

Redes de Comunicação 2023/2024

TP05

Programming with sockets and TCP

Jorge Granjal
University of Coimbra



TP05: Programming with sockets and TCP

Overview:

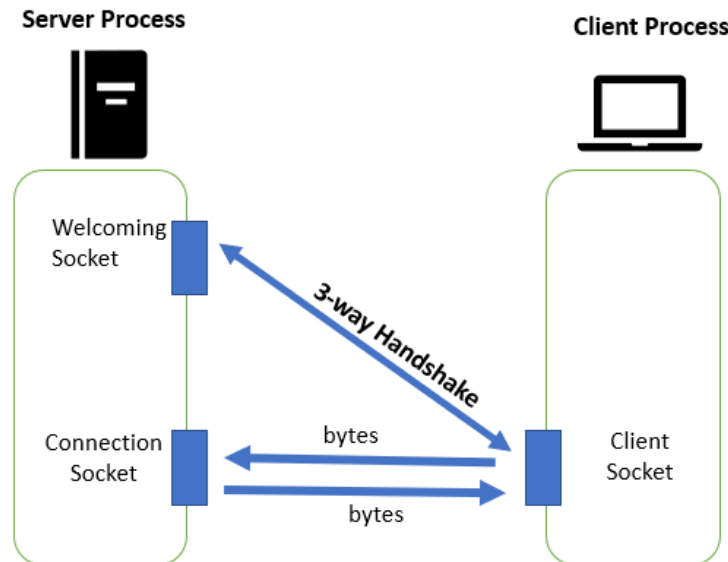
- Socket programming (TCP)
- Client-Server implementation structure
- Socket programming in C (functions)

Socket programming (TCP)

- TCP is a connection-oriented transport-layer protocol, it provides guaranteed, in-order and error-free packet communications between applications on *hosts*
- Before exchanging information, the client and server must establish a TCP connection
- When one side wants to send data to the other side it just drops data into the TCP connection via its socket:
 - Remember that with UDP you must attach a destination address to the packet before sending
 - A TCP socket is already identified by a 4-tuple (destination address and port, source address and port)

Socket programming (TCP)

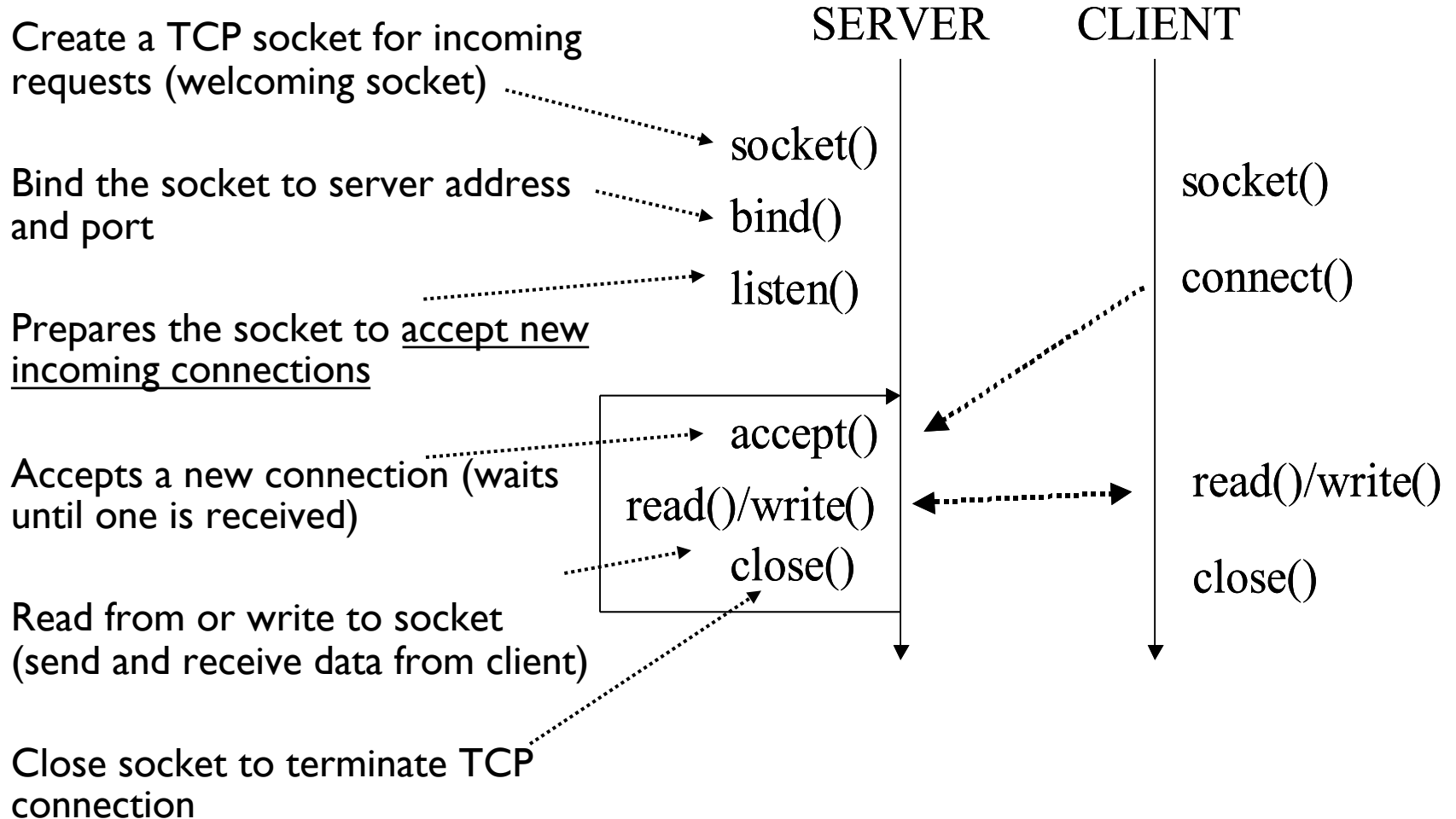
- A TCP Server uses two different types of sockets:
 - A *welcoming socket*, where new connections are received by the server: remains open so that server can receive connections from multiple clients
 - A *connection socket*: created/opened and closed as required, to handle communications with clients



