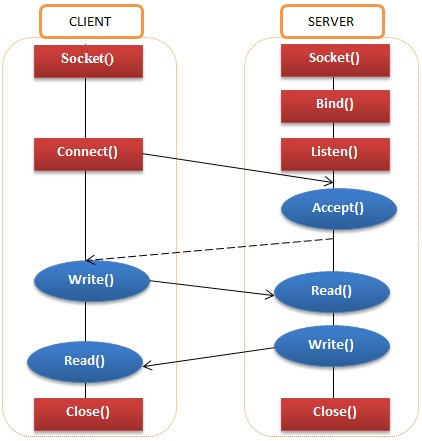
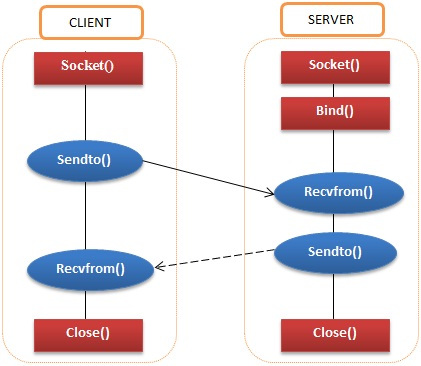
1. **Socket**
   * Socket là một cổng logic mà một chương trình sử dụng để kết nối với một chương trình khác chạy trên một máy tính khác trên Internet. Chương trình mạng có thể sử dụng nhiều Socket cùng một lúc, nhờ đó nhiều chương trình có thể sử dụng Internet cùng một lúc.
   * Có 2 loại Socket:
     + Stream Socket: Dựa trên giao thức TCP (Tranmission Control Protocol) việc truyền dữ liệu thì thực hiện giữa 2 quá trình đã 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 được truyền theo giao thức UDP không được tin cậy, có thể 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....
2. **Port**
   * Port xác định duy nhất một quá trình trên một máy trong mạng. Hay nói cách khác là cách mà phân biệt giữa các ứng dụng.
   * Một TCP/IP Socket gồm một địa chỉ IP kêt hợp với một port. Xác định duy nhất một tiến trình trên mạng. Hay nói cách khác luồng thông tin trên mạng dựa vào IP là để xác định một máy trên mạng còn port xác định một tiến trình trên một máy.
3. **Giao thức TCP và UDP**
   * Transmission Control Protocol (TCP): là một trong các giao thức cốt lõi của bộ giao thức TCP/IP. Sử dụng TCP, các ứng dụng trên các máy chủ được nối mạng có thể tạo các kết nối với nhau, mà qua đó có thể trao đổi dữ liệu hoặc các gói tin. Giao thức này đảm bảo chuyển giao dữ liệu tới nói nhận một cách đáng tin cậy và đúng thứ tự. TCP còn phân biệt giữa dữ liệu của nhiều ứng dụng (chẳng hạn, dịch vụ web và dịch vụ thư điện tử) đồng thời chạy trên cùng một máy chủ.  
       
       
       
     

Hình 1: Mô hình Socket với TCP

* + User Datagram Protocol (UDP): là một trong những giao thức cốt lõi của giao thức TCP/IP. Dùng UDP, chương trình trên mạng máy tính có thể gửi những dữ liệu ngắn được gọi là datagram tới máy khác. UDP không cung cấp sự tin cậy và thứ tự truyền nhận mà TCP làm; các gói dữ liệu có thể đến không đúng thứ tự hoặc bị mất mà không có thông báo. Tuy nhiên UDP nhanh và có hiệu quả hơn đối với các mục tiêu như kích thước nhỏ và yêu cầu khắt khe về thời gian. Do bản chất không trạng thái của UDP nên hữu dụng đối với việc trả lời các truy vấn nhỏ với số lượng lớn người yêu cầu.  
      
      
      
      
     Hình 2: Mô hình socket với UDP
  + TCP hoạt động theo hướng kết nối (connection-oriented), trước khi truyền dữ liệu giữa 2 máy, nó thiết lập một kết nối giữa 2 máy theo phương thức “bắt tay 3 bước (three-way-hand-shake)” bằng cách gửi gói tin ACK từ máy đích sang máy nhận, trong suốt quá trình truyền gói tin, máy gửi yêu cầu máy đích xác nhận đã nhận đủ các gói tin đã gửi, nếu có gói tin bị mất, máy đích sẽ yêu cầu máy gửi gửi lại, thường xuyên kiểm tra gói tin có bị lỗi không, ngoài ra còn cho phép quy định số lượng gói tin được gói tin được gửi trong một lần gửi (window-sizing), điều này đảm bảo máy nhận nhận được đầy đủ các gói tin mà máy gửi gửi đi. Vì vậy, TCP sẽ truyền dữ liệu chậm hơn UDP nhưng đáng tin cậy hơn UDP.
  + UDP hoạt động theo hướng không kết nối (connectionless), không yêu cầu kết nối giữa hai máy gửi và nhận, không có sự đảm bảo gói tin khi truyền đi cũng như không có thông báo về việc mất gói tin, không kiểm tra lỗi của gói tin. Vì vây, UDP truyền dữ liệu nhanh hơn TCP do cơ chế hoạt động có phần đơn giản tuy nhiên lại không đáng tin cậy bằng TCP.

