BVM Engineering College, VV Nagar

Gujarat Technological University

# EMBEDDED SYSTEMS

SEM 7
PRESENTATION

## Socket Functions

# Electronics & Communication Dept.

**Presented By :**

- Chaitanya Tejaswi (140080111013)
- Omkar Mudholkar (140080111031)

# **Introduction:** Need for Sockets

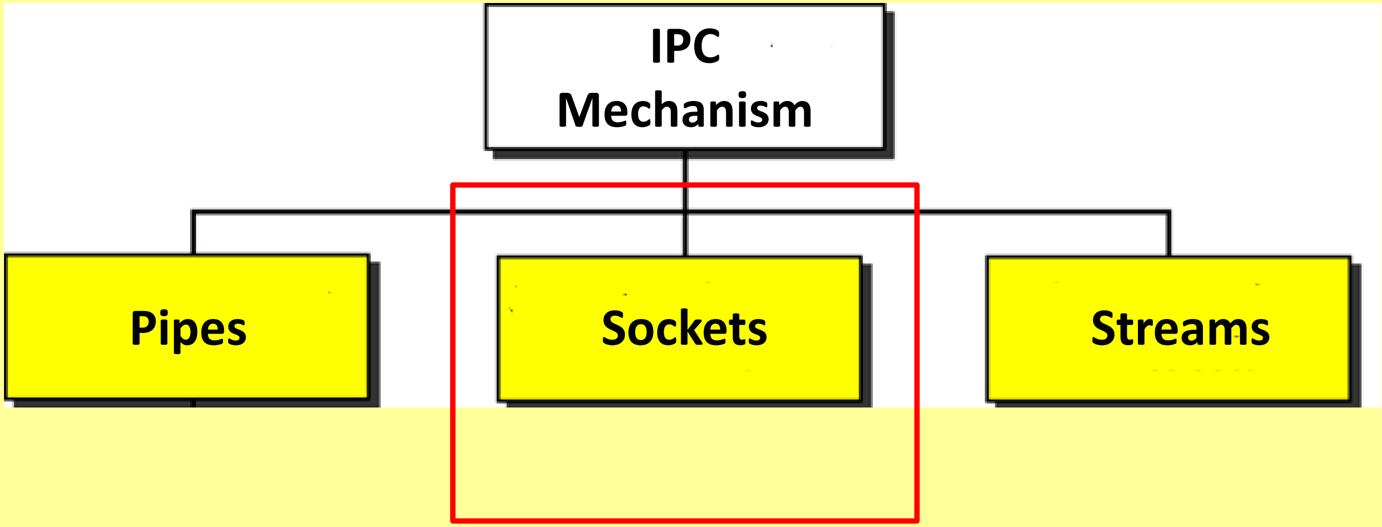
- Typical applications today consist of many cooperating processes either on the same host or on different hosts.

**Example:** Client-Server Application.

How to share (large amounts of ) data?

- Share files? How to avoid contention? What kind of system support is available?
- We want a general mechanism that will work for processes irrespective of their location.