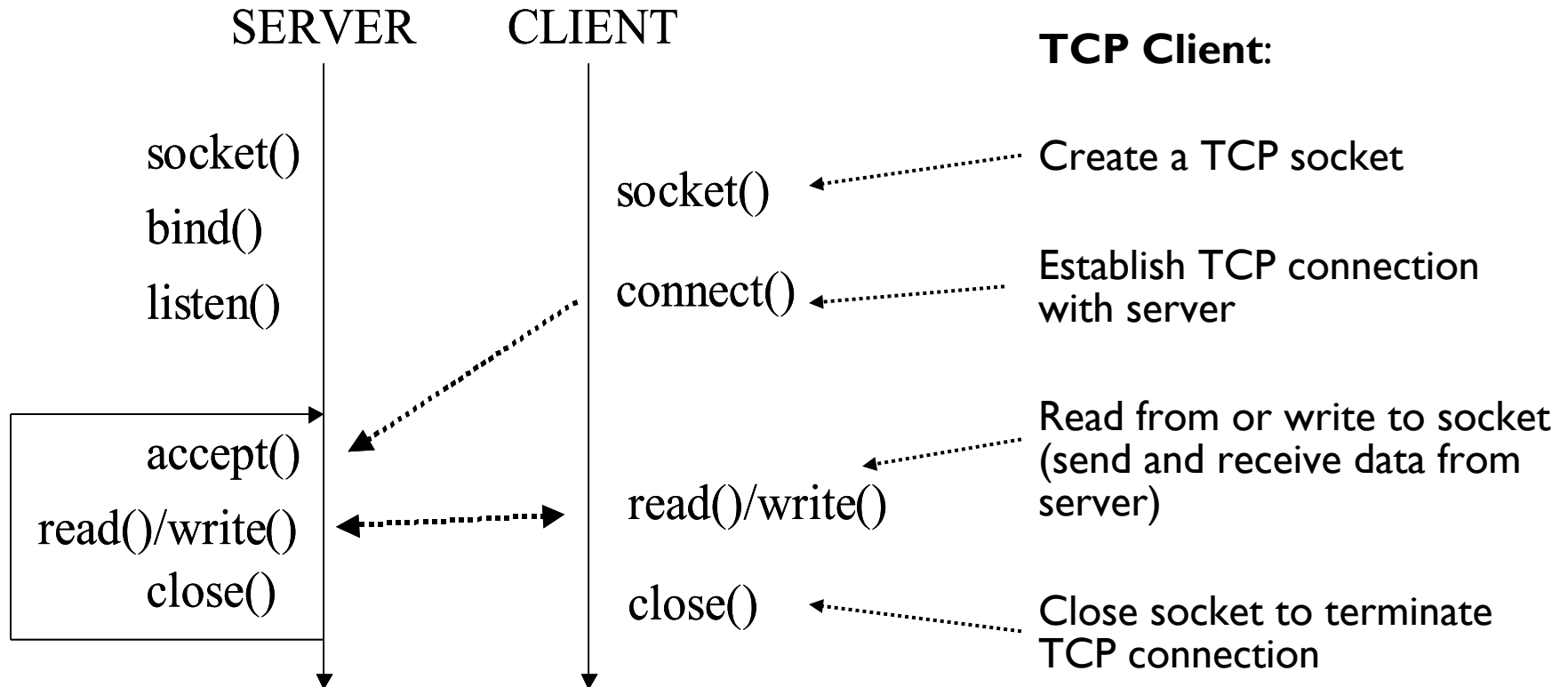
Client-socket, Welcoming-socket and Connection-socket

