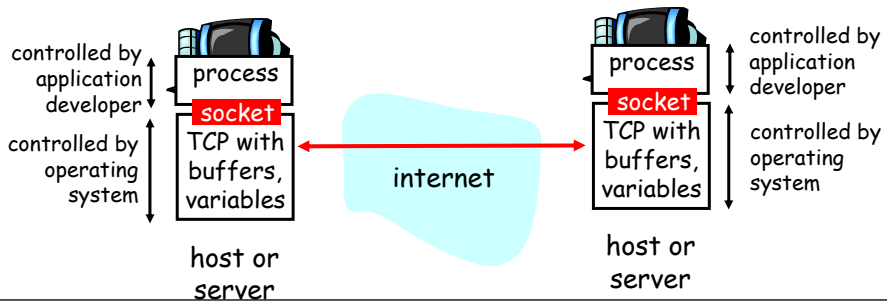
- Socket API
  - Được giới thiệu ở BSD4.1 UNIX, 1981
  - Được ứng dụng khởi tạo, sử dụng hay hủy bỏ
  - Dùng cơ chế client/server
  - Cung cấp hai dịch vụ chuyển dữ liệu thông qua socket API:
    - unreliable datagram
    - reliable, byte stream-oriented

1.3 Khái niệm về Socket(tt)

- Socket :
  - Là môi trường để các process ứng dụng giao tiếp với nhau, process ứng dụng có thể chạy trên cùng một máy hoặc trên hai máy khác nhau.
  - Được ứng dụng tạo ra và sử dụng tuy nhiên được hệ thống (hệ điều hành) kiểm soát.

1.3 Khái niệm về Socket(tt)

- Socket: "cửa" nằm giữa process ứng dụng và end-end-transport protocol (UCP or TCP)
- TCP service: dịch vụ truyền tin cậy chuỗi bytes giữa hai process



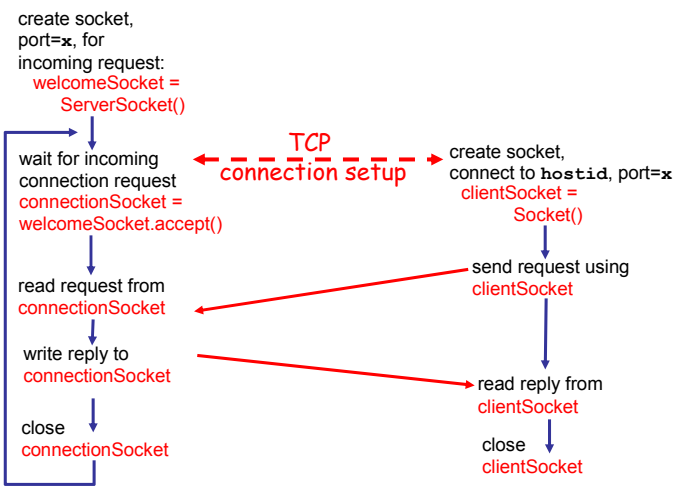
1.3 Khái niệm về Socket(tt)

- Lập trình socket với TCP
  - Client phải kết nối đến server
    - server process phải chạy trước (phần tử thụ động)
    - server phải tạo một socket để lắng nghe và chấp nhận các kết nối từ client
  - Client kết nối đến server bằng cách:
    - Khởi tạo TCP socket ở local
    - Xác định IP address, port number của server process và kết nối đến

1.3 Khái niệm về Socket(tt)

- Lập trình socket với TCP(tt)
  - Sau khi client khởi tạo socket, nó sẽ thiết lập kết nối đến server
  - Khi server nhận yêu cầu kết nối, nó sẽ chấp nhận yêu cầu và khởi tạo socket mới để giao tiếp với client.
    - Cho phép server chấp nhận nhiều client tại một thời điểm.

1.3 Khái niệm về Socket(tt)



Example: Java client (TCP)

```
import java.io.*;
import java.net.*;
class TCPClient {

   public static void main(String argv[]) throws Exception
   {
      String sentence;
      String modifiedSentence;

      BufferedReader inFromUser =
        new BufferedReader(new InputStreamReader(System.in));

      Socket clientSocket = new Socket("hostname", 6789);

      DataOutputStream outToServer =
        new DataOutputStream(clientSocket.getOutputStream());
```

Create input stream

Create client socket, connect to server

Create output stream attached to socket

Example: Java client (TCP), cont.

Create input stream attached to socket

```
BufferedReader inFromServer =
  new BufferedReader(new
InputStreamReader(clientSocket.getInputStream()));

sentence = inFromUser.readLine();
```

Send line to server

```
outToServer.writeBytes(sentence + '\n');
```

Read line from server

```
modifiedSentence = inFromServer.readLine();

System.out.println("FROM SERVER: " + modifiedSentence);

clientSocket.close();

}
}
```


Example: Java server (TCP)

```
import java.io.*;
import java.net.*;

class TCPServer {

  public static void main(String argv[]) throws Exception
   {
     String clientSentence;
     String capitalizedSentence;
```

Create welcoming socket at port 6789

```
     ServerSocket welcomeSocket = new ServerSocket(6789);
```

Wait, on welcoming socket for contact by client

```
     while(true) {

       Socket connectionSocket = welcomeSocket.accept();
```

Create input stream, attached to socket

```
      BufferedReader inFromClient =
        new BufferedReader(new
        InputStreamReader(connectionSocket.getInputStream()));
```

Example: Java server (TCP), cont

Create output stream, attached to socket

```
DataOutputStream  outToClient =
  new DataOutputStream(connectionSocket.getOutputStream());
```

Read in line from socket

```
clientSentence = inFromClient.readLine();

capitalizedSentence = clientSentence.toUpperCase() + '\n';
```

Write out line to socket

```
outToClient.writeBytes(capitalizedSentence);
            }
        }
    }
```

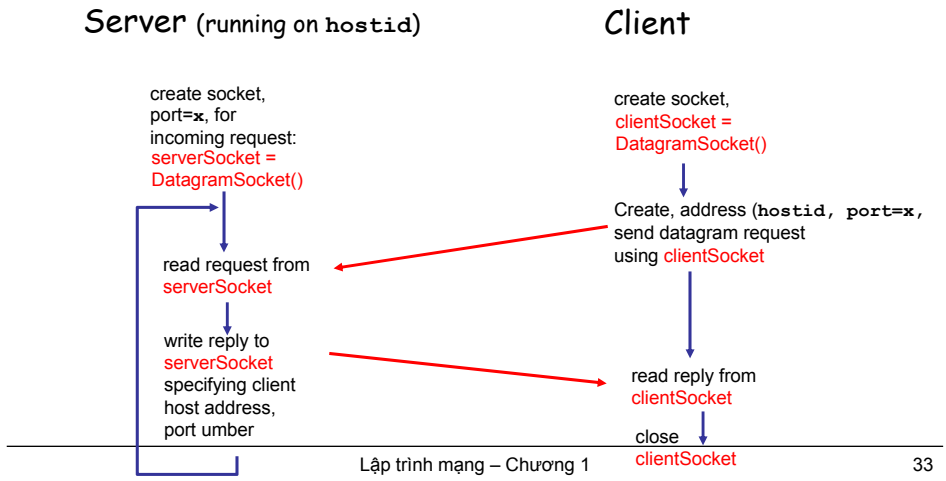
End of while loop, loop back and wait for another client connection

1.3 Khái niệm về Socket(tt)

- Lập trình socket với UTP
  - Cung cấp cơ chế truyền không tin cậy các nhóm các byte (datagrams) giữa client và server.
  - Không cần thiết lập kết nối giữa client với server.
  - Sender phải gởi kèm địa chỉ IP và port đích
  - Server khi nhận dữ liệu sẽ phân tích địa chỉ của sender để truyền lại.

1.3 Khái niệm về Socket(tt)

- Lập trình socket với UTP(tt)



Example: Java client (UDP)

```
import java.io.*;
import java.net.*;

class UDPClient {
   public static void main(String args[]) throws Exception
   {
```

Create input stream →

```
BufferedReader inFromUser =
  new BufferedReader(new InputStreamReader(System.in));
```

Create client socket →

```
DatagramSocket clientSocket = new DatagramSocket();
```

Translate hostname to IP address using DNS →

```
InetAddress IPAddress = InetAddress.getByName("hostname");

byte[] sendData = new byte[1024];
byte[] receiveData = new byte[1024];

String sentence = inFromUser.readLine();
sendData = sentence.getBytes();
```

Example: Java client (UDP), cont.

