LẬP TRÌNH SOCKET

Khoa Mạng máy tính & Truyền thông

- Đại học Công nghệ Thông tin -

Nội dung

- Giới thiệu lập trình socket
- TCP/IP
- Socket và TCP/IP
- Lập trình Winsock
- Cấu trúc chương trình ứng dụng
- Sử dụng IDE: Visual C++
- Tổng kết

Ký hiệu viết tắt

- IPC: InterProcess Communication
- BSD: Berkeley Software Distribution
- TCP: Transmission Control Protocol
- UDP: User Datagram Protocol
- IP: Internet Protocol

Giới thiệu về Sockets

- Socket là một trong những kỹ thuật cơ bản nhất trong truyền thông trên mạng máy tính
- Socket là một phương thức thực hiện truyền thông giữa các tiến trình được BSD đề xuất.
- Socket được sử dụng để một tiến trình “nói chuyện” với một tiến trình khác.
- Nhiều ứng dụng thông dụng hiện nay sử dụng kỹ thuật socket như: trình duyệt web, email client, …

Socket API

- Giao diện lập trình socket trên Windows, winsock, là một đặc tả của nhiều hãng phần mềm nhằm chuẩn hóa cách thức sử dụng TCP/IP trên Windows. Socket API dựa trên nền Berkeley socket.
- Trong BSD Unix, socket là một phần trong kernel của hệ điều hành. Nó cung cấp các dịch vụ IPC cục bộ hoặc giữa các tiến trình trên mạng
- Trong MS-DOS, Windows, MacOS và OS/2, sockets được cung cấp dưới dạng các thư viện lập trình.

Các loại socket

- Stream socket: TCP socket
- Datagram socket: UDP socket
- Raw socket: IP socket

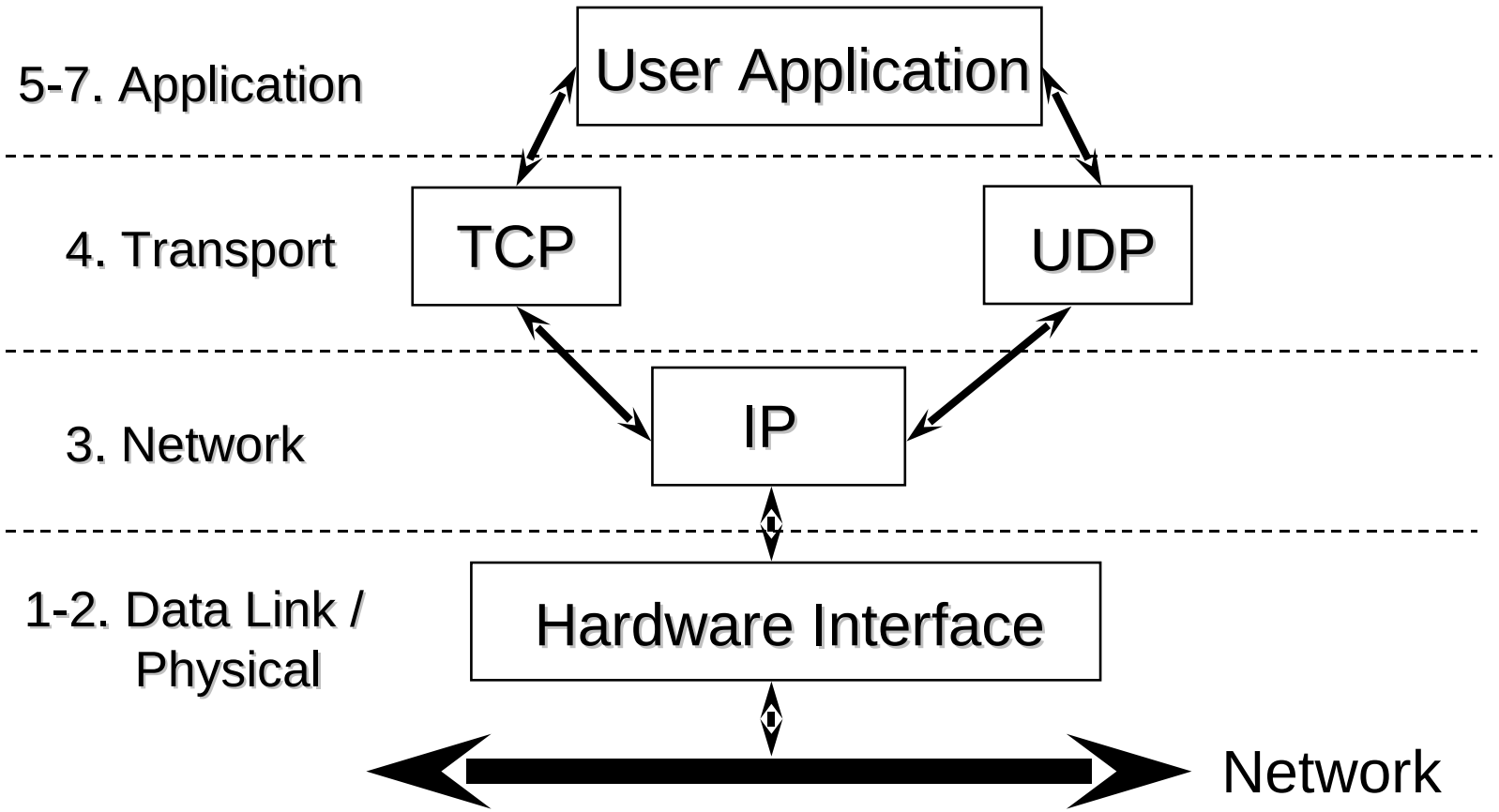
TCP/IP (1)

