BÀI 4.
LẬP TRÌNH WINSOCK NÂNG CAO

1

Nội dung

- Xây dựng ứng dụng yêu cầu lớn (Scalable Apps)
- Sử dụng raw socket
- Xây dựng ứng dụng broadcast và multicast

2

1. XÂY DỰNG ỨNG DỤNG YÊU CẦU LỚN

3

Ứng dụng yêu cầu lớn(Scalable Apps)

- Ứng dụng cần xử lý số lượng kết nối, yêu cầu rất lớn
- Sử dụng các hàm WinSock API hiệu năng cao
  - AcceptEx()
  - ConnectEx()
  - TransmitFile()
  - TransmitPacket()
  - …
- Sử dụng kỹ thuật vào ra Overlapped I/O với Completion Port
- Cần có các kỹ thuật kiểm soát số lượng kết nối, quản lý tài nguyên

4

Hàm AcceptEx()

- Chấp nhận một kết nối và (có thể) nhận gói tin đầu tiên
- Bộ đệm chứa dữ liệu nhận được và thông tin địa chỉ
- **`sAcceptSocket`** phải ở trạng thái chưa kết nối
- Trả về TRUE nếu thành công

```
BOOL AcceptEx(
  SOCKET sListenSocket,        //[IN] Socket nghe yêu cầu
  SOCKET sAcceptSocket,        //[IN] Socket chấp nhận kết nối
  PVOID  lpOutputBuffer,       //[IN] Bộ đệm nhận dữ liệu
  DWORD  dwReceiveDataLength, //[IN] Kích thước bộ đệm
  DWORD  dwLocalAddressLength,//[IN] Kích thước phần bộ đệm
                              // chứa địa chỉ local socket
  DWORD  dwRemoteAddressLength, //[IN] Kích thước phần bộ đệm
                                // chứa địa chỉ remote socket
  LPDWORD lpdwBytesReceived,          //[OUT] Kích thước dữ
                                      // liệu đã nhận
  LPOVERLAPPED lpOverlapped           //[IN] Kết quả vào ra
);
```

5

Sử dụng AcceptEx()

```
SOCKET  listenSock, accSock;
HANDLE  hCompPort;
LPFN_ACCEPTEX  lpfnAcceptEx=NULL;
GUID GuidAcceptEx=WSAID_ACCEPTEX;
PER_IO_DATA    ol;
SOCKADDR_IN saLocal;
DWORD dwBytes;
char buf[1024];
int buflen=1024;

// Create the completion port
hCompPort = CreateIoCompletionPort(INVALID_HANDLE_VALUE,
                                   NULL, (ULONG_PTR)0, 0);

// Create the listening socket
listenSock = socket(AF_INET, SOCK_STREAM, IPPROTO_TCP);

// Associate listening socket to completion port
CreateIoCompletionPort((HANDLE) listenSock, hCompPort,
                           (ULONG_PTR)0, 0);
```

6

Sử dụng AcceptEx()

```
// Bind the socket to the local port
salocal.sin_family = AF_INET;
salocal.sin_port   = htons(5150);
salocal.sin_addr.s_addr = htonl(INADDR_ANY);
bind(listenSock, (SOCKADDR *)&saLocal, sizeof(salocal));

// Set the socket to listening
listen(listenSock, 200);

// Load the AcceptEx function
WSAIoctl(listenSock,
       SIO_GET_EXTENSION_FUNCTION_POINTER,
       GuidAcceptEx,
       sizeof(GuidAcceptEx),
       &lpfnAcceptEx,
       sizeof(lpfnAcceptEx),
       &dwBytes, NULL, NULL);
```

Gọi hàm AcceptEx() khi chạy → hiệu năng tốt hơn

7

Sử dụng AcceptEx()

```
// Create the client socket for the accepted connection
accSock = socket(AF_INET, SOCK_STREAM, IPPROTO_TCP);

// Initialize our "extended" overlapped structure
memset(&ol, 0, sizeof(ol));
ol.operation = OP_ACCEPTEX;
ol.client    = accSock;

lpfnAcceptEx(listenSock, accSock, buf,
             buflen - ((sizeof(SOCKADDR_IN) + 16) * 2),
             sizeof(SOCKADDR_IN) + 16,
             sizeof(SOCKADDR_IN) + 16,
             &dwBytes, &ol.overlapped);

// Call GetQueuedCompletionStatus within the completion
//function After the AcceptEx() operation completes associate
//the accepted client socket with the completion port
```

