Network IPC: Sockets

华南理工大学软件学院
朱金辉

Abstract

1. Socket Address Structure
2. Byte Ordering Function
3. Address lookup Function
4. Address conversion Function
5. TCP Socket
6. Examples in textbook

1. Socket Address Structure

- Socket address = IP address + TCP or UDP port number
- Used in a socket function as a argument (as pointer).
- Each protocol define its own Socket Address Structure(IPv4, IPv6....)
- Begin with sockaddr_

  example: ipv4 - sockaddr_in

  ipv6 - sockaddr_in6

Generic Socket address structure

- Socket address structure are always passed by reference when passed as an argument to any of the socket function.
  - int bind(int , struct sockaddr * , socklen_t);

- <sys/socket.h> : Generic Socket address structure

```
struct sockaddr {
    sa_family_t   sa_family;
      /*address family: AF_xxx   value*/
    char   sa_data[14];   /*protocol specific address*/
}
```

Generic Socket address structure

- int bind(int , struct sockaddr * , socklen_t);
- Example:

```
struct sockaddr_in  serv;  /*IPv4 socket address structure*/
 /*  fill in serv{} */
bind(sockfd, (struct sockaddr *) &serv, sizeof(serv));
```

- Any calls to socket function must cast the pointer to the protocol-specific socket address structure to be a pointer to a generic socket address structure

IPv4 Socket Address Structure: sockaddr_in

/usr/include/netinet/in.h in: internet

```
struct sockaddr_in
{
  __SOCKADDR_COMMON (sin_);
  in_port_t sin_port;            /* Port number.  */
  struct in_addr sin_addr;       /* Internet address.  */

  /* Pad to size of `struct sockaddr'.  */
  unsigned char sin_zero[sizeof (struct sockaddr) -
              __SOCKADDR_COMMON_SIZE -
              sizeof (in_port_t) -
              sizeof (struct in_addr)];
};
```

```
#define __SOCKADDR_COMMON(sa_prefix) \
sa_family_t sa_prefix##family
```

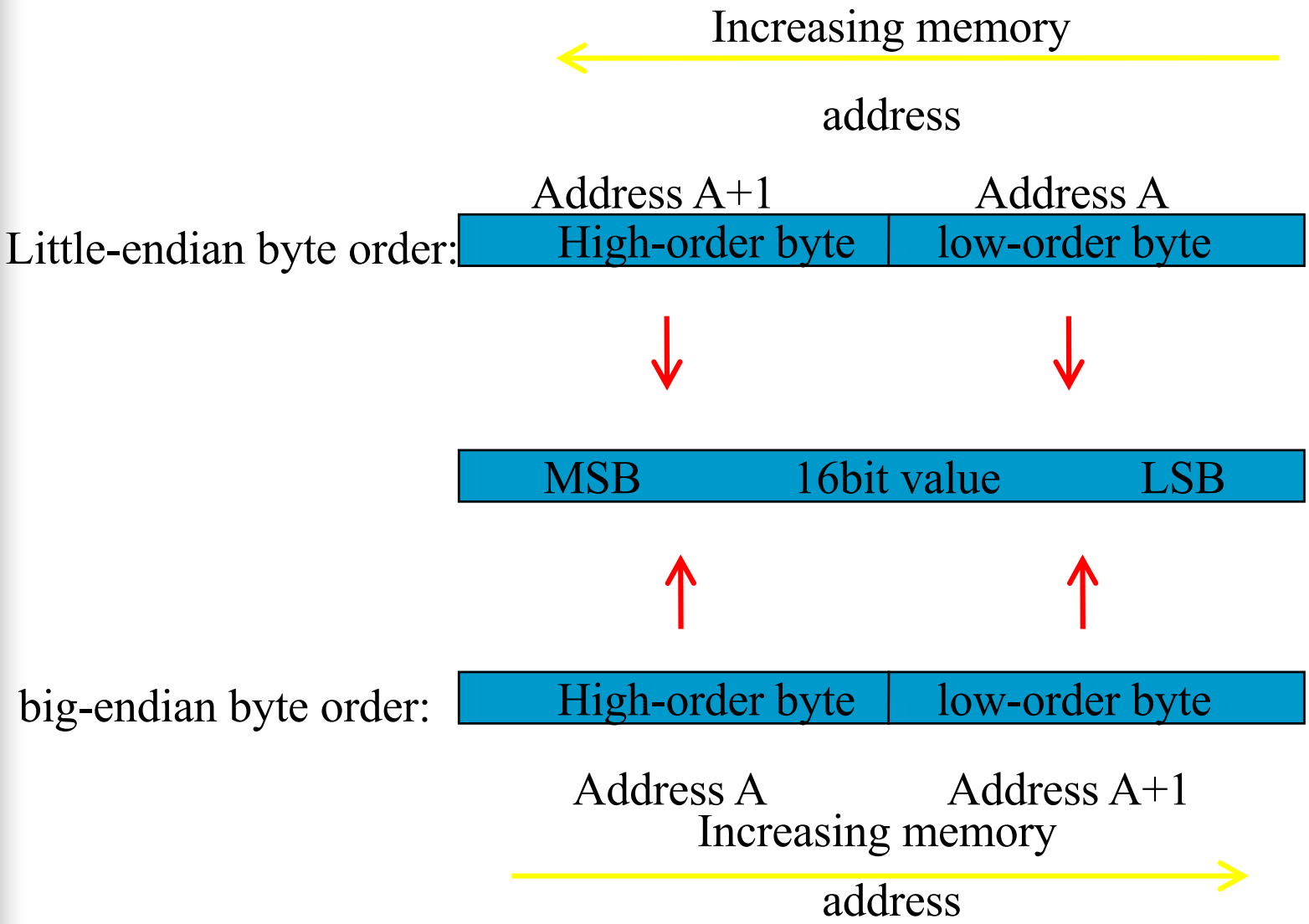
```
#define __SOCKADDR_COMMON_SIZE  (sizeof (unsigned short int))
```

