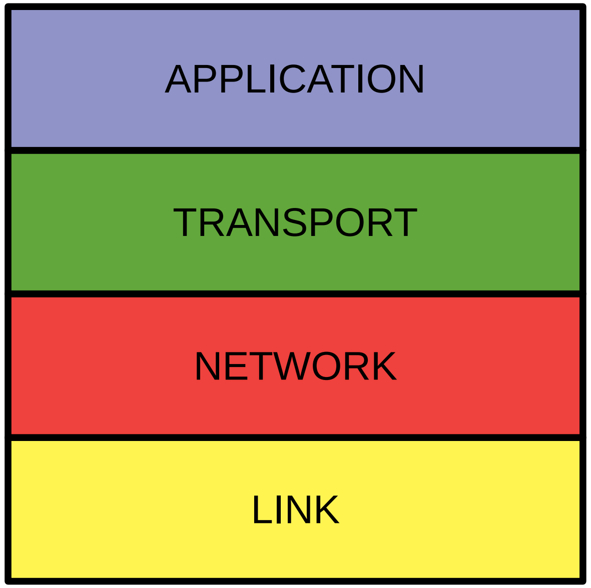
CS 348
Computer Networks
Lec 9

Spring 2020 IIT Goa

Course Instructor: Dr. Neha Karanjkar

Disclaimer: These slides are adapted from Computer Networking: A Top-down Approach by Kurose & Ross, 7th ed. and lecture slides of cs 168-2020 (http://cs168.io/) by Prof. Sylvia Ratnasamy

An overview of things to come …



An overview of things to come ...

LINK LAYER

- Ethernet and WiFi: how they work
- Routing withing local networks using MAC address, ARP
- How the link layer performs Error detection and correction

An overview of things to come …

NETWORK LAYER

- **How are Forwarding tables inside routers populated?**
  - Routing algorithms and protocols used in the Internet
- The IP protocol (IPv4 and IPv6)

An overview of things to come ...

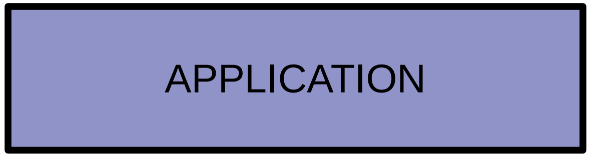
TRANSPORT LAYER

- **How can we implement reliable data transfer over an unreliable channel?**
- TCP, UDP: What they do, how they work
- Congestion control: how TCP does it

An overview of things to come ...

- **What is the “Interface” between applications and the Internet? How can applications use services of the layers below?**
- Some popular applications, how they work, protocols they use:
  - The Web and HTTP, Email, Peer-to-peer applications
- **How can “names” be translated to IP addresses? DNS**

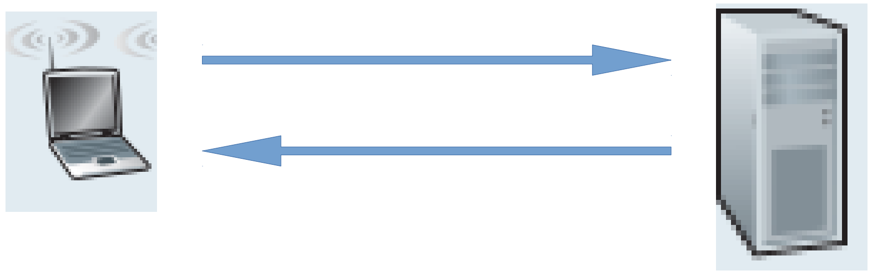
The Application Layer



- **Consists of:** Applications running on hosts communicating over the Internet.