# Making a Logical Connection to a (Remote) Process



# What are Sockets?

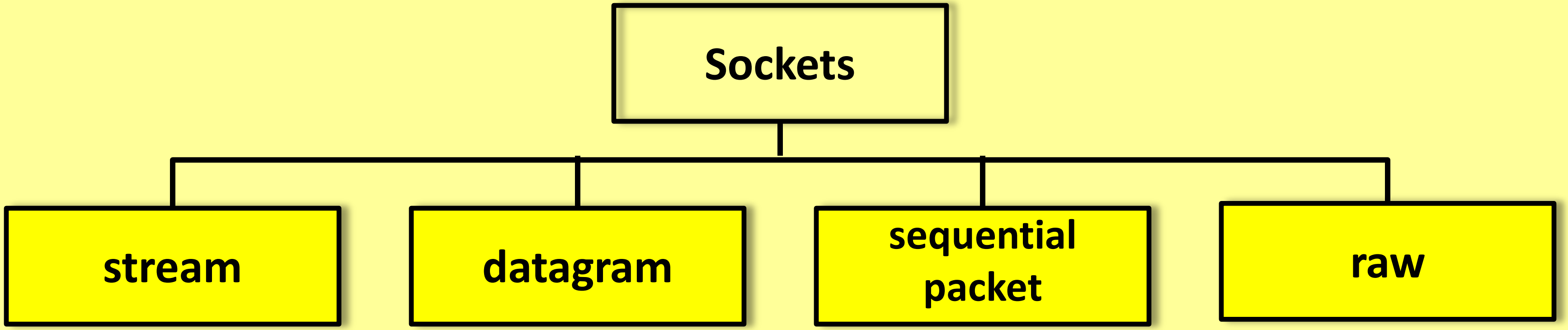
- **Socket** is an abstraction for an end point of communication that can be manipulated with a file descriptor.
- *It is an abstract object from which messages are sent and received.*

- **Sockets** are created within a ***communication domain*** just as **files** are created within a ***file system***.
- A **communication domain** is an abstraction introduced to bundle common properties of processes communicating through sockets.

  Example: UNIX domain, internet domain.

# Socket Types

**Socket types** define the communication properties visible to the application. Processes communicate only between sockets of the same type.

