LẬP TRÌNH MẠNG DÙNG SOCKET TRÊN SỬ DỤNG C

Bài Giảng 3

LẬP TRÌNH SOCKET TRÊN UNIX

- **Quản lý socket**
  - Unix: Integer

Primitives	Meaning
SOCKET	Create a new communication end point
BIND	Attach a local address to a socket
LISTEN	Announce willingness to accept connections; give queue size
ACCEPT	Block the caller until connection attempt arrives
CONNECT	Actively attempt to establish a connection
SEND	Send some data over the connection
RECEIVE	Receive some data from the connection
CLOSE	Release the connection

LẬP TRÌNH SOCKET TRÊN UNIX

– Cấu trúc địa chỉ Internet : định nghĩa dạng dữ liệu cấu trúc trong ngôn ngữ C. Cấu trúc này chỉ có 1 field kiểu u_long chứa địa chỉ IP 32 bit.

```
struct    in_addr
  {
            u_long    s_addr;
  };
```

in_addr

– Cấu trúc địa chỉ socket :

- địa chỉ này lưu trữ địa chỉ IP, chỉ số port, và dạng (family protocol)
- Tên cấu trúc là *sockaddr_in* được biểu diễn ở hình trong slide kế. Trong đó:
  - *sin_len:* lưu trữ chiều dài cấu trúc của sockaddr_in
  - *sin_family:* dạng protocol của socket
  - *sin_port:* chỉ số port
  - *sin_addr:* địa chỉ in Internet của socket
  - *sin_zero[8]:* không dùng, đặt giá trị = 0

LẬP TRÌNH SOCKET TRÊN UNIX

sockaddr_in

```
struct   sockaddr_in
{
        u_char          sin_len;
        u_short         sin_family;
        u_short         sin_port;
        struct  in_addr sin_addr;
        char            sin_zero[8];
};
```

