LẬP TRÌNH MẠNG DÙNG SOCKET

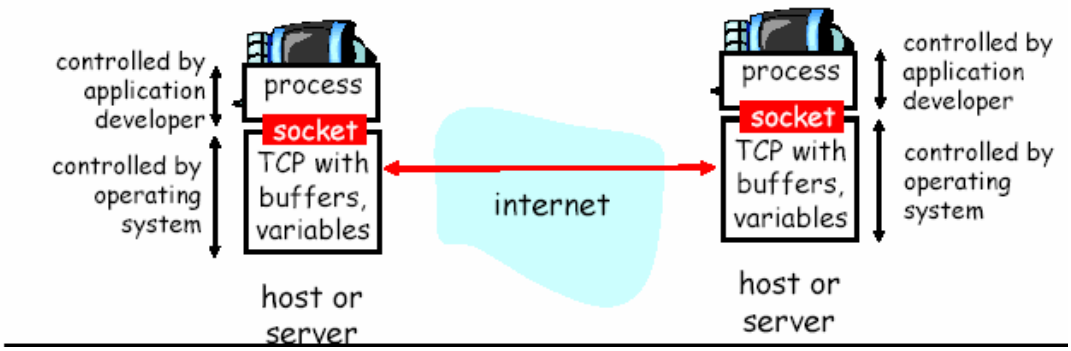
CHƯƠNG 1

KHÁI NIỆM VỀ SOCKET

- **Socket API**
  - Được giới thiệu ở BSD4.1 UNIX, 1981
  - Được ứng dụng khởi tạo, sử dụng và 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

KHÁI NIỆM VỀ SOCKET

- 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



THIẾT KẾ GIẢI THUẬT CLIENT/SERVER

- **Thiết kế giải thuật cho client**
  - Giải thuật cho chương trình client dùng UDP
    - Xác định địa chỉ server.
    - Tạo socket.
    - Gởi/nhận dữ liệu theo giao thức lớp ứng dụng đã thiết kế.
    - Đóng socket.
  - 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.

THIẾT KẾ GIẢI THUẬT CLIENT/SERVER

- **Thiết kế giải thuật cho Server**
  - Chương trình server có hai loại:
    - Lặp(iterative)
    - Đồng thời (concurrent).
  - Hai dạng giao thức chương trình server:
    - Connection-oriented
    - Connectionless.

THIẾT KẾ GIẢI THUẬT CLIENT/SERVER

- 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.

THIẾT KẾ GIẢI THUẬT CLIENT/SERVER

- 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.

THIẾT KẾ GIẢI THUẬT CLIENT/SERVER

- 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.

THIẾT KẾ GIẢI THUẬT CLIENT/SERVER

- 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.

THIẾT KẾ GIẢI THUẬT CLIENT/SERVER

- 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

THIẾT KẾ GIẢI THUẬT CLIENT/SERVER

- 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.

LẬP TRÌNH MẠNG TRÊN JAVA

- **Gói *java.net***
  - *InetAddress*
  - *ServerSocket*
  - *Socket*
  - *URL*
  - *URLConnection*
  - *DatagramSocket*

LẬP TRÌNH MẠNG TRÊN JAVA

- ***InetAddress* class**
  - Class mô tả về địa chỉ IP (Internet Protocol)
  - Các phương thức *getLocalHost*, *getByName*, hay *getAllByName* để tạo một *InetAddress* instance:
    - *public static InetAddess InetAddress.getByName(String hostname)*
    - *public static InetAddess [] InetAddress.getAllByName(String hostname)*
    - *public static InetAddess InetAddress.getLocalHost()*
  - Để lấy địa chỉ IP hay tên dùng các phương thức:
    - *getHostAddress()*
    - *getHostName()*

LẬP TRÌNH MẠNG TRÊN JAVA

- **In địa chỉ IP của localhost**

```
import java.net.*;
public class HostInfo {
   public static void main(String args[]) {
   HostInfo host = new HostInfo();
   host.init();
   }
   public void init() {
        try {
                InetAddress myHost = InetAddress.getLocalHost();
                System.out.println(myHost.getHostAddress());
                System.out.println(myHost.getHostName());
        } catch (UnknownHostException ex) {
                System.err.println("Cannot find local host");
        }
   }
}
```

