LẬP TRÌNH MẠNG

1 Lập trình WinSock

- Khởi tạo Socket
- Lập trình Socket cho ứng dụng TCP cơ bản
- Lập trình Socket cho ứng dụng UDP cơ bản
- Thiết kế giao thức ứng dụng

KHỞI TẠO SOCKET

- **SOCKET:**
  - Hỗ trợ cung cấp dịch vụ lớp giao vận
  - Các hàm API dựa trên khái niệm Socket
  - SOCKET: một kiểu dữ liệu độc lập

- **Hai hàm khởi tạo Socket**
  - Socket
  - WSASocket

KHỞI TẠO SOCKET ĐƠN GIẢN

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

- **Tham số:**
  - af: Họ địa chỉ giao thức được sử dụng;
    - Ví dụ: AF_INET (IPv4)
  - type: Kiểu socket của giao thức,
    - SOCK_STREAM: cho giao thức TCP/IP
    - SOCK_DGRAM: cho giao thức UDP/IP
  - protocol: kiểm tra đủ điều kiện giao vận
    - IPPROTO_TCP: dùng cho TCP
    - IPPROTO_UDP: dùng cho UDP

KHỞI TẠO SOCKET ĐƠN GIẢN

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

- **Kết quả trả về:**
  - Giá trị nguyên > 0: khởi tạo thành công
  - INVALID_SOCKET: khởi tạo thất bại

TCP SOCKET

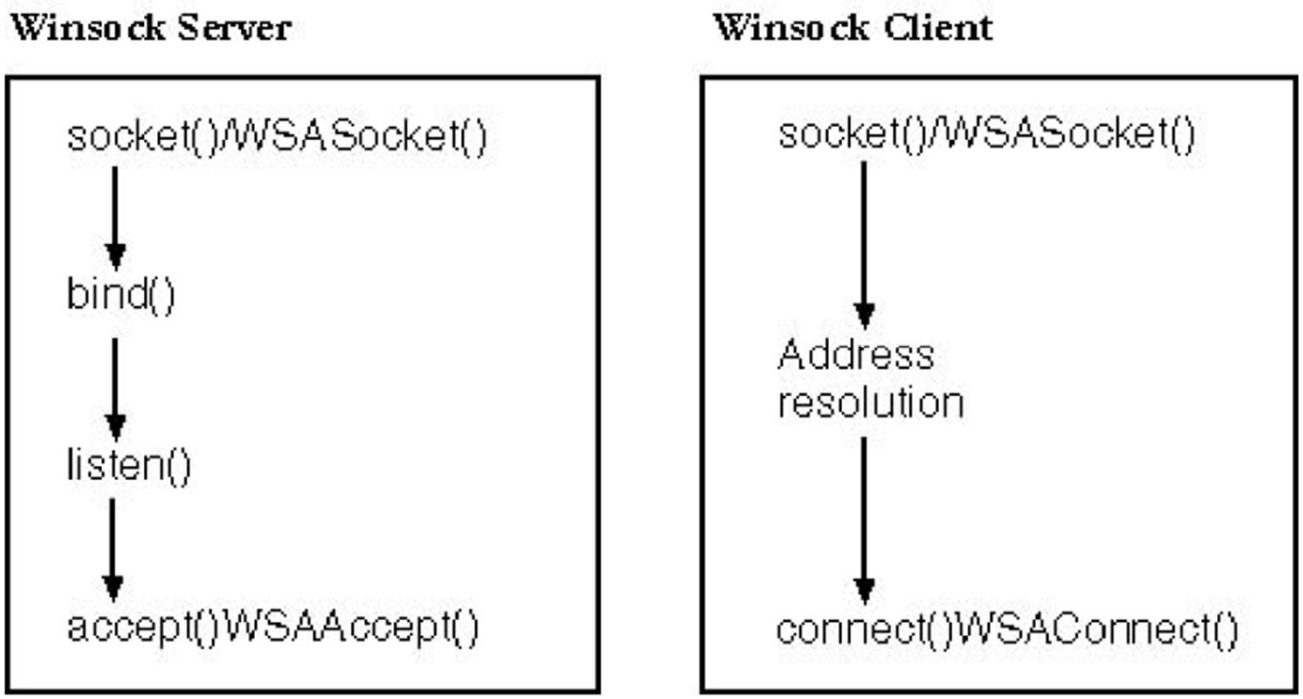
TCP SOCKET

- **Chức năng cần thiết của Winsock cho TCP/IP:**
  - Nhận kết nối
  - Thiết lập kết nối
  - Truyền dữ liệu trong phiên hướng kết nối
- **Phát triển Server:**
  - Nhận kết nối từ client
  - Thực hiện tiến trình chấp nhận/từ chối (accept/reject) kết nối
- **Phát triển Client:**
  - Khởi tạo kết nối đến server