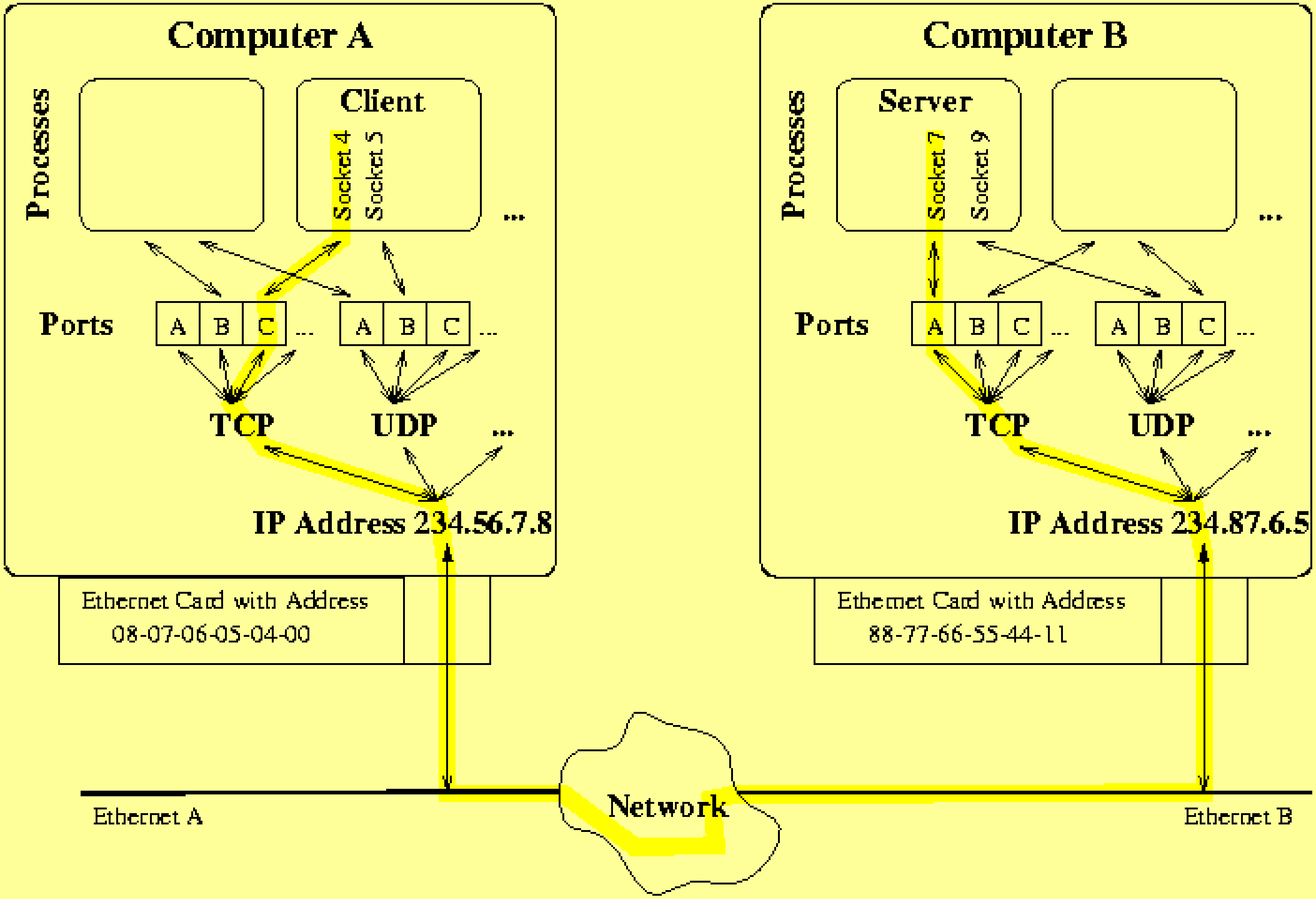


# Identifying A Socket

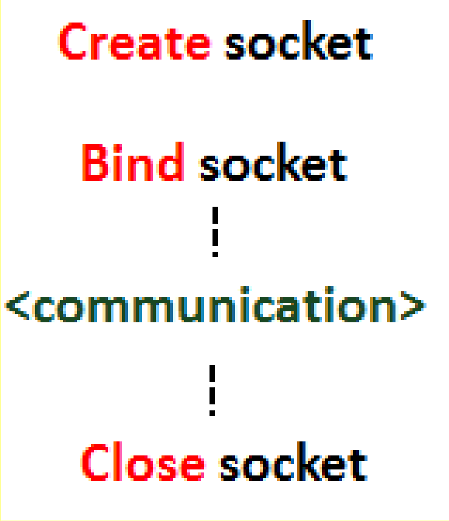
Each socket is identified by an address made up of:

**Global end-point Address** *(eg. IP address)*

**Local end-point Address** *(eg. Port Number)*



# Anatomy of a Socket #1



Create socket

Bind socket

<communication>

Close socket

**Create Socket**

1. A socket is created by a ***system call***, and has file-like semantics. Any process can do this, to communicate with any other process.
2. The ***system call*** returns a ***descriptor*** that is used as  a reference to the socket.

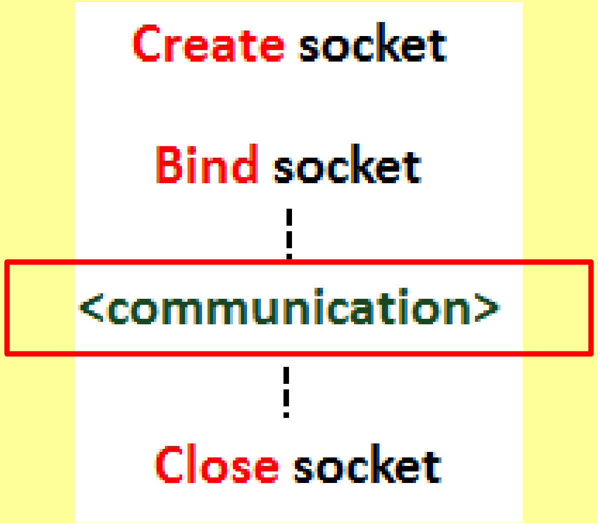
**Bind Socket**

1. Before communication can take place, the destination process must ***bind*** the ***descriptor*** to its ***socket address***. The sender may do so, if a reply is to be expected.
2. A ***system call*** is used to execute the binding.
3. Once bound, a socket address cannot be changed.