- TCP/IP là một bộ giao thức, được xây dựng dựa trên kỹ thuật “phi kết nối”. Dữ liệu được truyền theo từng dãy các gói tin đơn lẻ.
- TCP được sử dụng cho các dịch vụ với khả năng truyền dữ liệu lớn và một kết nối liên tục
- UDP thường được sử dụng cho các thao tác tìm kiếm nhanh hay các truy vấn đơn lẻ

TCP/IP (2)

- Thông thường, các ứng dụng TCP/IP sử dụng 4 lớp:
  - Một giao thức ứng dụng, chẳng hạn như email, ftp hay www
  - Một giao thức cung cấp các dịch vụ cần thiết cho các ứng dụng, chẳng hạn như TCP/UDP
  - IP cung cấp dịch vụ cơ bản chuyển giao các gói tin đến đúng địa chỉ đích
  - Các giao thức cần thiết để quản lý phương tiện truyền dẫn vật lý, chẳng hạn như Ethernet hay một đường nối điểm – điểm

So sánh OSI với TCP/IP



TCP

- Hướng kết nối
- Đảm bảo độ tin cậy trong quá trình truyền dữ liệu
- Phân đoạn dữ liệu truyền thành các gói tin đơn lẻ
- Dữ liệu truyền được đánh số thứ tự
- Cung cấp cơ chế phản hồi sau khi nhận được dữ liệu
- Sử dụng cơ chế phát hiện lỗi checksums

