Types of Processes

1. **Independent processes**:
   - Cannot affect or be affected by the execution of another process.
   - Any process that does not share data with any other process is also an independent process.

2. **Cooperating processes**:
   - Can affect or be affected by the execution of another process running on the same or different host.
   - Any process that shares data with other processes is a cooperating process.

Inter-Process Communication: Definition & Concept

- Methods for effective sharing of information among cooperating processes are collectively known as:
  *inter-process communication (**IPC**)*.

- IPC is a mechanism for processes to cooperate, communicate, and synchronize their actions.

- An operation system provides inter-process commination API to allow processes to exchange information.
  - Part of the ***System Call API*** of the underlying/host OS.

Advantages of Process Cooperation

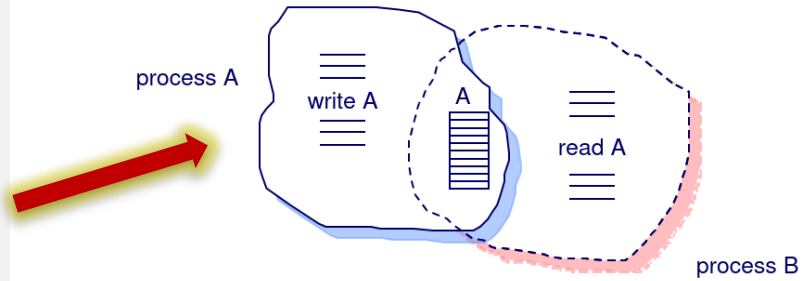
- Inter-process communication is useful for creating **cooperating processes**.

- **Advantages:**
  - **Information sharing**
  - Computation speed up
  - Modularity
  - Convenience

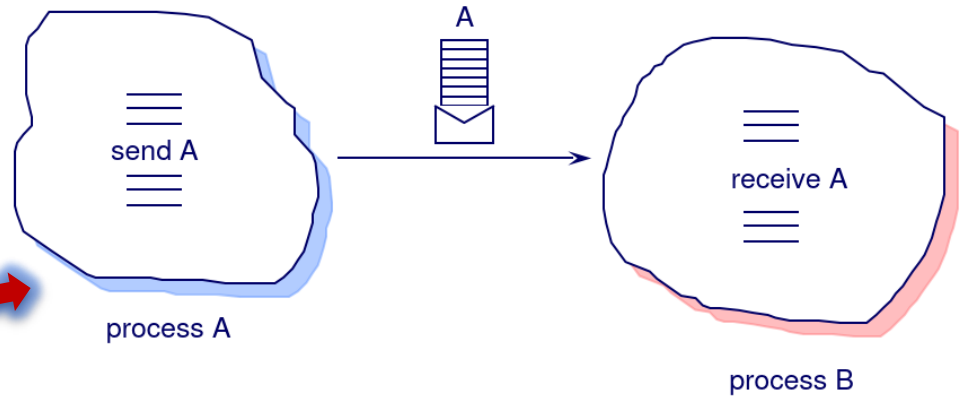
Inter-Process Communication: Models

- **Two** basic models are used:

1. **Shared storage**:
   - Shared memory or shared files.
   - Shared data are directly available to each process in their address spaces.



2. **Message passing**:
   - Sockets, pipes, etc.
   - Shared data are explicitly exchanged via ***messages***.
   - Processes interact with each other through ***messages*** with assistance from the underlying operating system.



Message Types and Structure

- In a message passing system, the information exchanged among processes is called a ***message***.

- There are ***two types*** of messages:
  - **Text based** (plain text)
  - **Binary based** (binary representation)

- A message can be a structured object and can have a fixed or variable length.

- There are two basic operations on IPC messages:
  1) `send()` → transmission of a message (initiated by a ***sender***)
  2) `receive()` → receipt of a message (handled by a ***receiver***)