sockaddr_in is expanded to:

```
struct sockaddr_in
{
  sa_family_t sin_family;
  in_port_t sin_port;            /* Port number.  */
  struct in_addr sin_addr;       /* Internet address.  */

  /* Pad to size of `struct sockaddr'.  */
  unsigned char sin_zero[sizeof (struct sockaddr) -
              (sizeof (unsigned short int) -
              sizeof (in_port_t) -
              sizeof (struct in_addr)];
};
```

- struct in_addr
  { in_addr_t s_addr; };

2. Byte Ordering Function



determine host byte order

```
#include "unp.h"
int  main(int argc, char **argv)
{
    union {  short  s;
             char   c[sizeof(short)];
    } un;
    un.s = 0x0102;
    printf("%s: ", CPU_VENDOR_OS);
    if (sizeof(short) == 2) {
        if (un.c[0] == 1 && un.c[1] == 2)       printf("big-endian\n");
          else if (un.c[0] == 2 && un.c[1] == 1)   printf("little-endian\n");
         else  printf("unknown\n");
    } else
        printf("sizeof(short) = %d\n", sizeof(short));
    exit(0);
}
```

Byte Conversion Function

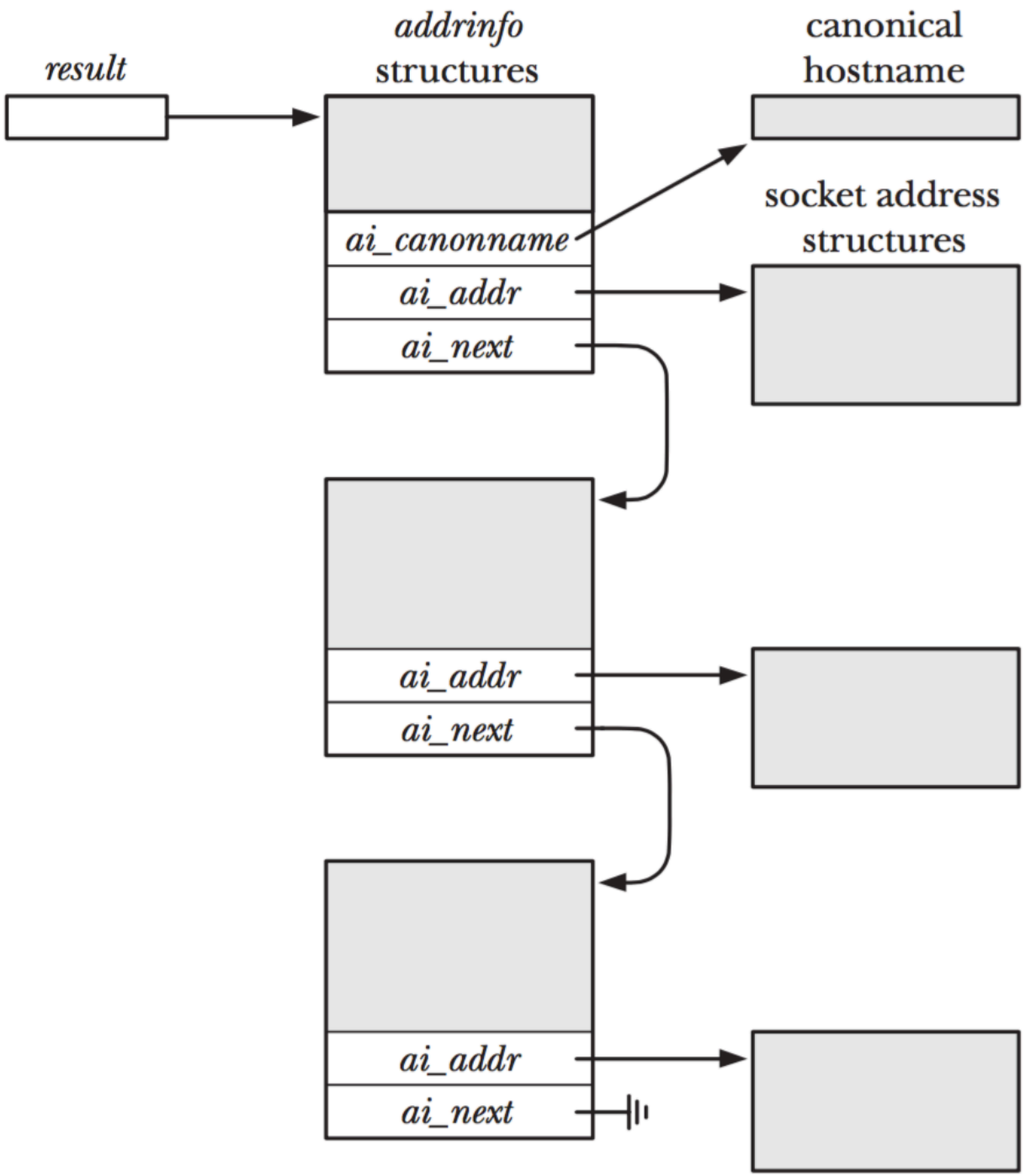
```
#include<netinet/in.h>
uint16_t    htons  (uint16_t    host16bitvalue);
uint32_t    htonl  (uint32_t    host32bitvalue);