UDP

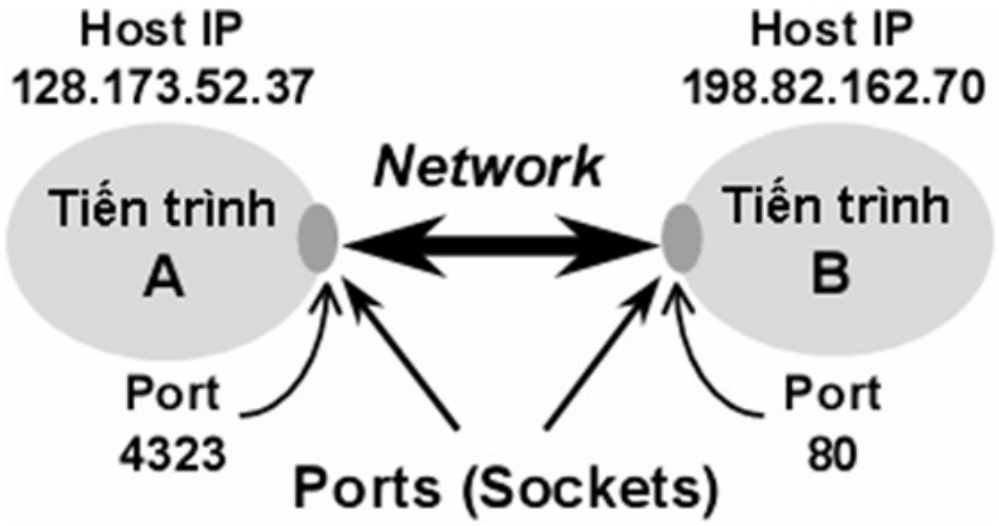
- Phi kết nối
- Không phân đoạn dữ liệu truyền
- Không cung cấp cơ chế tập hợp dữ liệu nhận cũng như đồng bộ quá trình truyền nhận dữ liệu
- Nếu có lỗi xảy ra, ứng dụng bắt buộc phải thực hiện truyền lại dữ liệu
- Không có cơ chế phản hồi sau khi nhận dữ liệu

Các ví dụ về TCP / UDP

Dịch vụ	Giao thức	Cổng dịch vụ
DNS lookup	UDP	53
FTP	TCP	21
HTTP	TCP	80
POP3	TCP	110
Windows shared printer name lookup	UDP	137
Telnet	TCP	23

Sockets

- Khi được tạo ra, một socket không có những thông tin chỉ định cách thức hoạt động
- Bộ giao thức TCP/IP sẽ định nghĩa một điểm kết nối trên socket, gồm có một địa chỉ IP và một số hiệu cổng dịch vụ



Địa chỉ socket

- Địa chỉ của một socket trên mạng TCP/IP gồm có hai phần:
  - Địa chỉ IP: một số nguyên 32 bits xác định duy nhất một card mạng trên máy tính (host)
  - Cổng dịch vụ: một số nguyên 16 bits xác định điểm kết nối với một ứng dụng trên một host. Các ứng dụng thương mại hay các dịch vụ thông dụng sử dụng các cổng dịch vụ chuẩn đã được đăng ký.

Passive/Active Socket

- Passive socket được sử dụng ở chương trình server để chờ nhận các kết nối đến từ client
- Active socket được sử dụng ở chương trình client để thiết lập kết nối đến chương trình server

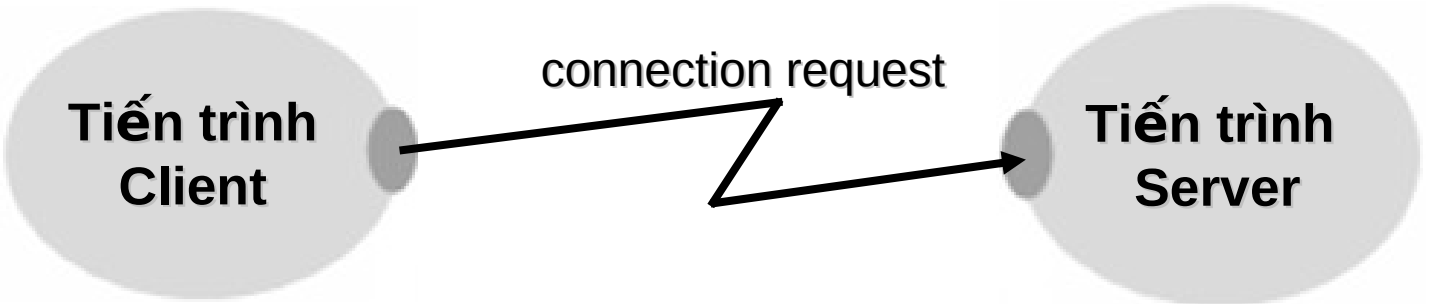
Giao thức hướng kết nối (1)

