Lập trình Socket

Giáo viên: Nguyễn Hoài Sơn
Bộ môn Mạng và Truyền thông máy tính
Khoa Công nghệ thông tin

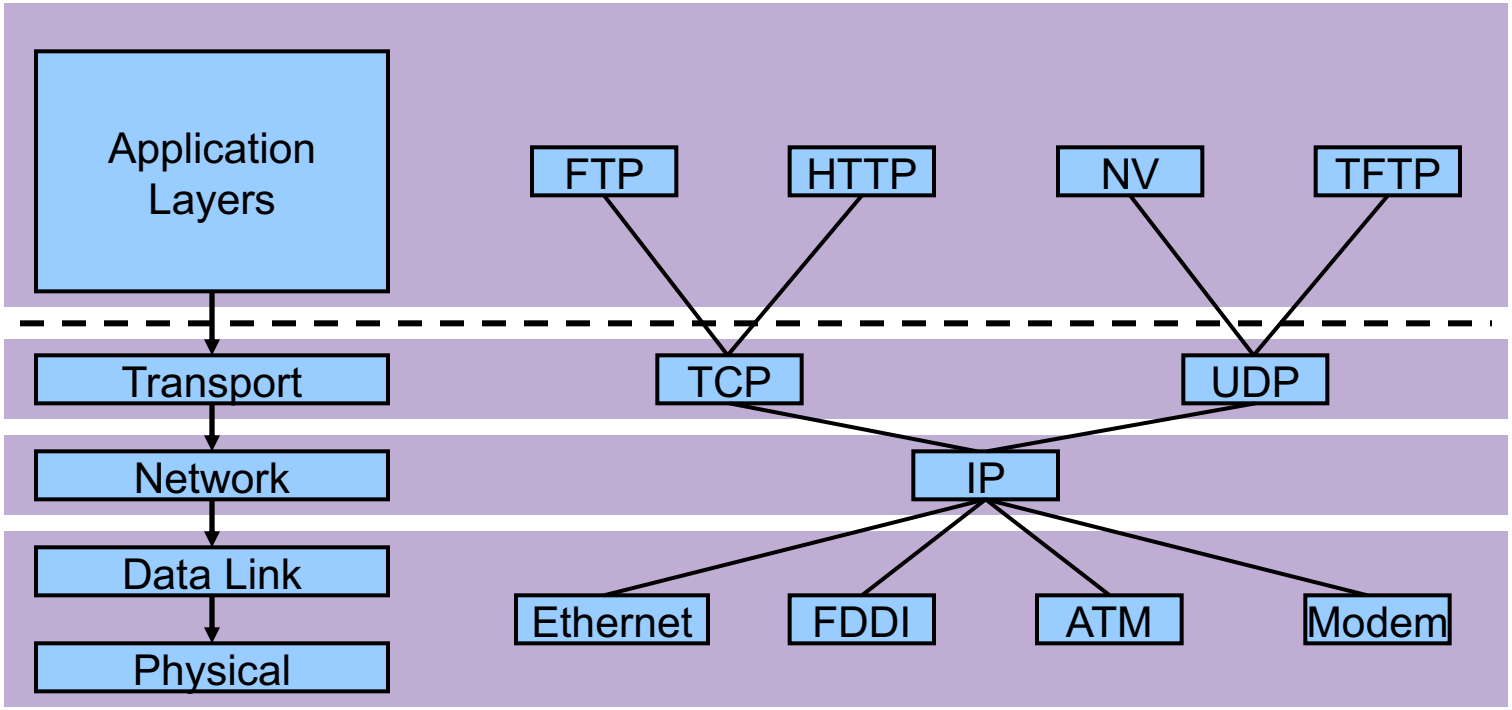
Nội dung

- Giới thiệu về socket
- Các hàm socket với mô hình khách chủ TCP
- Ví dụ về mô hình khách chủ TCP
- Các hàm socket với mô hình khách chủ UDP
- Ví dụ về mô hình khách chủ UDP
- Chuyển tên miền DNS thành địa chỉ IP

Nội dung

- Giới thiệu về socket
- Các hàm socket với mô hình khách chủ TCP
- Ví dụ về mô hình khách chủ TCP
- Các hàm socket với mô hình khách chủ UDP
- Ví dụ về mô hình khách chủ UDP
- Chuyển tên miền DNS thành địa chỉ IP

Giao thức TCP/IP



Giao diện giữa tầng ứng dụng và tầng TCP/IP

- Chuẩn TCP/IP không quy định giao diện của phần mềm ứng dụng với phần mềm thực thi giao thức TCP/IP
  - Việc thực thi giao thức TCP/IP tùy thuộc vào mỗi hệ thống
  - Giao diện này có thể khác nhau với mỗi hệ điều hành
    - Giao diện Socket (BSD Unix), mô hình STREAMS (UNIX System V)

Các chức năng của giao diện

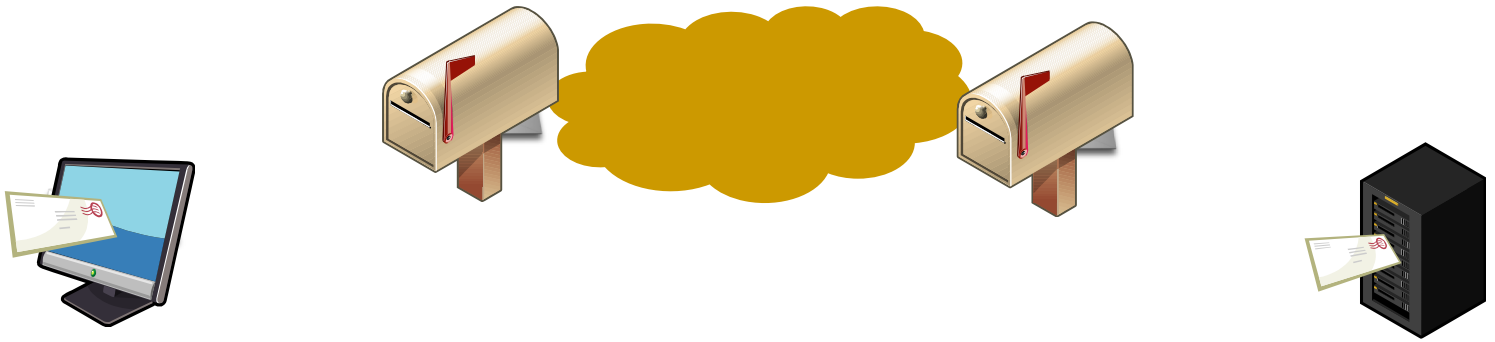
- Yêu cầu cấp phát tài nguyên cho một cuộc truyền tin
- Xác định địa chỉ của bên gửi và bên nhận
- Thiết lập trạng thái chờ một kết nối mới (thông báo mới) đến (máy chủ)
- Khởi tạo một cuộc truyền tin (máy khách)
- Gửi/nhận dữ liệu
- Kết thúc tốt đẹp một cuộc truyền tin
- …

Giao diện Socket

- Mô hình socket được sử dụng rộng rãi hiện nay
  - Được thực thi rộng rãi trên hệ điều hành BSD Unix
    - Xuất hiện lần đầu tiên với hệ điều hành BSD Unix 4.1c
    - Được phổ biến trên hệ điều hành BSD Unix 4.3 năm 1986
  - Hiện được dùng trên Unix, Windows, MAC, ...
- Cung cấp các chức năng cơ bản hỗ trợ việc truyền tin trên mạng với nhiều loại giao thức
  - Không chỉ giao thức TCP/IP

Socket là gì?