8

Hàm GetAcceptExSockaddrs()

- Lấy thông tin địa chỉ từ dữ liệu của hàm AcceptEx()

```
void GetAcceptExSockaddrs(
  PVOID lpOutputBuffer,        //[IN] Bộ đệm nhận dữ liệu
                               // sử dụng trong AcceptEx()
  DWORD dwReceiveDataLength,   //[IN] Kích thước dữ liệu trong
                               // bộ đệm = dwReceiveDataLength
  DWORD dwLocalAddressLength, //[IN] Kích thước phần bộ đệm
                               // chứa địa chỉ local socket
  DWORD dwRemoteAddressLength,//[IN] Kích thước phần bộ đệm
                               // chứa địa chỉ remote socket
  LPSOCKADDR *LocalSockaddr,  //[OUT] Địa chỉ local socket
  LPINT LocalSockaddrLength,  //[OUT] Kích thước địa chỉ local
                               // socket
  LPSOCKADDR *RemoteSockaddr, //[OUT] Địa chỉ remote socket
  LPINT RemoteSockaddrLength  //[OUT] Kích thước địa chỉ
                               // remote socket
);
```

9

Hàm TransmitFile()

- Truyền file qua TCP socket
- Trả về: TRUE nếu thành công
- Xem thêm TransmitPacket()

```
BOOL PASCAL TransmitFile(
   SOCKET hSocket,     //[IN] Socket
   HANDLE hFile,       //[IN] Handle của file
   DWORD nNumberOfBytesToWrite, //[IN] Kích thước dữ liệu sẽ gửi
   DWORD nNumberOfBytesPerSend, //[IN] Kích thước mỗi
                                // block(=0:default)
   LPOVERLAPPED lpOverlapped,   //[IN] Cấu trúc OVERLAPPED chứa
                                // kết quả
   //[IN] Dữ liệu cần truyền ngoài dữ liệu của file
   LPTRANSMIT_FILE_BUFFERS lpTransmitBuffers,
   DWORD dwFlags       //[IN] Cờ điều khiển
);
```

10

Hàm TransmitFile() – Cờ điều khiển

Cờ	Ý nghĩa
TF_DISCONNECT	Ngắt kết nối sau khi truyền xong file
TF_REUSE_SOCKET	Có thể sử dụng lại socket cho AcceptEx(). Cờ này chỉ dùng đồng thời với cờ TF_DISCONNECT
TF_USE_DEFAULT_WORKER	Sử dụng thread của hệ thống, thường dùng khi truyền file kích thước lớn
TF_USE_SYSTEM_THREAD	Sử dụng thread của hệ thống để xử lý
TF_USE_KERNEL_APC	Sử dụng Asyncronous Procedure Call để xử lý
TF_WRITE_BEHIND	Truyền file ngay, không qua trạng thái PENDING

11

Hàm ConnectEx()

- Yêu cầu thiết lập kết nối và gửi 1 gói tin ngay khi kết nối được thiết lập
- Chỉ dùng trên TCP socket
- Trả về: TRUE nếu thành công
- Đọc thêm: DisconnectEx()

```
BOOL PASCAL ConnectEx(
  SOCKET s,                    //[IN] Socket
  const struct sockaddr *name,//[IN] Địa chỉ của server
  int namelen,                 //[IN] Kích thước địa chỉ
  PVOID lpSendBuffer,          //[IN] Bộ đệm chứa dữ liệu cần gửi
  DWORD dwSendDataLength,      //[IN] Kích thước dữ liệu cần gửi
  LPDWORD lpdwBytesSent,       //[OUT] Kích thước dữ liệu đã gửi
  LPOVERLAPPED lpOverlapped    //[IN] Cấu trúc Overlapped chứa
                               //kết quả
);
```

12

Xử lý kết nối - AcceptEx()

- Hàm AcceptEx() xử lý việc chấp nhận một kết nối qua kỹ thuật I/O Overlapped
- Yêu cầu khởi tạo socket trước khi chấp nhận kết nối
- Gọi hàm AcceptEx() "đúng lúc": sử dụng WSAEventSelect() với sự kiện theo dõi FD_ACCEPT
- Hàm AcceptEx() chỉ trả về nếu kết nối được thiết lập thành công và nhận được tối thiểu 1 byte → kết nối trở thành kết nối "rác" nếu không có dữ liệu gửi tới
- Loại trừ các kết nối "rác": sử dụng hàm getsockopt() với thuộc tính tùy chọn SO_CONNECT_TIME để xác định thời gian kết nối đã được thiết lập. Nếu lớn hơn ngưỡng nào đó có thể đóng kết nối
- Số lần gọi AcceptEx() phụ thuộc vào thời gian một kết nối duy trì
  - Có thể sử dụng ngưỡng để kiểm soát