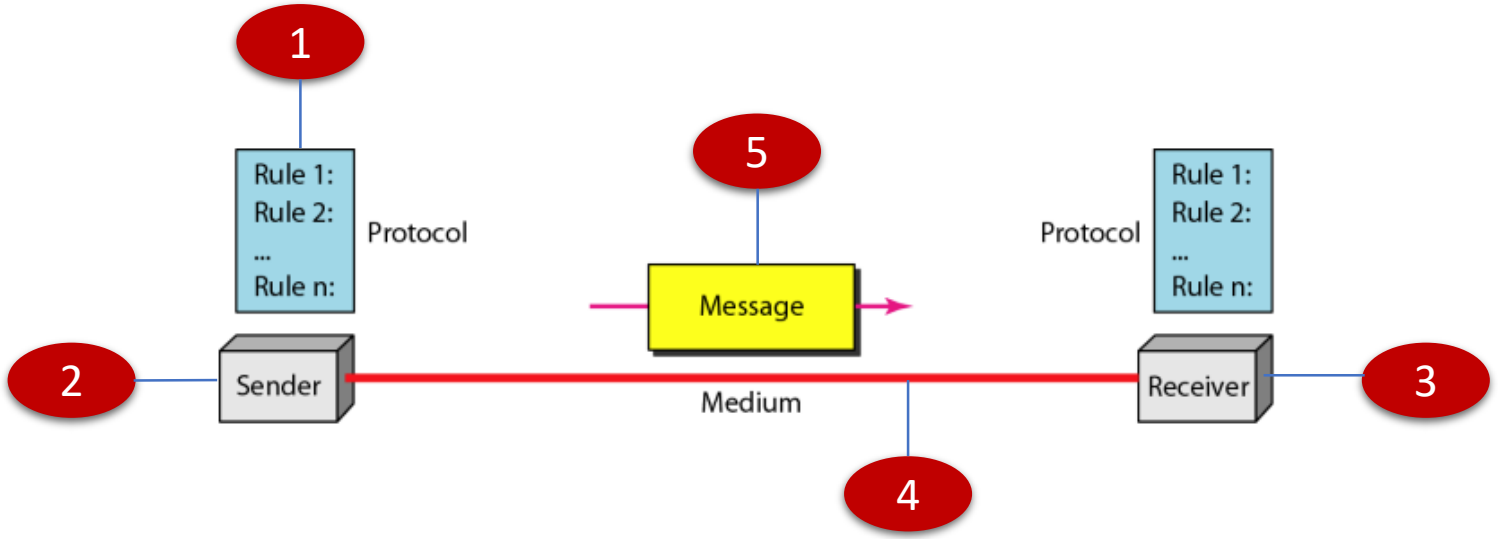
Message Passing Systems: Types of Communication

- The sender and receiver can communicate in either of the following **forms:**

- **Synchronous** (also known as *rendezvous*)**:**
  - Involved processes synchronize at every message.
  - Both ***send*** and ***receive*** are blocking operations.

- **Asynchronous:**
  - The ***send*** operation is almost always non-blocking.
  - The ***receive*** operation can have blocking (waiting) or non-blocking (polling) variants).

Components of Data Communication in a Message Passing System

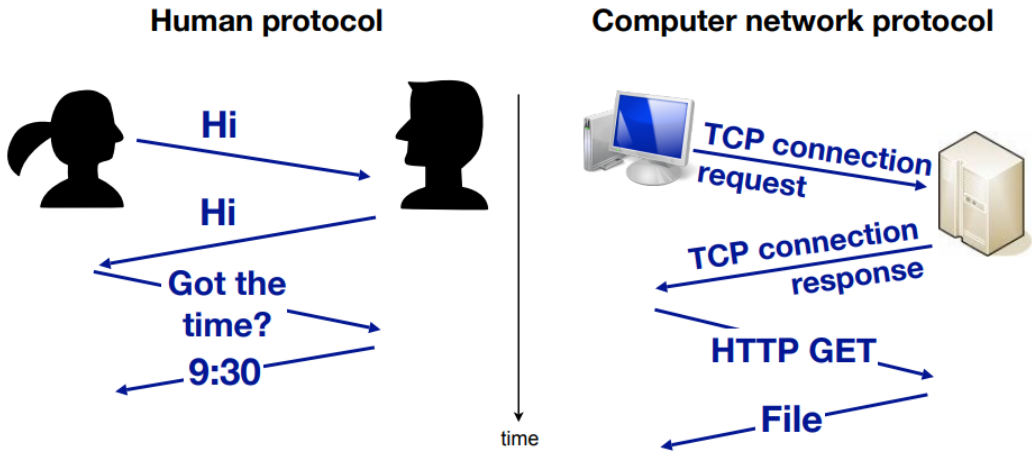


- Communication in distributed systems is always based on low-level message passing as offered by the underlying network.

What is Protocol?

