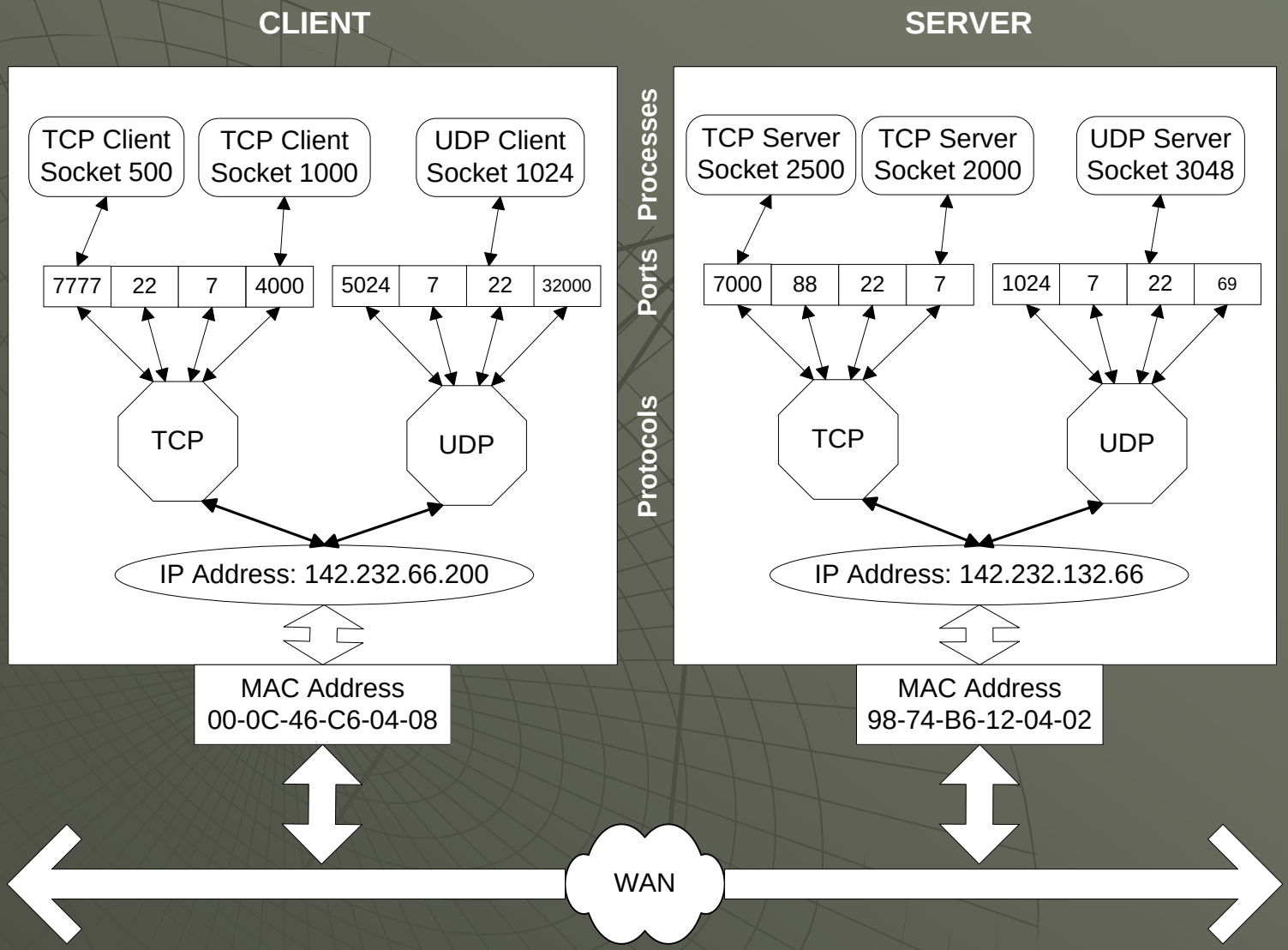
TCP/IP Programming Interface

- Network programming involves the implementation of “server” and “client” applications.
- We will use the Berkeley socket API to design “server” and “client” applications