Create datagram with data-to-send, length, IP addr, port

Send datagram to server

Read datagram from server

```
                DatagramPacket sendPacket =
                  new DatagramPacket(sendData, sendData.length, IPAddress, 9876);

                clientSocket.send(sendPacket);

                DatagramPacket receivePacket =
                   new DatagramPacket(receiveData, receiveData.length);

                clientSocket.receive(receivePacket);

                String modifiedSentence =
                    new String(receivePacket.getData());

                System.out.println("FROM SERVER:" + modifiedSentence);
                clientSocket.close();
                }
            }
```


Example: Java server (UDP)

Create datagram socket at port 9876

Create space for received datagram

Receive datagram

```
import java.io.*;
import java.net.*;

class UDPServer {
  public static void main(String args[]) throws Exception
    {
      DatagramSocket serverSocket = new DatagramSocket(9876);

      byte[] receiveData = new byte[1024];
      byte[] sendData  = new byte[1024];

      while(true)
       {
        DatagramPacket receivePacket =
            new DatagramPacket(receiveData, receiveData.length);
        serverSocket.receive(receivePacket);
```

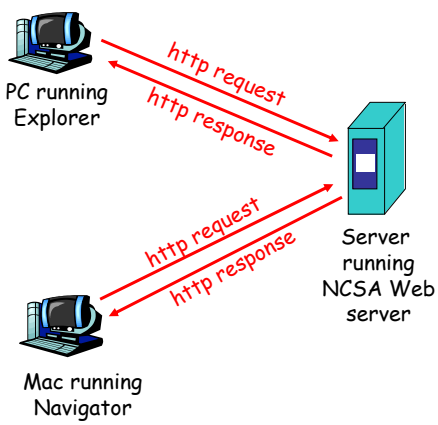
Example: Java server (UDP), cont

```
                String sentence = new String(receivePacket.getData());
Get IP addr
port #, of
sender          InetAddress IPAddress = receivePacket.getAddress();

                int port = receivePacket.getPort();

                        String capitalizedSentence = sentence.toUpperCase();

Create datagram
to send to client   sendData = capitalizedSentence.getBytes();

                DatagramPacket sendPacket =
                  new DatagramPacket(sendData, sendData.length, IPAddress,
                                port);
Write out
datagram
to socket       serverSocket.send(sendPacket);
              }
            }
          }
```

End of while loop, loop back and wait for another datagram

1.4 Một số ứng dụng mạng.

- World Wide Web (W W W)
  - Dùng giao thức **http**: hypertext transfer protocol
  - Web's application layer protocol
  - Mô hình client/server
    - *client:* browser gởi yêu cầu, nhận và hiển thị kết quả.
    - *server:* Web server gởi kết quả cho client đối với mỗi request.
  - http1.0: RFC 1945
  - http1.1: RFC 2068



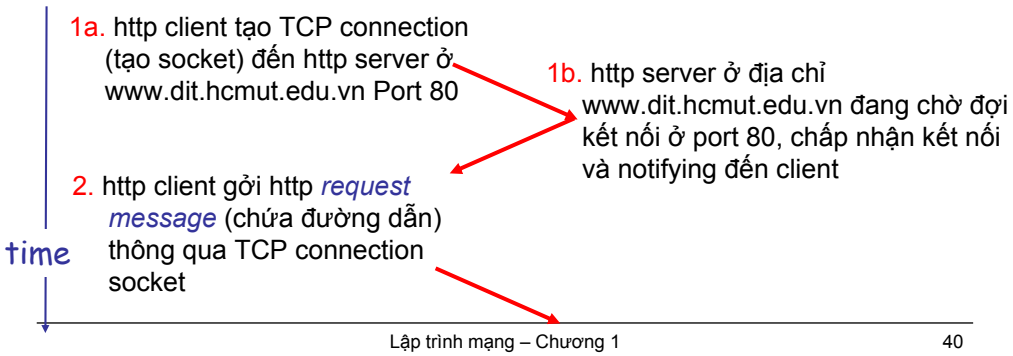
1.4 Một số ứng dụng mạng(tt)

- W W W (tt)
  - http: TCP transport service:
    - client khởi tạo TCP connection (tạo socket) đến server, port 80 (default)
    - server chấp nhận kết nối từ client
    - http messages (application-layer protocol messages) được trao đổi giữa browser (http client) và Web server (http server)
    - đóng TCP connection

1.4 Một số ứng dụng mạng(tt)

- W W W (tt) – Ví dụ
  - User đánh địa chỉ URL lên browser

http://www.dit.hcmut.edu.vn/~phu/courses/net-programming/index.html



1.4 Một số ứng dụng mạng(tt)

- W W W (tt) – Ví dụ

time

3. http server nhận yêu cầu, tạo *response message* (/~phu/courses/net-programming/index.html) trả kết quả, gởi message qua socket

4. http server đóng cầu nối TCP

5. http client nhận response message chứa file html và hiển thị. Phân tích file html để tìm các liên kết

6. Lặp lại các bước 1-5 cho mỗi liên kết(object)

1.4 Một số ứng dụng mạng(tt)

- W W W (tt)
  - Có hai dạng message trong http : *request, response*
  - http request message:
    - ASCII (human-readable format)

1.4 Một số ứng dụng mạng(tt)

- W W W (tt)
  - http request message:

request line (GET, POST, HEAD commands)

```
GET /~phu/index.html HTTP/1.0
User-agent: Mozilla/4.0
Accept: text/html, image/gif,image/jpeg
Accept-language:vn
```

header lines

(extra carriage return, line feed)

Carriage return, line feed indicates end of message

1.4 Một số ứng dụng mạng(tt)

- W W W (tt)
  - http response message:

status line (protocol status code status phrase)

```
HTTP/1.0 200 OK
Date: Thu, 06 Aug 1998 12:00:15 GMT
Server: Apache/1.3.0 (Unix)
Last-Modified: Mon, 22 Jun 1998 …...
Content-Length: 6821
Content-Type: text/html
```

header lines

```
data data data data data ...
```

data, e.g., requested html file

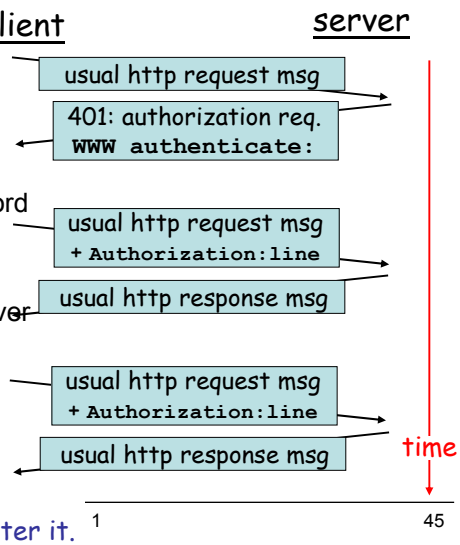
1.4 Một số ứng dụng mạng(tt)

Authentication goal: control access to server documents

- stateless: client must present authorization in each request
- authorization: typically name, password
  - `authorization:` header line in request
  - if no authorization presented, server refuses access, sends

    `WWW authenticate:`

    header line in response



Browser caches name & password so that user does not have to repeatedly enter it.

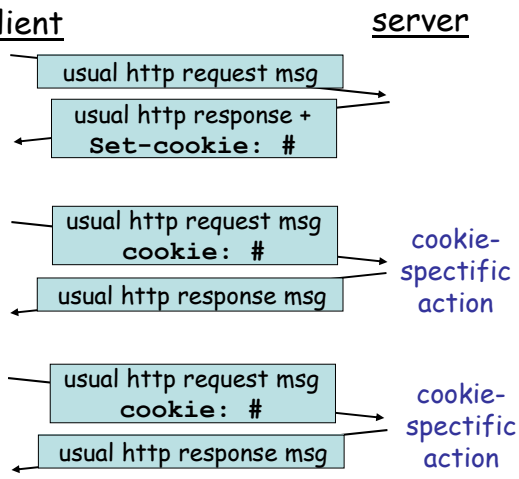
1.4 Một số ứng dụng mạng(tt)

Cookies

- server sends "cookie" to client in response mst

  `Set-cookie: 1678453`
- client presents cookie in later requests

  `cookie: 1678453`
- server matches presented-cookie with server-stored info
  - authentication
  - remembering user preferences, previous choices



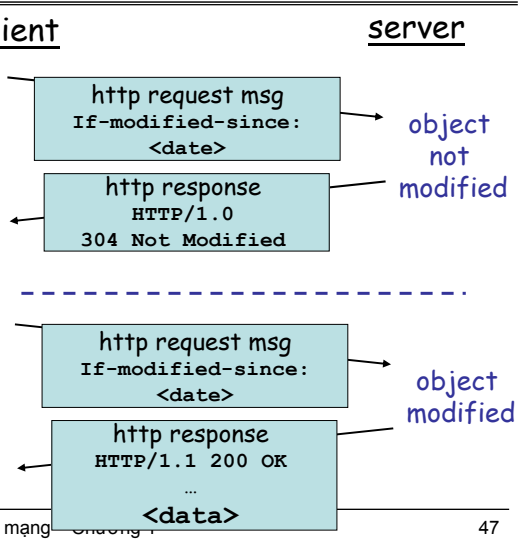
1.4 Một số ứng dụng mạng(tt)