- Các giao thức dựa trên phiên làm việc hay sự chuyển giao các gói tin có thứ tự
- Cung cấp dịch vụ kết nối hai chiều tin cậy dựa trên một phiên làm việc
- Các gói tin được đánh số thứ tự duy nhất
- Từng gói tin chuyển giao được xác nhận truyền/nhận thành công
- Các gói tin nhận trùng lắp được phát hiện và loại bỏ

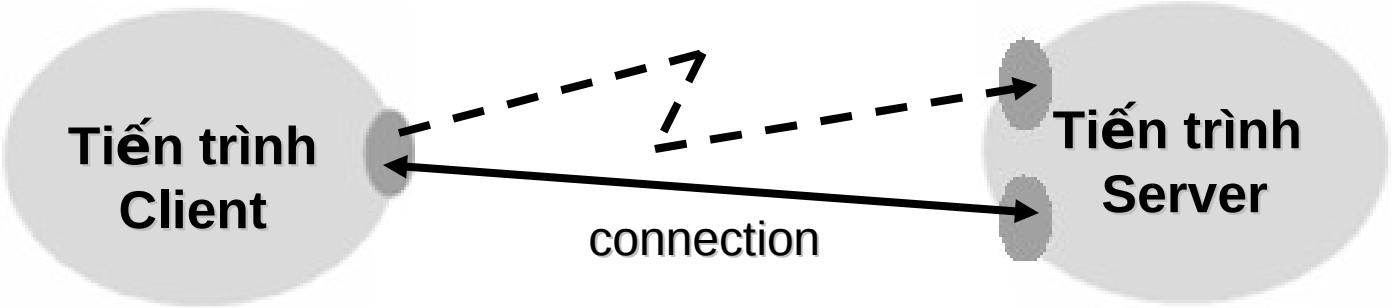
Giao thức hướng kết nối (2)

- Các giao thức hướng kết nối hoạt động theo ba giai đoạn
  - Thiết lập kết nối: hai tiến trình truyền/nhận thiết lập kết nối và thống nhất các tham số định nghĩa kết nối
  - Truyền nhận dữ liệu: hai tiến trình truyền nhận dữ liệu dựa trên kết nối đã được thiết lập
  - Giải phóng kết nối: kết nối giữa hai tiến trình được giải phóng