13

Xử lý kết nối - AcceptEx()

- Không đóng các socket khi hàm AcceptEx() chưa hoàn thành thiết lập kết nối
- Không gọi hàm AcceptEx() trong luồng xử lý của completion port
- Tránh thực hiện các thao tác tính toán chi phí lớn trong luồng xử lý của completion port

14

Xử lý kết nối(tiếp)

- Số lượng kết nối tối đa trong hàng đợi (backlog):
  - HĐH Windows cho máy trạm: 5
  - HĐH Windows cho máy chủ: 200
  - Có thể mở rộng lên 0x7FFFFFFF
  - Giá trị backlog phụ thuộc vào tần suất yêu cầu kết nối, thời gian thực thi hàm accept()
  - Tại sao không đặt backlog quá lớn, quá nhỏ?
  - Có thể thiết lập chế độ Dynamic Backlog

  https://support.microsoft.com/en-us/kb/142641

15

Truyền dữ liệu

- Sử dụng kỹ thuật Overlapped I/O với Completion Port
- Sử dụng TransmitFile(), TransmitPacket()
- Có cần thiết phải xóa bộ đệm dữ liệu?
  - Không làm tăng hiệu năng ứng dụng
  - Xóa bộ đệm gửi của ứng dụng không gây ảnh hưởng tới hiệu năng do bộ đệm bị "khóa" cho tới khi có dữ liệu chuyển xuống cho giao thức TCP xử lý
  - Xóa bộ đệm nhận của ứng dụng làm giảm hiệu năng của hệ thống. Khi đó, dữ liệu được lưu trong bộ đệm của TCP (đặt trong vùng nhớ không phân trang) chờ xử lý. Số lượng kết nối tăng lên làm tăng kích thước vùng nhớ cần cho bộ đệm
- Khi nào việc xóa bộ đệm nhận không ảnh hưởng tới hiệu năng?
  - Dữ liệu gửi tới với tần suất rất thấp → có thể sử dụng các hàm vào ra ở chế độ non-blocking thay cho Overlapped I/O

16

Quản lý tài nguyên

- Với các ứng dụng yêu cầu lớn, cần kiểm soát hai loại bộ nhớ:
  - Locked page
  - Non-page pool
- Các bộ đệm sử dụng trong thao tác vào ra Overlaped có thể bị khóa (locked page):
  - Gọi quá nhiều hàm vào ra làm bộ nhớ locked page vượt ngưỡng → báo lỗi WSAENOBUFS
  - Giải pháp:
    - Gọi hàm nhận dữ liệu với tham số kích thước vùng buffer nhận bằng 0()
    - Khi thao tác trên hoàn thành, gọi hàm nhận dữ liệu ở chế độ non-blocking tới khi trả về lỗi WSAEWOULDBLOCK
  - Do lock paged có kích thước là bội số của kích thước một trang bộ nhớ → lời gọi vào ra nên sử dụng bộ đệm có kích thước là bội số

17

Quản lý tài nguyên(tiếp)

- Non-paged pool:
  - tcpip.sys
  - Thông tin địa chỉ local của socket
  - Thông tin địa chỉ remote
  - Mỗi socket trả về từ hàm accept() được cấp 2 KB
  - Mỗi socket sử dụng trong hàm AcceptEx() được cấp 1.5 KB
  - Mỗi lời gọi overlapped sử dụng 500 bytes
- Cần kiểm thử để xác định số kết nối tối đa có thể phục vụ mà không xảy ra hiện tượng non-paged pool quá mức giới hạn của hệ thống(khoảng ¼ kích thước bộ nhớ chính, tối đa là 2GB trên HĐH 32-bit và 128/284 GB trên HĐH 64 bit)