**Close Socket**

- A socket lasts *until it is closed* **OR** until every process with the descriptor exits.

# Anatomy of a Socket #2



Create socket

Bind socket

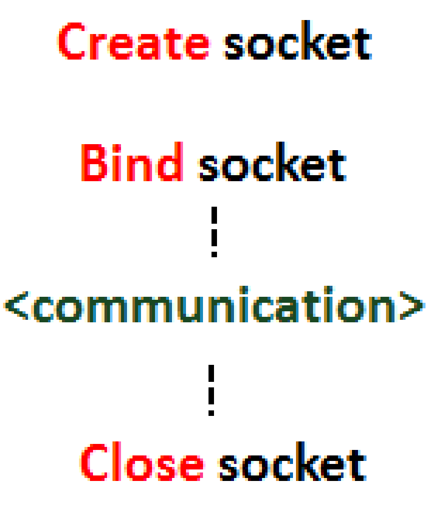
<communication>

Close socket

**Communication**

1. When communication begins, messages are **queued** at the sender-socket until transmitted by the associated device-driver.
2. Similarly for receiver-socket.

# Anatomy of a Socket – C/C++



Create socket

Bind socket

<communication>

Close socket

```
/* Create Socket */
int socket(int domain, int type, int protocol);

/* Bind Socket */
int bind(int s, const struct sockaddr *name, int
namelen);

/* Communication */
// Initiate
int listen(int s, int backlog)
int connect(int s, struct sockaddr *name, int namelen)
// Communicate
read();
write();
int send(int s, const char *msg, int len, int flags);
int recv(int s, char *buf, int len, int flags);

/* Bind Socket */
close();
```