1. AF\_INET
2. **Luồng (Thread)**
   * Luồng là đơn vị sử dụng CPU cơ bản, gồm : định danh luồng, bộ đếm chương trình, tập các thanh ghi, không gian stack.
   * Các luồng chia sẻ trong một tiến trình : đoạn mã lệnh, đoạn dữ liệu (đối tượng toàn cục), các tài nguyên của khác của hệ điều hành.
   * Các luồng có thể thực hiện cùng đoạn mã với ngữ cảnh (tập thanh ghi, bộ đếm chương trình, stack) khác nhau.
   * Luồng còn được gọi là tiến trình nhẹ (Lightweight Process). Một tiến trình có ít nhất một luồng.
   * Lợi ích của lập trình đa luồng:
     + Tăng tính đáp ứng với người dùng: Cho phép chương trình vẫn thực hiện ngay khi một phần đang chờ đợi (block) hoặc đang thực hiện tính toán tăng cường.
     + Chia sẻ tài nguyên: Các luồng chia sẻ bộ nhớ và tài nguyên của tiến trình chứa luồng đó. Cho phép một ứng đụng chứa nhiều luồng hoạt động trong cùng không gian địa chỉ.
     + Tính kinh tế: Các thao tác khởi tạo, hủy bỏ và luân chuyển luồng ít tốn kém.
     + Sử dụng kiến trúc nhiều vi xử lý: Các luồng chạy song song thực sự trên các bộ vi xử lý khác nhau.

AF\_INET: The Internet Protocol version 4 (IPv4) address family.

SOCK\_STREAM: A socket type that provides sequenced, reliable, two-way, connection-based byte streams with an OOB data transmission mechanism. This socket type uses the Transmission Control Protocol (TCP) for the Internet address family (AF\_INET or AF\_INET6).

IPPROTO\_TCP: The Transmission Control Protocol (TCP). This is a possible value when the af parameter is AF\_INET or AF\_INET6 and the type parameter is SOCK\_STREAM.

AI\_PASSIVE flag indicates the caller intends to use the returned socket address structure in a call to the [bind](https://msdn.microsoft.com/en-us/library/windows/desktop/ms737550(v=vs.85).aspx) function. When the AI\_PASSIVE flag is set and nodename parameter to the [getaddrinfo](https://msdn.microsoft.com/en-us/library/windows/desktop/ms738520(v=vs.85).aspx) function is a NULL pointer, the IP address portion of the socket address structure is set to INADDR\_ANY for IPv4 addresses or IN6ADDR\_ANY\_INIT for IPv6 addresses.

WSAStartup

Khởi tạo Winsock :

**int** WSAStartup(**WORD** wVersionRequested, LPWSADATA lpWSAData)**;**

wVersionRequested: phiên bản của thư viện

lpWSAData: là một số thông tin bổ sung **sẽ được trả về** sau khi gọi khởi tạo Winsock:

typedef struct WSAData {

WORD           wVersion;

WORD           wHighVersion;

char           szDescription[WSADESCRIPTION\_LEN+1];

char           szSystemStatus[WSASYS\_STATUS\_LEN+1];

unsigned short iMaxSockets;

unsigned short iMaxUdpDg;

char FAR       \*lpVendorInfo;

} WSADATA, \*LPWSADATA;

# MAKEWORD macro

Tạo một giá trị Word bằng cách ghép nối các giá trị được chỉ định.

WORD MAKEWORD(

BYTE bLow,

BYTE bHigh

);

## Thông số:

bLow

Byte thấp của giá trị mới.

bHigh

Byte cao của giá trị mới.

## Giá trị trả về

Kiểu: WORD

Giá trị trả về là một WORD value.

<https://msdn.microsoft.com/en-us/library/windows/desktop/ms740506(v=vs.85).aspx>

<https://msdn.microsoft.com/en-us/library/windows/desktop/ms742213(v=vs.85).aspx>

BOOL WINAPI GetWindowRect(

\_In\_  HWND   hWnd,

\_Out\_ LPRECT lpRect

);

Lấy kích thước của của sổ có thẻ là hWnd và lưu vào lpRect. Kích thước là tọa độ của màn hình góc trên bên trái và góc dưới bên phải.