18

Chiến lược xây dựng server

- Kết nối có thông lượng cao(VD: file server)
  - Giới hạn số lượng kết nối của mỗi client
  - Số lời gọi hàm vào ra phụ thuộc vào thông lượng tối đa của server
  - Nếu số lượng kết nối quá lớn, sử dụng hàng đợi và chỉ gọi hàm vào ra trên từng lượng kết nối nhất định
  - Kiểm soát các thao tác vào ra, chấp nhận kết nối chưa hoàn thành (outstanding operation)
- Tần suất kết nối lớn (VD: message server)
  - Khó kiểm soát do số lượng kết nối phải xử lý rất lớn
  - Có thể sử dụng kỹ thuật dùng bộ đệm kích thước 0

19

So sánh hiệu năng(Pentium 4 1.7 GHz Xeon, 768 MB RAM)

I/O Model	Attempted/Connected	Memory Used (KB)	Non-Paged Pool	CPU Usage	Threads	Throughput (Send/ Receive Bytes Per Second)
Blocking	7000/ 1008	25,632	36,121	10–60%	2016	2,198,148/ 2,198,148
	12,000/ 1008	25,408	36,352	5– 40%	2016	404,227/ 402,227
Non-blocking	7000/ 4011	4208	135,123	95–100%*	1	0/0
	12,000/ 5779	5224	156,260	95–100%*	1	0/0
WSAAsync Select	7000/ 1956	3640	38,246	75–85%	3	1,610,204/ 1,637,819
	12,000/ 4077	4884	42,992	90–100%	3	652,902/ 652,902
WSAEvent Select	7000/ 6999	10,502	36,402	65–85%	113	4,921,350/ 5,186,297
	12,000/ 11,080	19,214	39,040	50–60%	192	3,217,493/ 3,217,493
	46,000/ 45,933	37,392	121,624	80–90%	791	3,851,059/ 3,851,059
Overlapped (events)	7000/ 5558	21,844	34,944	65–85%	66	5,024,723/ 4,095,644
	12,000/12,000	60,576	48,060	35–45%	195	1,803,878/ 1,803,878
	49,000/48,997	241,208	155,480	85–95%	792	3,865,152/ 3,834,511
Overlapped (completion port)	7000/ 7000	36,160	31,128	40–50%	2	6,282,473/ 3,893,507
	12,000/12,000	59,256	38,862	40–50%	2	5,027,914/ 5,027,095
	50,000/49,997	242,272	148,192	55–65%	2	4,326,946/ 4,326,496

20

2. XÂY DỰNG ỨNG DỤNG BROADCAST

Xây dựng ứng dụng broadcast

- Sử dụng UDP socket
- Thiết lập tùy chọn SO_BROADCAST cho socket
- Địa chỉ đích: INADDR_BROADCAST

```
int setsockopt (
    SOCKET s,        //[IN]socket được thiết lập
    int level,       //[IN]giao thức của tùy chọn
    int optname,     // [IN]tên tùy chọn
    const char FAR * optval, //[IN]giá trị thiết lập
    int optlen       //[IN]kích thước của tham số optval
);
```

- level = SOL_SOCKET
- optname = SO_BROADCAST
- optval = 1

3. XÂY DỰNG ỨNG DỤNG MULTICAST

23

Địa chỉ multicast

Scope	IPv6	IPv4	
		TTL	Địa chỉ
Interface local	1	0	
Link local	2	1	224.0.0.0 – 224.0.0.255
Site-local	5	<32	239.255.0.0 – 239.255.255.255
Organization-local	8		239.192.0.0 – 239.192.255.255
Global	14	<= 255	224.0.1.0 – 238.255.255.255

- Một số địa chỉ đặc biệt:
  - 224.0.0.1: all-host group
  - 224.0.0.2: all-router group

24

Muticasting với IPv4

- Sử dụng setsockopt() để gia nhập hoặc rời nhóm multicast
- Socket nên được gán địa chỉ INADDR_ANY
- level: IPPROTO_IP
- otpname:
  - IP_ADD_MEMBERSHIP: gia nhập nhóm
  - IP_DROP_MEMBERSHIP: rời nhóm
- Cấu trúc giá trị thiết lập cho tùy chọn:

```
struct ip_mreq {
   struct in_addr imr_multiaddr;  // Địa chỉ nhóm
   struct in_addr imr_interface;  // Địa chỉ của nút mạng
};
```