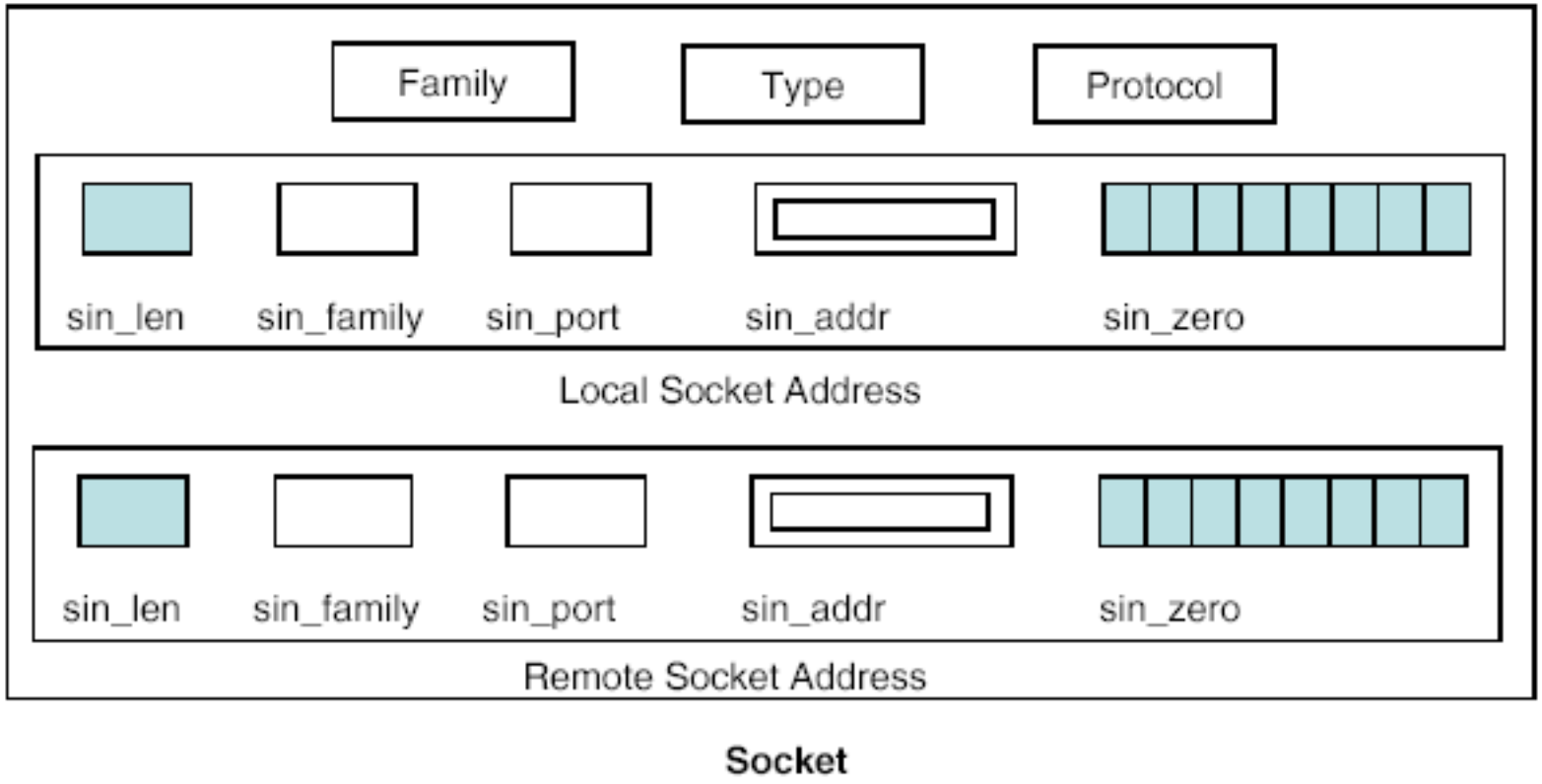
Hình - Cấu trúc địa chỉ socket

LẬP TRÌNH SOCKET TRÊN UNIX

– Cấu trúc socket :

- socket được định nghĩa trong hệ điều hành bằng một cấu trúc, được xem như điểm nối để hai procceses giao tiếp với nhau.
- Cấu trúc socket gồm 5 field được mô tả như hình trong slide kế:
  - *Family* : xác định protocol group
  - *Type* : xác loại socket, stream, datagram hay raw socket.
  - *Protocol* : là field thường gán giá trị bằng 0
  - *Local Socket Address* và *Remote Socket Address* : là địa chỉ socket của process cục bộ và từ xa.