Conditional GET

- Goal: don't send object if client has up-to-date stored (cached) version
- client: specify date of cached copy in http request

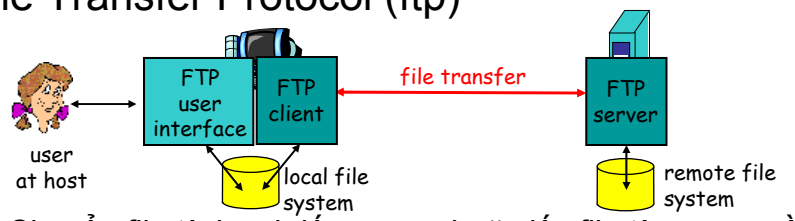
  `If-modified-since: <date>`
- server: response contains no object if cached copy up-to-date:

  `HTTP/1.0 304 Not Modified`



1.4 Một số ứng dụng mạng(tt)

- File Transfer Protocol (ftp)



  - Chuyển file từ local đến server hoặc lấy file từ server về local.
  - Hoạt động theo cơ chế client/server
  - FTP server chạy ở port 21.
  - Tham khảo : RFC 959

1.4 Một số ứng dụng mạng(tt)

- FTP (tt)
  - ftp client giao tiếp đến ftp server qua TCP ở port 21
  - Hai cầu nối TCP được thiết lập:
    - control: exchange commands, responses between client, server. “out of band control”
    - data: file data to/from server
  - ftp server hiện thực cơ chế “state”: current directory, earlier authentication

1.4 Một số ứng dụng mạng(tt)

Sample commands:

- sent as ASCII text over control channel
- **`USER`** ***`username`***
- **`PASS`** ***`password`***
- **`LIST`** return list of file in current directory
- **`RETR filename`** retrieves (gets) file
- **`STOR filename`** stores (puts) file onto remote host

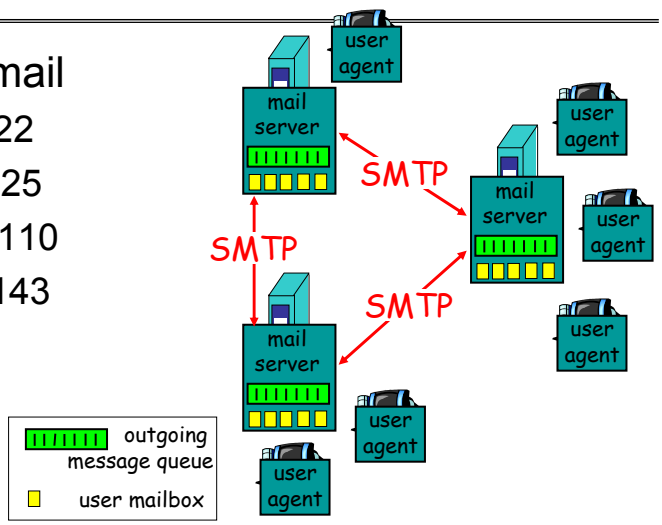
1.4 Một số ứng dụng mạng(tt)

Sample return codes

- status code and phrase (as in http)
- **331 Username OK, password required**
- **125 data connection already open; transfer starting**
- **425 Can't open data connection**
- **452 Error writing file**

1.4 Một số ứng dụng mạng(tt)

- Hệ thống E-mail
  - RFC 821, 822
  - SMTP: port 25
  - POP3: port 110
  - IMAP: port 143



1.4 Một số ứng dụng mạng(tt)

Hệ thống E-mail – Ví dụ về SMTP

S: 220 hamburger.edu
C: HELO crepes.fr
S: 250 Hello crepes.fr, pleased to meet you
C: MAIL FROM: <alice@crepes.fr>
S: 250 alice@crepes.fr... Sender ok
C: RCPT TO: <bob@hamburger.edu>
S: 250 bob@hamburger.edu ... Recipient ok
C: DATA
S: 354 Enter mail, end with "." on a line by itself
C: Do you like ketchup?
C: How about pickles?
C: .
S: 250 Message accepted for delivery
C: QUIT
S: 221 hamburger.edu closing connection

Ví dụ về POP3

- client commands:
  - **`user:`** declare username
  - **`pass:`** password
- server responses
  - **`+OK`**
  - **`-ERR`**

transaction phase, client:

- **`list:`** list message numbers
- **`retr:`** retrieve message by number
- **`dele:`** delete
- **`quit`**

```
S: +OK POP3 server ready
C: user alice
S: +OK
C: pass hungry
S: +OK user successfully logged on

C: list
S: 1 498
S: 2 912
S: .
C: retr 1
S: <message 1 contents>
S: .
C: dele 1
C: retr 2
S: <message 1 contents>
S: .
C: dele 2
C: quit
S: +OK POP3 server signing off
```

CHƯƠNG 2
THIẾT KẾ GIẢI THUẬT CHO CHƯƠNG TRÌNH CLIENT/SERVER

2.1 Giao tiếp socket (Socket Interface)

2.2 Thiết kế giải thuật cho chương trình client

2.3 Thiết kế giải thuật cho chương trình server

2.1 Giao tiếp socket

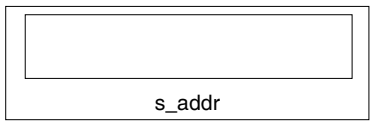
- Giao tiếp socket (Socket Interface) là các API dùng cho việc lập trình các ứng dụng mạng.
- Socket Interface được định nghĩa trong UNIX BSD, dựa trên việc mở rộng tập các system calls (access files).

  => Phần này chỉ giới thiệu các khái niệm, ý tưởng và các hàm, kiểu dữ liệu dùng cho lập trình mạng với Socket Interface.

2.1 Giao tiếp socket (tt)

- Một số cấu trúc dữ liệu
  - Cấu trúc địa chỉ Internet : định nghĩa dạng dữ liệu cấu trúc trong ngôn ngữ C. Cấu trúc này chỉ có 1 field kiểu u_long chứa địa chỉ IP 32 bit.

```
struct    in_addr
   {
              u_long    s_addr;
   };
```



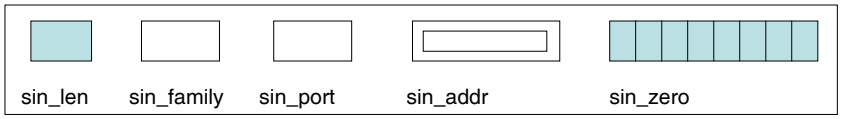
in_addr

Hình - cấu trúc địa chỉ Internet

2.1 Giao tiếp socket (tt)

- Một số cấu trúc dữ liệu (tt)
  - Cấu trúc địa chỉ socket :
    - địa chỉ này lưu trữ địa chỉ IP, chỉ số port, và dạng (family protocol)
    - Tên cấu trúc là *sockaddr_in* được biểu diễn ở hình trong slide kế. Trong đó:
      - *sin_len:* lưu trữ chiều dài cấu trúc của sockaddr_in
      - *sin_family:* dạng protocol của socket
      - *sin_port:* chỉ số port
      - *sin_addr:* địa chỉ in Internet của socket
      - *sin_zero[8]:* không dùng, đặt giá trị = 0

2.1 Giao tiếp socket (tt)



sockaddr_in

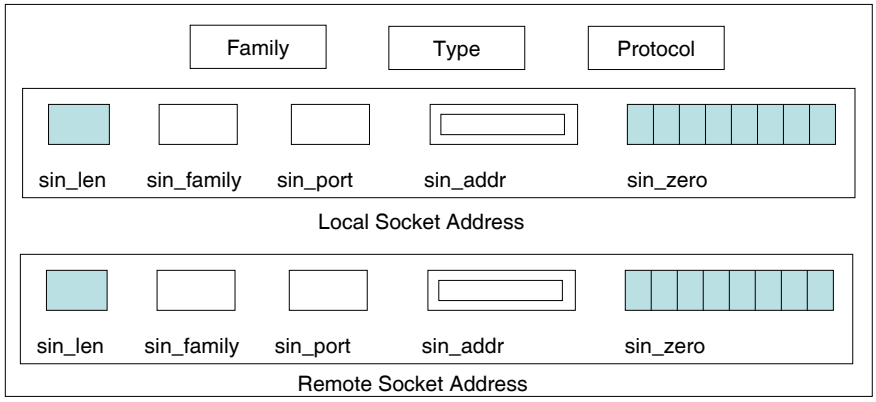
Hình - Cấu trúc địa chỉ socket

```
struct   sockaddr_in
{
        u_char          sin_len;
        u_short         sin_family;
        u_short         sin_port;
        struct  in_addr sin_addr;
        char            sin_zero[8];
};
```


2.1 Giao tiếp socket (tt)

- Một số cấu trúc dữ liệu (tt)
  - Cấu trúc socket :
    - socket được định nghĩa trong hệ điều hành bằng một cấu trúc, được xem như điểm nối để hai procceses giao tiếp với nhau.
    - Cấu trúc socket gồm 5 field được mô tả như hình trong slide kế:
      - *Family :* xác định protocol group
      - *Type :* xác loại socket, stream, datagram hay raw socket.
      - *Protocol :* là field thường gán giá trị bằng 0
      - *Local Socket Address* và *Remote Socket Address :* là địa chỉ socket của process cục bộ và từ xa.