25

Gia nhập nhóm

```
SOCKET s;
SOCKADDR_IN    localif;
struct ip_mreq mreq;

s = socket(AF_INET, SOCK_DGRAM, IPPROTO_UDP);

localif.sin_family = AF_INET;
localif.sin_port   = htons(5150);
localif.sin_addr.s_addr = htonl(INADDR_ANY);
bind(s, (SOCKADDR *)&localif, sizeof(localif));

mreq.imr_interface.s_addr = inet_addr("157.124.22.104");
mreq.imr_multiaddr.s_addr = inet_addr("234.5.6.7");

setsockopt(s, IPPROTO_IP, IP_ADD_MEMBERSHIP,
           (char *)&mreq,
sizeof(mreq));
```

26

Rời nhóm

```
mreq.imr_interface.s_addr = inet_addr("157.124.22.104");
mreq.imr_multiaddr.s_addr = inet_addr("234.5.6.7");

setsockopt(s, IPPROTO_IP, IP_DROP_MEMBERSHIP, (char
*)&mreq,
sizeof(mreq));
```

27

Multicasting với nguồn xác định

- Một nút khi gia nhập nhóm multicast có thể chỉ định nguồn dữ liệu muốn nhận
  - Chế độ INCLUDE: nhận dữ liệu từ nguồn chỉ định
  - Chế độ EXCLUDE: không nhận dữ liệu từ nguồn chỉ định(chặn)
- Cấu trúc giá trị:

```
struct ip_mreq_source {
   struct in_addr imr_multiaddr;   // Địa chỉ nhóm
   struct in_addr imr_sourceaddr;  // Địa chỉ nguồn
   struct in_addr imr_interface;   // Địa chỉ nút mạng
};
```

28

setsockopt()

- Nhóm INCLUDE:
  - optname: IP_ADD_SOURCE_MEMBERSHIP gia nhập nhóm và thêm nguồn
  - optname: IP_DROP_SOURCE_MEMBERSHIP xóa nguồn
- Nhóm EXCLUDE:
  - optname: IP_ADD_MEMBERSHIP gia nhập nhóm
  - optname: IP_BLOCK_SOURCE thêm nguồn bị chặn
  - optname: IP_UNBLOCK_SOURCE xóa nguồn bị chặn

29

Gia nhập nhóm INCLUDE

```
SOCKET                  s;
SOCKADDR_IN             localif;
struct ip_mreq_source mreqsrc;

s = socket(AF_INET, SOCK_DGRAM, IPPROTO_UDP);

localif.sin_family = AF_INET;
localif.sin_port   = htons(5150);
localif.sin_addr.s_addr = htonl(INADDR_ANY);
bind(s, (SOCKADDR *)&localif, sizeof(localif));

mreqsrc.imr_interface.s_addr = inet_addr("157.124.22.104");
mreqsrc.imr_multiaddr.s_addr = inet_addr("234.5.6.7");
mreqsrc.imr_sourceaddr.s_addr = inet_addr("172.138.104.10");
setsockopt(s, IPPROTO_IP, IP_ADD_SOURCE_MEMBERSHIP,
               (char *)&mreqsrc, sizeof(mreqsrc));

mreqsrc.imr_sourceaddr.s_addr = inet_addr("172.141.87.101");
setsockopt(s, IPPROTO_IP, IP_ADD_SOURCE_MEMBERSHIP,
               (char *)&mreqsrc, sizeof(mreqsrc));
```

30

Gia nhập nhóm EXCLUDE

```
SOCKET                  s;
SOCKADDR_IN             localif;
struct ip_mreq          mreq;
struct ip_mreq_source mreqsrc;

s = socket(AF_INET, SOCK_DGRAM, IPPROTO_UDP);

bind(s, (SOCKADDR *)&localif, sizeof(localif));

// Join a group - the filter is EXCLUDE none
mreq.imr_interface.s_addr = inet_addr("157.124.22.104");
mreq.imr_multiaddr.s_addr = inet_addr("234.5.6.7");

setsockopt(s, IPPROTO_IP, IP_ADD_MEMBERSHIP, (char *)&mreq,
               sizeof(mreq));

mreqsrc.imr_interface = mreq.imr_interface;
mreqsrc.imr_multiaddr = mreq.imr_multiaddr;
mreqsrc.imr_sourceaddr.s_addr = inet_addr("172.138.104.10");

setsockopt(s, IPPROTO_IP, IP_BLOCK_SOURCE, (char *)&mreqsrc,
               sizeof(mreqsrc));
```