TCP Connection



Thiết lập kết nối



Truyền nhận dữ liệu

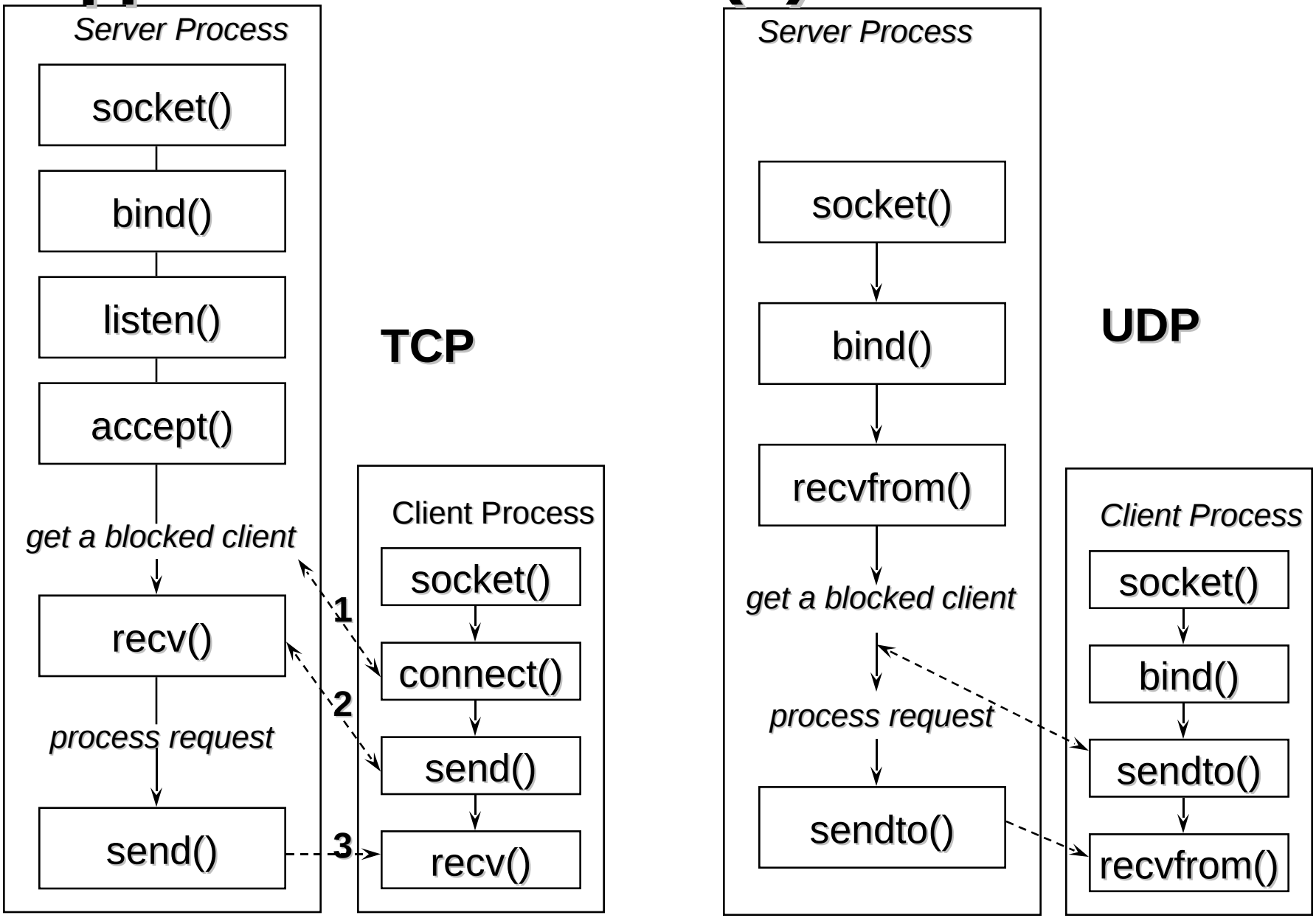
Giao thức phi kết nối (1)

- Đơn giản, nhưng không tin cậy. Không cung cấp cơ chế điều khiển đường truyền dựa trên việc đánh số thứ tự dữ liệu truyền hoặc cơ chế xác nhận
- Cung cấp tính năng broadcast thông tin
- Dữ liệu được truyền/nhận theo từng gói tin đơn lẻ: datagram hay packet.
- Một datagram là một thông điệp độc lập được gửi qua mạng -- không được đảm bảo đến đích, thời điểm đến đích và nội dung thông điệp

Giao thức phi kết nối (2)

- Thích hợp cho các ứng dụng broadcast
- Được sử dụng trong các môi trường không xác định được host nhận dữ liệu
- Khả năng truyền thông điệp nhanh -- sử dụng trong các ứng dụng không quan tâm đến việc dữ liệu được truyền đến đích đúng và đủ

Lập trình Winsock (1)



Lập trình Winsock (2)

- **Thư viện**
  - winsock2.h
- **Kiểu dữ liệu**
  - WSADATA
  - SOCKADDR_IN
  - LPSOCKADDR
- **Các giá trị hằng số**
  - AF_INET
  - SOCK_STREAM / SOCK_DGRAM
  - INVALID_SOCKET
  - SOCKET_ERROR