LẬP TRÌNH MẠNG TRÊN JAVA

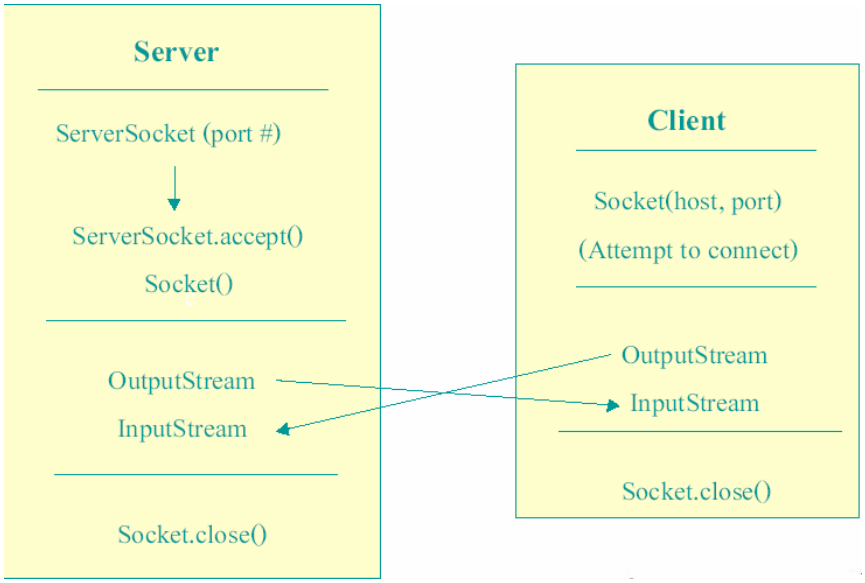
- **In địa chỉ IP của proxy.hcmut.edu.vn**

```
import java.net.*;
class kku{
    public static void main (String args[]) {
        try {
            InetAddress[] addresses =
            InetAddress.getAllByName(“proxy.hcmut.edu.vn");
            for (int i = 0; i < addresses.length; i++) {
                System.out.println(addresses[i]);
            }
        }
        catch (UnknownHostException e) {
            System.out.println("Could not find proxy.hcmut.edu.vn");
        }
    }
}
```


LẬP TRÌNH MẠNG TRÊN JAVA

- **Các chương trình đọc thêm**
  - Tạo một địa chỉ IP từ mảng byte, chuỗi String.
    - InetAddressFactory.java
  - Cho một địa chỉ tìm tên máy.
    - ReverseTest.java

LẬP TRÌNH MẠNG TRÊN JAVA



LẬP TRÌNH MẠNG TRÊN JAVA

- ***Socket* class**
  - Class mô tả về *socket*
  - Tạo một *socket*
    - **Socket**(InetAddress address, int port)
    - **Socket**(String host, int port)
    - **Socket**(InetAddress address, int port, InetAddress, localAddr, int localPort)
    - **Socket**(String host, int port, InetAddress, localAddr, int localPort)
    - **Socket**()

LẬP TRÌNH MẠNG TRÊN JAVA

- ***Socket* class (tiếp theo)**
  - **Lấy thông tin về một *socket***
    - InetAddress **getInetAddress**() : trả về địa chỉ mà socket kết nối đến.
    - int **getPort**() : trả về địa chỉ mà socket kết nối đến.
    - InetAddress **getLocalAddress**() : trả về địa chỉ cục bộ.
    - int **getLocalPort**() : trả về địa chỉ cục bộ.
  - **Sử dụng Streams**
    - public OutputStream **getOutputStream**() throws IOException
      Trả về một output stream cho việc viết các byte đến socket này.
    - public InputStream **getInputStream()** throws IOException
      Trả về một input stream cho việc đọc các byte từ socket này.

LẬP TRÌNH MẠNG TRÊN JAVA

- **Kết nối đên 1 số webserver**

```
import java.net.*;
import java.io.*;
public class getSocketInfo {
   public static void main(String[] args) {
     for (int i = 0; i < args.length; i++) {
     try {
                Socket theSocket = new Socket(args[i], 80);
                System.out.println("Connected to " +
                theSocket.getInetAddress() +
                " on port " + theSocket.getPort() + " from port " +
                theSocket.getLocalPort() + " of " +
                theSocket.getLocalAddress());
```

LẬP TRÌNH MẠNG TRÊN JAVA

- **Kết nối đến 1 số webserver (tiếp theo)**

```
    } catch (UnknownHostException e) {
                System.err.println("I can't find " + args[i]);
      } catch (SocketException e) {
                System.err.println("Could not connect to " + args[i]);
      } catch (IOException e) {
                System.err.println(e);
      }
    } // end for
  } // end main
} // end getSocketInfo
```


LẬP TRÌNH MẠNG TRÊN JAVA

- ***ServerSocket* class**
  - Class mô tả về *ServerSocket*
  - Tạo một *ServerSocket*
    - **ServerSocket**(int port) throws IOException
    - **ServerSocket**(int port, int backlog) throws IOException
    - **ServerSocket**(int port, int backlog, InetAddress bindAddr) throws IOException

LẬP TRÌNH MẠNG TRÊN JAVA

- ***ServerSocket* class**
  - Các phương thức trong *ServerSocket*
    - Socket **accept**() throws IOException : Lắng nghe một kết nối đến socket này và chấp nhận nó.
    - void **close**() throws IOException : Đóng socket.
    - InetAddress **getInetAddress**() : trả về địa chỉ cục bộ của socket
    - int **getLocalPort**() : Trả về port mà server đang lắng nghe.
    - void **setSoTimeout**(int timeout) throws SocketException
    - Enable/disable SO_TIMEOUT với khai báo timeout (milliseconds)