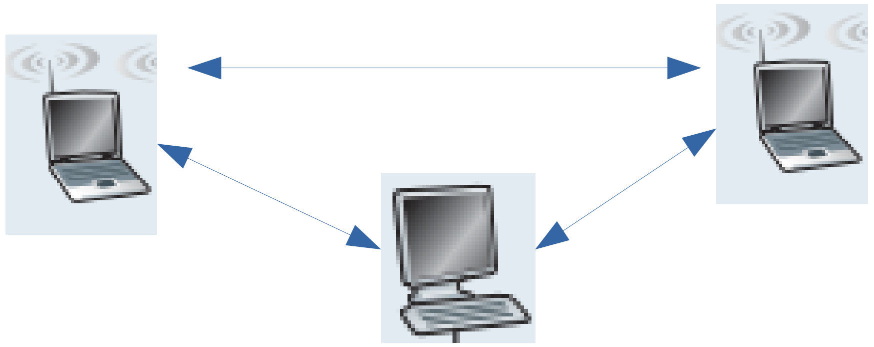
- **What is the “Interface” between applications and the Internet?**
- How can applications use services of the layers below?
  - For Reliable, connection-oriented data transfer? (TCP)
  - For Best-effort, connectionless data transfer? (UDP)

Client-Server Architecture



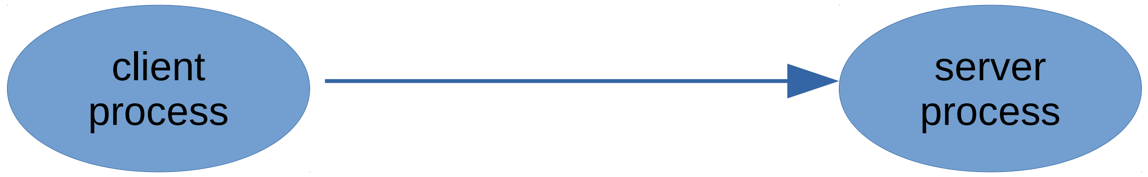
- **Server:** always ON, listening for requests.
  - Has a fixed, well-known IP address
- **Client:** initiates communication with the Server, makes requests
  - Need not have a well-known IP address
- Example: Web-Browser (Client) and Web-Server

Peer-to-Peer (P2P) Architecture



- **Minimal (or no) reliance on “always-ON” servers**
- Direct communication between pairs of hosts (that may have intermittent connectivity)
- Examples: BitTorrent, Skype