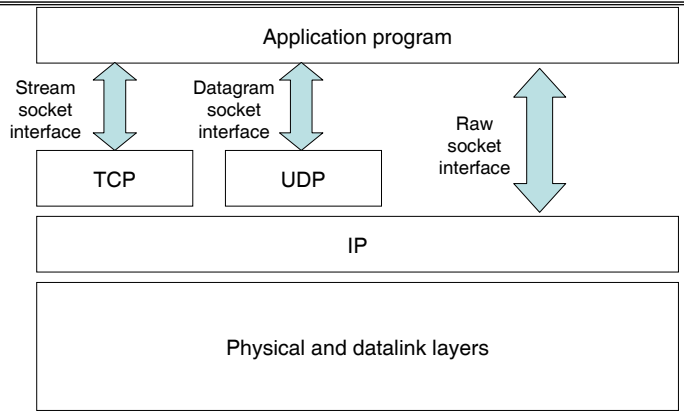
2.1 Giao tiếp socket (tt)



2.1 Giao tiếp socket (tt)

- Một số cấu trúc dữ liệu (tt)
  - Loại socket :
    - Giao tiếp socket định nghĩa 3 loại socket có thể dùng trên môi trường TCP/IP (hình ở slide kế).
    - Các loại socket gồm:
      - *Stream Socket:* dùng cho connection-oriented protocol như TCP.
      - *Datagram Socket:* dùng cho connectionless protocol như UDP.
      - *Raw Socket:* dùng cho một số protocol của một số ứng dụng đặc biệt, dùng các dịch vụ trực tiếp của lớp IP.

2.1 Giao tiếp socket (tt)



Hình - Các loại socket

- Một số cấu trúc dữ liệu (tt)
  - Thông tin remote host :
    - Thông tin được lưu trữ trong một cấu trúc *hostent* được trả về khi ứng dụng muốn ánh xạ địa chỉ tên miền bằng cách gọi hàm *gethostbyname():*

```
struct hostent * gethostbyname(const char * hostname);
struct hostent {
   char      *h_name;     char      **h_aliases;
   int       h_addrtype;  int       h_length;
   char      **h_addr_list;
}
```

2.1 Giao tiếp socket (tt)

- Một số cấu trúc dữ liệu (tt)
  - Byte Ordering
    - Big-Endian Byte Order : byte có trọng số lớn lưu trước.
    - Little -Endian Byte Order : byte có trọng số nhỏ lưu trước.
    - Tuỳ cấu trúc của mỗi máy, lưu trữ số theo một trong hai cách trên => khi giao tiếp mạng sẽ không đồng nhất.

2.1 Giao tiếp socket (tt)

- Một số cấu trúc dữ liệu (tt)
  - Byte Ordering (tt)
    - Network Byte Order : thứ tự lưu trữ dùng cho giao tiếp mạng.
    - Giao tiếp socket định nghĩa một số hàm để thực hiện các thao tác chuyển đổi :
      - *htons* và *htonl* : chuyển từ dạng lưu trữ của máy sang Network
      - *ntohs* và *ntohl* : chuyển từ dạng lưu trữ của Network sang dạng lưu trữ của máy.

2.1 Giao tiếp socket (tt)

- Các hàm dùng cho lập trình socket
  - Hàm *socket()* để tạo mới một socket

```
int socket (int family, int type, int protocol);
```

Hàm này tạo một socket, kết quả trả về là một số nguyên nhận dạng (socket descriptor), nếu có lỗi giá trị trả về là -1. Các thông số :

- *family:* họ socket
- *type:* kiểu socket (stream hay datagram)
- *protocol:* giao thức, thường đặt bằng 0

2.1 Giao tiếp socket (tt)

- Các hàm dùng cho lập trình socket (tt)
  - Hàm *bind()* để đăng ký với hệ thống

```
int bind (int sockfd, const struct sockaddr_in
  *localaddr, int localaddrlen);
```

Đăng ký socket đã khởi tạo với địa chỉ socket local. Trả về 0 nếu thành công, -1 nếu thất bại. Các thông số :

- *sockfd:* mô tả socket đã tạo bởi hàm ***socket()***
- *localaddr:*con trỏ chỉ đến địa chỉ socket của local
- *localaddrlen:* chiều dài của địa chỉ socket

2.1 Giao tiếp socket (tt)

- Các hàm dùng cho lập trình socket (tt)
  - Hàm *connect()* để kết nối đến server

```
int connect(int sockfd, const struct sockaddr_in
  *serveraddr, int serveraddrlen);
```

Dùng cho chương trình client thiết lập kết nối đến server. Trả về 0 nếu thành công, -1 nếu thất bại. Các thông số :

- *sockfd:* mô tả socket đã tạo bởi hàm ***socket()***
- *serveraddr:*con trỏ địa chỉ socket của server
- *serveraddrlen:* chiều dài của địa chỉ socket server

2.1 Giao tiếp socket (tt)

- Các hàm dùng cho lập trình socket (tt)
  - Hàm *listen()* để kết nối đến server

```
int listen(int sockfd, int backlog);
```

Hàm này dùng cho chương trình server connection-oriented để đặt socket ở trạng thái chờ, lắng nghe kết nối từ phía client. Trả về 0 nếu thành công, -1 nếu thất bại. Các thông số:

- *sockfd:* mô tả socket đã tạo bởi hàm ***socket()***
- *backlog:* số request có thể queued.

2.1 Giao tiếp socket (tt)

- Các hàm dùng cho lập trình socket (tt)
  - Hàm *accept()* : chấp nhận kết nối từ client đến.

```
int accept(int sockfd, const struct sockaddr_in
  *clientaddr, int *clientaddrlen);
```

Chấp nhận kết nối từ client, tạo socket mới. Giá trị là một socket descriptor của socket mới. Các thông số :

- *sockfd:* mô tả socket đã tạo bởi hàm ***socket()***
- *clientaddr:*con trỏ địa chỉ socket của client kết nối đến.
- *clientaddrlen:* chiều dài của clientaddr

2.1 Giao tiếp socket (tt)

- Các hàm dùng cho lập trình socket (tt)
  - Hàm *read()* để đọc dữ liệu từ socket

```
int read(int sockfd, const void *buf, int len);
```

Đọc dữ liệu từ connection vào bộ nhớ. Trả về số bytes đọc được nếu thành công, trả về 0 nếu không có dữ liệu, trả về -1 nếu thất bại. Các thông số :

- *sockfd:* mô tả socket đã tạo bởi hàm ***socket()***
- *buf:* con trỏ đến bộ đệm để lưu thông tin đọc được
- *len:* chiều dài của bộ đệm

2.1 Giao tiếp socket (tt)

- Các hàm dùng cho lập trình socket (tt)
  - Hàm *write()* để ghi dữ liệu

```
int write(int sockfd, const void *buf, int len);
```

Ghi dữ liệu từ bộ nhớ lên connection. Trả về số bytes ghi được nếu thành công, trả về -1 nếu thất bại. Các thông số :

- *sockfd:* mô tả socket đã tạo bởi hàm ***socket()***
- *buf:* con trỏ đến bộ đệm để lưu thông tin đọc được
- *len:* chiều dài của bộ đệm

2.1 Giao tiếp socket (tt)

- Các hàm dùng cho lập trình socket (tt)
  - Hàm *sendto()* để gởi dữ liệu

```
int sendto(int sockfd, const void *buf, int len, int
  flags, const struct sockaddr_in *toaddr, int
  toaddrlen);
```

Gởi dữ liệu đến một địa chỉ socket từ xa. Trả về số bytes gởi được nếu thành công, trả về -1 nếu thất bại. Các thông số :

- *sockfd, buf, len:* giống các hàm đã giới thiệu
- *flags: thường đặt bằng 0*
- *toaddr, toaddrlen:* địa chỉ socket đến và chiều dài.