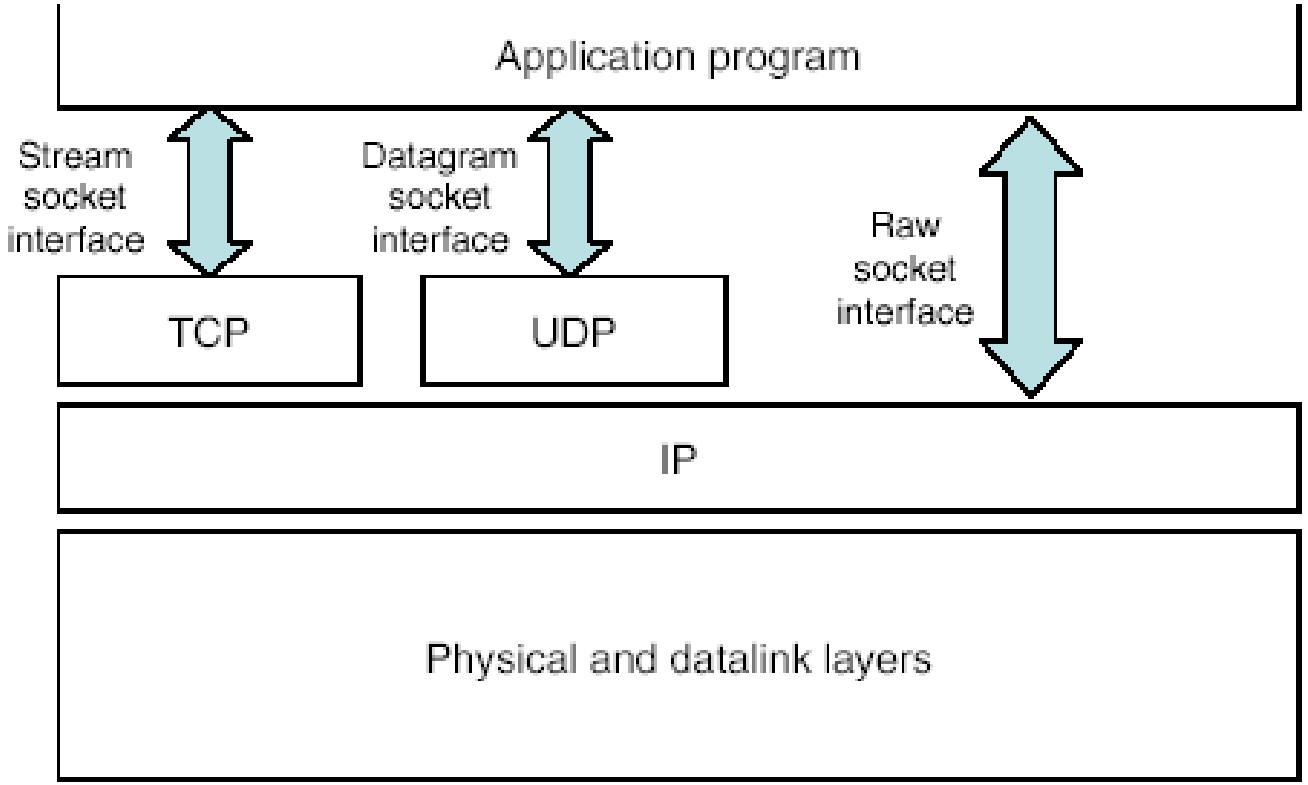
LẬP TRÌNH SOCKET TRÊN UNIX



LẬP TRÌNH SOCKET TRÊN UNIX

- Loại socket :
  - Giao tiếp socket định nghĩa 3 loại socket có thể dùng trên môi trường TCP/IP (hình ở slide kế).
  - Các loại socket gồm:
    - *Stream Socket:* dùng cho connection-oriented protocol như TCP.
    - *Datagram Socket:* dùng cho connectionless protocol như UDP.
    - *Raw Socket:* dùng cho một số protocol của một số ứng dụng đặc biệt, dùng các dịch vụ trực tiếp của lớp IP.

LẬP TRÌNH SOCKET TRÊN UNIX



Hình - Các loại socket

LẬP TRÌNH SOCKET TRÊN UNIX

- **Thông tin về các hàm dùng cho lập trình socket**

int socket(int domain, int type, int protocol);

Trong đó:

- domain (af): họ địa chỉ, thường sử dụng là AF_INET: Internet address
- type : Kiểu socket (SOCK_STREAM, SOCK_DGRAM)
- protocol : giao thức được dùng, default = 0

int bind(int sockfd, struct sockaddr *my_addr, int addrlen);

Trong đó:

- sockfd: là socket file descriptor trả về từ hàm socket
- *my_addr :* a pointer to a struct sockaddr *(chứa: address, port and IP address ...)*
- *addrlen* = sizeof(struct sockaddr).

LẬP TRÌNH SOCKET TRÊN UNIX

- **Thông tin về các hàm dùng cho lập trình socket**

int connect(int sockfd, struct sockaddr *serv_addr, int addrlen);

Trong đó:

- *sockfd* là socket file descriptor.
- *serv_addr là* struct sockaddr chứa port & IP address đích
- *addrlen* = sizeof(struct sockaddr).

int listen(int sockfd, int backlog);

Trong đó:

- *sockfd là* socket file descriptor.
- *backlog là số kết nối cho phép của hàng đợi.* Các yêu cầu connect của đối tác sẽ được lưu trong queue cho tới khi được accept

LẬP TRÌNH SOCKET TRÊN UNIX

- **Thông tin về các hàm dùng cho lập trình socket**

int accept(int sockfd, void *addr, int *addrlen);

Trong đó:

- *sockfd là* socket file descriptor.
- *addr là* pointer trỏ tới sockaddr_in. Xác định ai kết nối tới, kết nối từ port nào.
- *addrlen là biến int* = sizeof(struct sockaddr_in)

int send(int sockfd, const void *msg, int len, int flags);

int recv(int sockfd, void *buf, int len, unsigned int flags);

int read(int sockfd, const void *buf, int len);

int write(int sockfd, const void *buf, int len);

LẬP TRÌNH SOCKET TRÊN UNIX

- **Thông tin về các hàm dùng cho lập trình socket**

int sendto(int sockfd, const void *msg, int len, unsigned int flags,
const struct sockaddr *to, int tolen);

tolen có giá trị bằng sizeof(struct sockaddr).

int recvfrom(int sockfd, void *buf, int len, unsigned int flags,
struct sockaddr *from, int *fromlen);

fromlen khởi tạo bằng sizeof(struct sockaddr).

