Lecture 4.c

Sockets

Dr. Gabriele Pierantoni

10.02.2021

Sockets
and
Socket Programming

Message Passing: Sockets (1)

Socket

- **is an interface between an application and network that is created by the application**
- **is a construct (file descriptor) defined at OS level that supports read/write data from/to network**
- **initiates and accepts, and terminates connection**

Message Passing: Sockets (1)

Socket

- **acts as an endpoint in the communication where message destinations are specified as socket addresses where each socket address is a communication identifier defined by an Internet address and a port number (the port number distinguishes between services running on the same machine)**
  - **end point determined by**
    - **Host address: *IP address - Network Layer ID***
    - **Port number: *Transport Layer ID***
  - **two end-points determine a connection: socket pair**
    - **ex: 206.62.226.35,p21 + 198.69.10.2,p1500**
    - **ex: 206.62.226.35,p21 + 198.69.10.2,p1499**

Message Passing: Sockets (2)

Socket

- **Example:**
  - **206.62.226.35:80 – Unsecured HTTP Server**
  - **206.62.226.35:443 – Secured HTTPS Server**
  - **206.62.226.35:21 – Unsecured FTP Server**

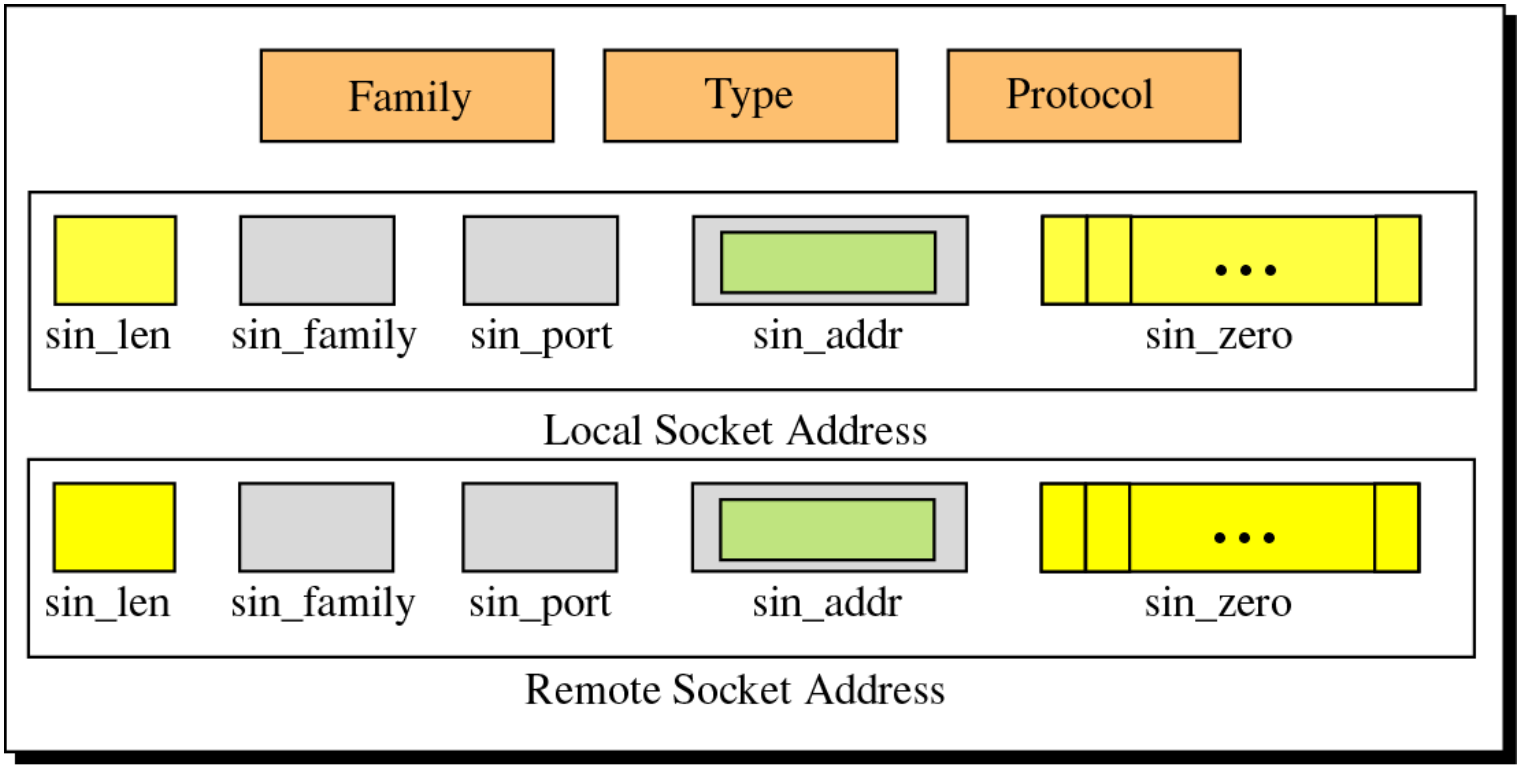
Standard Ports

- http: 80
- https: 443
- ftp: 20/21
- smtp: 25

Message Passing: Sockets (2)

Socket structure

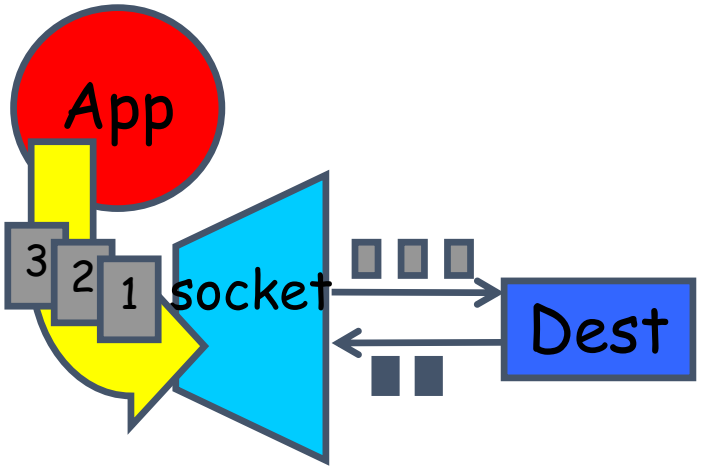
- **Family:** defines the protocol group (IPv4, IPv6, UNIX domain protocols)
- **Type:** defines the exchange-type (stream, packet, raw)
- **Protocol:** TCP/UDP (or IP)
- **Local address:** combination of remote IP and application port address
- **Remote address:** combination of remote IP and application port address



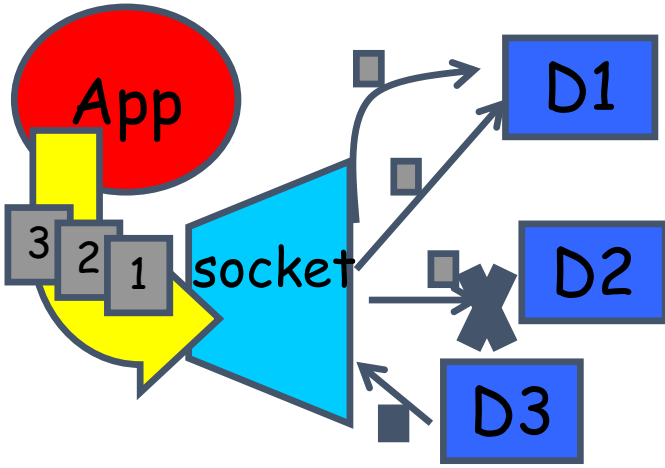
Message Passing: Sockets (3)

Socket types:

- **stream socket: SOCK_STREAM**
  - **connection oriented bi-directional communication**
  - **uses the TCP protocol**
  - **error free (reliable) delivery;**
  - **no out- of- order packets**
  - **applications: telnet/ssh, http, …**