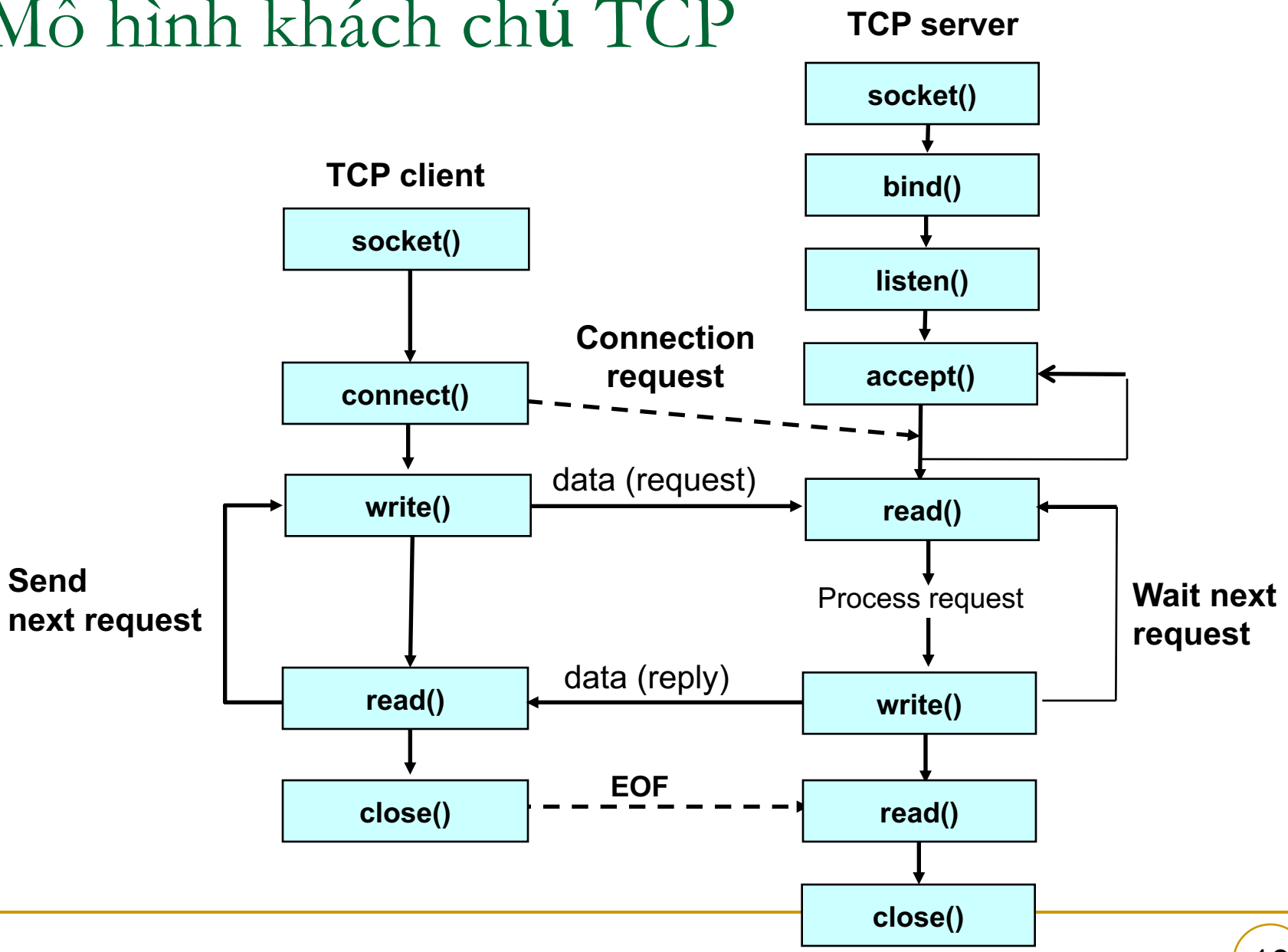
- Với một ứng dụng, socket là một mô tả file cho phép các ứng dụng đọc/ghi từ mạng
  - Máy chủ và máy khách liên lạc với nhau bằng cách đọc và viết vào mô tả socket
  - Sử dụng như các lệnh gọi I/O truyền thống
    - Các lệnh gọi I/O truyền thống: open, close, read, write, lseek, ioctl



Nội dung

- Giới thiệu về socket
- Các hàm socket với mô hình khách chủ TCP
- Ví dụ về mô hình khách chủ TCP
- Các hàm socket với mô hình khách chủ UDP
- Ví dụ về mô hình khách chủ UDP
- Chuyển tên miền DNS thành địa chỉ IP

Mô hình khách chủ TCP



socket() : Tạo socket

```
#include <sys/socket.h>
int socket (int family, int type, int protocol);

Returns: non-negative descriptor if OK, -1 on error
```

- family = Họ giao thức/không gian địa chỉ
  - AF_INET IPv4
  - AF_INET6 IPv6
  - AF_LOCAL Unix Domain
- type = Kiểu socket
  - SOCK_STREAM (TCP)
  - SOCK_DGRAM (UDP)
  - SOCK_RAW
- Protocol = kiểu giao thức
  - 0 nếu sử dụng mặc định của hệ thống
  - Cấn thiết lập trong trường hợp của SOCK_RAW (IP) sockets

Ví dụ về tạoTCP socket

```
int sockfd;                    /* socket descriptor */

if((sockfd = socket(AF_INET, SOCK_STREAM,0)) <
0) {
      perror("socket");
      exit(1);
}
```

- ***socket*** trả về giá trị nguyên là một mô tả socket
  - ***sockfd*** < 0 khi có lỗi tạo socket
- **AF_INET**: gán socket với họ giao thức IPv4
- **SOCK_STREAM:** gán kiểu socket là truyền hướng kết nối
- **0:** sử dụng giao thức mặc định (TCP)

bind() : Gán địa chỉ cho socket

```
#include <sys/socket.h>
int bind (int sockfd, const struct sockaddr *sockaddr, socklen_t addrlen);

                                                    Returns: 0 if OK,-1 on error
```

- *sockfd* = Mô tả socket được tạo ra bởi hàm socket
- *sockaddr* = Con trỏ trỏ đến cấu trúc địa chỉ socket
- *addrlen* = độ lớn địa chỉ

Cấu trúc địa chỉ socket

- Cấu trúc địa chỉ socket chung:
  - Dùng như là tham số địa chỉ trong các lệnh gọi connect, bind, and accept.
  - Cần thiết chỉ vì khi thiết kế giao diện socket, C không có con trỏ kiểu void*

```
struct sockaddr {
  uint8_t         sa_len;       /* length of socket address*/
  unsigned short  sa_family;    /* protocol family */
  char            sa_data[14];  /* address data.  */
};
```

Cấu trúc địa chỉ socket IPv4

```
struct in_addr {
    in_addr_t s_addr;          /* 32-bit IPv4 address */
                               /* network byte ordered */
};
```

- Phải gán (sockaddr_in *) thành (sockaddr *) trong các lệnh gọi connect, bind, and accept.

```
struct sockaddr_in  {
  uint8_t         sin_len;
  unsigned short  sin_family;  /* address family (always AF_INET) */
  unsigned short  sin_port;    /* port num in network byte order */
  struct in_addr  sin_addr;    /* IP addr in network byte order */
  unsigned char   sin_zero[8]; /* pad to sizeof(struct sockaddr) */
};
```