Client and Server Processes

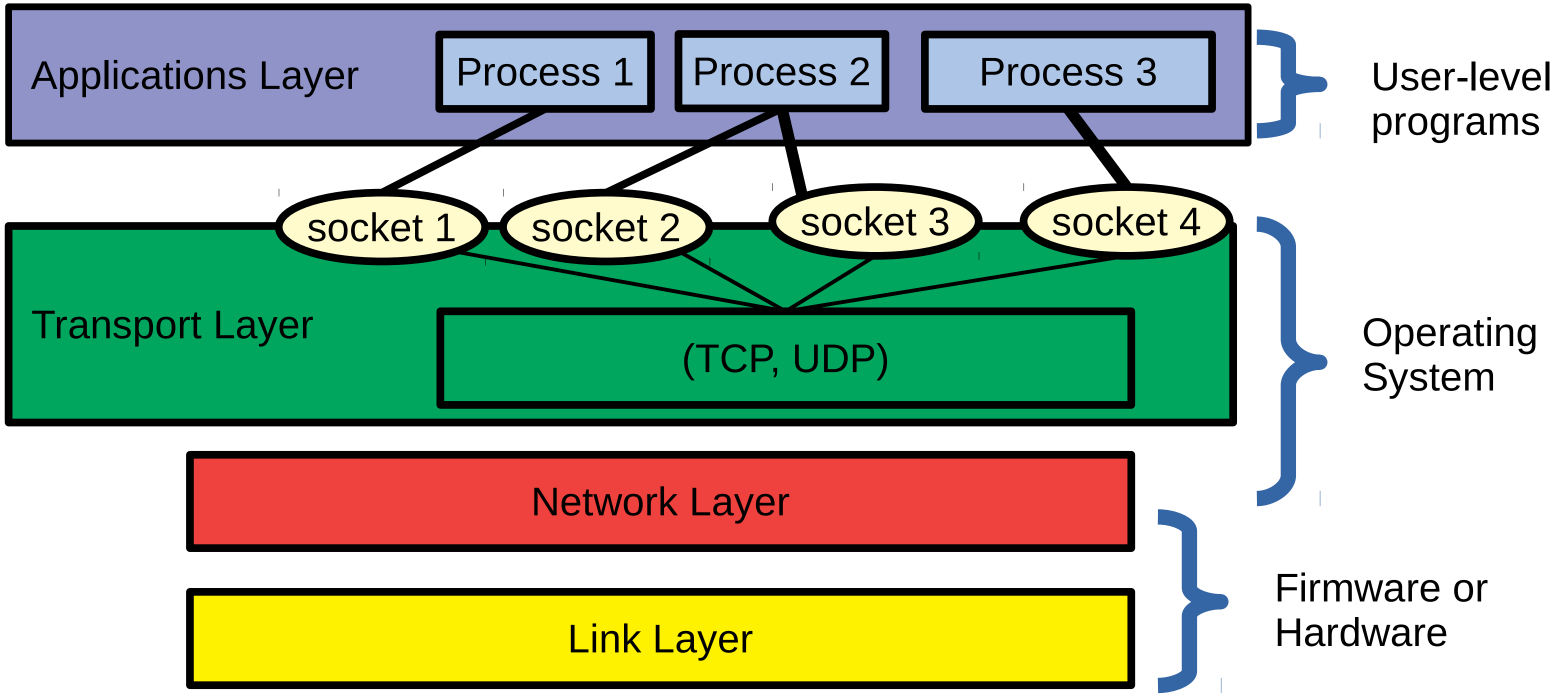


In the context of communicating processes ...

- **Client:** The process that initiates the communication.
  - Client needs to know the address of the server to contact it
- **Server:** The process that waits to be contacted to begin the communication session

