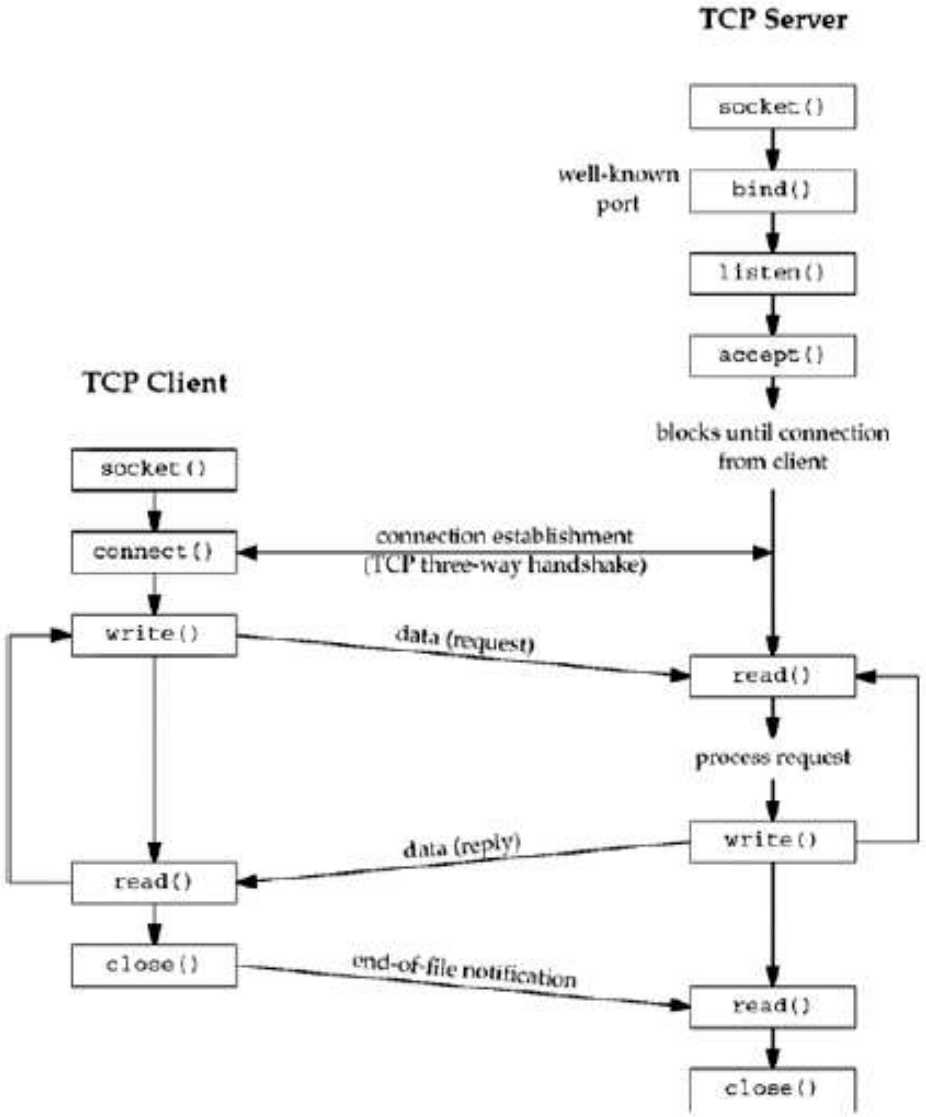
Elementary TCP Sockets

Socket functions for elementary TCP client/server



socket Function

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

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

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

- ***family*** specifies the protocol family and is one of the constants shown in Figure

family	Description
AF_INET	IPv4 protocols
AF_INET6	IPv6 protocols
AF_LOCAL	Unix domain protocols (Chapter 15)
AF_ROUTE	Routing sockets (Chapter 18)
AF_KEY	Key socket (Chapter 19)

type	Description
SOCK_STREAM	stream socket
SOCK_DGRAM	datagram socket
SOCK_SEQPACKET	sequenced packet socket
SOCK_RAW	raw socket

- The socket ***type*** is one of the constants shown in Figure
- The ***protocol*** argument to the socket function should be set to the specific protocol type found in Figure, or 0 to select the system's default for the given combination of *family* and *type*

Protocol	Description
IPPROTO_TCP	TCP transport protocol
IPPROTO_UDP	UDP transport protocol
IPPROTO_SCTP	SCTP transport protocol

AF_*xxx* Versus PF_*xxx*

- The "AF_" prefix stands for "address family"
- The "PF_" prefix stands for "protocol family“
- Historically, the intent was that a single protocol family might support multiple address families
- The PF_ value was used to create the socket
- The AF_ value was used in socket address structures
- But in actuality, a protocol family supporting multiple address families has never been supported
- The <sys/socket.h> header defines the PF_ value for a given protocol to be equal to the AF_ value for that protocol
- While there is no guarantee that this equality between the two will always be true

connect Function

- The connect function is used by a TCP client to establish a connection with a TCP server
- *sockfd* is a socket descriptor returned by the socket function
- The second and third arguments are a pointer to a socket address structure and its size
- The socket address structure must contain the IP address and port number of the server

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

Returns: 0 if OK, -1 on error

bind Function

- The bind function assigns a local protocol address to a socket (*sockfd)*
- With the Internet protocols, the protocol address is the combination of either a 32-bit IPv4 address or a 128-bit IPv6 address, along with a 16-bit TCP or UDP port number
- The second argument is a pointer to a protocol-specific address
- The third argument is the size of this address structure

```
#include <sys/socket.h>
int bind (intsockfd,const struct sockaddr *myaddr,socklen_taddrlen);
                                                    Returns: 0 if OK,-1 on error
```