LẬP TRÌNH SOCKET TRÊN UNIX

- **Thông tin về các hàm dùng cho lập trình socket**

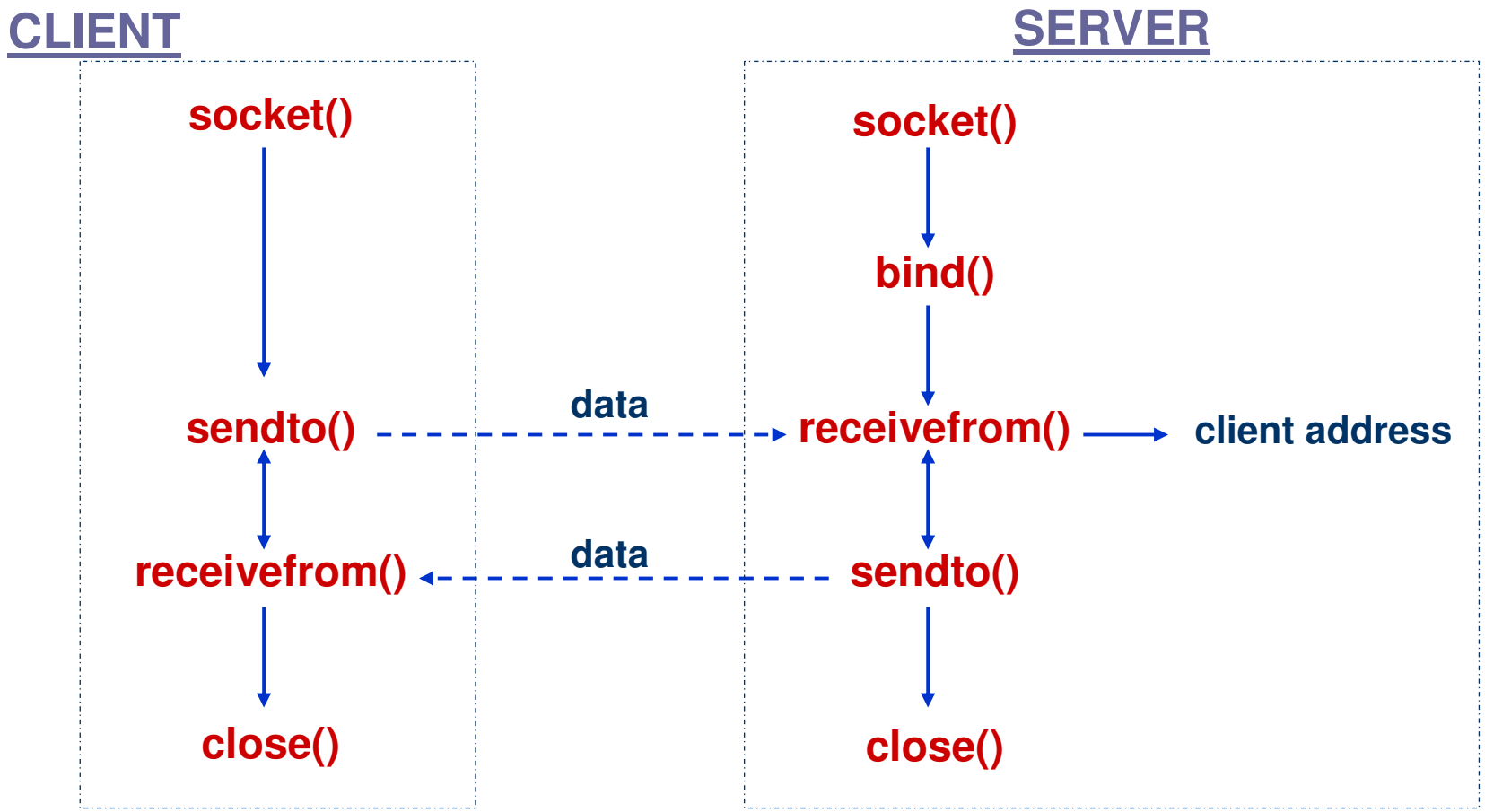
– Một số hàm dùng cho việc chuyển đổi

```
u_short     htons(u_short     host_short);
u_short     ntohs(u_short     network_short);
u_long      htonl(u_long      host_long);
u_long      ntohl(u_long      network_long);
char        *inet_ntoa(struct  in_addr  inaddr)
int         inet_aton(const  char *strptr,
                      struct  in_addr *addptr)
```