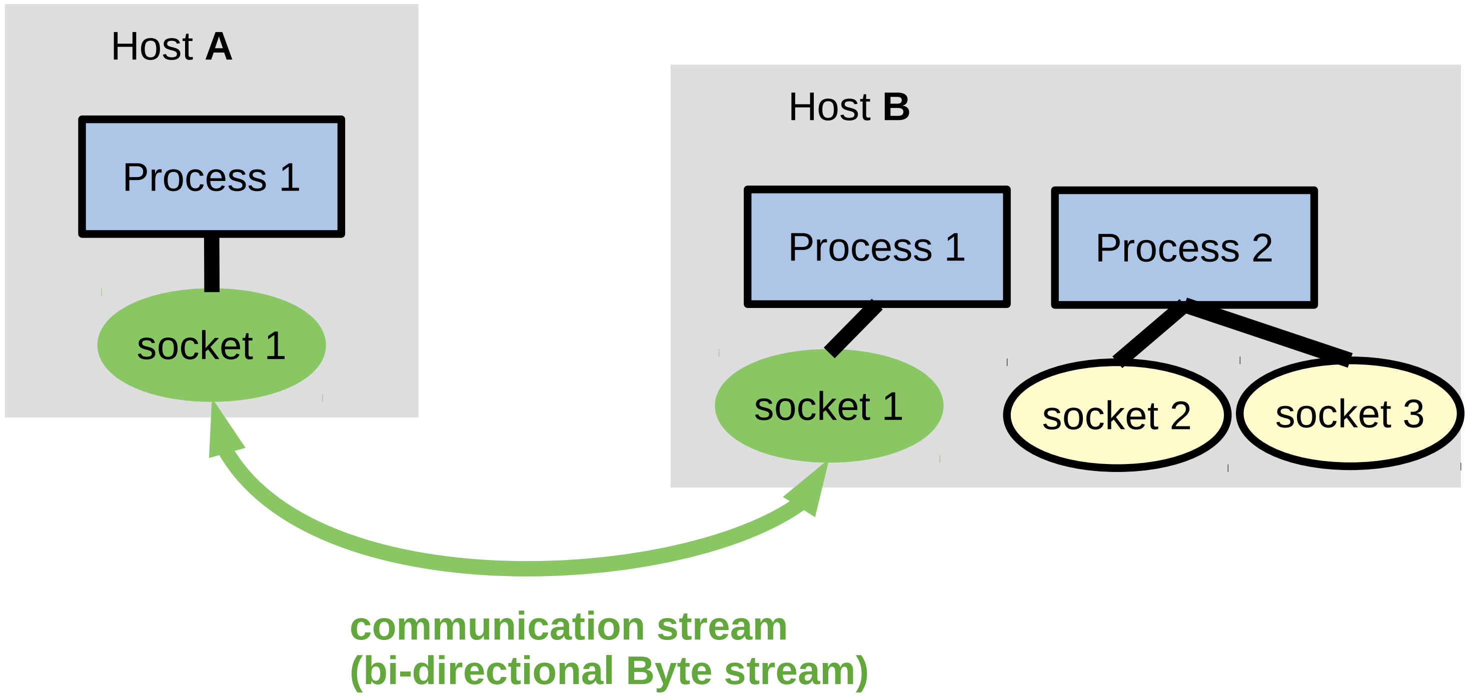
Sockets

- **Sockets serve as an Interface between Applications and the Transport layer**



Sockets

- **Sockets are end-points of a communication stream between two processes**



Sockets

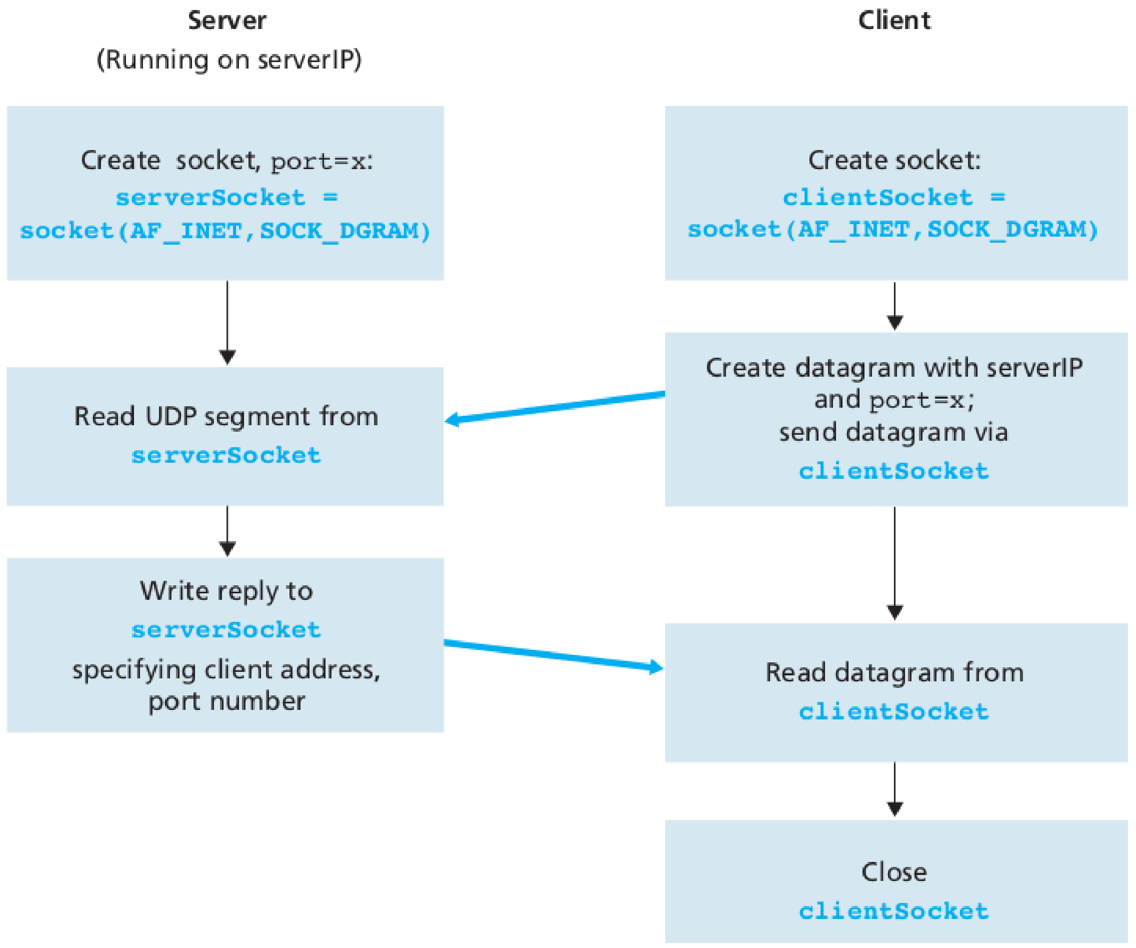
- **But what IS a socket?**
  - An object in memory: “state” variables + some functions such as read(), write()
  - Implemented in the Operating System (kernel)
- Two kinds:
  - **SOCK_STREAM** (For TCP)
  - **SOCK_DGRAM** (For UDP)