Các hàm chuyển đổi thứ tự Byte

'**h**' : host byte order

'**n**' : network byte order

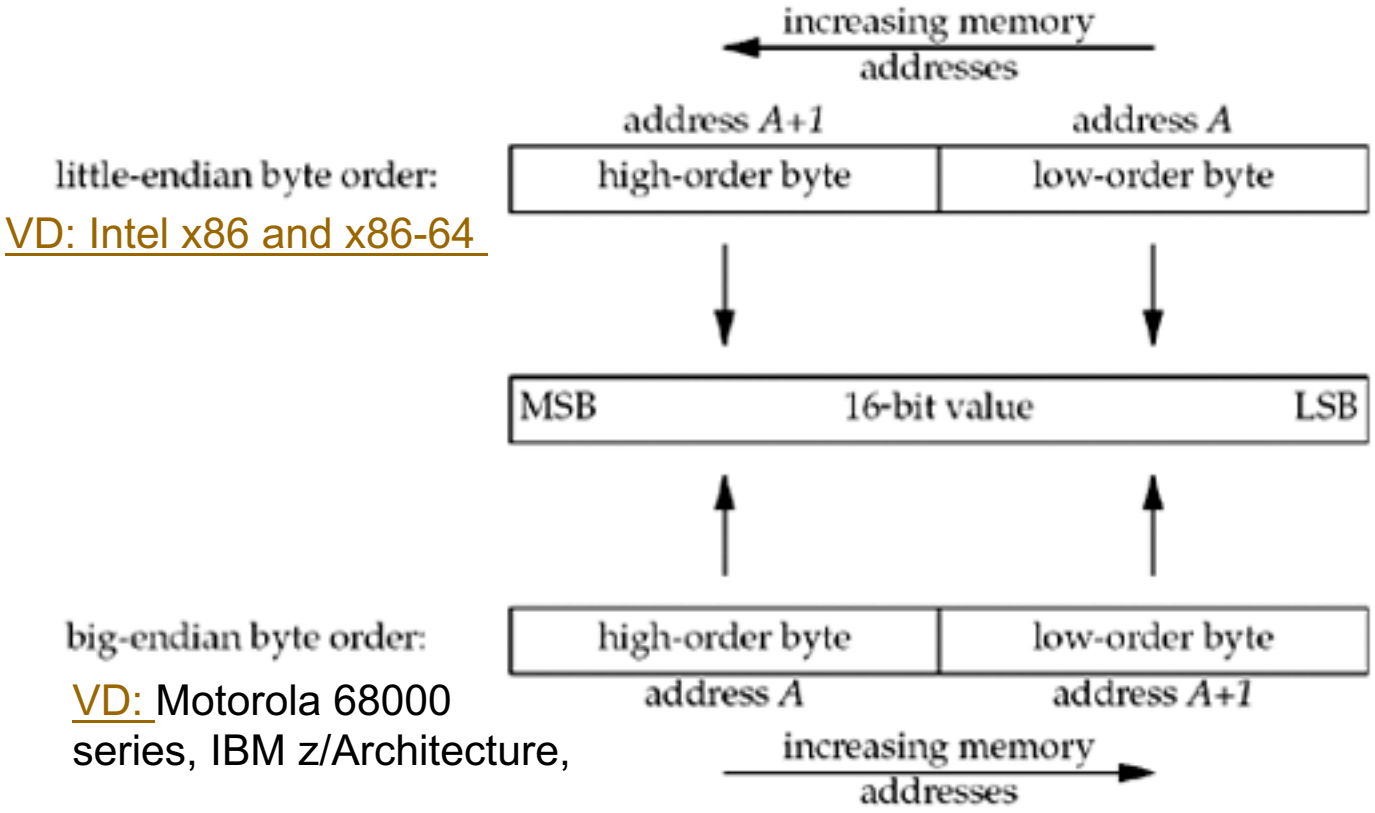
'**s**' : short (16bit)

'**l**' : long (32bit)

```
#include <netinet/in.h>

uint16_t htons(uint16_t);
uint16_t ntohs(uint_16_t);

uint32_t htonl(uint32_t);
uint32_t ntohl(uint32_t);
```

Sự cần thiết của việc chuyển đổi thứ tự byte

- Biểu diễn dữ liệu 16 bit



VD: Intel x86 and x86-64

VD: Motorola 68000 series, IBM z/Architecture,

Các hàm xử lý buffer

```
#include <strings.h>
void bzero(void *dest, size_t nbytes);
void bcopy(const void *src, void *dest, size_t nbytes);
int bcmp(const void *ptr1, const void *ptr2, size_t nbytes);
```

Returns: 0 if equal, nonzero if unequal

```
#include <string.h>
void *memset(void *dest, int c, size_t len);
void *memcpy(void *dest, const void *src, size_t nbytes);
int memcmp(const void *ptr1, const void *ptr2, size_t nbytes);
```

Returns: 0 if equal, <0 or >0 if unequal

Ví dụ về gán địa chỉ socket

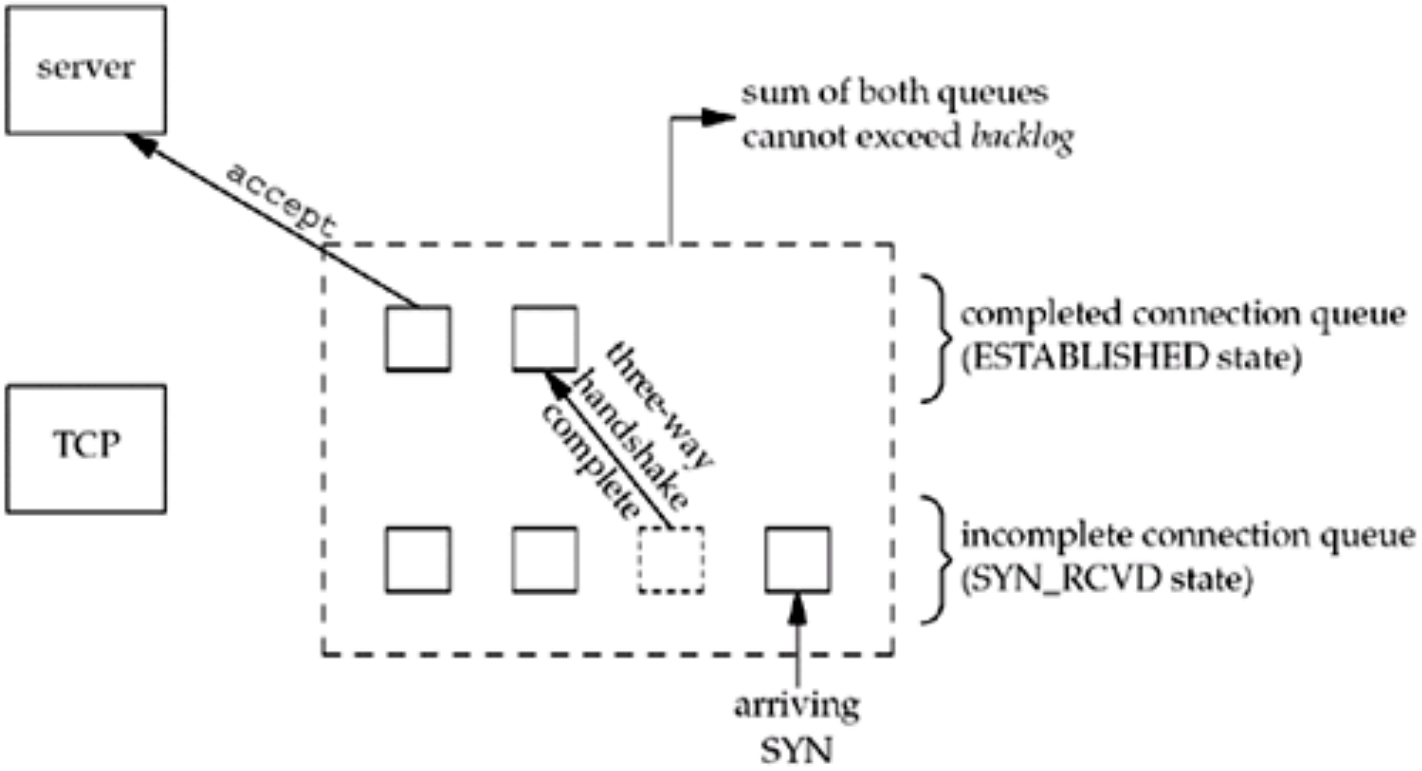
```
int sockfd;                          /* socket descriptor */
struct sockaddr_in sockaddr;         /* used by bind() */

/* create the socket */
bzero(&sockaddr, sizeof(sockaddr)); /*  */
sockaddr.sin_family = AF_INET; /* use the Internet addr family */

sockaddr.sin_port = htons(123); /* bind socket 'sockfd' to port
123*/

/* kernel choose IP address bound for the socket */
sockaddr.sin_addr.s_addr = htonl(INADDR_ANY);

if(bind(sockfd, (struct sockaddr*) &sockaddr, sizeof(sockaddr)) <
0) {
        perror("bind"); exit(1);
}
```

listen() : Chỉ định socket lắng nghe kết nối

```
#include <sys/socket.h>
int listen (int sockfd, int backlog);

                                    Returns: 0 if OK, -1 on error
```

- *sockfd* = Mô tả file của socket đã tạo
- *backlog* = Số lượng tối đa của các kết nối đang chờ
  - Cần thiết lập giá trị của *backlog* một cách thích hợp

Hai hàng đợi tại TCP socket



Ví dụ sử dụng hàm listen()