LẬP TRÌNH MẠNG TRÊN JAVA

- **DateTime Server**

```
import java.net.*;
import java.io.*;
import java.util.Date;
public class DayTimeServer {
    public final static int daytimePort = 5000;
    public static void main(String[] args) {
            ServerSocket theServer;
            Socket theConnection;
            PrintStream p;
            try {
                    theServer = new ServerSocket(daytimePort);
```

LẬP TRÌNH MẠNG TRÊN JAVA

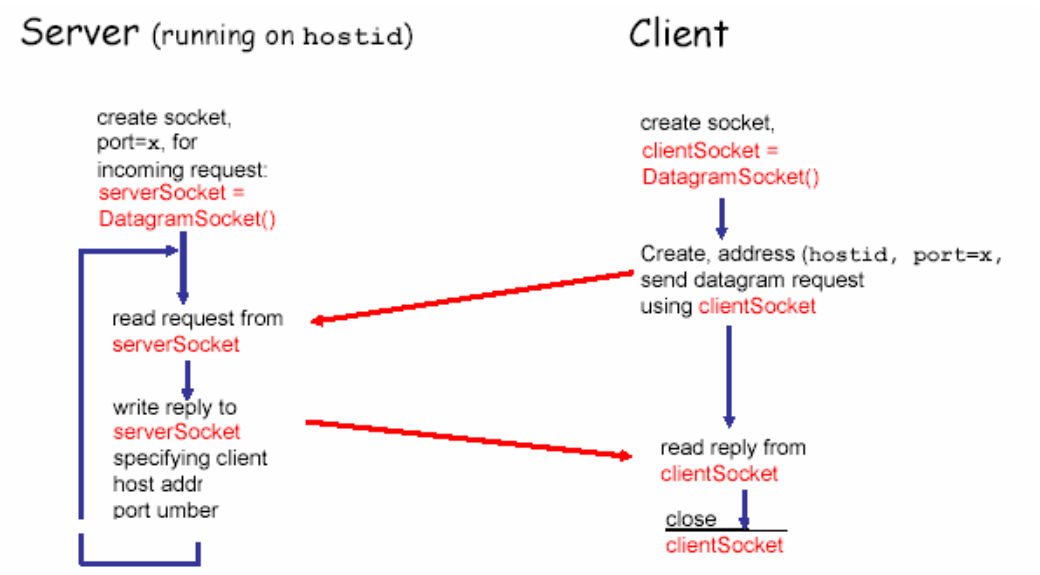
- **DateTime Server (tiếp theo)**

```
                    while (true) {
                        theConnection = theServer.accept();
                        p = new PrintStream(theConnection.getOutputStream());
                        p.println(new Date());
                        theConnection.close();
                        theServer.close();
                    }
                }catch (IOException e) {
                    System.err.println(e);
                }
            }
        }
```


LẬP TRÌNH SOCKET VỚI UDP

- Cung cấp cơ chế truyền không tin cậy giữa các nhóm các byte (datagrams) giữa client và server.
- Không cần thiết lập kết nối giữa client và 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.
- Có thể server chấp nhận nhiều client tại một thời điểm.

LẬP TRÌNH SOCKET VỚI UDP



VÍ DỤ (UDP Client)

```
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();
```

VÍ DỤ (UDP Client)

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();
}
}
```


VÍ DỤ (UDP Server)

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

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

Create datagram socket at port 9876

```
DatagramSocket serverSocket = new DatagramSocket(9876);

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

   while(true)
    {
```

Create space for received datagram

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

Receive datagram

```
serverSocket.receive(receivePacket);
```

VÍ DỤ (UDP Server)