2.1 Giao tiếp socket (tt)

- Các hàm dùng cho lập trình socket (tt)
  - Hàm *recvfrom()* để nhận dữ liệu

```
int recvfrom(int sockfd, const void *buf, int len, int
  flags, const struct sockaddr_in *fromaddr, int
  fromaddrlen);
```

Nhận dữ liệu từ một địa chỉ socket từ xa. Trả về số bytes gởi được nếu thành công, trả về -1 nếu thất bại. Các thông số :

- *sockfd, buf, len:* giống các hàm đã giới thiệu
- *fromaddr, fromaddrlen:* địa chỉ socket gởi đến và chiều dài.

2.1 Giao tiếp socket (tt)

- Các hàm dùng cho lập trình socket (tt)
  - Một số hàm dùng cho việc chuyển đổi

```
u_short     htons(u_short     host_short);
u_short     ntohs(u_short     network_short);
u_long      htonl(u_long      host_long);
u_long      ntohl(u_long      network_long);
char        *inet_ntoa(struct  in_addr  inaddr)
int         inet_aton(const  char *strptr,
                      struct  in_addr *addptr)
```

2.1 Giao tiếp socket (tt)

- Các hàm dùng cho lập trình socket (tt)
  - Một số hàm dùng cho việc thao tác dữ liệu

```
void   *memset ( void *dest, int chr, int len);
void   *memcopy ( void *dest, void *src, int len)
int      memcmp ( const void *first,
                  const void *second, int len)
```


2.2 Thiết kế giải thuật cho chương trình client

- Giải thuật cho chương trình client dùng TCP
  - Xác định địa chỉ server
  - Tạo socket.
  - Kết nối đến server.
  - Gởi/nhận dữ liệu theo giao thức lớp ứng dụng đã thiết kế.
  - Đóng kết nối.

2.2 Thiết kế giải thuật cho chương trình client (tt)

- Giải thuật cho chương trình client dùng UCP
  - Xác định địa chỉ server
  - Tạo socket.
  - Đăng ký socket với hệ thống.
  - Gởi/nhận dữ liệu theo giao thức lớp ứng dụng đã thiết kế đến server theo địa chỉ đã xác định.
  - Đóng kết nối.

2.3 Thiết kế giải thuật cho chương trình server

- Chương trình server có hai loại đơn giản : lặp (iterative) và đồng thời (concurrent).
- Hai dạng giao thức chương trình server có thể sử dụng là connection-oriented hoặc connectionless.
- Các slide kế tiếp trình bày cách thiết kế giải thuật cho các loại server kết hợp các đặc điểm trên

2.3 Thiết kế giải thuật cho chương trình server (tt)

- Giải thuật cho chương trình server iterative, connection-oriented:
  - Tạo socket, đăng ký địa chỉ socket với hệ thống.
  - Đặt socket ở trạng thái lắng nghe, chờ và sẵn sàng cho việc kết nối từ client.
  - Chấp nhận kết nối từ client, gởi/nhận dữ liệu theo giao thức lớp ứng dụng đã thiết kế.
  - Đóng kết nối sau khi hoàn thành, trở lại trạng thái lắng nghe và chờ kết nối mới.

2.3 Thiết kế giải thuật cho chương trình server (tt)

- Giải thuật cho chương trình server iterative, connectionless:
  - Tạo socket và đăng ký với hệ thống.
  - Lặp công việc đọc dữ liệu từ client gởi đến, xử lý và gởi trả kết quả cho client theo đúng giao thức lớp ứng dụng đã thiết kế.

2.3 Thiết kế giải thuật cho chương trình server (tt)

- Các yêu cầu cho concurrent Server:
  - Tại một thời điểm có thể xử lý nhiều yêu cầu từ client.
  - Chương trình concurrent server có thể chạy trên máy chỉ có 1 CPU.
  - Hệ thống phải hỗ trợ multi-tasking

2.3 Thiết kế giải thuật cho chương trình server (tt)

- Giải thuật cho chương trình concurrent, connectionless server:
  - Tạo socket, đăng ký với hệ thống.
  - Lặp việc nhận dữ liệu từ client, đối với một dữ liệu nhận, tạo mới một process để xử lý. Tiếp tục nhận dữ liệu mới từ client.
  - Công việc của process mới :
    - Nhận thông tin của process cha chuyển đến, lấy thông tin socket
    - Xử lý và gởi thông tin về cho client theo giao thức lớp ứng dụng đã thiết kế.
    - Kết thúc.

2.3 Thiết kế giải thuật cho chương trình server (tt)

- Giải thuật cho chương trình concurrent, connection-oriented server:
  - Tạo socket, đăng ký với hệ thống.
  - Đặt socket ở chế độ chờ, lắng nghe kết nối.
  - Khi có request từ client, chấp nhận kết nối, tạo một process con để xử lý. Quay lại trạng thái chờ, lắng nghe kết nối mới.
  - Công việc của process mới gồm:
    - Nhận thông tin kết nối của client.
    - Giao tiếp với client theo giao thức lớp ứng dụng đã thiết kế.
    - Đóng kết nối và kết thúc process con.

2.3 Thiết kế giải thuật cho chương trình server (tt)

- Multi-protocol Server (TCP,UDP)
  - Dùng một chương trình , mở một master socket cho cả TCP và UDP.
  - Dùng hàm hệ thống (*select*)để chọn lựa TCP socket hay UDP socket sẵn sàng.
  - Tùy vào protocol (TCP, UDP) để xử lý gởi nhận thông điệp theo đúng giao thức của lớp ứng dụng.
  - Tham khảo thêm RFC 1060

2.3 Thiết kế giải thuật cho chương trình server (tt)

- Multi-service Server
  - Tạo một điểm giao tiếp chung.
  - Với mỗi request, xem loại dịch vụ cần xử lý.
  - Với mỗi loại dịch vụ, xử lý riêng biệt
  - Có thể kết hợp Multi-service và Multi-protocol để thiết kế cho chương trình server.

CHƯƠNG 3
LẬP TRÌNH MẠNG TRÊN CÁC MÔI TRƯỜNG PHỔ DỤNG

3.1 Lập trình mạng trong UNIX

3.2 Các hàm hỗ trợ lập trình mạng trong UNIX

3.3 Lập trình mạng trong Windows với TCP/IP

3.4 Các hàm hỗ trợ lập trình mạng trong Windows

2.1 Lập trình mạng trong UNIX

- Lập trình mạng trong môi trường UNIX dùng socket có các hàm giống BSD Socket Interface đã giới thiệu.

3.2 Các hàm hỗ trợ lập trình mạng trong UNIX

- Địa chỉ socket trên Internet và địa chỉ IP:

```
#include <netinet/in.h>
struct sockaddr_in {
  short   sin_family;
  u_short sin_port;
  struct  in_addr sin_addr;
  char    sin_zero[8];
};
struct in_addr{
  u_long  s_addr;
}
```

3.2 Các hàm … (tt)

- Địa chỉ socket tổng quát:

```
#include <sys/socket.h>
struct sockaddr {
  short   sa_family;
  char    sa_data[14];
};
```

- Họ địa chỉ socket được định nghĩa trong <sys/socket.h>:

```
#define AF_UNIX 1/* local to host (pipes, portals) */
#define AF_INET 2 /* internetwork: UDP, TCP, etc. */
```


3.2 Các hàm … (tt)

- Cấu trúc địa chỉ máy từ xa.

```
struct hostent {
  char    *h_name;
  char    **h_aliases;
  int     h_addrtype;
  int     h_length;
  char    **h_addr_list;
  #define h_addr h_addr_list[0];
}
```

3.2 Các hàm … (tt)

- Tạo socket:

  #include <sys/types.h>

  #include <sys/socket.h>

  int **socket**(int *family*, int *type*, int *protocol*);

  Ví dụ tạo socket:

```
int sockfd;
//Tạo stream socket
sockfd = socket(AF_INET,SOCK_STREAM,0);
//Tạo datagram socket
sockfd = socket(AF_INET,SOCK_DGRAM,0);
```


3.2 Các hàm … (tt)

- Liên kết socket với địa chỉ socket(đăng ký)

  int **bind**(int *sockfd*, struct sockaddr **myaddr*,
  int *myaddrlen*);

  Ví dụ bind socket vừa tạo với địa chỉ socket:

```
struct sockaddr_in myaddr;
bzero((char*)&myaddr,sizeof(myaddr));
myaddr.sin_family = AF_INET;
myaddr.sin_port = htons(portno);
myaddr.sin_addr.s_addr = htonl(INADDR_ANY);
if (bind(sockfd, (struct sockaddr *) &myaddr,
  sizeof(myaddr)) < 0) error("ERROR on binding");
```