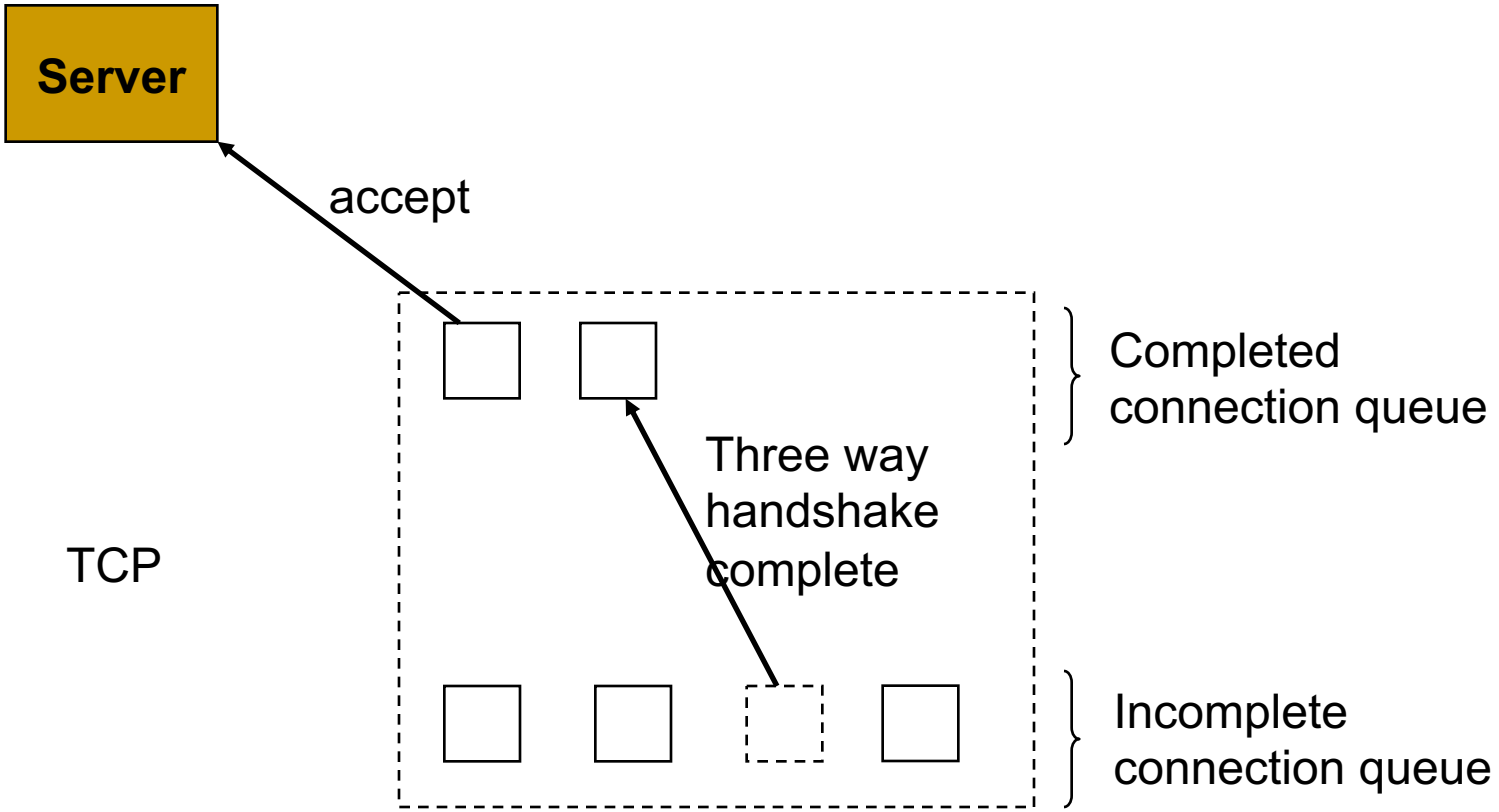
```
int sockfd;                    /* socket descriptor */
struct sockaddr_in sockaddr;     /* used by bind()
*/

/* 1) create the socket */
/* 2) bind the socket to a port */

if(listen(sockfd, 5) < 0) {
      perror("listen");
      exit(1);
}
```

accept (): Chờ/chấp nhận kết nối đến

```
#include <sys/socket.h>
int accept (int sockfd, struct sockaddr *cliaddr, socklen_t *addrlen);

                                    Returns: non-negative descriptor if OK, -1 on error
```

- sockfd = Mô tả socket
- cliaddr = Con trỏ tới cấu trúc địa chỉ socket của tiến trình kết nối đến
- addrlen = Độ lớn của cấu trúc địa chỉ
- Hàm ***accept*** gửi trả về một socket mới có các tham số giống như socket chờ (***sockfd***)
  - Máy chủ có thể sử dụng socket mới này để trao đổi dữ liệu với máy khách bằng các lệnh gọi ***read/write***

accept() (2)


Server
accept
Completed
connection queue
Three way
handshake
complete
TCP
Incomplete
connection queue

Ví dụ về hàm accept ()

```
int sockfd;                          /* socket descriptor */
struct sockaddr_in cliaddr;                /* used by accept() */
int newfd;                          /* returned by accept() */
int cliaddr_len = sizeof(cli);     /* used by accept() */

/* 1) create the socket */
/* 2) bind the socket to a port */
/* 3) listen on the socket */

newfd = accept(sockfd, (struct sockaddr*) &cliaddr, &cliaddr_len);
if(newfd < 0) {
        perror("accept");    exit(1);
}
```

- Máy chủ biết thông tin máy khách bằng cách nào?
  - **cliaddr.sin_addr.s_addr** = địa chỉ IP của máy khách
  - **cliaddr.sin_port** = cổng của máy khách

connect(): Thiết lập một kết nối với máy chủ TCP

```
#include <sys/socket.h>
int connect (int sockfd, const struct sockaddr *servaddr, socklen_t addrlen);

                                                    Returns: 0 if OK, -1 on error
```

sockfd = Mô tả file của socket đã tạo ra

servaddr = Con trỏ tới cấu trúc địa chỉ socket của máy chủ kết nối

addrlen = kích thước cấu trúc địa chỉ

Làm việc với địa chỉ IP

- Địa chỉ IP thường được viết bởi chuỗi ký tự (“128.2.35.50”), nhưng trong chương trình địa chỉ IP được biểu diễn bằng chuỗi ký tự bit.

Chuyển đổi chuỗi ký tự thành địa chỉ dạng số:

```
#include <arpa/inet.h>
int         inet_aton(const char *strptr, struct in_addr *addrptr);
                                          Returns: 1 if string was valid, 0 on error
in_addr_t     inet_addr(const char *strptr);
                        Returns: 32-bit binary network byte ordered IPv4 address;
                                    INADDR_NONE if error
```

strptr = địa chỉ IP dạng chuỗi ký tự
addrptr = địa chỉ IP dạng chuỗi byte

Chuyển đổi địa chỉ dạng số thành chuỗi ký tự:

```
char *inet_ntoa(struct in_addr inaddr);

                                          Returns: pointer to dotted-decimal string
```

Làm việc với địa chỉ IP(2)

- Làm việc với cả IPv4 and IPv6

```
#include <arpa/inet.h>
int inet_pton (int family, const char *strptr, void *addrptr);
                        Returns: 1 if OK, 0 if input not a valid presentation format,
                                 -1 on error
const char *inet_ntop (int family, const void *addrptr, char *strptr, size_t len);
                        Returns: pointer to result if OK, NULL on error
```