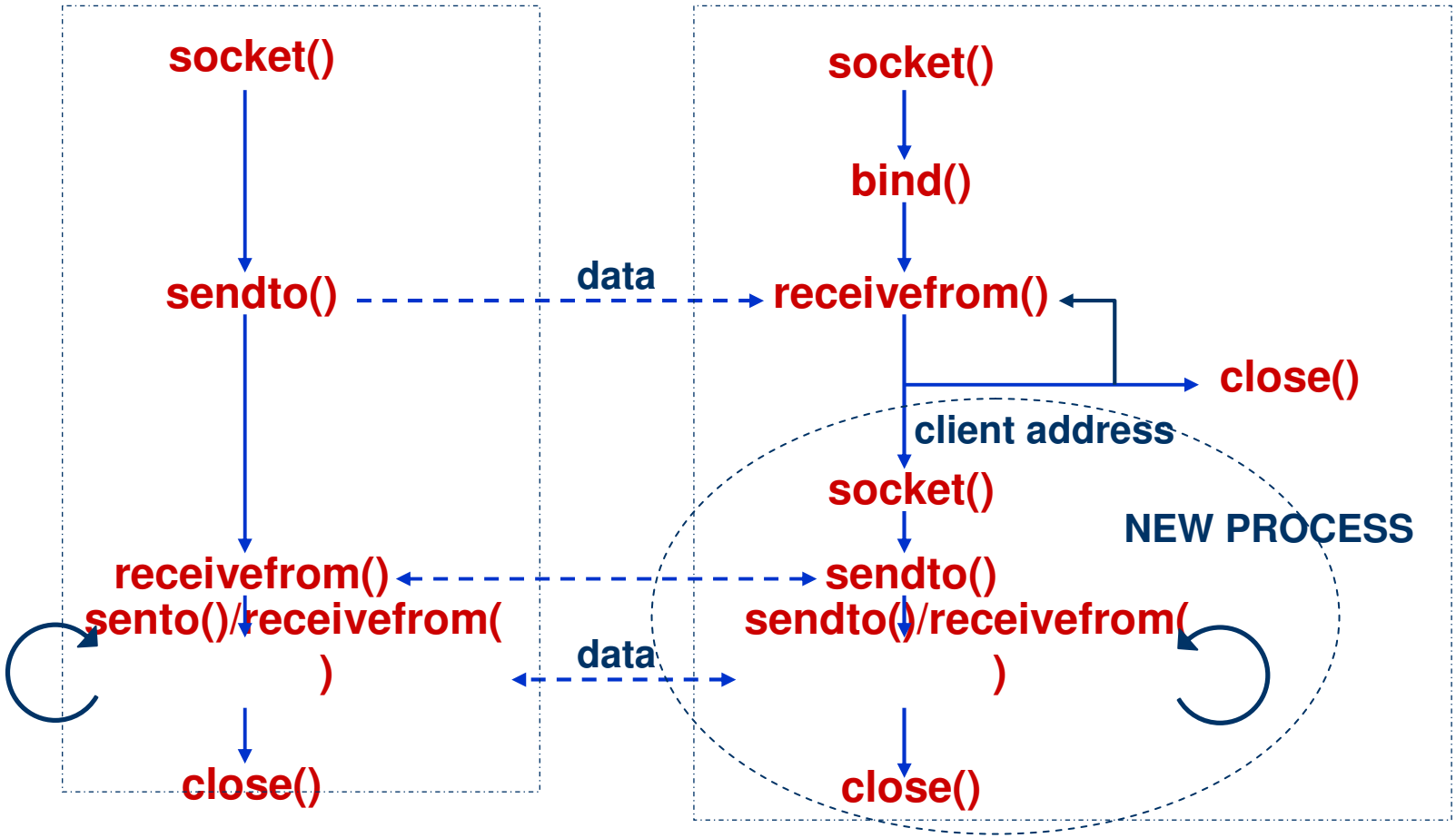
– Một số hàm dùng cho việc thao tác dữ liệu

```
void   *memset ( void *dest, int chr, int len);
void   *memcopy ( void *dest, void *src, int len)
int     memcmp ( const void *first,
                 const void *second, int len)
```

