**Winsock là gì?**

Winsock (viết tắt cho Windows Sockets) là một đặc điểm kỹ thuật định nghĩa cách thức mà phần mềm Windows truy cập vào dịch vụ mạng, đặc biệt là TCP

**TCP là gì?**  
TCP, viết tắt của Transmission Control Protocol, là một trong các giao thức cốt lõi của bộ giao thức Internet, thườngđược gọi đơn giản là TCP / IP. Sử dụng TCP, các ứng dụng trên máy chủ mạng có thể tạo các kết nối với nhau, mà họcó thể trao đổi các dòng dữ liệu bằng cách sử dụng Stream Sockets. Giao thức bảo đảm cung cấp đáng tin cậy vàtrong trật tự của dữ liệu từ người gửi đến người nhận. TCP cũng phân biệt dữ liệu cho nhiều kết nối bởi các ứng dụng đồng thời (ví dụ, máy chủ Web và máy chủ e-mail) đang chạy trên cùng một máy chủ lưu trữ.

**Socket là gì?**  
-Socket là một cổng logic mà một chương trình sử dụng để kết nối với một chương trình khác chạy trên một máy tính khác trên Internet. Chương trình mạng có thể sử dụng nhiều Socket cùng một lúc, nhờ đó nhiều chương trình có thể sử dụng Internet cùng một lúc.  
Có 2 loại Socket:  
+Stream Socket: Dựa trên giao thức TCP( Tranmission Control Protocol) việc truyền dữ liệu chỉ thực hiện giữa 2 quá trình **đã thiết lập kết nối**. Giao thức này đảm bảo dữ liệu được truyền đến nơi nhận một cách đáng tin cậy, đúng thứ tự nhờ vào cơ chế quản lý luồng lưu thông trên mạng và cơ chế chống tắc nghẽn.  
+Datagram Socket: Dựa trên giao thức UDP( User Datagram Protocol) việc truyền dữ liệu **không yêu cầu có sự thiết lập kết nối giữa 2 quá trình**. Ngược lại với giao thức TCP thì dữ liệu được truyền theo giao thức UDP không được tin cậy, có thế không đúng trình tự và lặp lại. Tuy nhiên vì nó không yêu cầu thiết lập kết nối không phải có những cơ chế phức tạp nên tốc độ nhanh…ứng dụng cho các ứng dụng truyền dữ liệu nhanh như chat, game…..  
**Port là gì ?**  
– Port xách định duy nhất một quá trình (process) trên một máy trong mạng. Hay nói cách khác là cách mà phân biệt giữa các ứng dụng.  
VD: Khi máy bạn chạy nhiều ứng dụng mạng như Yahoo,Firefox, game online… .Ví dụ chương Yahoo sử dụng ( port 5150 hay 5050)  thì khi ai đó gửi tin nhắn đến cho bạn, lúc tin nhắn đến máy bạn nó sẽ dựa vào port để nhận biết đó là chương trình Yahoo ( port 5150) chứ ko pải là chương trình khác. Sau đó thông tin sẽ đc xử lý và hiễn thị tin nhắn lên.  
– Một TCP/IP Socket gồm một địa chỉ IP kết hợp với một port ? Xác định duy nhất một tiến trình (process ) trên mạng.Hay nói cách khác Luồng thông tin trên mạng dựa vảo IP là để xác định máy một máy trên mạng còn port xác định 1 tiến trình trên 1 máy.  
**Ứng dụng Client – Server là gì**  
– Trước tới giờ, các bạn lập trình với mục đích là tạo ra được một ứng dụng. Nhưng ứng dụng đó chỉ hoạt động độc lập 1 mình riêng lẽ. Mục tiêu lập trình mạng sẽ đưa ra những ứng dụng dạng Client – Server. Tức là sẽ có 2 loại ứng dụng chính đó là Client và Server.  
– Quy trình hoạt động của ứng dụng Server – Client như sau: Server có nhiệm vụ của là **lắng nghe, chờ đợi kết nối từ Client trên địa chỉ IP của mình với PORT được quy định sẵn**. Khi client gởi dữ liệu tới Server thì nó phải giải quyết một công việc là **nhận dữ liệu đó -> xử lý -> trả kết quả lại cho Client**.  
– Client là ứng dụng được phục vụ, nó chỉ gởi truy vấn và chờ đợi kết quả từ Server

**2.1.1. Khởi động Winsock**

int WSAStartup(WORD wVersionRequested, LPWSADATA lpWSAData);

Các tham số:  
– wVersionRequested là phiên bản thư viện mà mình sử dụng. Ở đây sẽ là giá trị 0x0202 có nghĩa là phiên bản 2.2.  
– lpWSData là một số thông tin bổ sung sẽ được trả về sau khi gọi khởi tạo Winsock