TCP/IP Model


CLIENT
SERVER
Processes
Ports
Protocols
TCP Client Socket 500
TCP Client Socket 1000
UDP Client Socket 1024
7777
22
7
4000
5024
7
22
32000
TCP
UDP
IP Address: 142.232.66.200
MAC Address 00-0C-46-C6-04-08
TCP Server Socket 2500
TCP Server Socket 2000
UDP Server Socket 3048
7000
88
22
7
1024
7
22
69
TCP
UDP
IP Address: 142.232.132.66
MAC Address 98-74-B6-12-04-02
WAN

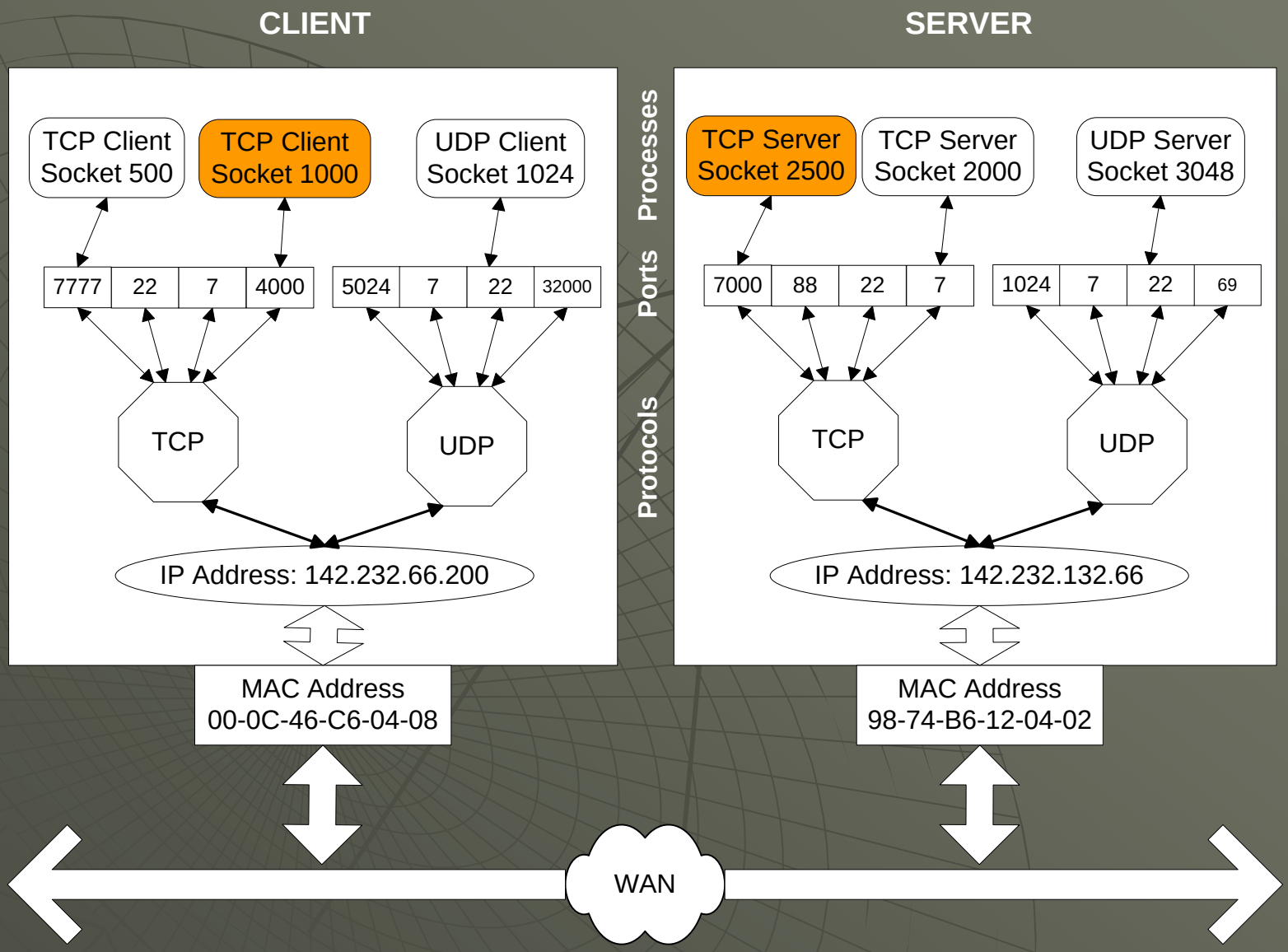
Socket

- Generalization of the **UNIX file access system** designed to incorporate **network protocols**.
- **Communications channel** between a pair of **'sockets'**
- A socket is defined by a group of four integers:
  - The remote host identification number or address
  - The remote host port number
  - The local host identification number or address
  - The local host port number