Client-Server implementation

TCP Server:



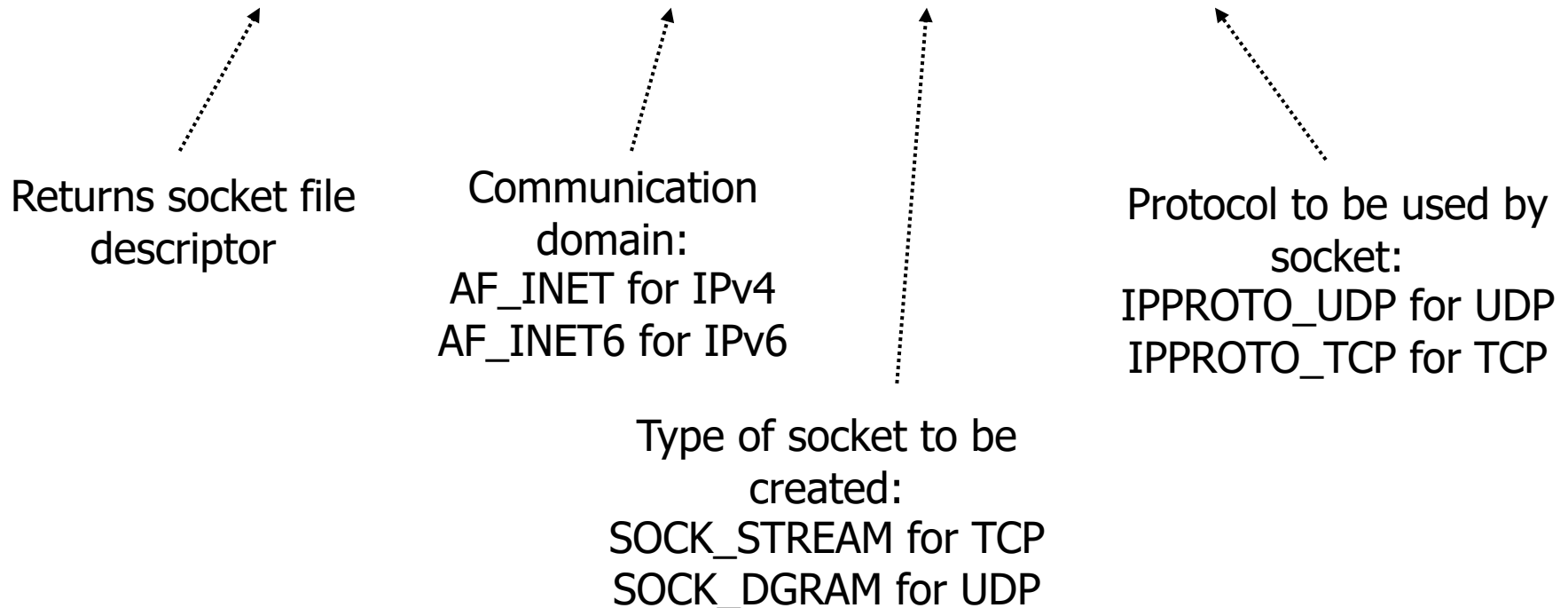
Client-Server implementation



Socket programming in C (functions)

- The application invokes the **socket** function to create an UDP or TCP socket

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



Socket programming in C (functions)

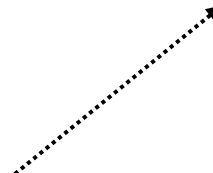
- The **bind** function is used by the server to assign an address to the unbound socket

int bind(int sockfd, const struct sockaddr *addr, socklen_t addrlen)

File descriptor of
socket to be
binded



Structure in which
address to be
binded to is
specified



Size of addr structure




Socket programming in C (functions)