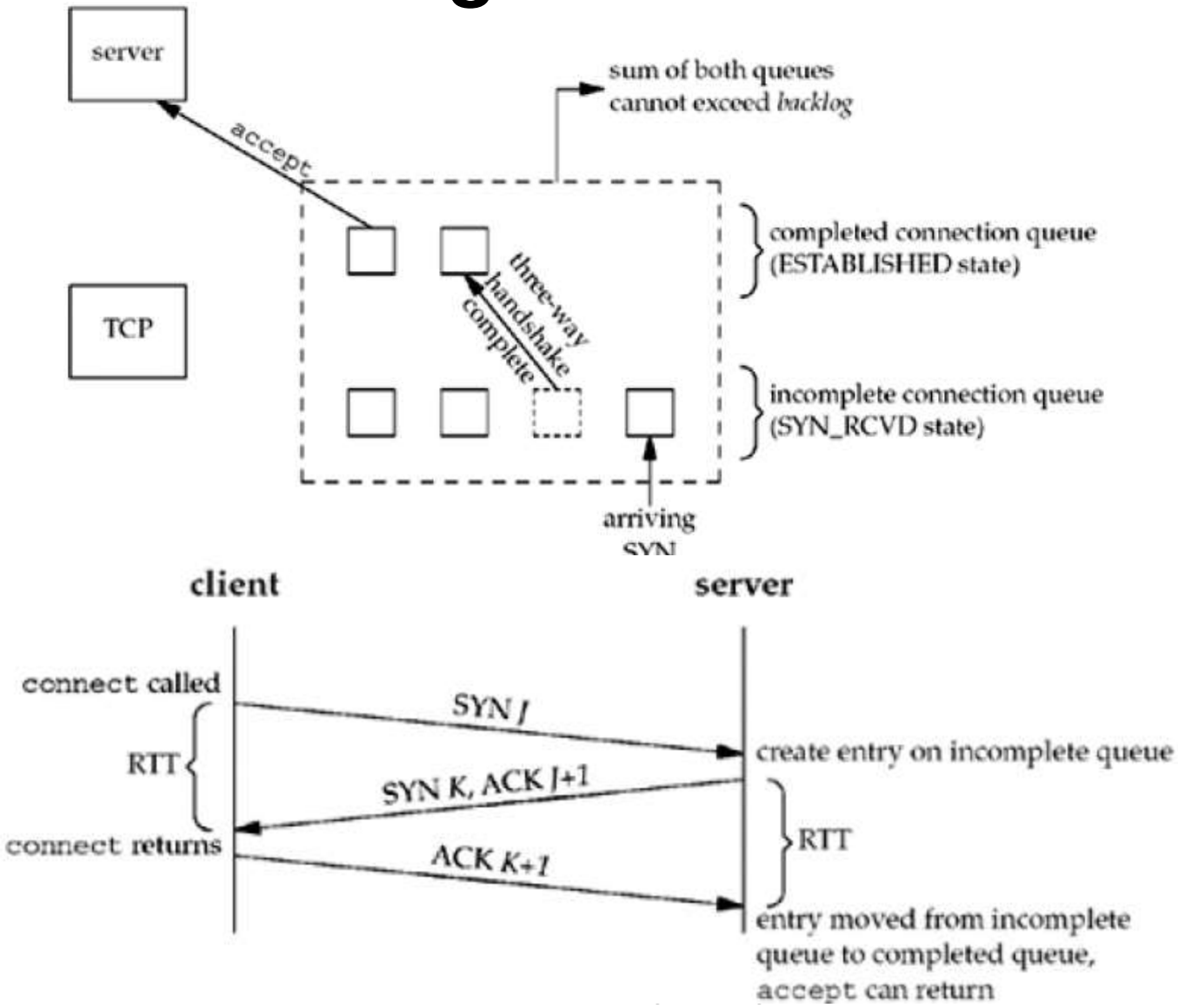
listen Function

- The listen function is called only by a TCP server and it performs two actions:
- When a socket is created by the socket function, it is assumed to be an active socket, that is, a client socket that will issue a connect
- The listen function converts an unconnected socket into a passive socket, indicating that the kernel should accept incoming connection requests directed to this socket
- The second argument to this function specifies the maximum number of connections the kernel should queue for this socket

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

Returns: 0 if OK, -1 on error

The two queues maintained by TCP for a listening socket



accept Function

- accept is called by a TCP server to return the next completed connection from the front of the completed connection queue
- If the completed connection queue is empty, the process is put to sleep (assuming the default of a blocking socket)

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

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

- The *cliaddr* and *addrlen* arguments are used to return the protocol address of the connected peer process (the client)
- *addrlen* is referred by the integer value to the size of the socket address structure pointed to by *cliaddr*
- If accept is successful, its return value is a brand-new descriptor automatically created by the kernel that refers to the TCP connection with the client

Daytime server that prints client IP address and port

- See Code from: *intro/daytimetcpsrv1.c*

solaris % **daytimetcpcli 127.0.0.1**
Thu Sep 11 12:44:00 2003
solaris % **daytimetcpcli 192.168.1.20**
Thu Sep 11 12:44:09 2003

- We first specify the server's IP address as the loopback address (127.0.0.1) then as its own IP address (192.168.1.20)
- Here is the corresponding server output:

solaris # **daytimetcpsrv1**
connection from 127.0.0.1, port 43388
connection from 192.168.1.20, port 43389

close Function

- The normal Unix close function is also used to close a socket and terminate a TCP connection

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

Returns: 0 if OK, -1 on error

Thank you