Create Socket

socket(family, type, protocol)

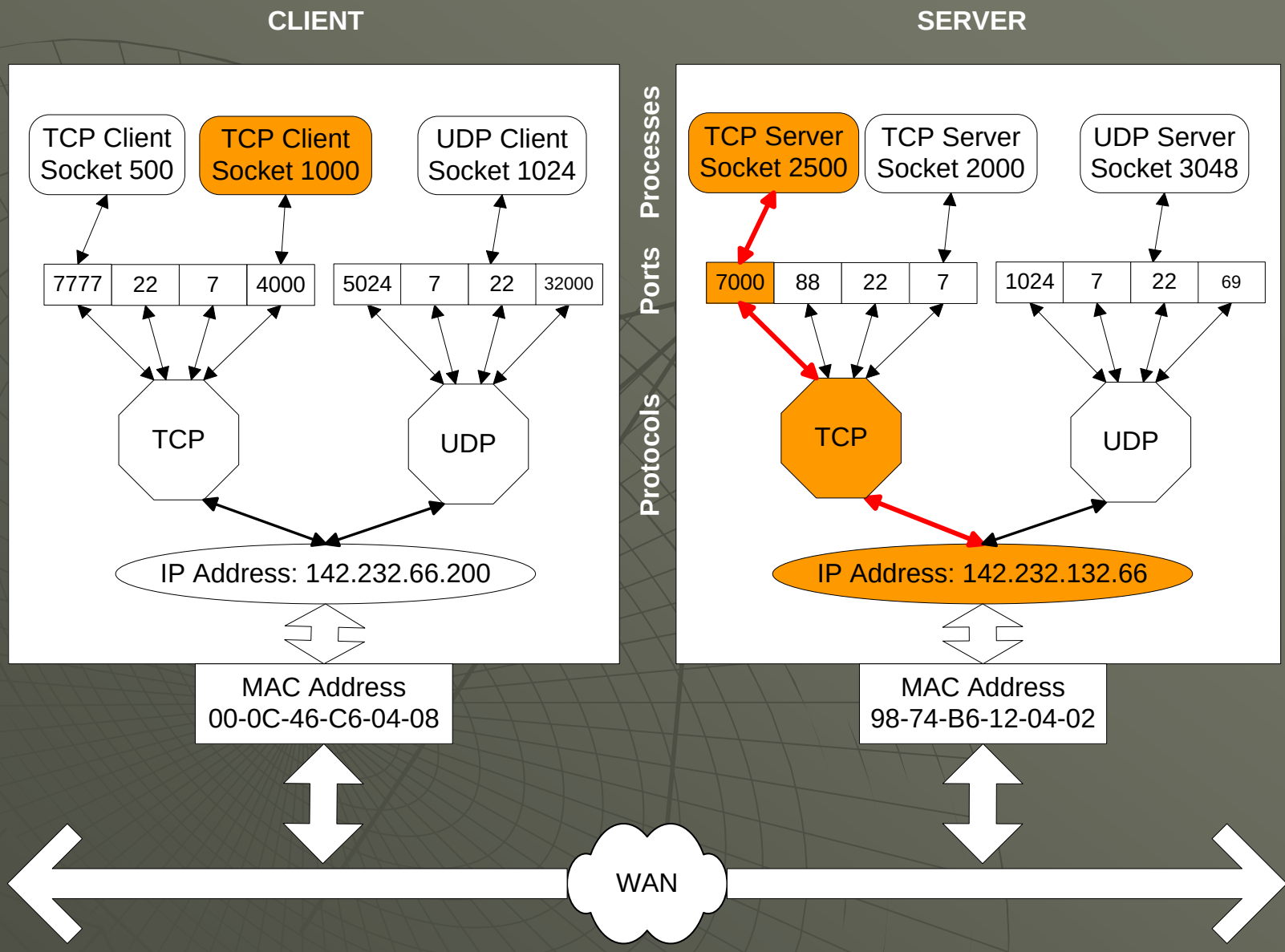
- family: the protocol family
  (e.g. **PF_INET, PF_UNIX)**
- type: the abstract type of the communication desired
  (e.g. **SOCK_STREAM, SOCK_DGRAM**)
- protocol: specific protocol desired
  (e.g. **TCP** or **UDP**)
- Returns an **socket file descriptor** (int)