```
String sentence = new String(receivePacket.getData());
InetAddress IPAddress = receivePacket.getAddress();
int port = receivePacket.getPort();
String capitalizedSentence = sentence.toUpperCase();
sendData = capitalizedSentence.getBytes();
DatagramPacket sendPacket =
  new DatagramPacket(sendData, sendData.length, IPAddress,
                     port);
serverSocket.send(sendPacket);
}
}
}
```

Get IP addr port #, of sender

Create datagram to send to client

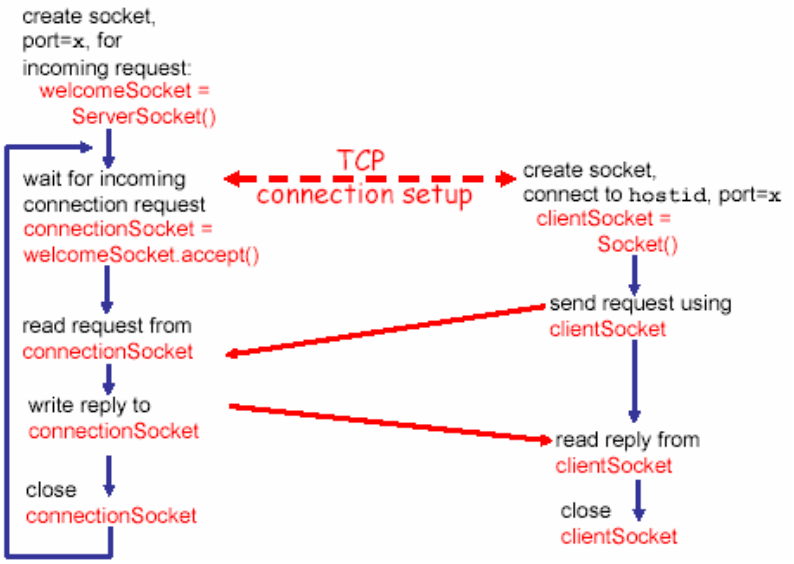
Write out datagram to socket

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

LẬP TRÌNH SOCKET VỚI TCP

- **Server**
  - Server process phải chạy trước.
  - Server phải tạo một socket để lắng nghe và chấp nhận các kết nối từ client.
- **Client**
  - Khởi tạo TCP socket.
  - Xác định IP address, port number của server.
  - 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.
  - Có thể server chấp nhận nhiều client tại một thời điểm.

LẬP TRÌNH SOCKET VỚI TCP



VÍ DỤ (TCP Client)

```
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

VÍ DỤ (TCP Client tiếp theo)

Create input stream attached to socket

Send line to server

Read line from server

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

sentence = inFromUser.readLine();

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

modifiedSentence = inFromServer.readLine();

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

clientSocket.close();

  }
}
```


VÍ DỤ (TCP Server)

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

class TCPServer {

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

     ServerSocket welcomeSocket = new ServerSocket(6789);

     while(true) {

       Socket connectionSocket = welcomeSocket.accept();

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

Create welcoming socket at port 6789

Wait, on welcoming socket for contact by client

Create input stream, attached to socket

VÍ DỤ (TCP Server – tiếp theo)

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

BÀI TẬP

- **Viết chương trình trên Java/C tương tự như nslookup:**
  - Cho 1 tên tìm ra địa chỉ IP.
  - Cho 1 địa chỉ IP tìm ra tên.
  - Giao diện tương tự như sau:

BÀI TẬP

- Viết chương trình *echo client* trên Java.
  - echo : 7/tcp, 7/udp.
- Viết chương trình *finger client* trên Java.
  - Finger: 79/tcp.
- Viết chương trình *echo server* trên Java.

LẬP TRÌNH SOCKET TRÊN UNIX

Primitives	Meaning
SOCKET	Create a new communication end point
BIND	Attach a local address to a socket
LISTEN	Announce willingness to accept connections; give queue size
ACCEPT	Block the caller until connection attempt arrives
CONNECT	Actively attempt to establish a connection
SEND	Send some data over the connection
RECEIVE	Receive some data from the connection
CLOSE	Release the connection

LẬP TRÌNH SOCKET TRÊN UNIX

- **Một so sánh tương tự**
  - socket: Điện thoại
  - bind: Gán một số cho điện thoại
  - listen: Bật chế độ chuông
  - connect: quay số
  - accept: trả lời điện thoại
  - read/write: nói chuyện
  - close: kết thúc

LẬP TRÌNH SOCKET TRÊN UNIX

- **Để gởi**
  - socket, connect, write
- **Để nhận**
  - socket, bind, listen, accept, read
- **Endpoint**
  - Địa chỉ IP + chỉ số port

LẬP TRÌNH SOCKET TRÊN UNIX

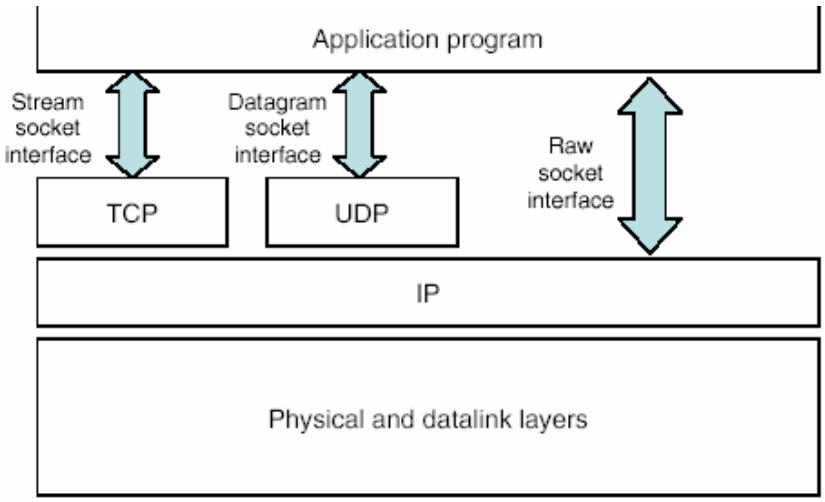
- **Hàm *socket***

int socket(int domain, int type, int protocol);

Trong đó:

- domain : họ địa chỉ, thường sử dụng là AF_INET
- type : Kiểu socket (SOCK_STREAM, SOCK_DGRAM, …)
- protocol : giao thức được dùng, default = 0

LẬP TRÌNH SOCKET TRÊN UNIX



Hình - Các loại socket

LẬP TRÌNH SOCKET TRÊN UNIX

- **Ví dụ (ex1.c)**