uint16_t    ntohs  (uint16_t    net16bitvalue);
uint32_t    ntohl  (uint32_t    net32bitvalue);
```

- h: host
- n: network
- s: short(16bit)
- l: long(32bit)

3. Address Lookup

- #include <sys/socket.h>
- #include <netdb.h>
- int getaddrinfo(const char *restrict *host*,
  const char *restrict *service*,
  const struct addrinfo *restrict *hint*,
  struct addrinfo **restrict *res*);
- Returns: 0 if OK, nonzero error code on error

- void freeaddrinfo(struct addrinfo **ai*);


result
addrinfo structures
canonical hostname
socket address structures
ai_canonname
ai_addr
ai_next
ai_addr
ai_next
ai_addr
ai_next

4. Address conversion function

- Convert internet address between ASCII string and network byte ordered binary values(big endian)
- For example:

  Socket address structure

  <==>”203.255.74.129”

IPV4 Address conversion function

- #include<arpa/inet.h>

  int inet_aton(const char *strptr, struct in_addr *addrptr);

  /* return : 1 if string was valid,0 on error */

  Deprecated

  in_addr_t inet_addr(const char *strptr);

  /* return : 32bit binary network byte ordered IPv4 address;

  INADDR_NONE if error */

  char *inet_ntoa(struct in_addr inaddr); /*return pointer to dotted-decimal string*/