CLIENT
SERVER
Processes
Ports
Protocols
TCP Client Socket 500
TCP Client Socket 1000
UDP Client Socket 1024
7777
22
7
4000
5024
7
22
32000
TCP
UDP
IP Address: 142.232.66.200
MAC Address 00-0C-46-C6-04-08
TCP Server Socket 2500
TCP Server Socket 2000
UDP Server Socket 3048
7000
88
22
7
1024
7
22
69
TCP
UDP
IP Address: 142.232.132.66
MAC Address 98-74-B6-12-04-02
WAN

Bind Socket with Address - server

bind (socket,
sockaddr,
sockaddrlen)

- socket: socket descriptor
- sockaddr: **pointer** to the **address** to which the socket should be bound.
- sockaddrlen: the **size** of the **address**.
- Returns -1 and sets **errno** for error


CLIENT
SERVER
Processes
Ports
Protocols
TCP Client Socket 500
TCP Client Socket 1000
UDP Client Socket 1024
7777
22
7
4000
5024
7
22
32000
TCP
UDP
IP Address: 142.232.66.200
MAC Address 00-0C-46-C6-04-08
TCP Server Socket 2500
TCP Server Socket 2000
UDP Server Socket 3048
7000
88
22
7
1024
7
22
69
TCP
UDP
IP Address: 142.232.132.66
MAC Address 98-74-B6-12-04-02
WAN

listen() - server

Marks socket as 'listening' and sets the maximum number of listen queue

int listen(int s, int backlog)

- s: socket
- backlog: max length of the queue of unprocessed connection requests

Addressing

- **common framework** for all addresses to support multiple protocol families
- In the **Internet family**, transport addresses are **6 octets** long:
  - **4 octets** for the **Internet address**
  - **2 octets** for the **port number**.
- **<netinet/in.h>** defines the appropriate structures to use for the Internet family

```
struct sockaddr_in
{
   short  sin_family;
   unsigned short  sin_port;
   struct in_addr  sin_addr;
   unsigned char sin_zero[8];   /* unused */
}
```

sin_family: address family
sin_port: 16-bit port number
sin_addr: 32-bit Internet address

```
struct in_addr
{
   u_long  s_addr;          /* 32-bit IP Address */
}
```

Get Server Information

gethostbyname(host_name)

- Maps the request into a **DNS** (**Domain Name System**) query that it sends to the **resolver** running on the local machine.
- Returns a **pointer** to a **hostent structure** that contains the requested addresses.
- The related function **gethostbyaddr()** takes an **Internet address** as an argument instead.

```
struct hostent
{
        char *h_name;
        char **h_alias;
        int h_addrtype;
        int h_length;
        char **h_addr_list;
};
```