BOOL WINAPI MoveWindow(

\_In\_ HWND hWnd,

\_In\_ int  X,

\_In\_ int  Y,

\_In\_ int  nWidth,

\_In\_ int  nHeight,

\_In\_ BOOL bRepaint

);

Thay đổi vi trí và kích thước cửa sổ. X: ví trị bên trái mới của cửa sổ. Y: vị trí bên trên mới của cửa sổ, nWidth: chiều rộng mới của cửa sổ, nHeight: chiều cao mới của cửa sổ

typedef struct \_CONSOLE\_SCREEN\_BUFFER\_INFO {

COORD      dwSize;

COORD      dwCursorPosition;

WORD       wAttributes;

SMALL\_RECT srWindow;

COORD      dwMaximumWindowSize;

} CONSOLE\_SCREEN\_BUFFER\_INFO;

Chứa thông tin về vùng đệm của màn hình console. dwSize : kích thước của vùng đệm, dwCursorPosition: vị trí của con trỏ trên vùng đệm, wAttributes: chứa thông tin về các ký tự, srWindow: chứa thông tin về tọa độ góc trên bên trái và góc dưới bên phải của cửa sổ, dwMaximumWindowSize: kích thước tối đa của cửa sổ.

BOOL WINAPI GetConsoleScreenBufferInfo(

\_In\_  HANDLE                      hConsoleOutput,

\_Out\_ PCONSOLE\_SCREEN\_BUFFER\_INFO lpConsoleScreenBufferInfo

);

Lấy thông tin về vùng đệm màn hình console. hConsoleOutput: thẻ của console buffer, lpConsoleScreenBufferInfo: con trỏ trỏ đến cấu trúc chứa thông tin về vùng đệm.

void ZeroMemory(

[in] PVOID  Destination,

[in] SIZE\_T Length

);

Chuyển cả khối nhớ về 0. Destination: con trỏ trỏ đến khối nhớ, Length: kích thước của khối nhớ.

int WSAAPI getaddrinfo(

\_In\_opt\_       PCSTR      pNodeName,

\_In\_opt\_       PCSTR      pServiceName,

\_In\_opt\_ const ADDRINFOA  \*pHints,

\_Out\_          PADDRINFOA \*ppResult

);

Lấy thông tin về IP address thông qua tên của máy. pNodeName: tên máy, pServiceName: tên cổng, \*pHints: con trỏ trỏ đến cấu trúc addinfo chứa thông tin về địa chỉ, \*ppResult: danh sách các kết quả.

HANDLE WINAPI GetStdHandle(

\_In\_ DWORD nStdHandle

);

Lấy thẻ về thiết bị chuẩn. nStdHandle: thiết bị chuẩn : **STD\_INPUT\_HANDLE ,** **STD\_OUTPUT\_HANDLE, STD\_ERROR\_HANDLE**

BOOL WINAPI SetConsoleTextAttribute(

\_In\_ HANDLE hConsoleOutput,

\_In\_ WORD   wAttributes

);

Thiết lập các thuộc tính của ký tự được viết. hConsoleOutput: thẻ vùng đệm màn hình console, wAttributes: chứa các thuộc tính.

typedef struct hostent {

char FAR      \*h\_name;

char FAR FAR \*\*h\_aliases;

short         h\_addrtype;

short         h\_length;

char FAR FAR \*\*h\_addr\_list;

} HOSTENT, \*PHOSTENT, FAR \*LPHOSTENT;

Cấu trúc lưu các thông tin về máy chủ. \*h\_name: tên máy chủ, \*\*h\_aliases: danh sách tên thay thế, h\_addrtype: loại địa chỉ, h\_length: độ dài địa chỉ, \*\*h\_addr\_list: danh sách địa chỉ của máy chủ

int gethostname(

\_Out\_ char \*name,

\_In\_  int  namelen

);

Lấy tên máy tính: \*name: tên máy nhận được, namelen: độ dài tên

struct hostent\* FAR gethostbyname(

\_In\_ const char \*name

);

Lấy thông tin của máy chủ theo tên \*name

char\* FAR inet\_ntoa(

\_In\_ struct in\_addr in

);

Chuyển đổi IP sang kiếu xâu ký tự

BOOL WINAPI SetConsoleCursorPosition(

\_In\_ HANDLE hConsoleOutput,

\_In\_ COORD  dwCursorPosition

);

Thiết lập vị trí con trỏ. hConsoleOutput: thẻ vùng đệm của màn hình, dwCursorPosition: tạo độ con trỏ.