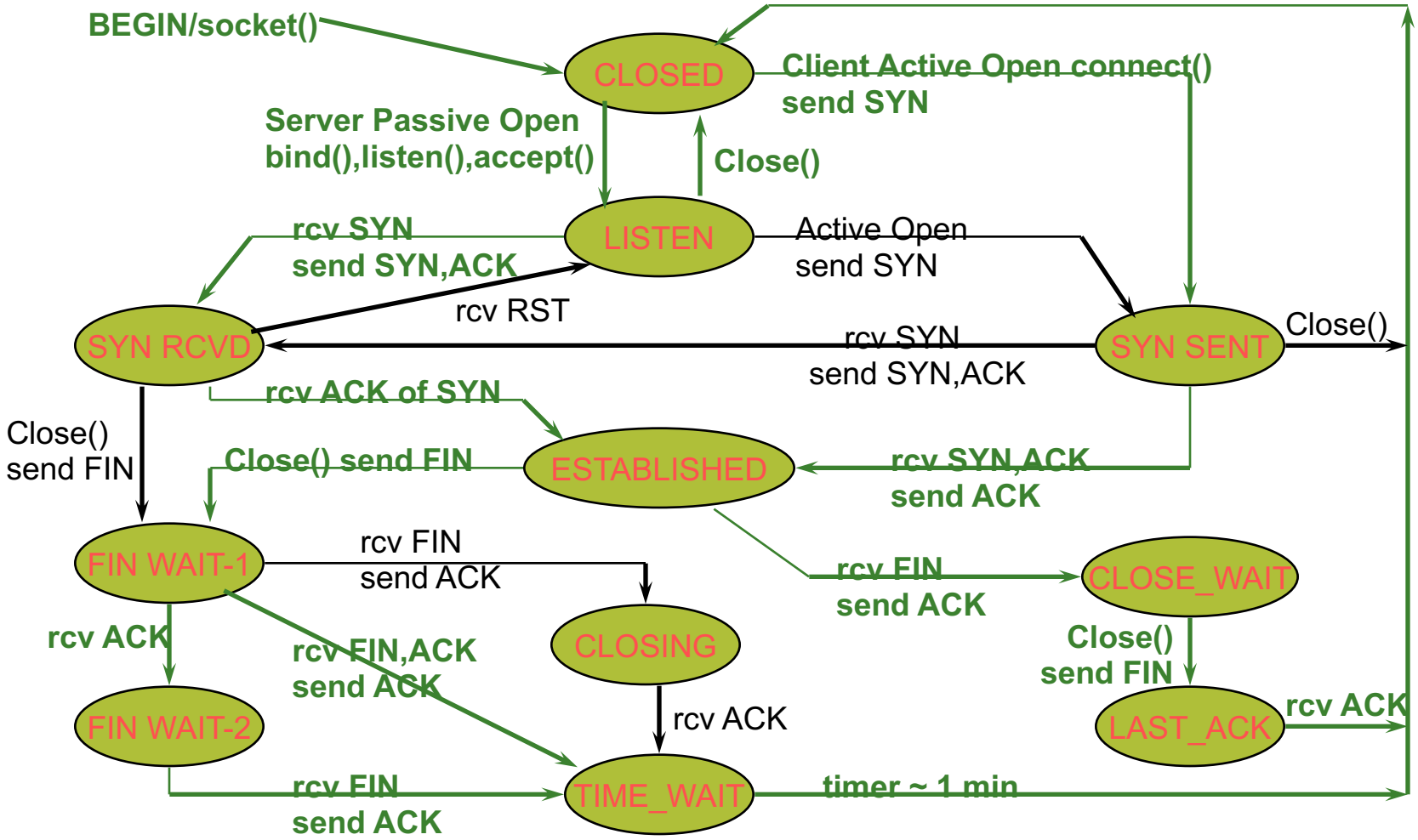
  - family = họ địa chỉ:
    - AF_INET
    - AF_INET6
  - len = kích thước buffer chuỗi ký tự địa chỉ
    - INET_ADDRSTRLEN for IPv4
    - INET6_ADDRSTRLEN for IPv6

Một ví dụ về connect()

- ***connect*** cho phép một máy khách kết nối với máy chủ …

```
int fd;                                 /* socket descriptor */
struct sockaddr_in servaddr;            /* used by connect() */

/* create the socket */

/* connect: use the Internet address family */
servaddr.sin_family = AF_INET;

/* connect: socket 'fd' to port 9876 */
servaddr.sin_port = htons(9876);

/* connect: connect to IP Address "192.168.0.1" */
servaddr.sin_addr.s_addr = inet_addr("192.168.0.1");

if(connect(fd, (struct sockaddr*) &servaddr, sizeof(servaddr)) <
0) {
        perror("connect"); exit(1);
}
```

Các trạng thái kết nối TCP



Socket I/O: read()

- *read* có thể sử dụng với socket
- *read* blocks đợi dữ liệu từ máy khách Hàm *read* không đảm bảo kích thước dữ liệu đọc được bằng kích thước dữ liệu chỉ định sizeof(buf)

```
int newfd;                          /* socket descriptor */
char buf[512];                      /* used by read() */
int nbytes;                        /* used by read() */

/* 1) create the socket */
/* 2) bind the socket to a port */
/* 3) listen on the socket */
/* 4) accept the incoming connection */

if((nbytes = read(newfd, buf, sizeof(buf))) < 0) {
        perror("read"); exit(1);
}
```

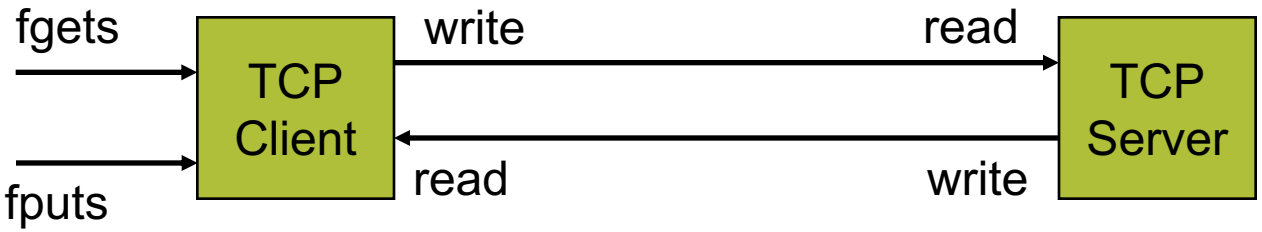
Socket I/O: write()

- Lệnh ***write*** có thể dùng với một socket

```
int fd;                              /* socket descriptor */
struct sockaddr_in srv;              /* used by connect() */
char buf[512];                       /* used by write() */
int nbytes;                          /* used by write() */

/* 1) create the socket */
/* 2) connect() to the server */

/* Example: A client could "write" a request to a server
*/
if((nbytes = write(fd, buf, sizeof(buf))) < 0) {
      perror("write");
      exit(1);
}
```

Ví dụ về máy chủ/khách TCP: echo

1. Máy khách đọc một dòng text từ bàn phím và gửi dòng text đó cho máy chủ
2. Máy khách nhận dòng text phản hồi từ máy chủ và xuất ra màn hình



Usage:

%server <listen port>

%client <server's IP address> <server's listen port number>

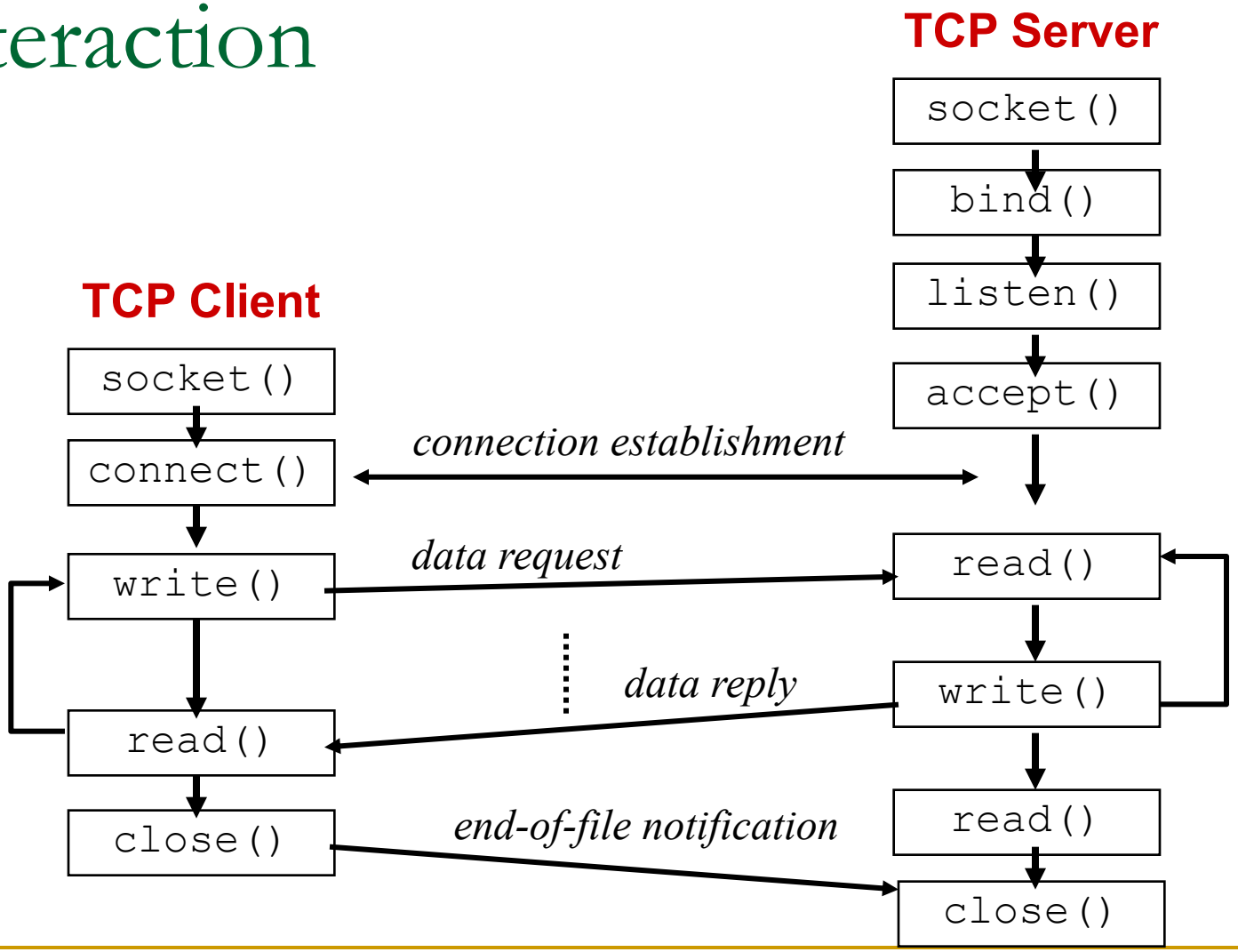
Chương trình máy chủ Echo

- tcpserv01.c
- str_echo.c

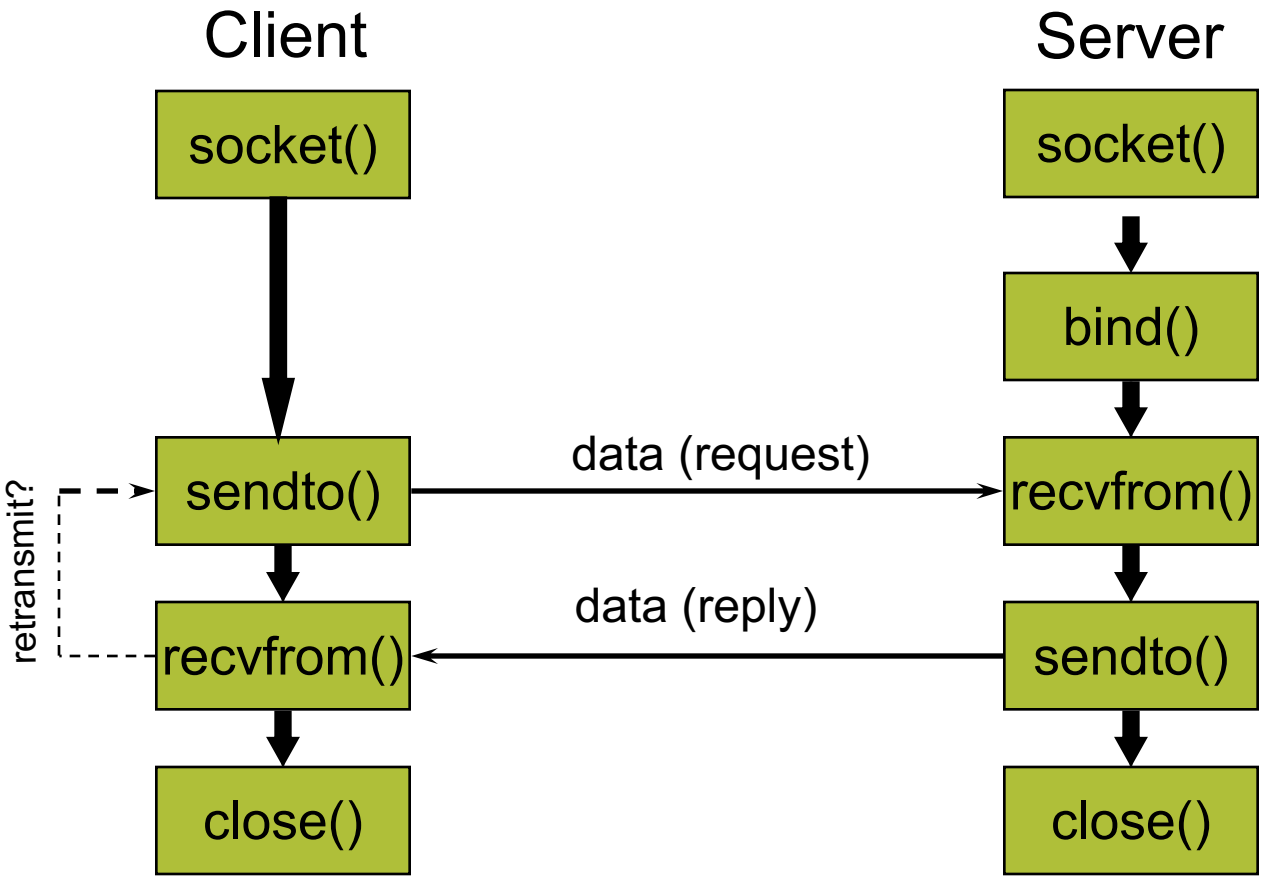
Chương trình máy khách Echo

- tcpcli01.c
- str_cli.c

Review: TCP Client-Server Interaction



Datagram (UDP) Client/Server

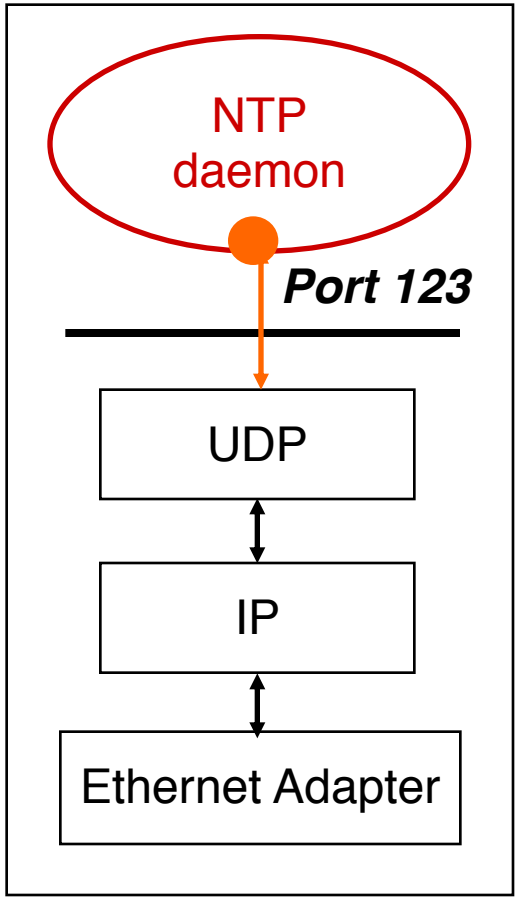


recvfrom() and sento()

```
#include <sys/socket.h>

ssize_t recvfrom(int sockfd, void *buff, size_t nbytes, int flags,
                                                    struct sockaddr *from, socklen_t *addrlen);
ssize_t sendto(int sockfd, const void *buff, size_t nbytes, int flags,
                                                    const struct sockaddr *to, socklen_t addrlen);
                                        Both return: number of bytes read or written if OK, –1 on error
```

- sockfd = Mô tả file của socket
- buff = Con trỏ trỏ đến buffer để nhận/gửi dữ liệu
- nbytes = Số lượng byte nhận/gửi
- flags = dùng với các hàm xuất nhập I/O (mặc định là 0)
- from = Con trỏ trỏ đến cấu trúc địa chỉ socket chứa thông tin của bên gửi gói tin
- to = Con trỏ trỏ đến cấu trúc địa chỉ socket chứa thông tin của bên nhận gói tin
- addrlen = kích thước của cấu trúc địa chỉ socket

Ví dụ về máy chủ UDP



- Máy chủ cung cấp thời gian mạng NTP
- **Máy chủ *UDP cần làm gì để máy khách UDP có thể kết nối đến?***

Socket I/O: socket()