- Agreement on information exchange in distributed networking.

- **A protocol defines:**
  - ***Format*** for valid messages (syntax)
  - ***Rules*** for data exchange (grammar)
  - A ***vocabulary*** of messages and their meaning (semantic).



Client-Server Paradigm

- The client-server model is the most used communication paradigms in networked systems.
- It supports bidirectional communication:
  - Tow-way, request-response based communication between processes running on the same or different hosts.
- A ***server process***, running on a server ***host***, provides access to a service.
- A ***client process***, running on a client host, accesses the service via the server process.
- The interaction of the processes proceeds according to a ***protocol***.
- An application based on the client-server paradigm is a client-server application.

Client-Server Communication Model

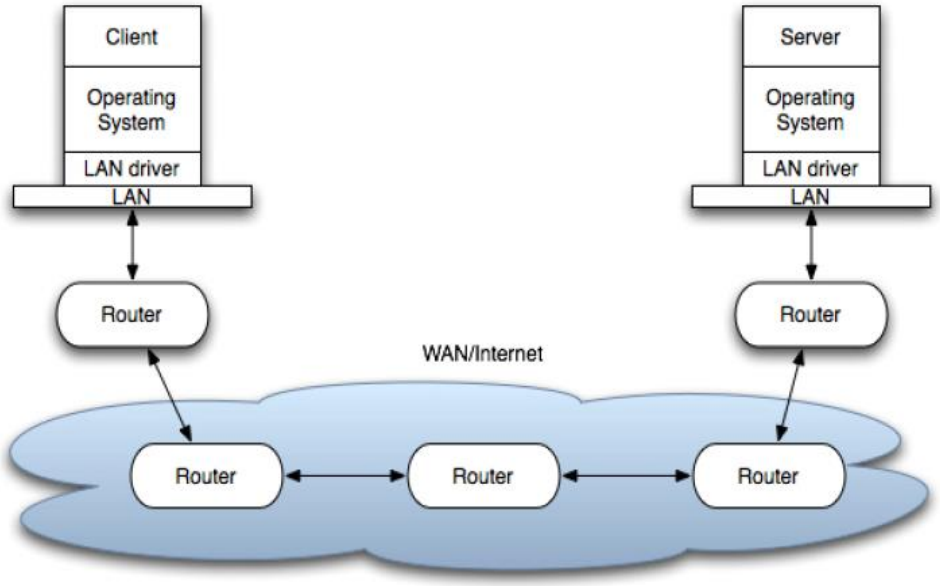
- Clients and servers communicate by means of multiple layers of network protocols

- **Server**:
  - Always on host (must be up and running)
  - Has a static (permanent) IP address
  - Passively waits for and responds to clients
  - Uses passive socket
  - A server is any application that **provides a service** and allows clients to communicate with it.

- **Client:**
  - Initiates the communication
  - Has a dynamic IP address
  - Must know the IP address and the port number of the server in advance
  - Uses an active socket
  - A client is any application that **requests a service** from a server.

TCP & Client-Server Communication Model

- The TCP protocol provides reliable point-to-point communication.
- Using TCP, the client and server must establish a connection in order to communicate.
- To do this, each program binds a socket to its end of the connection.
- A ***socket*** is one endpoint of a two-way communication link between 2 programs running on the network.
- A socket is bound to a port number so that the TCP layer can identify the application to which the data is to be sent.



What Is a Socket?

- A Socket is a software abstraction used to represent the ***terminals*** of a connection between to hosts.
  - A socket is an endpoint for communication between two processes.