**2.2.1. Tạo socket**

SOCKET s = socket(AF\_INET,SOCK\_STREAM,IPPROTO\_IP);

Các tham số:  
      – af: [in] mô tả họ địa chỉ.  
      – type: [in] kiểu của socket.  
              + SOCK\_STREAM: TCP socket  
              + SOCK\_DGRAM: UDP socket  
      -protocol: [in] nghi thức được sử dụng trên socket.  
              + SOCK\_DGREAM -> protocol là: IPPROTO\_UDP  
              + SOCK\_STREAM -> protocol là: IPPROTO\_IP  
              +  SOCK\_RAW -> protocol có thể là: IPPROTO\_RAW hay IPPROTO\_ICMP

--Getsockname()

Hàm này sử dụng để lấy địa chỉ cục bộ của socket. Tham số đầu tiên là socket đ~ được kết nối, hai tham số sau l{ đầu ra của hàm, chứa thông tin về địa chỉ cục bộ của socket.

**2.3. TCP**

**2.3.1. Gắn địa chỉ cho socket**

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

Các tham số  
*• s: [in] socket chưa được gắn kết địa chỉ.*  
*• name: [in] địa chỉ được gán cho socket, một cấu trúc SOCKADDR.*  
*• namelen: [in] kích thước của giá trị tham số name.*  
\* Giá trị trả về  
*• 0, nếu thành công*  
*• SOCKET\_ERROR, nếu có lỗi.*  
**2.3.2. Lắng nghe kết nối**

int listen( SOCKET s, int backlog );

*\** Các tham số  
*• s: [in] socket đã được gắn địa chỉ nhưng chưa kết nối.*  
*• backlog: [in] kích thước tối đa của hàng đợi thiết lập kết nối. Giá trị tối đa được chỉ định bằng hằng số SOMAXCONN.*  
**2.3.3. Chấp nhật thiết lập một kết nối**

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

\* Các tham số  
*• s: [in] socket đang lắng nghe yêu cầu kết nối.*  
*• addr: [out] địa chỉ của socket ở máy client đang thực hiện kết nối.*  
*• addrlen: [out] chiều dài thực sự của addr. Phải khởi tạo giá trị ban đầu là kích thước của addr*  
\* Giá trị trả về  
*• Một SOCKET để giao tiếp thực sự với client, nếu thành công*  
*• INVALID\_SOCKET, nếu có lỗ*i  
**2.3.4. Thiết lập một kết nối**

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

\* Các tham số  
*• s: [in] socket chưa kết nối.*  
*• name: [in] socket cần kết nối đến.*  
*• namelen: [in] kích thước của name.*  
\* Giá trị trả về  
*• 0, nếu thành công*  
*• SOCKET\_ERROR, nếu có lỗi*  
**2.3.5. Gửi dữ liệu**

int send( SOCKET s, const char FAR \* buf, int len, int flags )

\* Các tham số  
*• s: [in] socket đã kết nối.*  
*• buf: [in] vùng đệm chứa dữ liệu cần gửi.*  
*• len: [in] chiều dài dữ liệu trong buf.*  
*• flags: [in] chỉ định cách thức truyền dữ liệu, truyền dữ liệu bình thường, thiết lập giá trị 0.*  
\* Giá trị trả về  
*• số byte đã gửi đi, nếu thành công*  
*• SOCKET\_ERROR, nếu có lỗi.*  
**2.3.6. Nhận dữ liệu**

int recv( SOCKET s, char FAR\* buf, int len, int flags )

\* Các tham số  
*• s: [in] socket đã kết nối.*  
*• buf: [out] vùng đệm để lưu dữ liệu nhận.*  
*• len: [in] kích thước vùng đệm buf.*  
*• flags: [in] chỉ định cách thức nhận dữ liệu, nhận dữ liệu bình thường, thiết lập giá trị 0.*  
\* Giá trị trả về  
*• số byte dữ liệu nhận được, nếu thành công*  
*• SOCKET\_ERROR, nếu có lỗi.*  
2.3.7 Shutdown

int shutdown( SOCKET s, int how )

\* Các tham số  
*• s: [in] socket cần shutdown.*  
*• how: [in] chỉ định những loại thao tác nào không thực hiện nữa.*  
*o SD\_RECEIVE: không cho phép gọi các hàm recv() trên socket.*  
*o SD\_SEND: không cho phép gọi các hàm send() trên socket.*  
*o SD\_BOTH: không cho phép gọi cả send() và recv() trên socket.*  
\* Giá trị trả về  
*• 0, nếu thành công*  
*• SOCKET\_ERROR, nếu có lỗi*  
**2.3.7 Đóng socket**

int closesocket (SOCKET s)

\* Các tham số  
*• s: [in] socket cần đóng.*