- Máy chủ UDP phải tạo socket **datagram** …

```
int fd;             /* file descriptor for socket*/

if((fd = socket(AF_INET, SOCK_DGRAM, 0)) < 0) {
      perror("socket");
      exit(1);
}
```

- **SOCK_DGRAM: sử dụng giao thức UDP**

Socket I/O: bind()

- Gán cổng cho ***socket*** …

```
int fd;                          /* socket descriptor */
struct sockaddr_in servaddr; /* used by bind() */

/* create the socket */

servaddr.sin_family = AF_INET; //  use the Internet address family


/* bind: socket 'fd' to port 123*/
servaddr.sin_port = htons(123);

/* bind: a client may connect to any of my addresses */
servaddr.sin_addr.s_addr = htonl(INADDR_ANY);

if(bind(fd, (struct sockaddr*) &servaddr, sizeof(servaddr)) < 0) {
        perror("bind"); exit(1);
}
```

- **Bây giờ máy chủ UDP server đã sẵn sàng nhận packet…**

Socket I/O: recvfrom()

```
int fd;                              /* socket descriptor */
struct sockaddr_in servaddr;  /* used by bind() */
struct sockaddr_in cliaddr;   /* used by recvfrom() */
char buf[512];                       /* used by recvfrom() */
int cliaddr_len = sizeof(cliaddr);  //used by recvfrom()
int nbytes;                          /* used by recvfrom() */

/* 1) create the socket */
/* 2) bind to the socket */

nbytes = recvfrom(fd, buf, sizeof(buf), 0 /* flags */,
          (struct sockaddr*) &cliaddr, &cliaddr_len);
if(nbytes < 0) {
      perror("recvfrom"); exit(1);
}
```

Socket I/O: recvfrom() continued...

```
nbytes = recvfrom(fd, buf, sizeof(buf), 0 /* flags */,
                  (struct sockaddr*) cli, &cli_len);
```

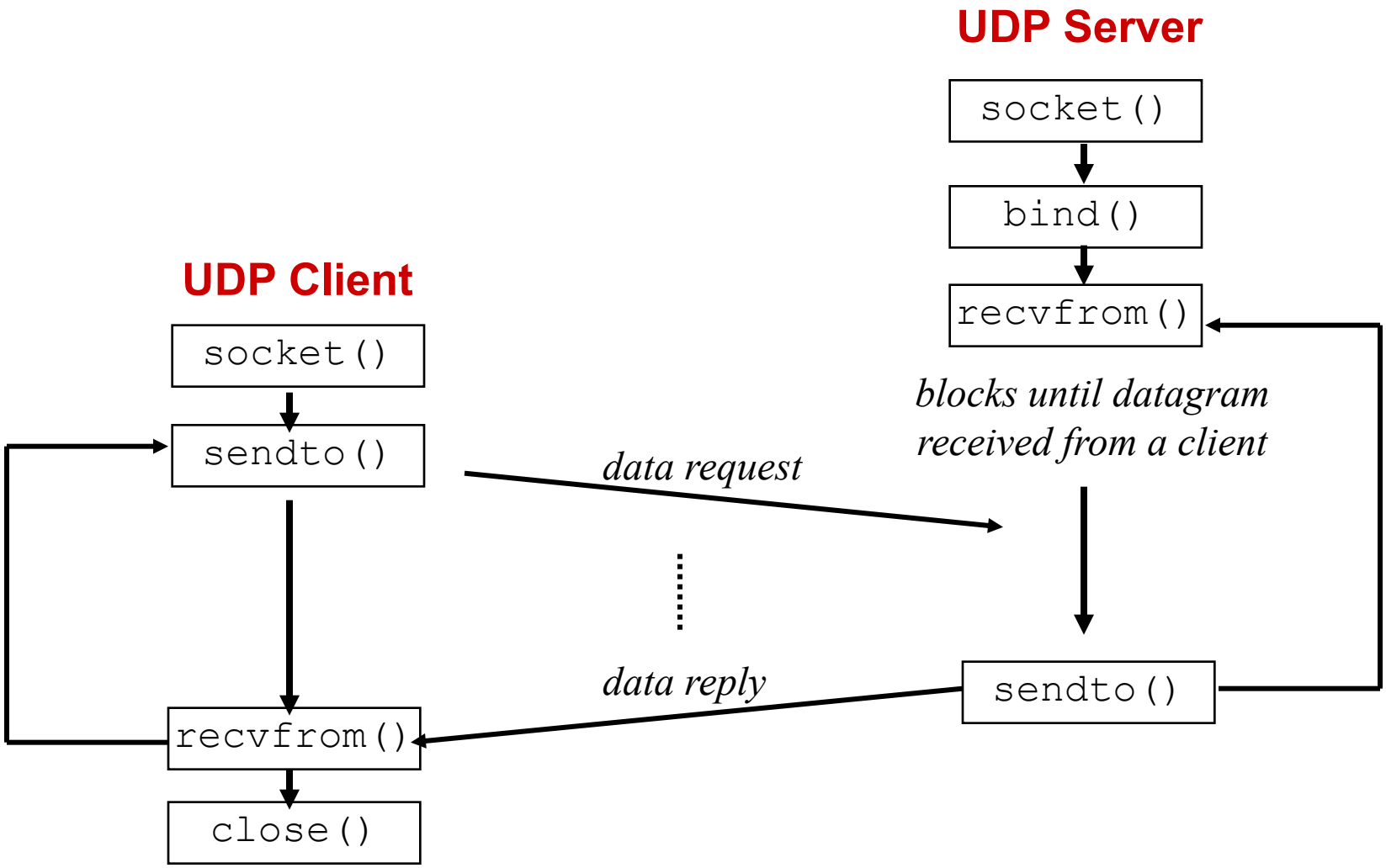
- Những thực hiện bởi lệnh ***recvfrom***
  - copy ***nbytes*** dữ liệu vào ***buf***
  - trả về số byte đã nhận (***nbytes***)
  - gán cấu trúc địa chỉ socket của máy khách mà con trỏ ***cli*** trỏ đến
  - gán kích thước địa chỉ socket của máy khách

Socket I/O: sendto()