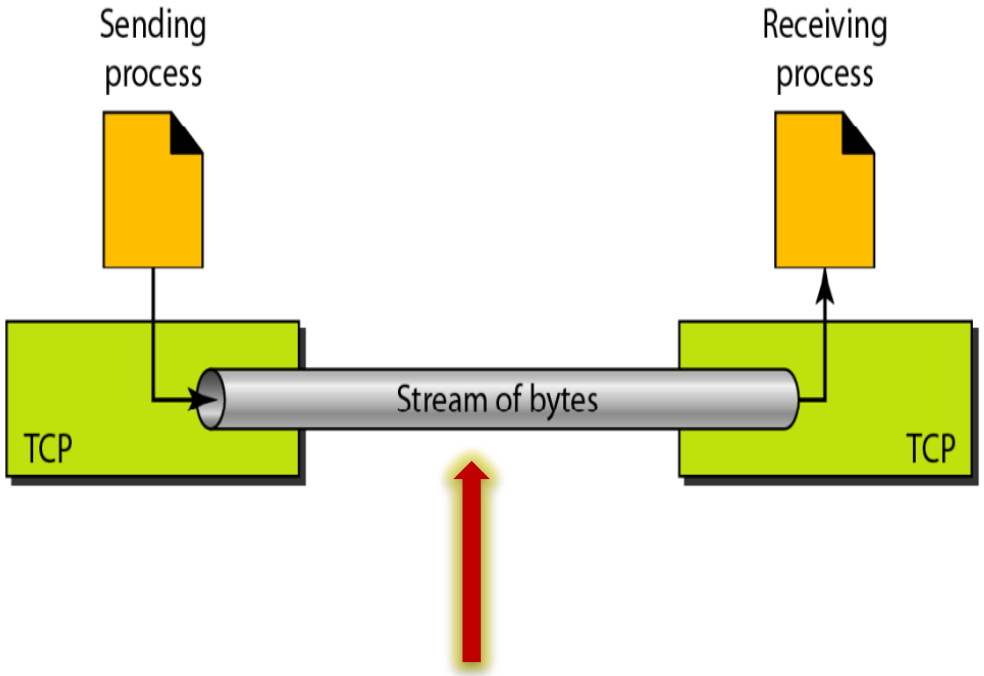
- Sockets enable the ***exchange of information*** between processes on the same machine or across a network, distribute work to the most efficient machine, and easily allow access to centralized data.

- Using a TCP Socket API, Inter-process communication consists of transmitting a ***message*** between a socket in one process and a socket in another process.

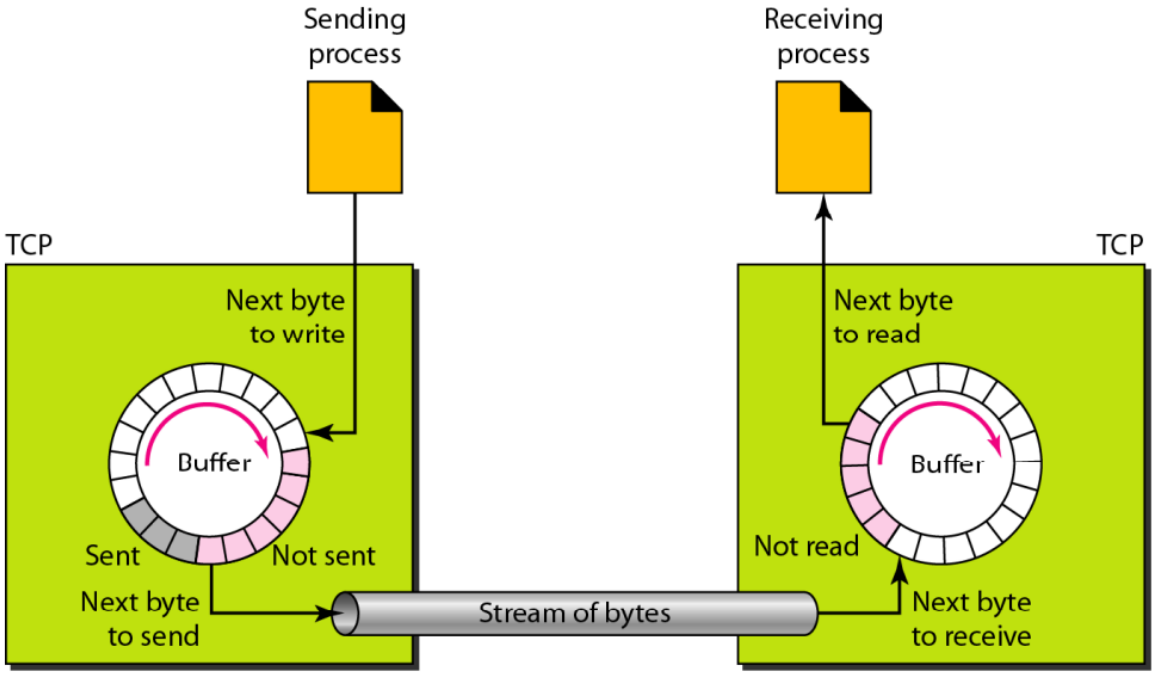
- A socket API is an Inter-processing Communication (IPC) programming interface originally provided as part of the Berkeley UNIX operating system.