IPV4/IPV6 Address conversion function

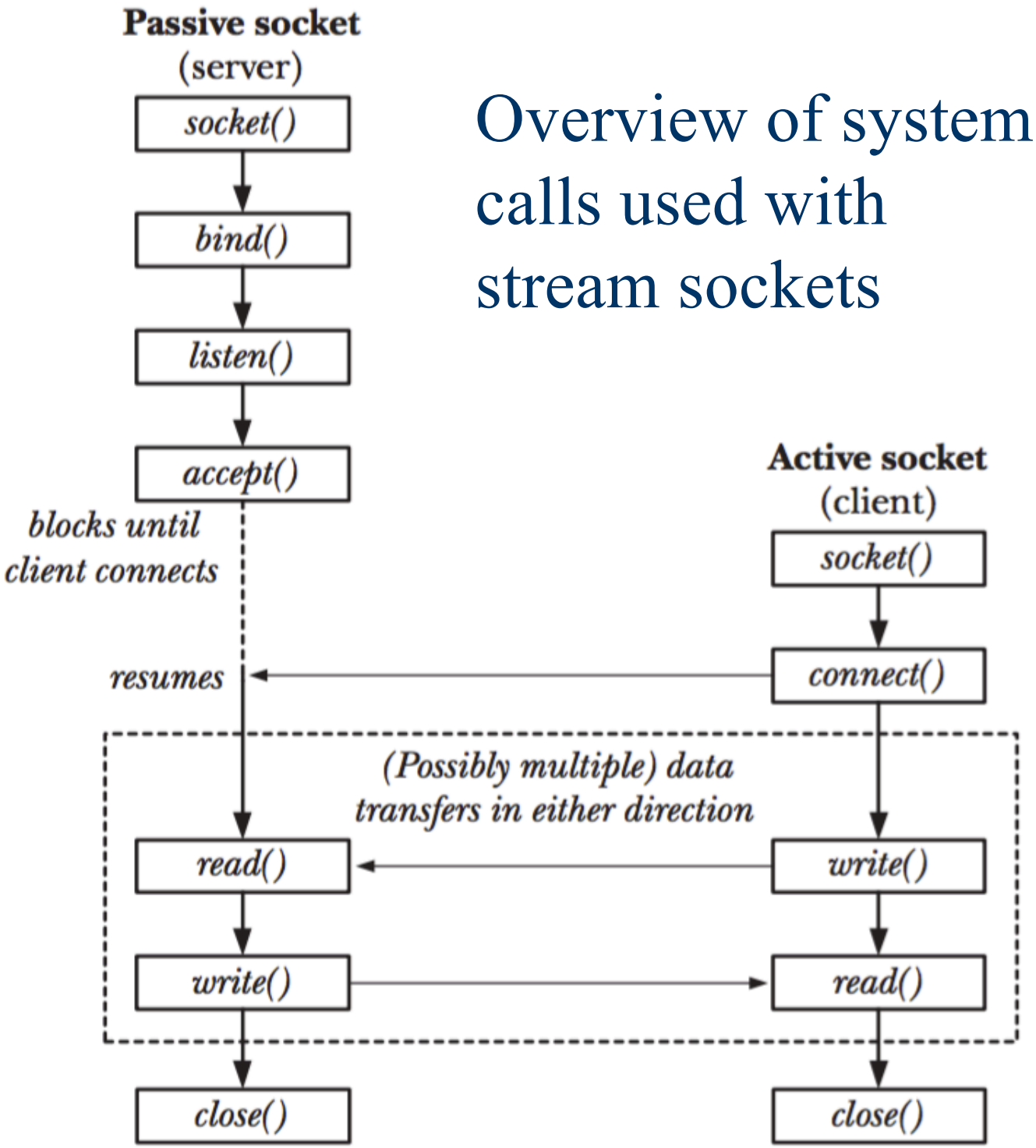
- p : presentation(string)
- n : numeric(binary)

```
#include<arpa/inet.h>
 int inet_pton(int family, const char *strptr, void *addrptr);
/* return: 1 if OK, 0 if input not a valid presentation format, -1 onerror
 */
/* string  TO  binary */

 const char *inet_ntop(int family, const void *addrptr, char *strpt,
 size_t len);
/* return : pointer to result if OK, NULL onerror */
/* len : size of the destination */
/* binary TO string */
```

5. Elementary TCP Socket

- socket function
- connect function
- bind function
- listen function
- accept function
- close function

Overview of system calls used with stream sockets



create Socket

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

returns:nonnegative descriptor if OK, -1 on error

Normaly the protocol argument to the socket function is set to 0 except for raw socket.