h_name: the official, fully qualified **domain name**

h_alias: a null-terminated list of alternate names **(aliases)**

h_addrtype: protocol family the address belongs to

h_length: the **length** of the address

h_addr_list: a null-terminated **array of addresses**

Address Manipulation Functions

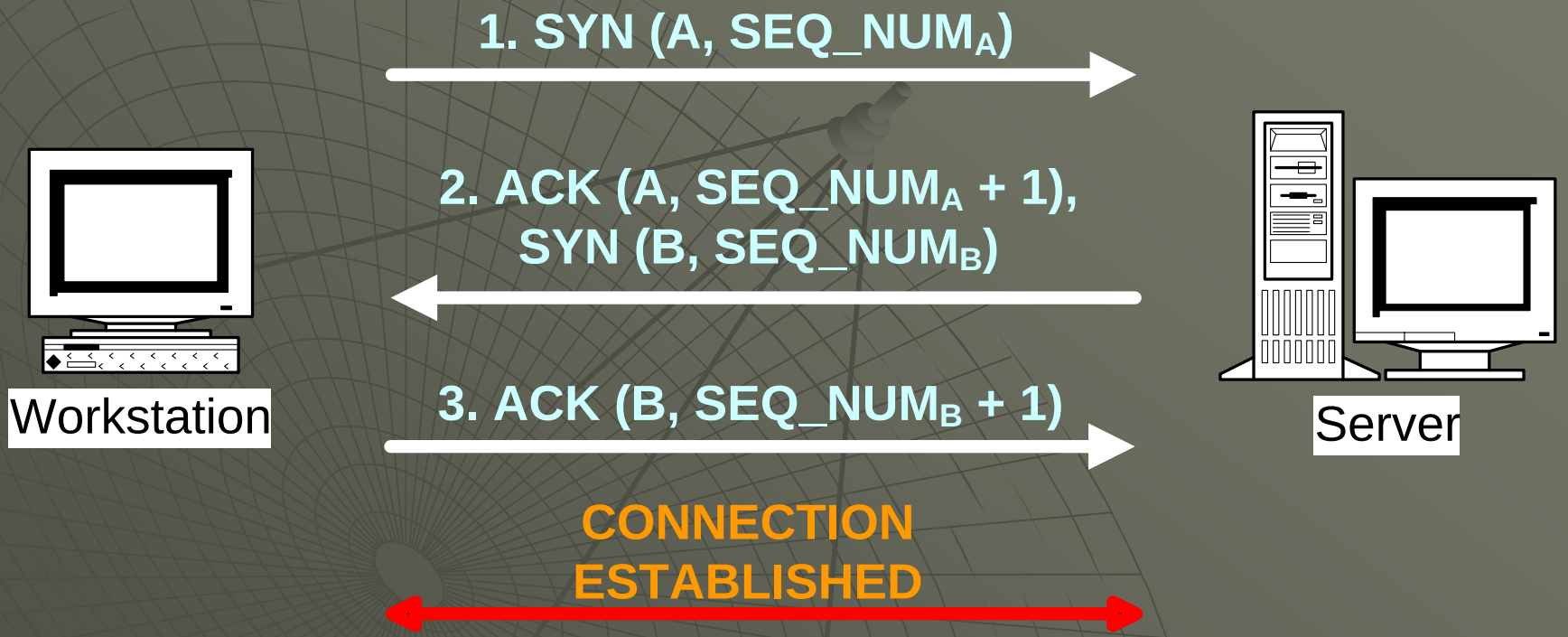
inet_aton() / inet_network()

- Takes strings representing Internet addresses in **dotted notation**
- Returns numbers suitable for use in **sockaddr_in** structures.

inet_ntoa()

- Takes a 32-bit IP address in **network byte order** Internet address
- Returns the corresponding address in dotted-decimal notation.