TCP SOCKET

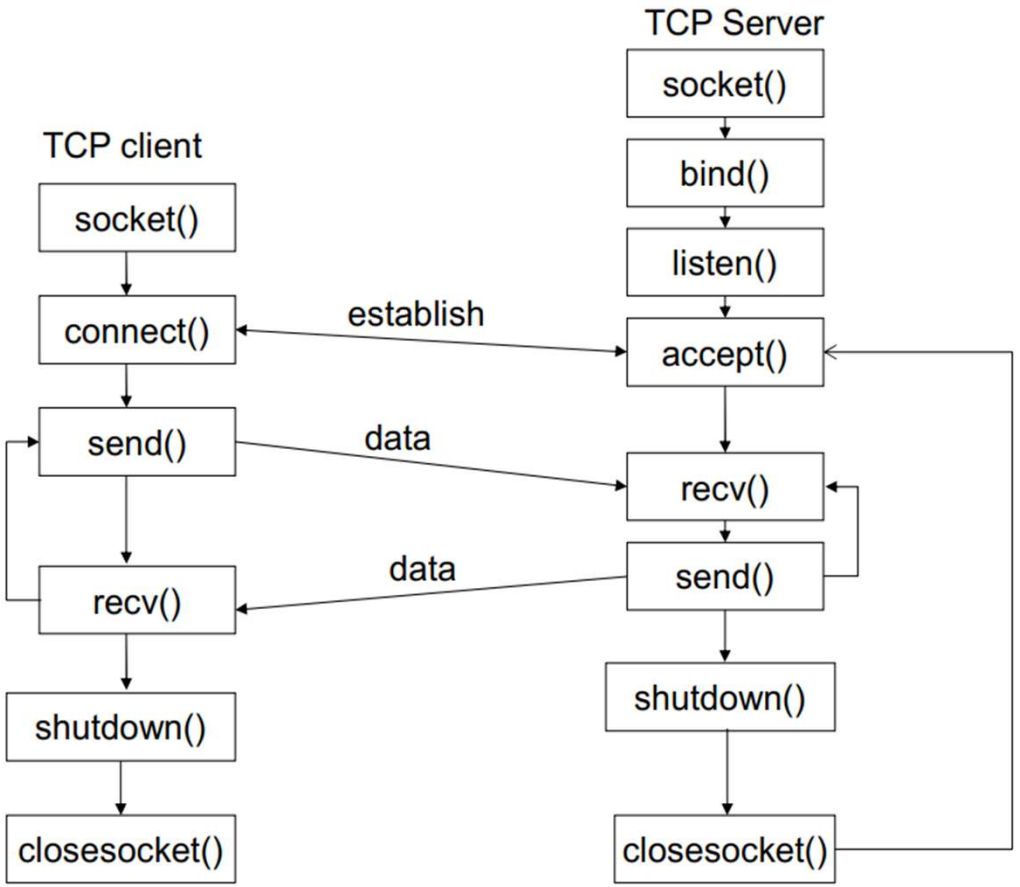
- **Giao thức TCP/IP:**
  - Cung cấp quá trình truyền dữ liệu tin cậy giữa hai máy tính
  - Khi ứng dụng giao tiếp thông qua TCP
    - Một kết nối ảo được thiết lập giữa máy nguồn và đích
  - Khi kết nối được thiết lập
    - Dữ liệu được trao đổi giữa các máy tính
    - Dòng byte được truyền theo hai chiều

TCP SOCKET



- **Tiến trình**
  - Hai tiến trình khác nhau được thực thi trên Server và Client

TCP SOCKET


TCP Server
socket()
bind()
listen()
accept()
recv()
send()
shutdown()
closesocket()
TCP client
socket()
connect()
send()
recv()
shutdown()
closesocket()
establish
data
data

HÀM API DÙNG CHO TCP > SERVER

- **Tiến trình Server:**
  - Chờ nhận yêu cầu (“request”) từ bất cứ số lượng kết nối client
  - Nghe yêu cầu kết nối và được định danh thông qua:
    - Địa chỉ IP (IP address)
    - Số hiệu cổng (port number)
  - Mỗi giao thức có phương pháp định danh khác nhau

- **Các bước thực hiện**
  - Bước 1: Khởi tạo socket
  - Bước 2: “bind API” – gắn socket với định danh
  - Bước 3: “listen API” – đặt socket vào “listening mode”
  - Bước 4: Client yêu cầu kết nối thì Server chấp nhận kết nối
    - Gọi “accept” hoặc “WSAAccept”

HÀM API DÙNG CHO TCP > SERVER

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

- **Tham số:**
  - "s": socket cho server
  - "name": kiểu struct sockaddr, địa chỉ trỏ đến một bộ đệm
  - "namelen": kích thước của bộ đệm được trỏ bởi con trỏ "name"

HÀM API DÙNG CHO TCP > SERVER

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

- **Hàm *listen()*: Đặt trạng thái lắng nghe cho socket**
- **Tham số:**
  - "s": socket cho server đã được tạo
  - "backlog": chiều dài tối đa của hàng đợi chờ kết nối, ví dụ; backlog = 5
- **Trả về:**
  - 0: Thành công
  - SOCKET_ERROR: Thất bại

HÀM API DÙNG CHO TCP > SERVER

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

- **Hàm *accept()*: Khởi tạo một socket cho TCP trong hàng đợi**
- **Tham số:**
  - "s": socket cho server đã được tạo
  - "addr": con trỏ đến địa chỉ socket của Client
  - "addrlen": kích thước bộ đệm được trỏ bởi "addr"
- **Trả về:**
  - SOCKET: Thành công, trả về một socket với kết nối TCP
  - SOCKET_ERROR: Thất bại