31

Truyền thông điệp multicast

- Nhận: phải gia nhập nhóm multicast để nhận thông điệp
- Gửi: không cần gia nhập nhóm
  - Chỉ định cổng gửi: thiết lập tùy chọn IP_MULTICAST_IF cho socket
  - Chỉ định giá trị TTL: thiết lập tùy chọn IP_MULTICAST_TTL
- Sử dụng sendto() và rcvfrom()

32

Xây dựng ứng dụng multicast theo mô hình truyền thông tin cậy

- Cài đặt thành phần MSMQ(Microsoft Message Queuing)
- Chỉ hỗ trợ IPv4
- Gửi:
  - B1: Khởi tạo socket để gửi thông tin multicast(SOCK_RDM)
  - B2: Gán địa chỉ INADDR_ANY cho socket
  - B3: Thiết lập tùy chọn RM_SET_SEND_IF
  - B4: Kết nối socket tới nhóm và gửi dữ liệu
- Nhận:
  - B1: Khởi tạo socket để gửi thông tin multicast(SOCK_RDM)
  - B2: Gán địa chỉ multicast cho socket
  - B3: Thiết lập tùy chọn RM_ADD_RECEIVE_IF để xác định địa chỉ nguồn nếu cần
  - B4: Gọi hàm listen()
  - B5: Gọi hàm accept()
  - B6: Nhận dữ liệu

33

Tùy biến với reliable multicast

- Thay đổi kích thước cửa sổ gửi dữ liệu
  - optname: RM_RATE_WINDOW_SIZE
  - optval: _RM_SEND_WINDOW

```
typedef struct _RM_SEND_WINDOW{
    ULONG RateKbitsPerSec;  //Tốc độ gửi
    ULONG WindowSizeInMSecs;//Thời gian dữ liệu nằm trong
                            //cửa sổ trước khi bị xóa
    ULONG WindowSizeInBytes;//Kích thước cửa sổ
} RM_SEND_WINDOW;
```

34

Tùy biến với reliable multicast

- Cơ chế FEC:
  - Gửi: tạo ra H gói tin parity từ K gói tin dữ liệu và gửi đi N = H + K
  - Nhận: Tái tạo K gói dữ liệu từ K gói bất kỳ trong N gói
- Chế độ:
  - Pro-active: tính FEC cho mọi gói tin
  - On-demand: chỉ tính FEC khi có NAK
- Thiết lập tùy chọn
  - optname: RM_USE_FEC
  - optvalue:

```
typedef struct _RM_FEC_INFO{
    USHORT FECBlockSize;            //Số gói tin tối đa
    USHORT FECProActivePackets;     //Số gói tin parity
    UCHAR  FECGroupSize;            // Số gói tin dữ liệu
    BOOLEAN fFECOnDemandParityEnabled;    //On-demand?
} RM_FEC_INFO;
```

35

4. RAW SOCKET

36

Raw Socket

- Cho phép truyền dữ liệu ở mức dưới tầng giao vận
- Thêm khóa registry

HKEY_LOCAL_MACHINE\System\CurrentControlSet\Services\AFD\Parameters\DisableRawSecurity

- Chạy chương trình với quyền Administrator
- Khởi tạo:

```
socket(s, SOCK_RAW, protocol)
```

protocol: giao thức điều khiển truyền dữ liệu trên raw socket

  - IPPROTO_UDP, IPPROTO_IP, IPPROTO_RAW: cần thiết lập tùy chọn IP_HDRINCL cho socket

37

Sử dụng Raw socket

- Trao đổi dữ liệu: sendto(), recvfrom(), WSASento(), WSARecvfrom()
- Thiết lập tùy chọn: tùy thuộc tình huống. Ví dụ:
  - Bắt tất cả gói tin:

```
int optval = 1
(WSAIoctl(sniffer, SIO_RCVALL, & opt, sizeof(optval), 0, 0,
(LPDWORD) &in , 0 , 0)
```

  - Gửi gói tin UDP, IP với nội dung tùy ý

```
setsockopt(s, IPPROTO_IP, IP_HDRINCL, (char *)&optval,
sizeof(optval));
```

  - Truyền thông điệp ICMP

```
IPV4_OPTION_HDR ipopt;
setsockopt(s, IPPROTO_IP, IP_OPTIONS, (char *)&ipopt,
sizeof(ipopt));
```

38

Tự học

- Thư viện libpcap/WinPcap
- Raw socket và socket tầng 2 trên Linux
- Network programming trong Python

39