```
#include <sys/types.h>
#include <sys/socket.h>
int main(void)
{
  int sockfd;
  if ((sockfd = socket(AF_INET, SOCK_STREAM, 0)) ==
  -1) {
        perror("socket");
        exit(1);
  }
     printf("Sockfd : %d \n", sockfd);
}
```


LẬP TRÌNH SOCKET TRÊN UNIX

- **Hàm *bind***

int bind(int sockfd, struct sockaddr *my_addr, int addrlen);

Trong đó:

- sockfd: *socket file descriptor* trả về từ hàm socket
- my_addr: Một con trỏ đến một cấu trúc sockaddr ???
- *addrlen* = sizeof(struct sockaddr).

LẬP TRÌNH SOCKET TRÊN UNIX

```
struct sockaddr_in {
   short sin_family;            // e.g. AF_INET
   unsigned short sin_port;     // e.g. htons(3490)
   struct in_addr sin_addr;     // see struct in_addr,
   below
   char sin_zero[8];            // zero this if you want to
};

struct in_addr {
   unsigned long s_addr;        // load with inet_aton()
};
```


LẬP TRÌNH SOCKET TRÊN UNIX

- **Ví dụ 2 (ex2.c)**

```
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
int main(void)
{
   struct sockaddr_in myaddr;
   int sockfd;

   myaddr.sin_family = AF_INET;
   myaddr.sin_port = htons(3490);

   // you can specify an IP
   address:
   //inet_aton("63.161.169.137",
   &myaddr.sin_addr.s_addr);

   // or you can let it automatically
   select one:
   myaddr.sin_addr.s_addr = INADDR_ANY;