Sockets

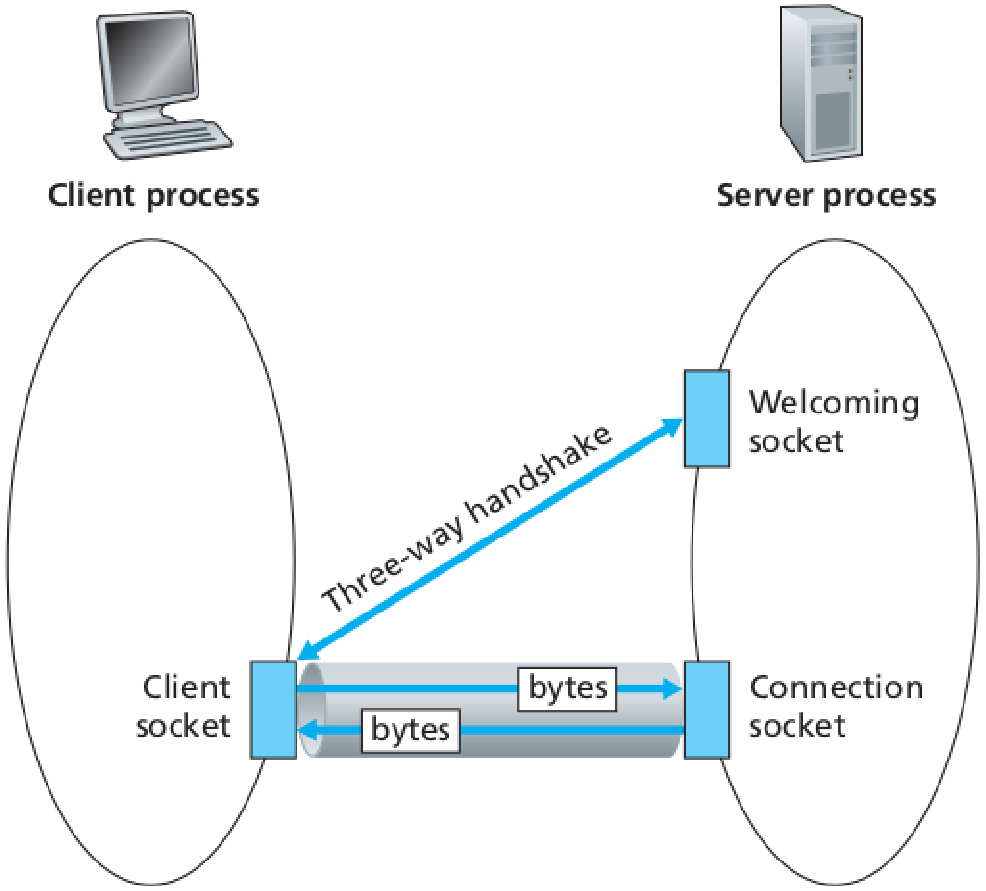
- **To uniquely identify a receiving socket, the sender needs to specify:**
  - **IP address** (uniquely identifies a host)
  - "**Port number**" (uniquely identifies a socket within a host)
    - 16 bit number
    - Specific port-numbers in the range 0-1023 **reserved** for well-known server-side processes (Port 80: HTTP server, port 443: HTTPS, Port 22: SSH)
    - Higher-numbered ports are available for general use by applications (typically clients) and are known as ephemeral ports.