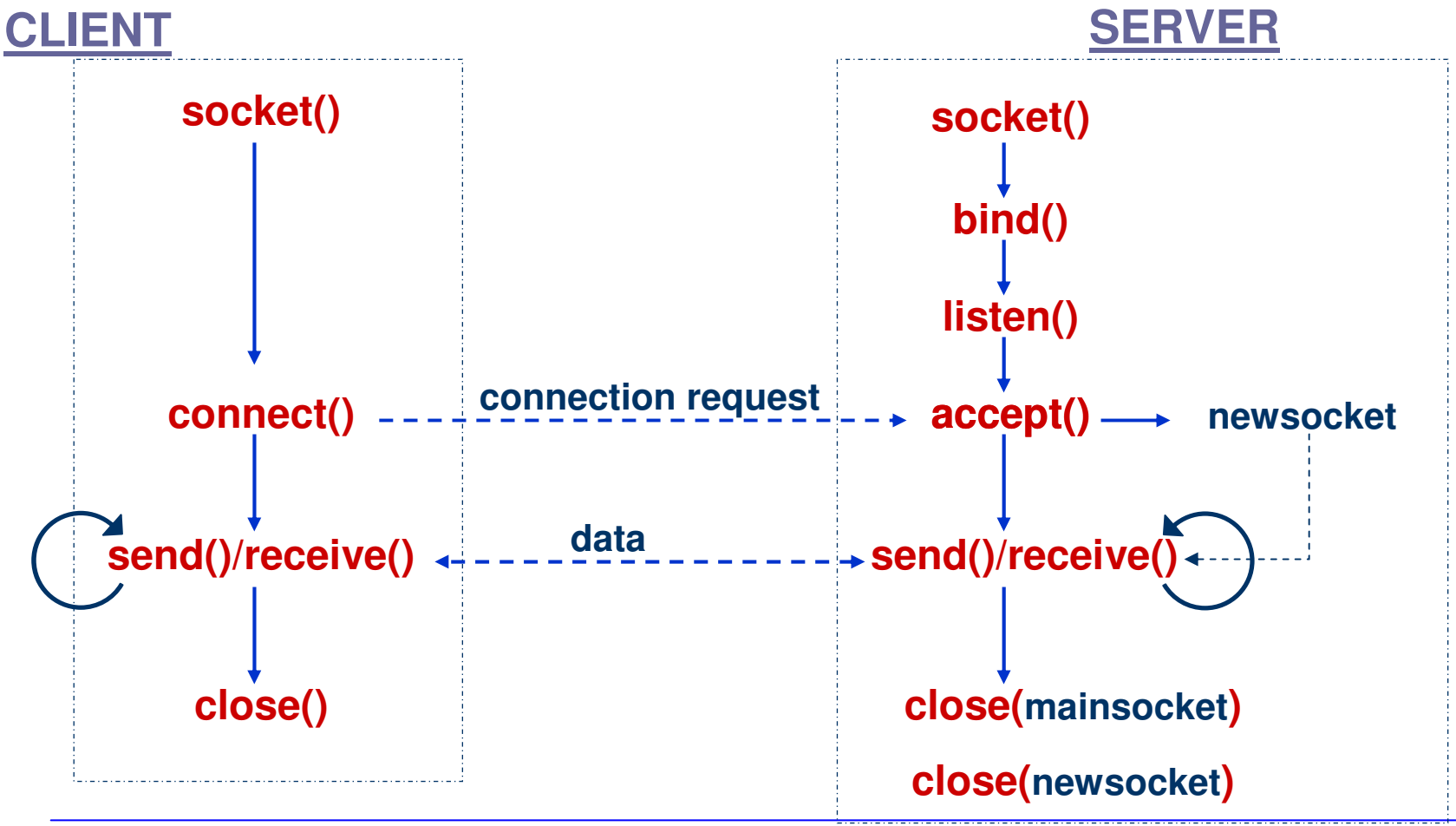
LẬP TRÌNH SOCKET VỚI UDP



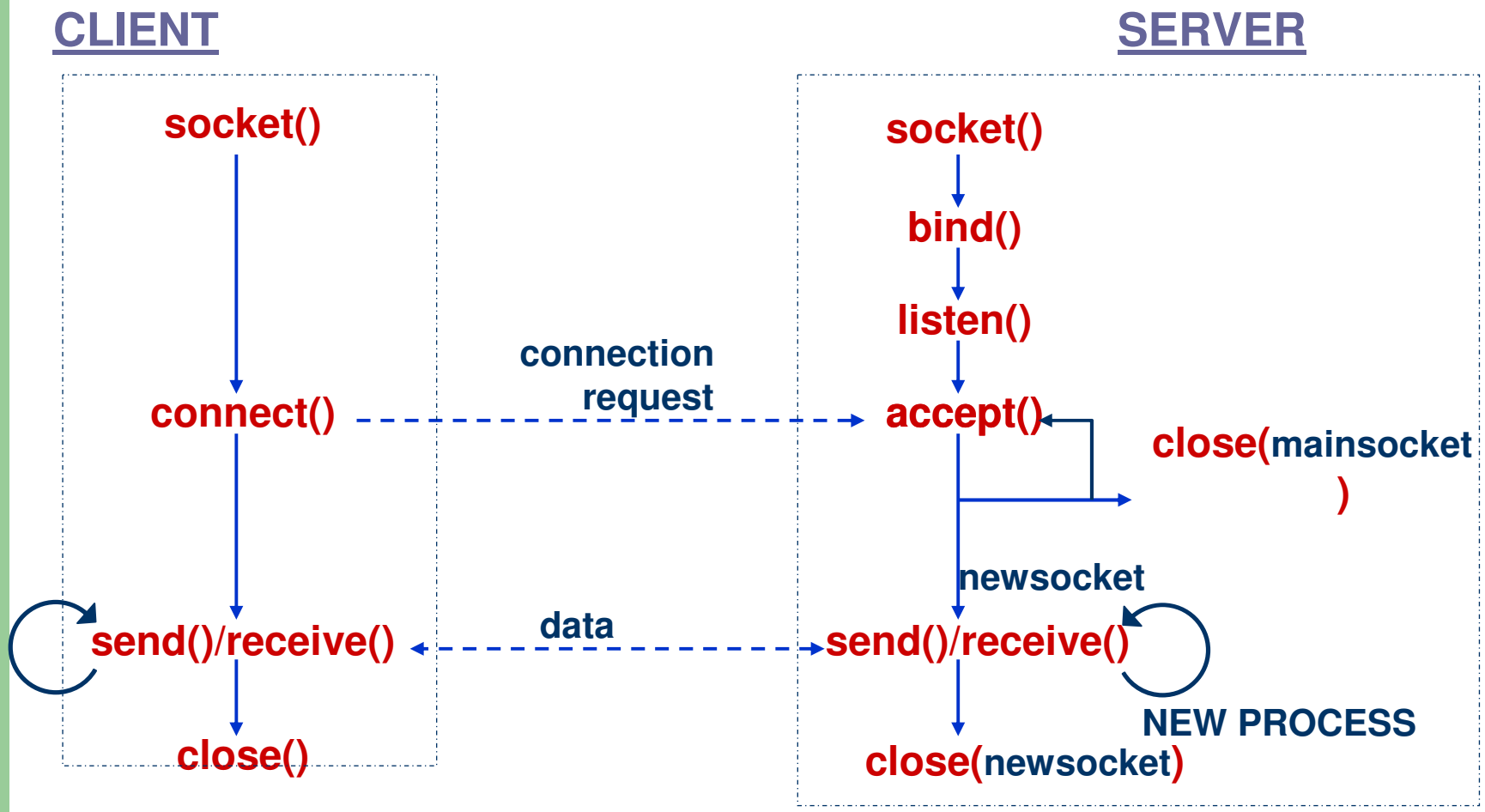
LẬP TRÌNH SOCKET VỚI UDP



LẬP TRÌNH SOCKET VỚI TCP



LẬP TRÌNH SOCKET VỚI TCP



LẬP TRÌNH SOCKET VỚI TCP

- **DateTime Client**

```
#include <sys/types.h>
#include <sys/socket.h>
int main(int argc, char **argv) {
        int  sockfd, n;
        char recvline[MAXLINE + 1];
        struct sockaddr_in servaddr;
    if( argc != 2 )
      printf("Usage : gettime <IP address>"); exit(1);
    /* Create a TCP socket */
    if ( (sockfd = socket (AF_INET, SOCK_STREAM, 0)) < 0)
    {
      perror("socket");
      exit(2);
     }
```

LẬP TRÌNH SOCKET VỚI TCP

- **DateTime Client (tiếp theo)**

```
/* Specify server's IP address and port */
   bzero (&servaddr, sizeof(servaddr));
   servaddr.sin_family = AF_INET;
   servaddr.sin_port = htons ( 13 );
   if (inet_pton (AF_INET, "127.0.0.1",
   &servaddr.sin_addr) <= 0) {
       perror("inet_pton"); exit(3);
   }

/* Connect to the server */
   if ( connect(  sockfd, (struct sockaddr *) &servaddr,
   sizeof(servaddr)) < 0 ) {