   if ((sockfd =
   socket(AF_INET,SOCK_STREAM,0)) == -1)
   {
        perror("socket");
        exit(1);
   }
   if(bind(sockfd, (struct sockaddr
   *)&myaddr, sizeof myaddr)== -1) {
        perror("bind");
        exit(1);
   }
   printf("socket, bind \n");
}
```

LẬP TRÌNH SOCKET TRÊN UNIX

- **Hàm *listen***

int listen(int sockfd, int backlog);

Trong đó:

- *backlog:* Số kết nối cho phép của hàng đợi. Các yêu cầu kết nối của đối tác sẽ được lưu trong queue cho tới khi được accept.

- **Hàm *accept***

int accept(int sockfd, void *addr, int *addrlen);

Trong đó:

- *addr:* con trỏ trỏ tới sockaddr_in (Xác định từ đâu kết nối tới ?)
- *addrlen:* biến int = sizeof(struct sockaddr_in)

LẬP TRÌNH SOCKET TRÊN UNIX

- **Hàm *connect***

int connect(int sockfd, struct sockaddr *serv_addr, int addrlen);

Trong đó:

- *serv_addr*: struct sockaddr chứa port & IP address đích
- *addrlen* = sizeof(struct sockaddr)

- **Các hàm *gởi nhận***

int send(int sockfd, const void *msg, int len, int flags);

int recv(int sockfd, void *buf, int len, unsigned int flags);

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

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