3.2 Các hàm … (tt)

- Chuyển socket về trạng thái chờ kết nối.

  int **listen**(int *sockfd*, int *backlog*);

- Chấp nhận yêu cầu kết nối từ client.

  int **accept**(int *sockfd*, struct sockaddr_in **peer*,
  int **addrlen*);

```
struct sockaddr_in cli_addr;int newsockfd,clilen;
listen(sockfd,5);
clilen = sizeof(cli_addr);
newsockfd = accept(sockfd,
         (struct sockaddr*)&cli_addr, &clilen);
if (newsockfd < 0) error("ERROR on accept");
```


3.2 Các hàm … (tt)

- Hàm kết nối đến server

  int **connect**(int *sockfd*, struct sockaddr *s*ervaddr*,
  int **addrlen*);

  Ví dụ:

```
struct sockaddr_in servaddr;
bzero((char*)&servaddr,sizeof(myaddr));
servaddr.sin_family = AF_INET;
servaddr.sin_port = htons(portno);
servaddr.sin_addr.s_addr = inet_addr(serverIP);
if (connect(sockfd, (struct sockaddr *) &servaddr,
  sizeof(servaddr)) < 0)
  error("Can not connect to server");
```

3.2 Các hàm … (tt)

- Các hàm truyền nhận dữ liệu:

int **read**(int *fd*,char **buf*, int *nbytes*);

int **write**(int *fd,char *buf,*int *nbytes*);

int **send**(int *sockfd*, char **buf,*int *nbytes,*int *flags*);

int **recv**(int *sockfd*, char **buf,*int *nbytes,*int *flags*);

int **sendto**(int *sockfd*, char **buf*, int *len*, int *flags*,
struct sockaddr_in **toaddr*, int *toaddrlen*);

int **recvfrom**(int *sockfd*, char **buf*, int *len*, int *flags*,
struct sockaddr_in **fromaddr*, int *fromaddrlen*);

3.2 Các hàm … (tt)

- Tạo process con để xử lý từng kết nối:

int fork(void);

```
int pid;
while(1){
    newsockfd = accept(sockfd, (struct sockaddr*)
                &cli_addr, &clilen);
    if ((pid=fork())==0){
          close(sockfd);
          process(newsockfd);
          close(newsockfd)
          exit(0);
    }
    close(newsockfd);
}
```

3.3 Lập trình mạng trong Windows với TCP/IP

- Dùng thư viện WinSock API (Windows Sockets Application Programming Interface) để hiện thực.
- Cần có thư viện WINSOCK.DLL hoặc WINSOCK32.DLL (32-bit Windows).
- Cần include các hàm và cấu trúc từ WINSOCK.H hoặc WINSOCK2.H
- Có thể biên dịch dạng dòng lệnh :

cl –o dest-file src-file ws2_32.lib

3.3 Lập trình mạng trong Windows với TCP/IP

- WinSock hiện thực Berkeley Sockets Interface trên môi trường Windows.
- WinSock có nhiều mở rộng thêm so với Berkeley Sockets.
  - Hỗ trợ kiến trúc Windows Message-Driven hay event-driven.
  - Hỗ trợ kiến trúc nonpreemptive của Windows

3.4 Các hàm hỗ trợ lập trình mạng trong Windows

- Khởi tạo WinSock:

  int WSAStartup(WORD wVersionRequired,
  LPWSADATA lpWSAData);

  Ví dụ :

```
WORD wVerRequested = MAKEWORD(0,1);
WSADATA wsaData;
if(WSAStartup(wVerRequested,&wsaData)!=0){
// process error
}
```


3.4 Các hàm WinSock (tt)

- Kết thúc WinSock

  int WSAClearup();
- Hàm lấy thông tin lỗi :

  int WSAGetLastError(void);

3.4 Các hàm WinSock (tt)

- Cám hàm dùng cho chuyển đổi:
  - Chuyển địa chỉ IP dạng chuỗi sang nhị phân:

```
unsigned long inet_addr(const char
                        FAR *cp);
```

  - Chuyển địa chỉ IP dạng nhị phân sang dạng chuỗi:

```
char FAR *inet_ntoa(struct in_addr in);
```

  - Lấy địa chỉ máy cục bộ:

```
int gethostname(char FAR*name,int len);
```

  - Lấy địa chỉ máy từ xa:

```
struct hostent FAR *gethostbyname(const
 char FAR *name);
```


3.4 Các hàm WinSock (tt)

Ví dụ về lấy địa chỉ

```
PHOSTENT phe =
   gethostbyname(condlg.m_remotehost);
char szTemp[128];
if (phe == NULL) {
   wsprintf(szTemp,"Not exist '%s'",
       condlg.m_remotehost);
   MessageBox(szTemp); return;
}
memcpy((char FAR *)&(ser_addr.sin_addr), phe-
>h_addr,phe->h_length);
```

3.4 Các hàm WinSock (tt)

- Hàm tạo socket

SOCKET socket (int *af*, **int** *type*, **int** *protocol* **);**

af :họ socket (thường dùng AF_INET : Internet)

type : loại socket (SOCK_STREAM, SOCK_DGRAM)

protocol : giao thức, thường đặt = 0 để lấy giá trị default

trả về giá trị INVALID_SOCKET nếu có lỗi

Ví dụ về hàm tạo socket :

```
ser_sock=socket(AF_INET,SOCK_STREAM,0);
if(ser_sock==INVALID_SOCKET) {
 MessageBox("Can not create socket");
 return TRUE;
}
```


3.4 Các hàm WinSock (tt)

- Hàm đăng ký địa chỉ socket với hệ thống

int bind (SOCKET *s*, **const struct sockaddr FAR** **addr*,
int *addrlen* **);**

s : mô tả socket đã được khởi tạo.

addr : địa chỉ socket.

addrlen : chiều dài *addr*

Nếu có lỗi trả về giá trị SOCK_ERROR

3.4 Các hàm WinSock (tt)

Ví dụ về lệnh bind :

```
//…
char message[100];
SOCKADDR_IN addr;
addr.sin_family=AF_INET;
addr.sin_port=htons(2000);
addr.sin_addr.s_addr=htonl(INADDR_ANY);
if(bind(s,(LPSOCKADDR)&addr,sizeof(addr))==
SOCKET_ERROR){
    wsprintf(message, "Can not bind socket : %d",
    WSAGetLastError());
    MessageBox(message);
    return TRUE;
}
```


3.4 Các hàm WinSock (tt)

- Hàm chuyển socket về trạng thái chờ

int listen (SOCKET *s*, **int** *backlog* **);**

backlog : chiều dài hàng đợi

trả về giá trị SOCKET_ERROR nếu có lỗi

Ví dụ về hàm listen:

```
if(listen(s,5)==SOCKET_ERROR) {
 wsprintf(message, "Can not listen : %d",
WSAGetLastError());
 MessageBox(message);
 return TRUE;
}
```

3.4 Các hàm WinSock (tt)

- Hàm chấp nhận kết nối từ client.

  SOCKET accept (SOCKET *s*, **struct sockaddr FAR ***addr**,**
  int FAR ***addrlen* **);

  s : là mô tả socket của server.

  addr : con trỏ địa chỉ socket của client kết nối đến.

  addrlen : chiều dài của *addr*

3.4 Các hàm WinSock (tt)

Ví dụ về hàm accept :

```
SOCKADDR_IN client_addr;SOCKET cli_s;
IN_ADDR clientIP;
int len=sizeof(client_addr);
cli_S=accept(s,(LPSOCKADDR)&client_addr,&len);
   if(sock==INVALID_SOCKET) {
     MessageBox("Can not eccept");
     return TRUE;    }
   else {
     memcopy(&clientIP,&client_addr.sin_addr.s_addr,4);
     wsprintf(message,"Client IP= %s and port= %d",
       inet_ntoa(clientIP),ntohs(cli_s.sin_port));
       …
   }
```

3.4 Các hàm WinSock (tt)

- Hàm thiết lập kết nối đến server.

int connect (SOCKET *s*, **const struct sockaddr FAR** ***name*, **int** *namelen* **);**

s : socket của chương trình local

name : địa chỉ socket của server.

namelen : chiều dài của name

Trả về giá trị SOCKET_ERROR nếu có lỗi

3.4 Các hàm WinSock (tt)

Ví dụ về hàm connect ➔

```
…
SOCKADDR_IN ser_addr;
ser_addr.sin_family=AF_INET;
ser_addr.sin_port=htons(2000);
ser_addr.sin_addr.s_addr=
          inet_addr("172.28.10.20");
if(connect(s,(LPSOCKADDR)&ser_addr,
   sizeof(ser_addr))==SOCKET_ERROR){
  MessageBox("Can not connect to server");
}
```