Make a Connection to Server

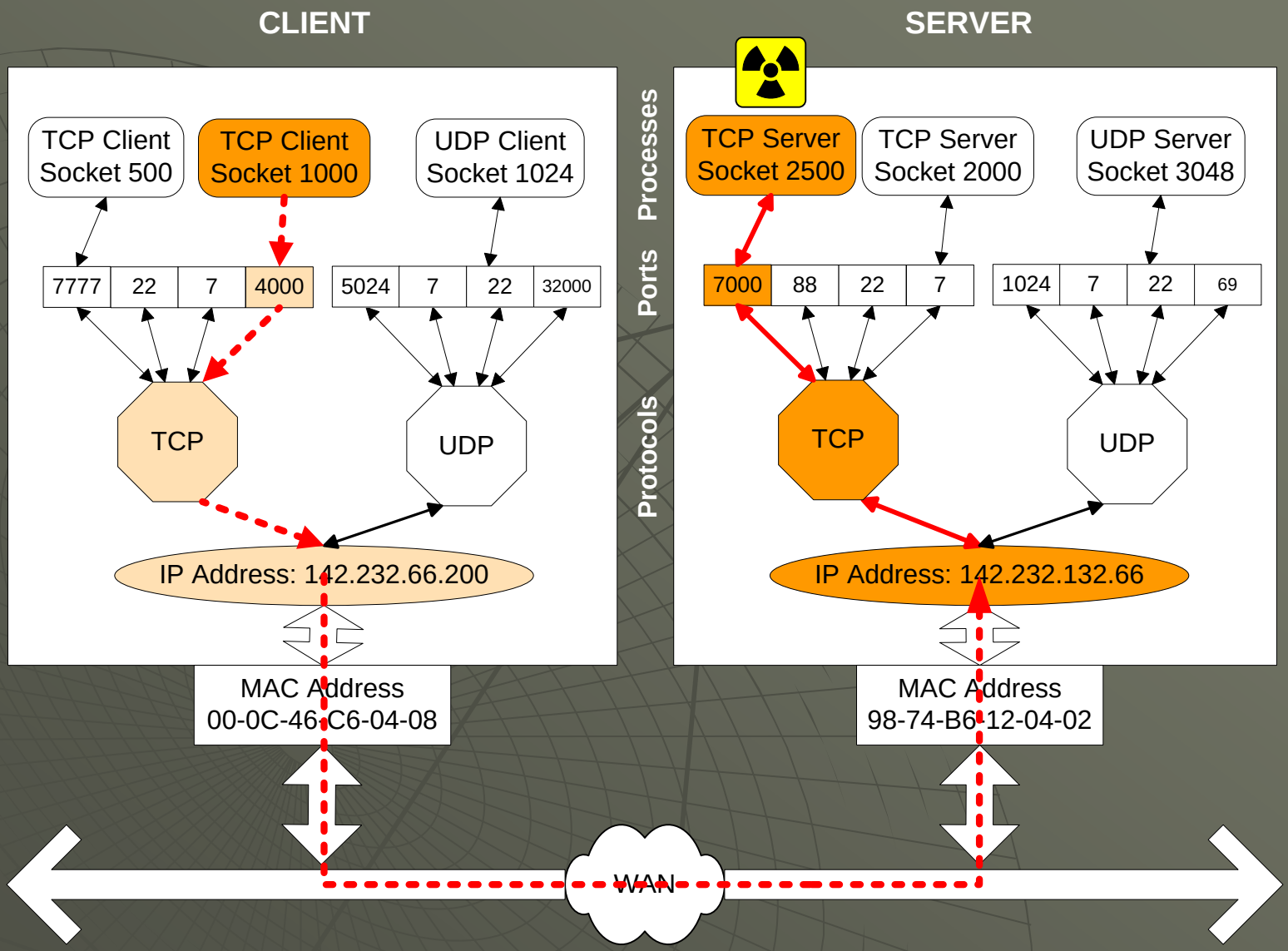


TCP 3-Way Handshake

Send Connect Request (SYN)

```
connect( int s,
         struct sockaddr *name,
         int namelen );
```