       perror("connect"); exit(4);
   }
```

LẬP TRÌNH SOCKET VỚI TCP

- **DateTime Client (tiếp theo)**

```
/* Read the date/time from socket */
   while ( (n = read ( sockfd, recvline, MAXLINE)) > 0) {
      recvline[n] = '\0';          /* null terminate */
       printf("%s", recvline);
   }
   if (n < 0) {
      perror("read"); exit(5);
   }
   close ( sockfd );
}
```

LẬP TRÌNH SOCKET VỚI TCP

- **DateTime Server**

```
#include <sys/types.h>
#include <sys/socket.h>
int main (int argc, char **argv) {
    int   listenfd, connfd;
    struct sockaddr_in servaddr, cliaddr;
    char buff[MAXLINE];
    time_t ticks;
    /* Create a TCP socket */
    listenfd = socket (AF_INET, SOCK_STREAM, 0);
    /* Initialize server's address and well-known port */
    bzero (&servaddr, sizeof(servaddr));
    servaddr.sin_family      = AF_INET;
    servaddr.sin_addr.s_addr = htonl (INADDR_ANY);
    servaddr.sin_port        = htons (13);
```

LẬP TRÌNH SOCKET VỚI TCP

- **DateTime Server (tiếp theo)**

```
/* Bind server's address and port to the socket */
bind (listenfd, (struct sockaddr*) &servaddr, sizeof( servaddr));