- **Demo:** A Client-Server Example

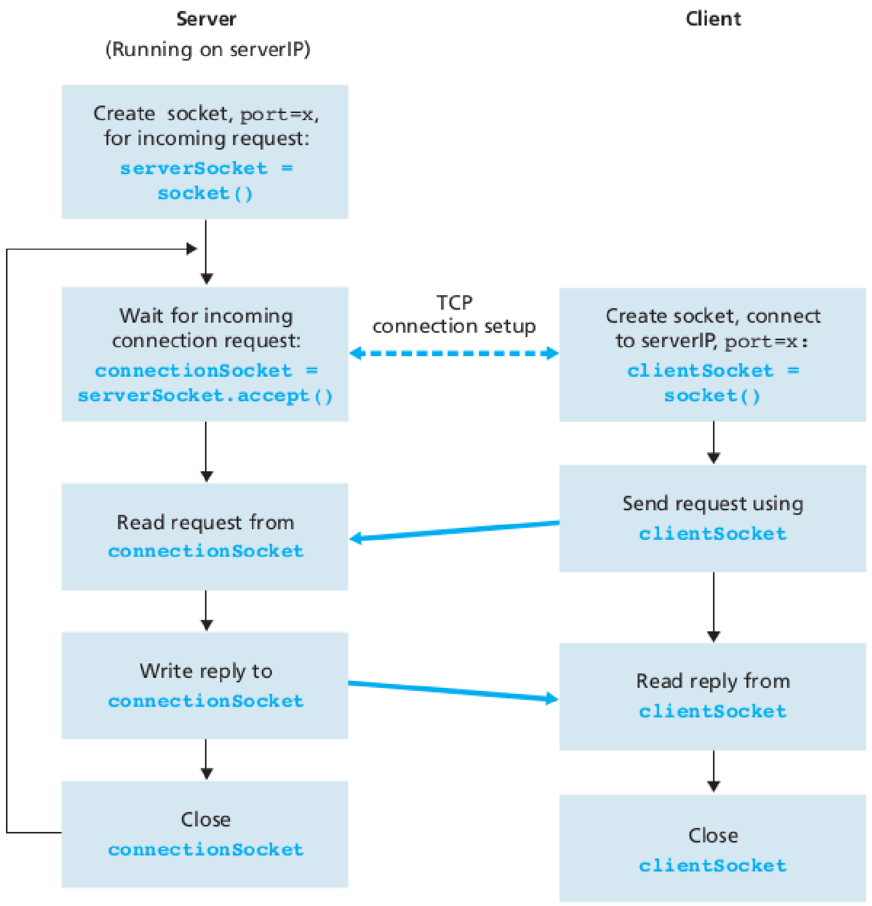
UDP Sockets


Server
(Running on serverIP)
Client
Create socket, port=x:
serverSocket =
socket(AF_INET,SOCK_DGRAM)
Create socket:
clientSocket =
socket(AF_INET,SOCK_DGRAM)
Create datagram with serverIP
and port=x;
send datagram via
clientSocket
Read UDP segment from
serverSocket
Write reply to
serverSocket
specifying client address,
port number
Read datagram from
clientSocket
Close
clientSocket

TCP Sockets


Client process
Server process
Welcoming socket
Three-way handshake
Client socket
bytes
bytes
Connection socket

TCP Sockets


Server
(Running on serverIP)
Client
Create socket, port=x, for incoming request:
serverSocket = socket()
Wait for incoming connection request:
connectionSocket = serverSocket.accept()
TCP connection setup
Create socket, connect to serverIP, port=x:
clientSocket = socket()
Read request from connectionSocket
Send request using clientSocket
Write reply to connectionSocket
Read reply from clientSocket
Close connectionSocket
Close clientSocket

Questions

- How should the applications interpret the Byte stream?
  - Application level protocols: Stateful vs Stateless
- What is the Web? What does it consist of?
- DNS: How are names/urls translated to IP addresses?

Reference and Reading Assignment

- Kurose and Ross 6th ed
  - Section 2.1: Principles of Networked Applications
  - Section 2.7: Socket Programming
- Tutorial on Socket Programming using Python:

  https://realpython.com/python-sockets/