3.4 Các hàm WinSock (tt)

Lệnh gởi dữ liệu
int send (SOCKET *s*, **const char FAR *** *buf*, **int** *len*,
int *flags* **);**
buf : chuỗi dữ liệu cần gởi
len : chiều dài của *buf*
flags : thường đặt giá trị 0
Trả về số byte dữ liệu gởi được, nếu lỗi trả về SOCKET_ERROR

```
//...
char buf[255];
lstrcpy(mesg,"Hello World");
if (send(s,buf,strlen(buf),0) ==SOCKET_ERROR) {
  MessageBox("Can not send data");
  return;
}
```


3.4 Các hàm WinSock (tt)

int recv (SOCKET *s*, **char FAR*** *buf*, **int** *len*, **int** *flags* **);**
Các thông số tương tự hàm *send*

```
//…
#define BUFSIZE (100)
char buf[BUFSIZE];
int nByteRecv;
nByteRecv = recv(s,buf, BUFSIZE,0);
if (nByteRecv == SOCKET_ERROR) {
  MessageBox("Error receive data");
  return;
} //…
```

3.4 Các hàm WinSock (tt)

- Các hàm dùng cho UDP

int sendto (SOCKET *s*, **const char FAR** **buf*, **int** *len*,
int *flags,* **const struct sockaddr FAR** **to*, **int** *tolen*);

to : địa chỉ socket của process muốn gởi đến

int recvfrom (SOCKET *s*, **char FAR*** *buf*, **int** *len*,
int *flags,***const struct sockaddr FAR**
***from*, int FAR *fromlen ***);**

from : địa chỉ socket của process gởi dữ liệu đến

3.4 Các hàm WinSock (tt)

Ví dụ về hàm sendto:

```
#define BUFSIZE (100)
char buf[BUFSIZE];
int nByteSend;
SOCKADDR_IN to;
to.sin_family = AF_INET;
to.sin_port = 2000;
to.sin_addr.s_addr = inet_addr("127.0.0.1");
lstrcpy(buf,"Hello World");
nByteSend = sendto(s,buf,lstrlen(buf),0,
                   (LPSOCKADDR)&to,sizeof(to));
if(nByteSend == SOCKET_ERROR )
//…
```

3.4 Các hàm WinSock (tt)

Ví dụ về hàm recvfrom

```
#define BUFSIZE (100)
char buf[BUFSIZE];
int nByteRecv;
SOCKADDR_IN from;
int fromlen;
nByteRecv = recvfrom(s,buf,BUFSIZE,0,
                (LPSOCKADDR)&from,&fromlen);
if(nByteRecv == SOCKET_ERROR )
//…
```


3.4 Các hàm WinSock (tt)

Hàm khai báo nhận event từ network cho socket.

int WSAAsyncSelect (SOCKET *s*, **HWND** *hWnd*, **unsigned int** *wMsg*, **long** *lEvent*);

hWnd : cửa sổ nhận sự kiện.

wMsg: thông điệp gởi đến.

lEvent : sự kiện của socket cần xử lý.

• Khi dùng hàm này, socket sẽ được chuyển về trạng thái nonblocking.

• Đối với mỗi socket thì chỉ khai báo một thông điệp đến. Có thể khai báo nhiều sự kiện bằng phép OR (|)

3.4 Các hàm WinSock (tt)

Ví dụ về hàm **WSAAsyncSelect**

```
BOOL CServerDlg::OnInitDialog() {
  //s là socket đã được tạo,
  //đã sử dụng các hàm bind và listen
  if(WSAAsyncSelect(s,m_hWnd,WM_USER+1,
                FD_ACCEPT)==SOCKET_ERROR) {
     return TRUE;
  }
  return FALSE;
}
```


3.4 Các hàm WinSock (tt)

- Sau khi dùng hàm WSAAsyncSelect, ta phải khai báo hàm để xử lý biến cố tương ứng.

```
BEGIN_MESSAGE_MAP(CServerDlg, CDialog)
   //{{AFX_MSG_MAP(CServerDlg)
  ON_MESSAGE(WM_USER+1,OnAccept)
  //...
  //}}AFX_MSG_MAP
END_MESSAGE_MAP()
```

- Viết hàm xử lý biến cố tương ứng

```
LONG CServerDlg::OnAccept(WPARAM wParam,
                          LPARAM lParam){
}
```

3.4 Các hàm WinSock (tt)

- Có thể viết code cho hàm WindowProc để xử lý sự kiện network.

```
LRESULT CServerDlg::WindowProc(UINT message, WPARAM wParam,
  LPARAM lParam) {
switch (message) {
  case WM_USER+1 :
     OnAccept(); return 1;
  case WSA_RDCLOSE :
     if (WSAGETSELECTEVENT(lParam) == FD_READ)
          Read_Process(wParam);
```

Socket desciptor

3.4 Các hàm WinSock (tt)

- Hàm đóng socket :

  int closesocket (SOCKET s);

  Hàm trả về giá trị 0 nếu thành công, nếu thất bại trả về giá trị SOCKET_ERROR

3.4 Các hàm WinSock (tt)

- Lập trình trên mạng trên Windows bằng MFC : dùng các lớp CWinSock, CDatagramSocket, CStreamSocket.
- Tham khảo thêm MSDN

CHƯƠNG 4
LẬP TRÌNH MẠNG VỚI JAVA

4.1 Giới thiệu ngôn ngữ Java
4.2 Ví dụ về lập trình mạng bằng Java
4.3 Khái niệm Stream và Multithreading
4.4 Thư viện java.net.*

4.1 Giới thiệu ngôn ngữ Java

- Là ngôn ngữ lập trình hướng đối tượng trong sáng, ra đời vào khoảng năm 1995 do Sun MicroSystems xây dựng.
- Là ngôn ngữ thông dịch, chạy trên kiến trúc máy ảo (Java Virtual Machine).
- Hỗ trợ mạng mẽ lập trình mạng, bảo mật, multi-thread…
- Sử dụng thư viện chuẩn JDK

4.1 Giới thiệu ngôn ngữ Java

- JVM hỗ trợ trên nhiều flatform => Java có tính portable “write one-run everywhere”.
- Hiện có rất nhiều công cụ hỗ trợ lập trình Java như : JBuilder (5), Visual Café, Microsoft Visual J++…
- JDK (Java Development Kit) phiên bản mới 1.4.1 trên http://java.sun.com/j2se/1.4.1/index.html

4.1 Giới thiệu ngôn ngữ Java

- Cài đặt : download chương trình và cài đặt lên máy tính theo hướng dẫn. VD:
  - Windows : C:\JDK
  - Unix : /usr/local/jdk
- Đặt biến môi trường PATH đến thư mục BIN trong thư mục cài đặt:
  - Windows : SET PATH=c:\jdk\bin;%PATH%
  - Unix :
    - PATH=$PATH:/usr/local/jdk/bin
    - export PATH

4.1 Giới thiệu ngôn ngữ Java

- Đặt biến môi trường CLASSPATH đến các package có sử dụng trong chương trình. VD:
  - set CLASSPATH=c:\lib\jdbc.zip;c\lib\xml4j.jar;.
- Lập trình : có thể dùng trình soạn thảo bất kỳ, lưu với tên file .java
- Biên dịch :
  - javac file-name.java
- Chạy :
  - java file-name

4.2 Ví dụ LTM với Java

Chương trình client/server Echo

- Chương trình Client

```
1. //file Client.java
2. import java.net.*;
3. import java.io.*;
4. public class Client{
5.      public static void main(String args[]) throws Exception{
6.              Socket clientsock;
7.              DataOutputStream output;
8.              BufferedReader input;//bộ đệm đọc dữ liệu
9.              clientsock = new Socket("127.0.0.1",2000);
10.             input = new BufferedReader(new
11.             InputStreamReader(clientsock.getInputStream()));
12.             output = new DataOutputStream(
13.                             clientsock.getOutputStream());
```


4.2 Ví dụ… (tt)

- Chương trình Client

```
14.             BufferedReader keyInput = new BufferedReader(new
15.                              InputStreamReader(System.in));
16.     System.out.print("Enter sentence to send to server:");
17.             String data = keyInput.readLine();
18.             output.writeBytes(data+"\n");
19.             int recvByte;
20.             System.out.print("Data receicved: ");
21.             System.out.println(input.readLine());
22.             clientsock.close();
23.     }//main
24. }//class
```

- Biên dịch: javac Client.java
- Thực thi : java Client

4.2 Ví dụ… (tt)

- Chương trình Server

```
1. //file Server.java
2. import java.net.*;
3. import java.io.*;
4. public class Server{
5.      public static void main(String args[]) throws Exception{
6.              ServerSocket serversock = new ServerSocket(2000);
7.              DataOutputStream output;//stream xuat du lieu
8.              BufferedReader input;//stream doc du lieu
9.              for(;;){
10.               Socket client = serversock.accept();
11.               output = new DataOutputStream(
12.                           client.getOutputStream());
```