LẬP TRÌNH SOCKET TRÊN UNIX

- **Các hàm gởi nhận (tiếp theo)**

int sendto(int sockfd, const void *msg, int len, unsigned int flags, const struct sockaddr *to, int tolen);

tolen có giá trị bằng sizeof(struct sockaddr).

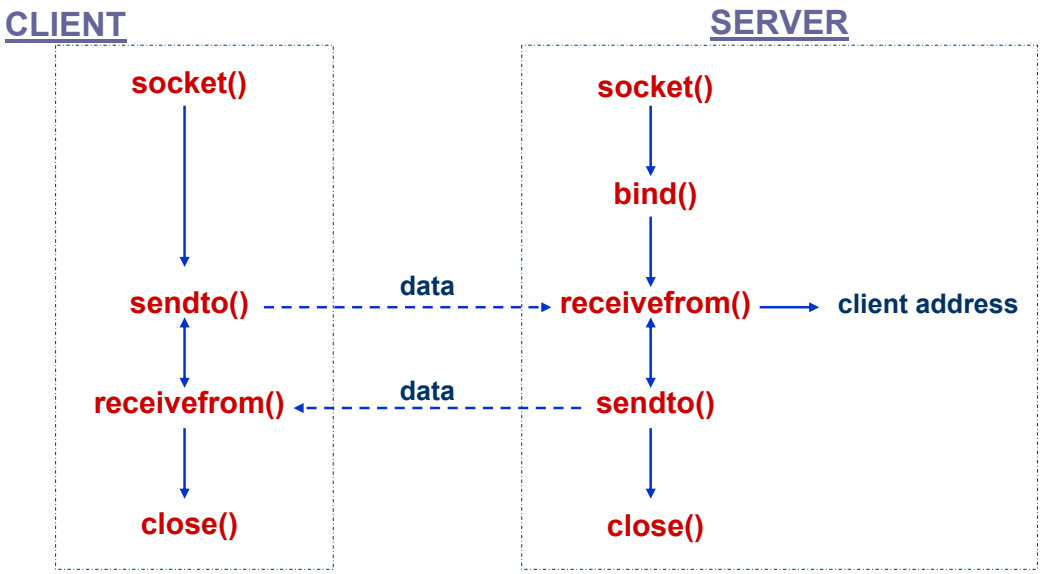
int recvfrom(int sockfd, void *buf, int len, unsigned int flags, struct sockaddr *from, int *fromlen);

fromlen khởi tạo bằng sizeof(struct sockaddr).

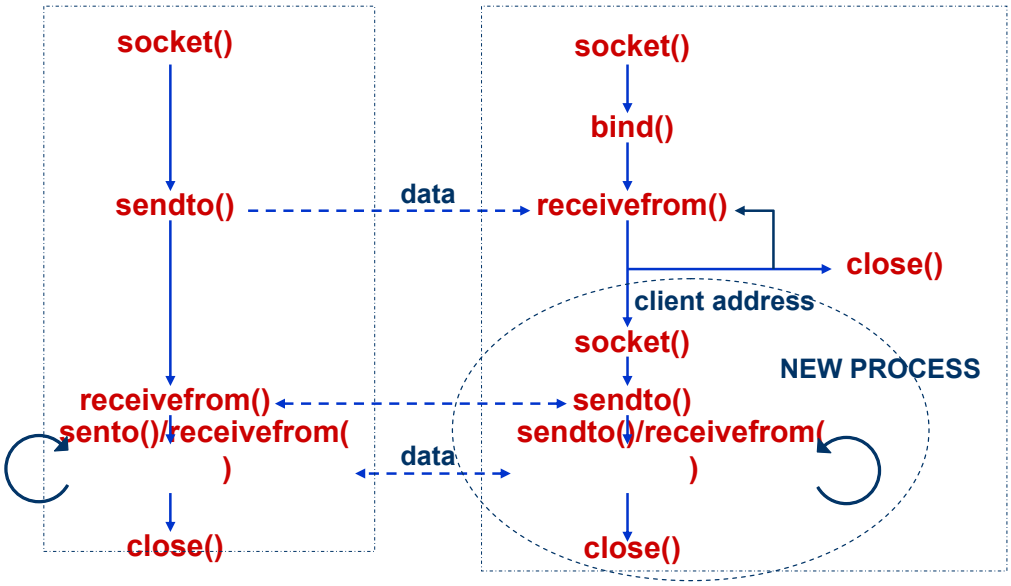
- **Hàm *close***

int close(int sockfd);

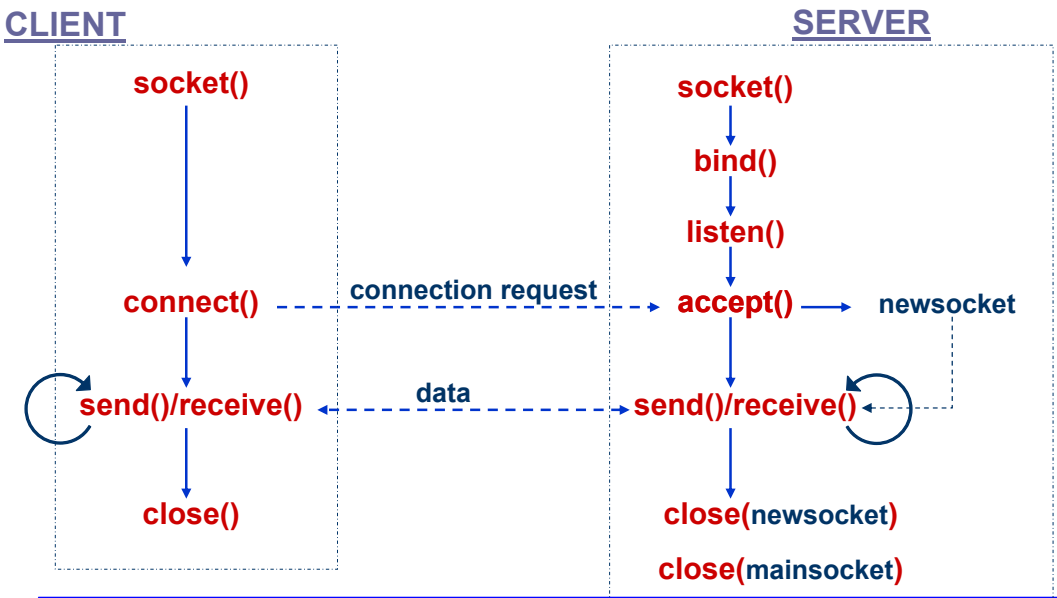
LẬP TRÌNH SOCKET VỚI UDP



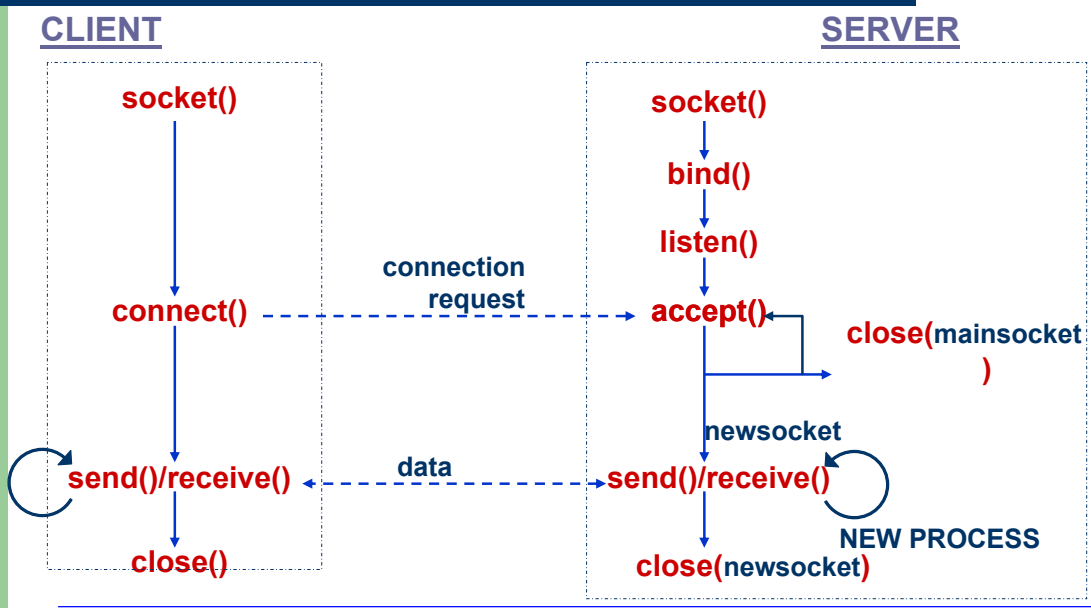
LẬP TRÌNH SOCKET VỚI UDP



LẬP TRÌNH SOCKET VỚI TCP



LẬP TRÌNH SOCKET VỚI TCP



LẬP TRÌNH SOCKET TRÊN UNIX

- **Một số hàm liên quan tên máy và địa chỉ**

int gethostname(char **name*, size_t *len*);

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

struct hostent *gethostbyaddr(const char **addr*, int *len*, int *type*);

char *inet_ntoa(struct in_addr *in*);

int inet_aton(const char **cp*, struct in_addr **inp*);

in_addr_t inet_addr(const char **cp*);

LẬP TRÌNH SOCKET TRÊN UNIX

- **Ví dụ về các hàm *inet_**