TCP Client

- Xác định địa chỉ IP và cổng dịch vụ ở server
- Tạo một socket với cổng dịch vụ cục bộ bất kỳ do TCP chỉ định
- Kết nối socket đến server
- Gửi yêu cầu và nhận thông tin phản hồi từ server
- Đóng kết nối

Lập trình: TCPClient (1)

```
void main()
{
    /* Khai báo biến */
    WSADATA wsaData;
    SOCKADDR_IN rAddr;
    char buffer[20];
    int iRc;

    /* Nạp thư viện và khởi tạo socket */
    iRc = WSAStartup(MAKEWORD(2, 2), &wsaData);
    SOCKET sk = socket(AF_INET, SOCK_STREAM, 0);
```

Lập trình: TCPClient (2)

```
    /* Kết nối đến Server */
    rAddr.sin_family = AF_INET;
    rAddr.sin_addr.s_addr = inet_addr("127.0.0.1");
    rAddr.sin_port = htons(1024);
    iRc = connect(sk, (LPSOCKADDR)&rAddr, sizeof(rAddr));

    /* Nhận và gửi thông điệp */
    printf("Message: ");
    gets(buffer);
    iRc = send(sk, buffer, strlen(buffer), 0);

    /* Đóng socket và giải phóng tài nguyên */
    closesocket(sk);
    WSACleanup();
} // end of main function
```

TCP Server

- Tạo một socket và gán cho socket một địa chỉ cố định (IP và port) để cung cấp dịch vụ
- Đặt socket ở chế độ passive
- Chấp nhận yêu cầu thiết lập kết nối từ client và nhận socket mới từ hệ thống
- Thực hiện quá trình lặp nhận yêu cầu và gửi thông tin phản hồi
- Khi client kết thúc, đóng kết nối và trở về trạng thái chờ chấp nhận yêu cầu thiết lập kết nối từ các client

Lập trình: TCPServer (1)

```
void main()
{
    /* Khai báo biến */
    WSADATA wsaData;
    SOCKADDR_IN serAddr, cliAddr;
    int iRc;
    int adsize;
    char buffer[256];

    /* Nạp thư viện và khởi tạo socket */
    iRc = WSAStartup(MAKEWORD(2, 2), &wsaData);
    SOCKET ssk = socket(AF_INET, SOCK_STREAM, 0);
```

Lập trình: TCPServer (2)

```
/* Thiết lập cấu trúc địa chỉ cho socket */
serAddr.sin_family  = AF_INET;
serAddr.sin_addr.s_addr = inet_addr("127.0.0.1");
serAddr.sin_port = htons(1024);

/* Gắn địa chỉ cho socket và thực hiện chờ thiết lập kết nối */
iRc = bind(ssk, (LPSOCKADDR)&serAddr, sizeof(serAddr));
iRc = listen(ssk, 5);
adsize=sizeof(cliAddr);
SOCKET sk = accept(ssk, (LPSOCKADDR)&cliAddr, &adsize);
```

Lập trình: TCPServer (3)

```
    /* Nhận dữ liệu từ Client */
    iRc = recv(sk, buffer, 256, 0);
    if (iRc == SOCKET_ERROR)
        printf("Could not receive data from server.\n");
    else {
        buffer[iRc]=0;
        printf("Message from client: %s\n", buffer);
    }

    /* Đóng socket và giải phóng tài nguyên */
    closesocket(sk);
    closesocket(ssk);
    WSACleanup();
} // end of main function
```

UDP Client

- Xác định địa chỉ IP và cổng dịch vụ ở server
- Tạo một socket với cổng dịch vụ cục bộ bất kỳ do UDP chỉ định
- Chỉ định server cần giao tiếp
- Gửi yêu cầu và nhận thông tin phản hồi từ server
- Đóng socket