4.2 Ví dụ… (tt)

- Chương trình Server (tt)

```
13.               input = new BufferedReader(new
14.             InputStreamReader(client.getInputStream()));
15.               String data = input.readLine();
16.               System.out.println("Recv from client: "+data);
17.               output.writeBytes(data+"\n");
18.               output.flush();
19.             }//for
20.     } //main
21. }//class
```

- Dịch : javac Server.java
- Chạy : java Server

4.3 Stream và Multithreading

- Khái niệm stream trong ngôn ngữ Java:
  - Stream : hỗ trợ chức năng truy xuất I/O trong ngôn ngữ Java.
  - Các công việc truy xuất I/O có thể kể đến như file, kết nối mạng, bàn phím(thiết bị nhập chuẩn), màn hình (tb xuất chuẩn)…
  - Stream là môi trường dẫn dữ liệu, không quan tâm đến định dạng của dữ liệu
  - Các lớp stream được cung cấp ở gói java.io.*;

4.3 Stream và Multithreading (tt)

- Khái niệm stream ...(tt):
  - Được chia ra làm hai loại chính input stream là stream chứa dữ liệu nhập; output stream là stream chứa dữ liệu xuất.
  - Hai lớp cơ bản trong Java xử lý nhập xuất là InputStream và OutputStream.
  - Các lớp dẫn xuất thường dùng của InputStream : BufferedInputStream, DataInputStream, ByteArrayInputStream, StringBufferInputStream.

4.3 Stream và Multithreading (tt)

- Khái niệm stream ...(tt):
  - Các lớp dẫn xuất thường dùng của OutputStream : BufferedOutputStream, DataOutputStream, ByteArrayOutputStream
  - Các lớp thường dùng cho truy xuất tập tin : File, RadomAcessFile…
  - Chi tiết lập trình xem thêm Java docs của JDK

4.3 Stream và Multithreading (tt)

- Thread và Multithread trong Java.
  - Thread : là một đối tượng có thể chạy nhiều phiên bản đồng thời.
  - Java hỗ trợ lập trình thread trong bản thân ngôn ngữ.
  - Có hai cách để tạo Thread :
    - Xây dựng class extends Thread.
    - Implements interface Runable

4.3 Stream và Multithreading (tt)

- Multithreading
  - Ví dụ :

```
public static void main(String args[]){
  //...
  while(true){
    Socket newsock = server.accept();
    ClientThread ct = new ClientThread(newsock);
    ct.start();
  }
}
```


4.3 Stream và Multithreading (tt)

```
class ClientThread extends Thread{
  Socket sock;
  public ClientThread(Socket sock){
    this.sock = sock;
  }
  public void run(){
    //xu ly
  }
}
```

4.4 Thư viện java.net.*

- Lớp InetAddress : dùng để thao tác về địa chỉ Internet, các phương thức thường dùng:
  - public byte[] **getAddress**(): Returns the raw IP address of this object
  - public static InetAddress[] **getAllByName**(String host) throws UnknownHostException
  - public String getHostAddress()
    - Returns the IP address string "%d.%d.%d.%d".
  - public static InetAddress getByName(String host) throws UnknownHostException

4.4 Thư viện java.net.*

- Lớp Socket : dùng cho chương trình client kết nối đến máy chủ
  - public **Socket**(String host, int port) throws UnknownHostException, IOException
    - Creates a stream socket and connects it to the specified port number on the named host.
  - public **Socket**(InetAddress address, int port) throws IOException
    - Creates a stream socket and connects it to the specified port number at the specified IP address.
  - public **Socket**(String host, int port, boolean stream) throws IOException
    - Creates a stream socket and connects it to the specified port number on the named host.

4.4 Thư viện java.net.*

- Lớp Socket (tt):
  - public InputStream **getInputStream**() throws IOException
  - public InetAddress **getInetAddress**()
    - Returns the address to which the socket is connected.
  - int **getPort**()
    - Returns the remote port to which this socket is connected.
  - public OutputStream **getOutputStream**() throws IOException
    - mReturns an output stream for this socket.

4.4 Thư viện java.net.*

- Lớp ServerSocket : dùng cho chương trình server tao socket và chấp nhận kết nối.
  - **ServerSocket**(int port)
    - Creates a server socket on a specified port.
  - **ServerSocket**(int port, int backlog)
    - Creates a server socket and binds it to the specified local port number, with the specified backlog.
  - **ServerSocket**(int port, int backlog, InetAddress bindAddr)
    - Create a server with the specified port, listen backlog, and local IP address to bind to.

4.4 Thư viện java.net.*

- Lớp ServerSocket (tt)
  - public Socket **accept**() throws IOException
    - Listens for a connection to be made to this socket and accepts it. The method blocks until a connection is made.
  - public void **close**() throws IOException
    - Closes this socket.
  - public void **setSoTimeout**(int timeout) throws SocketException
    - Enable/disable SO_TIMEOUT with the specified timeout, in milliseconds.
  - public String **toString**()
    - Returns the implementation address and implementation port of this socket as a String.

4.4 Thư viện java.net.*

- Lớp DatagramSocket : sử dụng cho chương trình dùng UDP
  - public **DatagramSocket**() throws SocketException
    - Constructs a datagram socket and binds it to any available port on the local host machine.
  - public **DatagramSocket**(int port) throws SocketException
    - Constructs a datagram socket and binds it to the specified port on the local host machine.
  - public **DatagramSocket**(int port, InetAddress laddr) throws SocketException
    - Creates a datagram socket, bound to the specified local address. The local port must be between 0 and 65535 inclusive.

4.4 Thư viện java.net.*

- Lớp DatagramSocket (tt) :
  - public void **connect**(InetAddress address, int port)
    - Connects the socket to a remote address for this socket.
  - public void **disconnect**()
    - Disconnects the socket. This does nothing if the socket is not connected.
  - public void **receive**(DatagramPacket p) throws IOException
    - Receives a datagram packet from this socket
  - public void **send**(DatagramPacket p) throws IOException
    - Sends a datagram packet from this socket.

4.4 Thư viện java.net.*

- Lớp DatagramPacket : dùng xây dựng các gói tin để trao đổi theo giao thức UDP.
  - public **DatagramPacket**(byte[] buf, int length)
    - Constructs a DatagramPacket for receiving packets of length length.
  - public **DatagramPacket**(byte[] buf, int length, InetAddress address, int port)
    - Constructs a datagram packet for sending packets of length length to the specified port number on the specified host.
  - public **DatagramPacket**(byte[] buf, int offset, int length)
    - Constructs a DatagramPacket for receiving packets of length length, specifying an offset into the buffer

4.4 Thư viện java.net.*

- Lớp DatagramPacket (tt)
  - public InetAddress **getAddress**()
    - Returns the IP address of the machine to which this datagram is being sent or from which the datagram was received.
  - public byte[] **getData**()
    - Returns the data received or the data to be sent.
  - public int **getLength**()
    - Returns the length of the data to be sent or the length of the data received.
  - public int **getPort**()
    - Returns the port number on the remote host.

4.4 Thư viện java.net.*

- Lớp DatagramPacket (tt)
  - public void **setAddress**(InetAddress iaddr)
    - Sets the IP address of the machine to which this datagram is being sent.
  - public void **setPort**(int iport)
    - Sets the port number on the remote host to which this datagram is being sent.
  - public void **setData**(byte[] buf)
    - Set the data buffer for this packet.
  - public void **setData**(byte[] buf, int offset, int length)
    - Set the data buffer for this packet.

4.4 Thư viện java.net.*

- Lớp URL : kết nối đến một tài nguyên Internet.
  - public **URL**(String spec) throws MalformedURLException
    - Creates a URL object from the String representation.
  - public **URL**(String protocol, String host, String file) throws MalformedURLException
    - Creates a URL from the specified protocol name, host name, and file name. The default port for the specified protocol is used.
  - public **URL**(String protocol, String host, int port, String file) throws MalformedURLException
    - Creates a URL object from the specified protocol, host, port number, and file. Specifying a port number of -1 indicates that the URL should use the default port for the protocol.

4.4 Thư viện java.net.*

- Lớp URL(tt)
  - public final Object **getContent**() throws IOException
    - Returns the contents of this URL.
  - public String **getFile**()
    - Returns the file name of this URL.
  - public URLConnection **openConnection**() throws IOException
    - Returns a URLConnection object that represents a connection to the remote object referred to by the URL.
  - public final InputStream **openStream**() throws IOException
    - Opens a connection to this URL and returns an InputStream for reading from that connection.