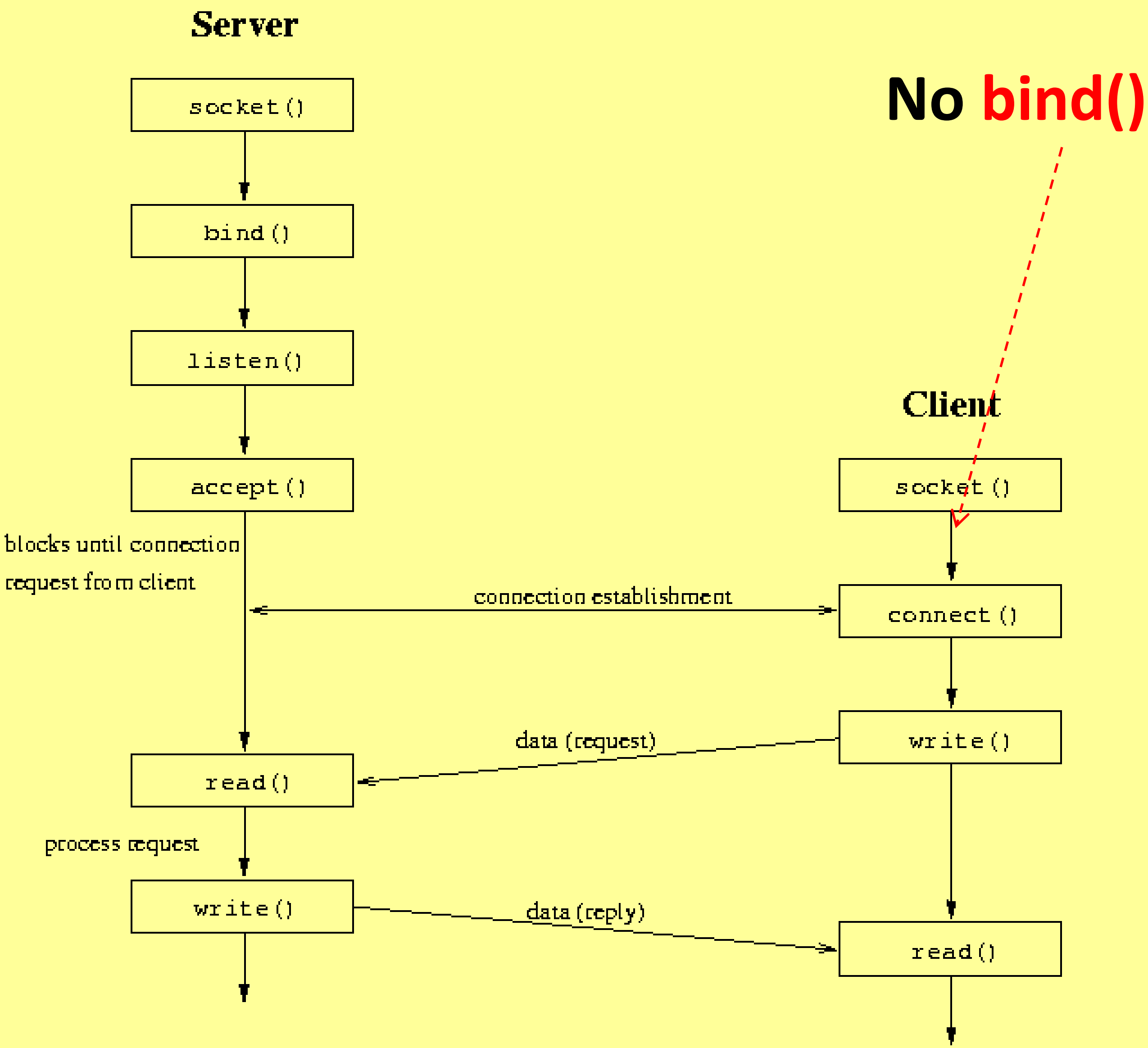
# Examples

## Socket System Calls:

Connection-Oriented Protocol

Connection-less Protocol

# Example: Connection-oriented Protocol



Server

socket()

bind()

listen()

accept()

blocks until connection request from client

read()

process request

write()

Client

socket()

connect()

write()

read()