TCP Process-to-Process Communication

Data exchange between two processes



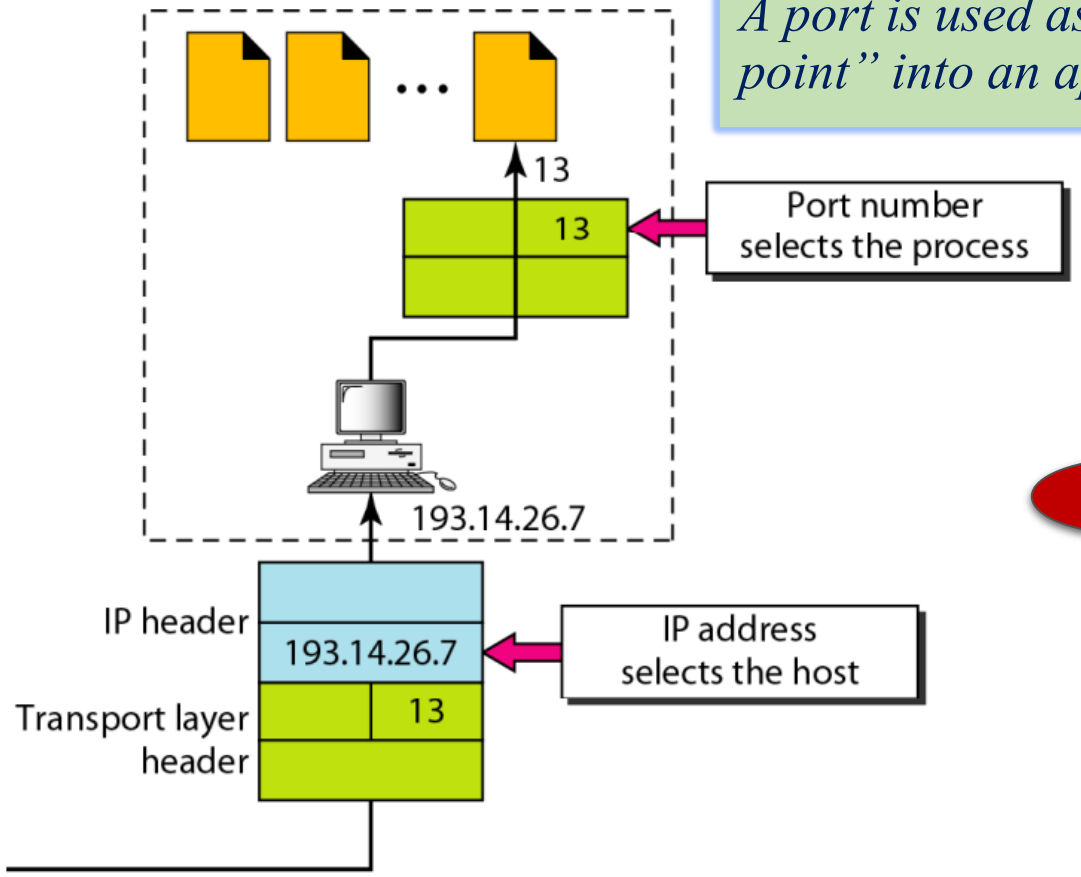
Sending and receiving buffers



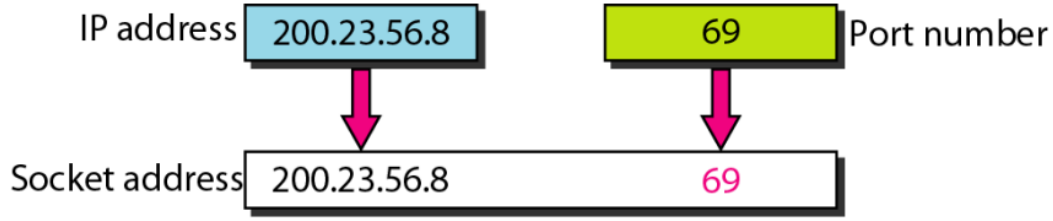
*Most clients and servers communicate by sending **streams of bytes** over connections*

IP Addresses vs Port Numbers vs Socket

A port is used as a gateway or "entry point" into an application.



VS



A socket runs on a specific machine and is bound to a specific port number