- The **listen** function is used in the server to prepare the socket to receive new connections

int listen(int sockfd, int backlog)

File descriptor of
socket to listen for
new connection
requests



How many clients are
keep in the queue
waiting for accept

Socket programming in C (functions)

- The **accept** function is used in the server to accept a new connection (wait if none in the queue)

int accept(int sockfd, const struct sockaddr *address, socklen_t *address_len)

Returns file descriptor of socket for new connection

File descriptor of welcoming socket

Structure containing information about the new connection is returned

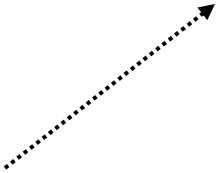
Variable in which size of address structure is returned

Socket programming in C (functions)

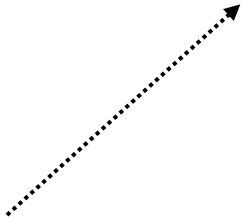
- The **connect** function is used in the client to initiate a new TCP connection

int connect(int sockfd, const struct sockaddr *address, socklen_t address_len)


File descriptor of
socket to initiate a
new connection



Structure containing
information about the
address and port of the
server

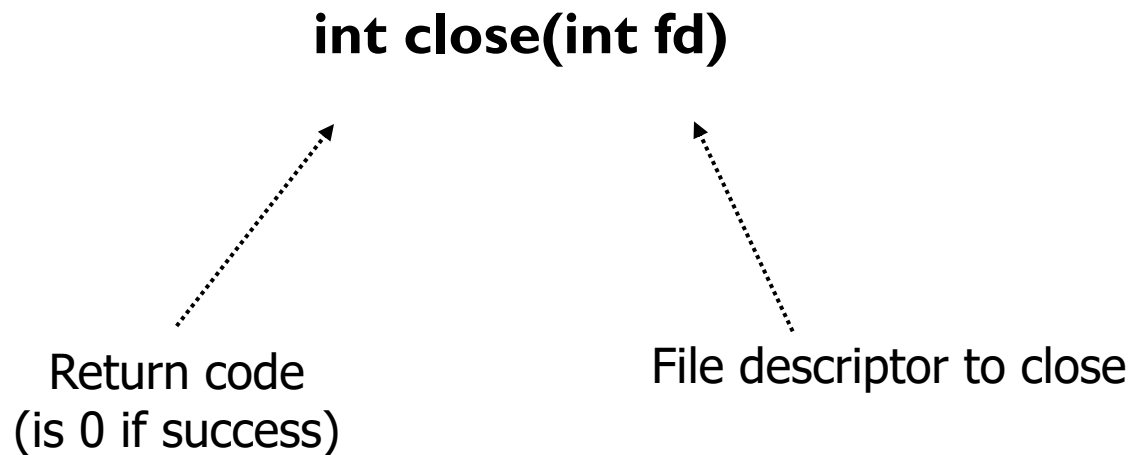


Variable with size of the
address structure



Socket programming in C (functions)

- The **close** function is used to close a file descriptor




Socket address structures

- Most socket functions require a pointer to a socket address structure
- The names of these structures begin with *sockaddr_* and end with a unique suffix for each protocol suite

```
#include <arpa/inet.h>
```

```
struct in_addr {  
    in_addr_t  s_addr;  
  
};
```

Generic structure (used in declarations,
to deal with address structures from any
supported protocol families)




```
/* 32-bit IPv4 address */  
/* network byte ordered */
```

```
struct sockaddr_in {  
    uint8_t      sin_len;           /* length of structure (16) */  
    sa_family_t  sin_family;      /* AF_INET */  
    in_port_t    sin_port;        /* 16-bit TCP or UDP port number */  
                                     /* network byte ordered */  
    struct in_addr sin_addr;      /* 32-bit IPv4 address */  
                                     /* network byte ordered */  
    char         sin_zero[8];      /* unused */  
  
};
```

Other useful functions

- Convert host name to IP address:

```
struct hostent * gethostbyname(const char* name)
```

 Return structure with IP address
corresponding to name

 Name of host (server)

- Network byte order** is *big endian*. Convert to network byte order when sending data and using addresses and ports:

```
uint32_t htonl(uint32_t hostlong);  
uint16_t htons(uint16_t hostshort);  
uint32_t ntohl(uint32_t netlong);  
uint16_t ntohs(uint16_t netshort);
```

TP05: Summary

What we have covered here?

- Socket programming (TCP)
- Client-Server implementation structure
- Socket programming in C (functions)