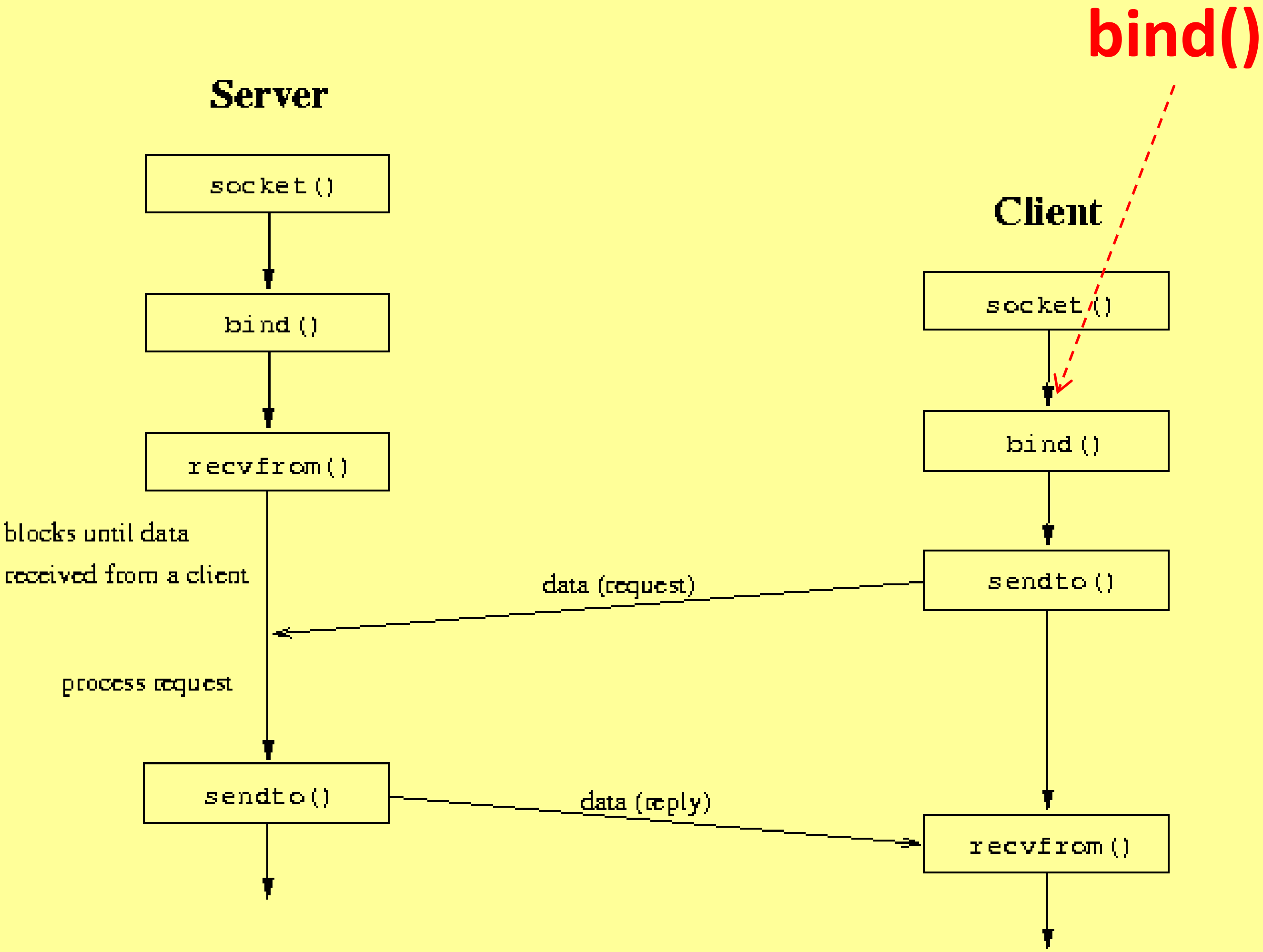
connection establishment

data (request)

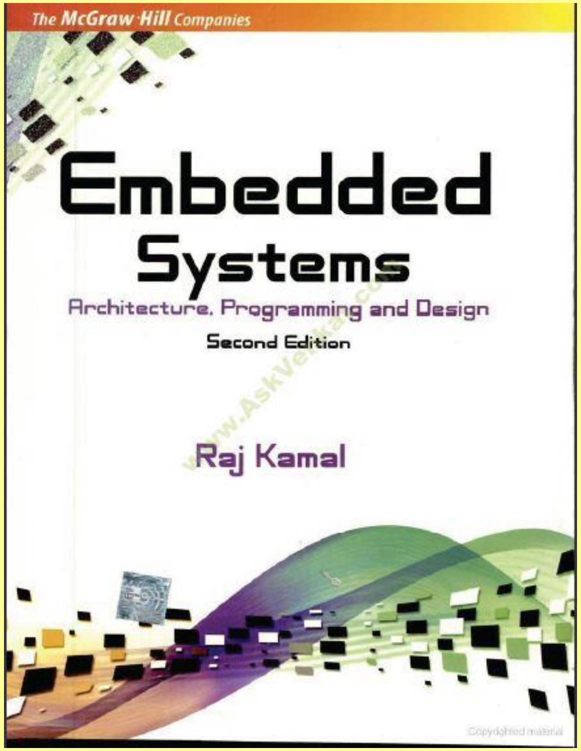
data (reply)

No bind()

# Example: Connection-less Protocol



# Bibliography


The McGraw·Hill Companies
Embedded
Systems
Architecture, Programming and Design
Second Edition
Raj Kamal
Copyrighted material

**Embedded Systems: Architecture, Programming and Design**
(Raj Kamal)

**Links:**
https://users.cs.cf.ac.uk/Dave.Marshall/C/node28.html
http://www.gerhardmueller.de/docs/UnixCommunicationFacilities/ip/node9.html