/* Convert socket to a listening socket */
listen (listenfd, 100);
for ( ; ; ) {
/* Wait for client connections and accept them */
    clilen = sizeof(cliaddr);
    connfd = accept( listenfd, (struct sockaddr *)&cliaddr,
    &clilen);
    ticks = time(NULL);
     snprintf( buff, sizeof(buff),"%.24s\r\n",ctime(&ticks));
    /* Write to socket */
     write( connfd, buff, strlen(buff) );
    /* Close the connection */
     close( connfd );
  }
}
```

BÀI TẬP

- Viết chương trình *nslookup* bằng C trên Unix/Linux
- Viết *echo Client/Server* bằng C trên Unix/Linux
- Viết một Web Server có những đặc điểm sau:
  - Hỗ trợ phương thức GET (GET xxx.html HTTP/1.0)
  - HTTP
  - Đáp ứng của Server có header như ExServer/b1.0
  - Hỗ trợ multithread
  - Ví dụ
    - ***Browser Request***:
      GET /intro.html HTTP/1.0 *WebServer Response*
    - ***Server Reponse***
      case 1: HTTP/1.0 200 OK
      case 2: HTTP/1.0 404 File Not Found
      case 3: HTTP/1.0 501 Not Implemented

TỔNG KẾT

- Khái niệm socket
- Thiết kế giải thuật cho client và server
- Lập trình mạng trên Java
- Lập trình socket trên UNIX