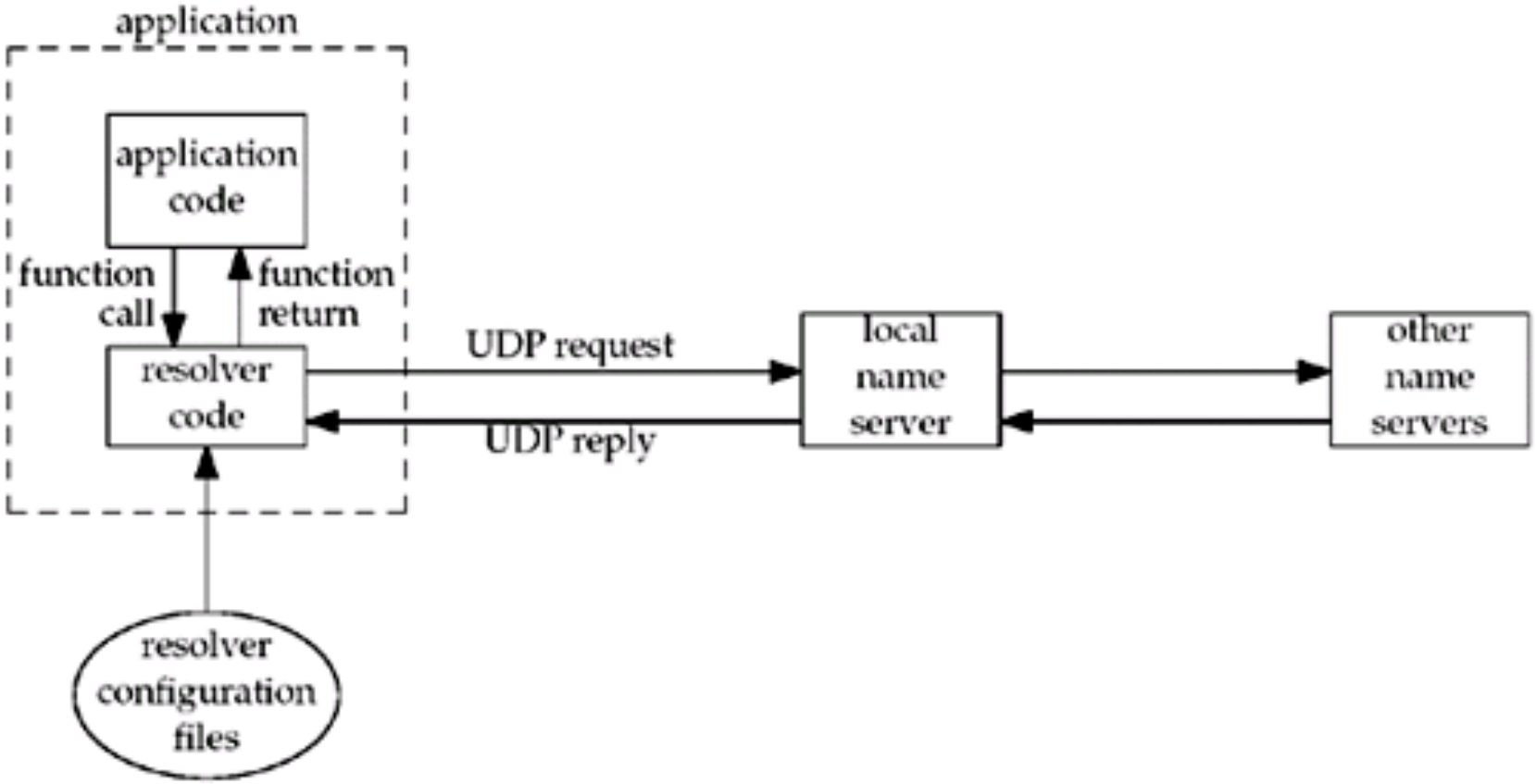
- ***Không sử dụng lệnh write***
- Máy khách UDP không gán số hiệu cổng của máy chủ cho socket
  - Số hiệu cổng được gán động khi lệnh ***sendto*** được gọi lần đầu tiên

```
int fd;                                  /* socket descriptor */
struct sockaddr_in srv;                  /* used by sendto() */

/* 1) create the socket */

/* sendto: send data to IP Address "198.0.0.50" port 123 */
srv.sin_family = AF_INET;
srv.sin_port = htons(123);
srv.sin_addr.s_addr = inet_addr("198.0.0.50");

nbytes = sendto(fd, buf, sizeof(buf), 0 /* flags */,
                  (struct sockaddr*) &srv, sizeof(srv));
if(nbytes < 0) {
       perror("sendto");    exit(1);
}
```

Review: UDP Client-Server Interaction



DNS – Chuyển đổi tên miền và địa chỉ IP



Chuyển đổi tên miền thành địa chỉ IP

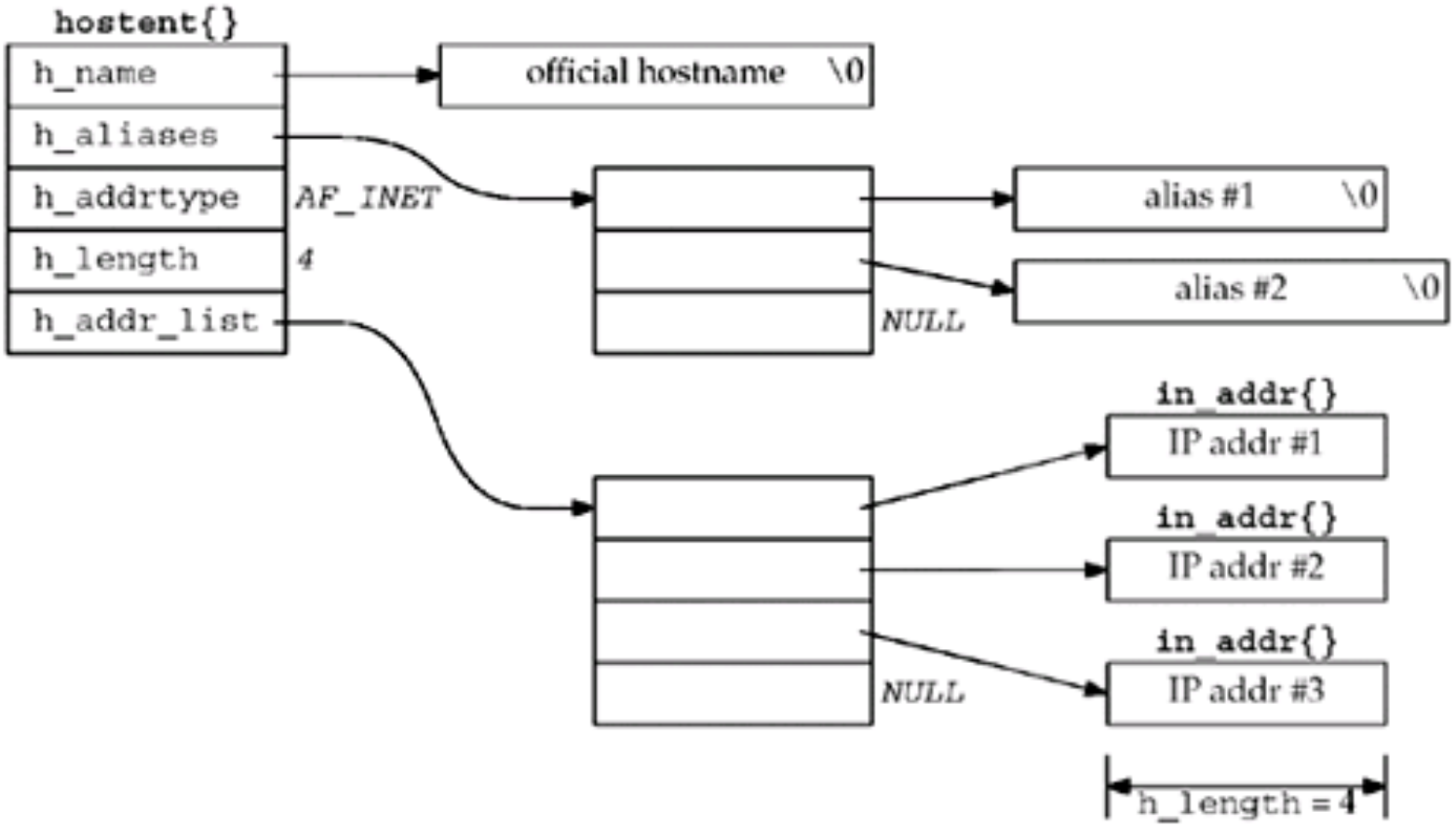
```
#include <netdb.h>
struct hostent *gethostbyname (const char *hostname);
```

Returns: non-null pointer if OK,NULL on error with h_errno set

- Hàm cơ bản để chuyển đổi địa chỉ
- Trả về con trỏ đến cấu trúc hostent
- struct hostent {

```
        char *h_name; /* Tên miền chính thức */
        char **h_aliases; /* Con trỏ trỏ tới chuỗi con trỏ tên miền alias */

         int h_addrtype; /* Họ giao thức: AF_INET */
        int h_length; /* Kích thước địa chỉ: 4 */
        char **h_addr_list; /* Con trỏ trỏ tới chuỗi con trỏ địa chỉ IPv4*/
};
```

Cấu trúc hostent



Ví dụ về cách sử dụng hàm gethostbyname

- hostent.c

Một số hàm khác

- gethostbyaddr
  - struct hostent *gethostbyaddr (const char *addr, socklen_t len, int family);
    - addr: Con trỏ trỏ tới cấu trúc in_addr chứa địa chỉ IPv4
  - trả về con trỏ trỏ tới cấu trúc `hostent` với tên miền tương ứng
- getservbyname
  - trả về mô tả dịch vụ (thường là số hiệu cổng) tương ứng với tên dịch vụ (thường được định nghĩa trong /etc/services)
  - Trả về con trỏ trỏ tới cấu trúc `servent`
    - struct servent {

```
        char *s_name; /* official service name */
        char **s_aliases; /* alias list */
        int s-port; /* port number, network-byte order */
        char *s_proto; /* protocol to use */
};
```

Ví dụ

```
struct servent *sptr;
sptr = getservbyname("domain", "udp");      /* DNS using UDP */
sptr = getservbyname("ftp", "tcp");         /* FTP using TCP */
sptr = getservbyname("ftp", NULL);          /* FTP using TCP */
sptr = getservbyname("ftp", "udp");         /* this call will fail */
```