```
struct sockaddr_in antelope;
char *some_addr;
inet_aton("10.0.0.1", &antelope.sin_addr);

// store IP in antelope
   some_addr = inet_ntoa(antelope.sin_addr);
   printf("%s\n", some_addr); // prints "10.0.0.1"

// and this call is the same as the inet_aton() call, above:
antelope.sin_addr.s_addr = inet_addr("10.0.0.1");
```


LẬP TRÌNH SOCKET TRÊN UNIX

- **Ví dụ về hàm *inet_ntoa* và *gethostbyname***

```
struct hostent *he;
// get the addresses of www.yahoo.com:
he = gethostbyname("www.yahoo.com");

// print information about this host:
printf("Official name is: %s\n", he->h_name);
printf("IP address: %s\n", inet_ntoa(*(struct
  in_addr*)he->h_addr));
printf("All addresses: ");
addr_list = (struct in_addr **)he->h_addr_list;
for(i = 0; addr_list[i] != NULL; i++) {
  printf("%s ", inet_ntoa(*addr_list[i]));
}
```

LẬP TRÌNH SOCKET TRÊN UNIX

- **Ví dụ về hàm *inet_aton* và *gethostbyaddr***

```
// get the host name of 66.94.230.32:
inet_aton("66.94.230.32", &addr);
he = gethostbyaddr(&addr, sizeof addr, AF_INET);
printf("Host name: %s\n", he->h_name);
```


LẬP TRÌNH SOCKET TRÊN UNIX

- **Một số hàm chuyển đổi số**

  uint32_t htonl(uint32_t *hostlong*);

  uint16_t htons(uint16_t *hostshort*);

  uint32_t ntohl(uint32_t *netlong*);

  uint16_t ntohs(uint16_t *netshort*);

LẬP TRÌNH SOCKET TRÊN UNIX

- **Ví dụ về các hàm chuyển đổi số**

```
uint32_t some_long = 10;
uint16_t some_short = 20;
uint32_t network_byte_order; // convert and send
network_byte_order = htonl(some_long);
send(s, &network_byte_order, sizeof(uint32_t), 0);

some_short == ntohs(htons(some_short)); // this
expression is true
```


LẬP TRÌNH SOCKET TRÊN UNIX

– 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)
```

LẬP TRÌNH SOCKET VỚI TCP

- **DateTime Server**

```
#include <sys/types.h>
#include <sys/socket.h>
int main (int argc, char **argv) {
    int   listenfd, connfd;
    struct sockaddr_in servaddr, cliaddr;
    char buff[MAXLINE];
    time_t ticks;
    /* Create a TCP socket */
    listenfd = socket (AF_INET, SOCK_STREAM, 0);
    /* Initialize server's address and well-known port */
    bzero (&servaddr, sizeof(servaddr));
    servaddr.sin_family      = AF_INET;
    servaddr.sin_addr.s_addr = htonl (INADDR_ANY);
    servaddr.sin_port        = htons (13);
```


LẬP TRÌNH SOCKET VỚI TCP

- **DateTime Server (tiếp theo)**

```
/* Bind server's address and port to the socket */
bind (listenfd, (struct sockaddr*) &servaddr, sizeof( servaddr));
/* Convert socket to a listening socket */
listen (listenfd, 100);
for ( ; ; ) {
/* Wait for client connections and accept them */
    clilen = sizeof(cliaddr);
    connfd = accept( listenfd, (struct sockaddr *)&cliaddr,
    &clilen);
    ticks = time(NULL);
     snprintf( buff, sizeof(buff),"%.24s\r\n",ctime(&ticks));
    /* Write to socket */
     write( connfd, buff, strlen(buff) );
    /* Close the connection */
     close( connfd );
  }
}
```

LẬP TRÌNH SOCKET VỚI TCP

- **DateTime Client**

```
#include <sys/types.h>
#include <sys/socket.h>
int main(int argc, char **argv) {
        int  sockfd, n;
        char recvline[MAXLINE + 1];
        struct sockaddr_in servaddr;
    if( argc != 2 )
      printf("Usage : gettime <IP address>"); exit(1);
    /* Create a TCP socket */
    if ( (sockfd = socket (AF_INET, SOCK_STREAM, 0)) < 0)
    {
      perror("socket");
      exit(2);
     }
```


LẬP TRÌNH SOCKET VỚI TCP

- **DateTime Client (tiếp theo)**

```
/* Specify server's IP address and port */
   bzero (&servaddr, sizeof(servaddr));
   servaddr.sin_family = AF_INET;
   servaddr.sin_port = htons ( 13 );
   if (inet_pton (AF_INET, "127.0.0.1",
   &servaddr.sin_addr) <= 0) {
       perror("inet_pton"); exit(3);
   }

/* Connect to the server */
   if ( connect(  sockfd, (struct sockaddr *) &servaddr,
   sizeof(servaddr)) < 0 ) {
       perror("connect"); exit(4);
   }
```

LẬP TRÌNH SOCKET VỚI TCP

- **DateTime Client (tiếp theo)**

```
/* Read the date/time from socket */
   while ( (n = read ( sockfd, recvline, MAXLINE)) > 0) {
      recvline[n] = '\0';        /* null terminate */
       printf("%s", recvline);
   }
   if (n < 0) {
      perror("read"); exit(5);
   }
   close ( sockfd );
}
```


BÀI TẬP

- Viết chương trình *nslookup* bằng C trên Unix/Linux
- Viết *echo Client/Server* bằng C trên Unix/Linux
- Viết một Web Server có những đặc điểm sau:
  - Hỗ trợ phương thức GET (GET xxx.html HTTP/1.0)
  - HTTP
  - Đáp ứng của Server có header như ExServer/b1.0
  - Ví dụ
    - ***Browser Request***:
      GET /intro.html HTTP/1.0 *WebServer Response*
    - ***Server Reponse***
      case 1: HTTP/1.0 200 OK
      case 2: HTTP/1.0 404 File Not Found
      case 3: HTTP/1.0 501 Not Implemented

TỔNG KẾT

- Khái niệm socket
- Thiết kế giải thuật cho client và server
- Lập trình mạng trên Java
- Lập trình socket trên UNIX