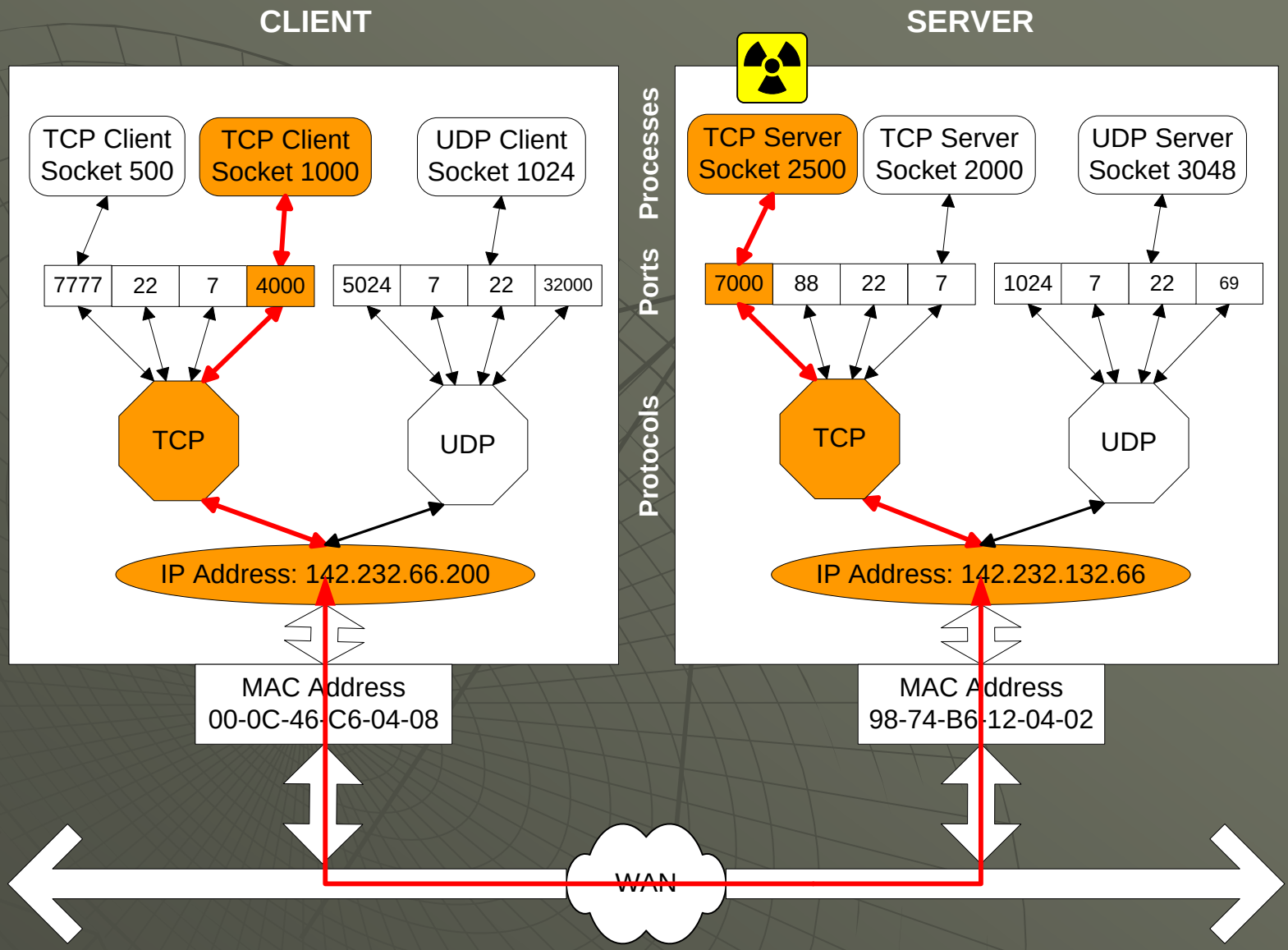
- Establishes communication with a remote entity
  - initiates a **three-way handshake** for TCP connection
  - normally only used for **SOCK_STREAM** sockets.
- s: client socket
- name: points to an area of memory containing the address information for the remote entity
- namelen: gives the length of 'name'


CLIENT
SERVER
Processes
Ports
Protocols
TCP Client Socket 500
TCP Client Socket 1000
UDP Client Socket 1024
7777
22
7
4000
5024
7
22
32000
TCP
UDP
IP Address: 142.232.66.200
MAC Address 00-0C-46-C6-04-08
TCP Server Socket 2500
TCP Server Socket 2000
UDP Server Socket 3048
7000
88
22
7
1024
7
22
69
TCP
UDP
IP Address: 142.232.132.66
MAC Address 98-74-B6-12-04-02
WAN

Accept Connect Request (SYN/ACK)

```
accept ( int s,
         struct sockaddr *addr,
         int *addrlen)
```

- s: base socket descriptor
- addr: points to a struct sockaddr of the remote (calling) system.
- addrlen: size of addr
- Returns a new socket descriptor that will be used for all subsequent communication with the remote host.


CLIENT
SERVER
Processes
Ports
Protocols
TCP Client Socket 500
TCP Client Socket 1000
UDP Client Socket 1024
7777
22
7
4000
5024
7
22
32000
TCP
UDP
IP Address: 142.232.66.200
MAC Address 00-0C-46-C6-04-08
TCP Server Socket 2500
TCP Server Socket 2000
UDP Server Socket 3048
7000
88
22
7
1024
7
22
69
TCP
UDP
IP Address: 142.232.132.66
MAC Address 98-74-B6-12-04-02
WAN

Send/Receive Data

send(socket, message, length, flags)
recv(socket, message, length, flags)

- message: a pointer to the data to be written.
- length: data length.
- flags: to send **flags** to the **underlying protocol**.
  - flags may be formed by ORing MSG_OOB and MSG_DONTROUTE.
- send() and recv() may only be used with **SOCK_STREAM** type sockets.

Send/Receive Data - UDP

sendto(socket, message, length, flags, dest, destlength)

recvfrom(socket, message, length, flags, from, fromlength)

- The two additional arguments are **pointers** to a **sockaddr** structure and its **length**.