Lập trình: UDPClient (1)

```
void main()
{
    /* Khai báo biến */
    WSADATA wsaData;
    SOCKET sk;
    WORD _toPort;
    DWORD _toIP;
    SOCKADDR_IN _toAddr;
    int iRc, adsize;
    char msg[512];

    /* Nạp thư viện và khởi tạo socket */
    iRc = WSAStartup(MAKEWORD(2, 2), &wsaData);
    sk = socket(AF_INET, SOCK_DGRAM, 0);
```

Lập trình: UDPClient (2)

```
/* Nhận thông điệp cần gửi */
printf("Message: ");
scanf("%s", &msg);

/* Gửi dữ liệu đến server */
_toAddr.sin_family = AF_INET;
_toAddr.sin_port = htons(_toPort);
_toAddr.sin_addr.s_addr = inet_addr(_toIP);
iRc = sendto(sk, msg, strlen(msg), 0,
                    (LPSOCKADDR)&_toAddr, sizeof(_toAddr));
```

Lập trình: UDPClient (3)

```
    /* Đóng socket và giải phóng tài nguyên */
    closesocket(sk);
    WSACleanup();
} // end of main function
```

UDP Server

- Tạo một socket và gán cho socket một địa chỉ cố định (IP và port) để cung cấp dịch vụ
- Thực hiện quá trình lặp nhận yêu cầu và gửi thông tin phản hồi

Lập trình: UDPServer (1)

```
void main()
{
    /* Khai báo biến */
    WSADATA wsaData;
    SOCKET sk;
    SOCKADDR_IN _locAddr, _frAddr;
    int iRc, adsize;
    char msg[512];

    /* Nạp thư viện và khởi tạo socket */
    iRc = WSAStartup(MAKEWORD(2, 2), &wsaData);
    sk = socket(AF_INET, SOCK_DGRAM, 0);
```

Lập trình: UDPServer (2)

```
/* Gắn địa chỉ cho socket */
_locAddr.sin_family = AF_INET;
_locAddr.sin_port = htons(1024);
_locAddr.sin_addr.s_addr = htonl(INADDR_ANY);
iRc = bind(sk, (LPSOCKADDR)&_locAddr, sizeof(_locAddr));

/* Nhận dữ liệu từ Client */
adsize = sizeof(_frAddr);
iRc = recvfrom(sk, msg, 512, 0,
                        (LPSOCKADDR)&_frAddr, &adsize);
```

Lập trình: UDPServer (3)

```
    /* Hiển thị dữ liệu lên màn hình */
    if (iRc == SOCKET_ERROR)
            printf("Could not receive data from server.\n");
    else {
            msg[iRc] = 0;
            printf("\nMessage from sender: %s\n", msg);
    }

    /* Đóng socket và giải phóng tài nguyên */
    closesocket(sk);
    WSACleanup();
}
```

Visual C++ với Winsock (1)

Include **Winsock** header:

- Khởi động VC++
- Chọn New > Windows Console Application
- Chọn loại ứng dụng “Hello World”
- Thiết lập vùng cửa sổ trái sang chế độ File View
- Trong mục “Header Files”, mở file **`StdAfx.h`**
- Thêm dòng **`#include <winsock2.h>`**
- Lưu file **`StdAfx.h`**
- Thêm dòng **`#include “StdAfx.h”`** trong tập tin chương trình

Visual C++ với Winsock (2)

Liên kết thư viện Winsock:

- Chọn Project > Settings trên menu
- Chọn tab Link
- Thêm **`wsock32.lib`** vào ô "Object/Library modules"

- Để xác định lỗi trong chương trình, sử dụng hàm WSAGetLastError()
  - Ví dụ: với lỗi 10037, sử dụng lệnh:
    - **`net helpmsg 10037`**

Tổng kết

- Socket và bộ giao thức TCP/IP
- Lập trình Winsock: cấu trúc chương trình, thư viện sử dụng, IDE
- Viết các chương trình ứng dụng sử dụng socket, với hai giao thức TCP và UDP