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

Domain	Description
AF_INET	IPv4 Internet domain
AF_INET6	IPv6 Internet domain (optional in POSIX.1)
AF_UNIX	UNIX domain
AF_UNSPEC	unspecified

Figure 16.1 Socket communication domains

Type	Description
SOCK_DGRAM	fixed-length, connectionless, unreliable messages
SOCK_RAW	datagram interface to IP (optional in POSIX.1)
SOCK_SEQPACKET	fixed-length, sequenced, reliable, connection-oriented messages
SOCK_STREAM	sequenced, reliable, bidirectional, connection-oriented byte streams

Figure 16.2 Socket types

Protocol	Description
IPPROTO_IP	IPv4 Internet Protocol
IPPROTO_IPV6	IPv6 Internet Protocol (optional in POSIX.1)
IPPROTO_ICMP	Internet Control Message Protocol
IPPROTO_RAW	Raw IP packets protocol (optional in POSIX.1)
IPPROTO_TCP	Transmission Control Protocol
IPPROTO_UDP	User Datagram Protocol

Figure 16.3 Protocols defined for Internet domain sockets

connect function

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

- Returns:0 if OK, -1 on error

bind function

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

Returns: 0 if OK, -1 on error

==>this function assigns a local protocol address to a socket.

Process specifies		Result
IP address	port	
wildcard	0	kernel chooses IP address and port
wildcard	nonzero	kernel chooses IP address, process specifies port
local IP address	0	process specifies IP address, kernel chooses port
local IP address	nonzero	process specifies IP address and port

Figure 4.5 Result when specifying IP address and/or port number to `bind`.

listen function

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

Returns:0 if OK, -1 on error

- backlog =>specify the maximum number of connections that the kernel should queue for this socket.
- If the queues are full when client SYN arrives, TCP server ignore the SYN.

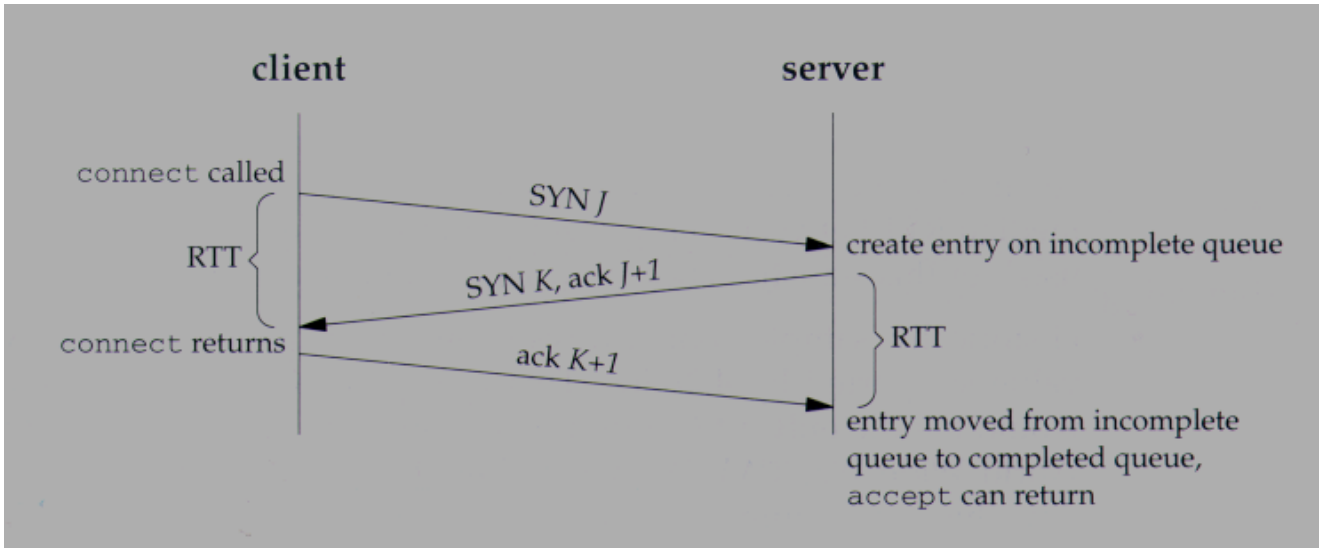


Figure 4.7 TCP three-way handshake and the two queues for a listening socket.

accept function

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

Returns:nonnegative descriptor if OK, -1 on error

⇒return the next completed connection from the front of the completed connection queue.

If queue is empty, the process is put to sleep.

close function

```
#include <unistd.h>
int close(int  sockfd);
```

returns:0 if OK, -1 on error

6. Examples in textbook

- **Connection-Oriented Client**

  Fig 16.16
- **Connection-Oriented Server**

  Fig 16.17
- **Alternative Connection-Oriented Server**

  Fig 16.18
- **Connectionless Client**

  Fig 16.19
- **Connectionless Server**

  Fig 16.20