Close Connection

shutdown (socket, how)

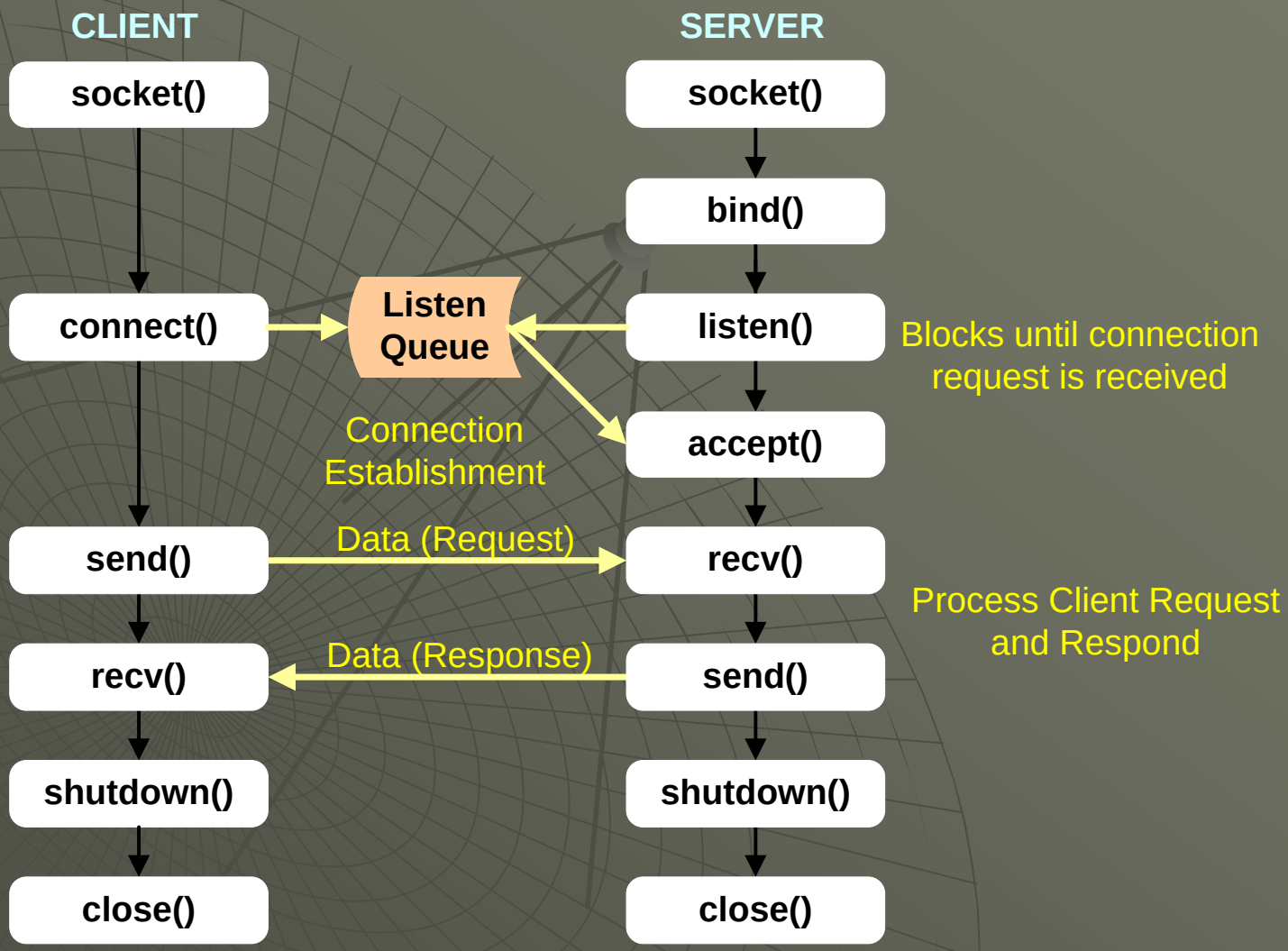
to shut down all or part of a full-duplex connection

- how: **direction** of the connection to be shutdown:
  - 0 - additional receives are shutdown.
  - 1 - additional sends are disallowed.
  - 2 - additional sends and receives will be disabled.

close(socket)

to close a socket

TCP/IP Client Server Model



UDP Client Server Model