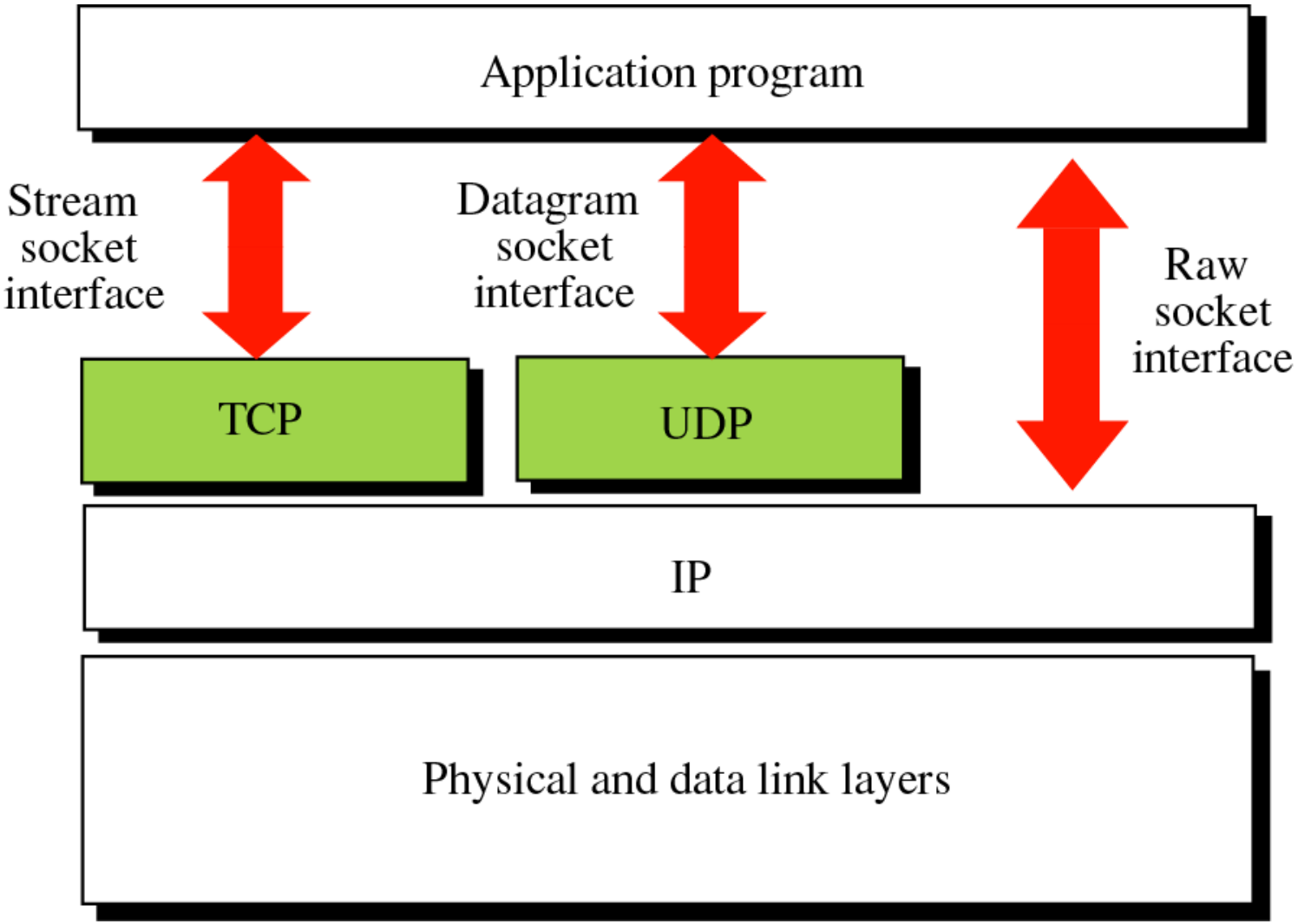


- **datagram socket: SOCK_DGRAM**
  - **connectionless communication**
  - **uses the UDP protocol**
  - **packets may be lost and may arrive out of order**
  - **applications: streaming audio/ video (realplayer), …**

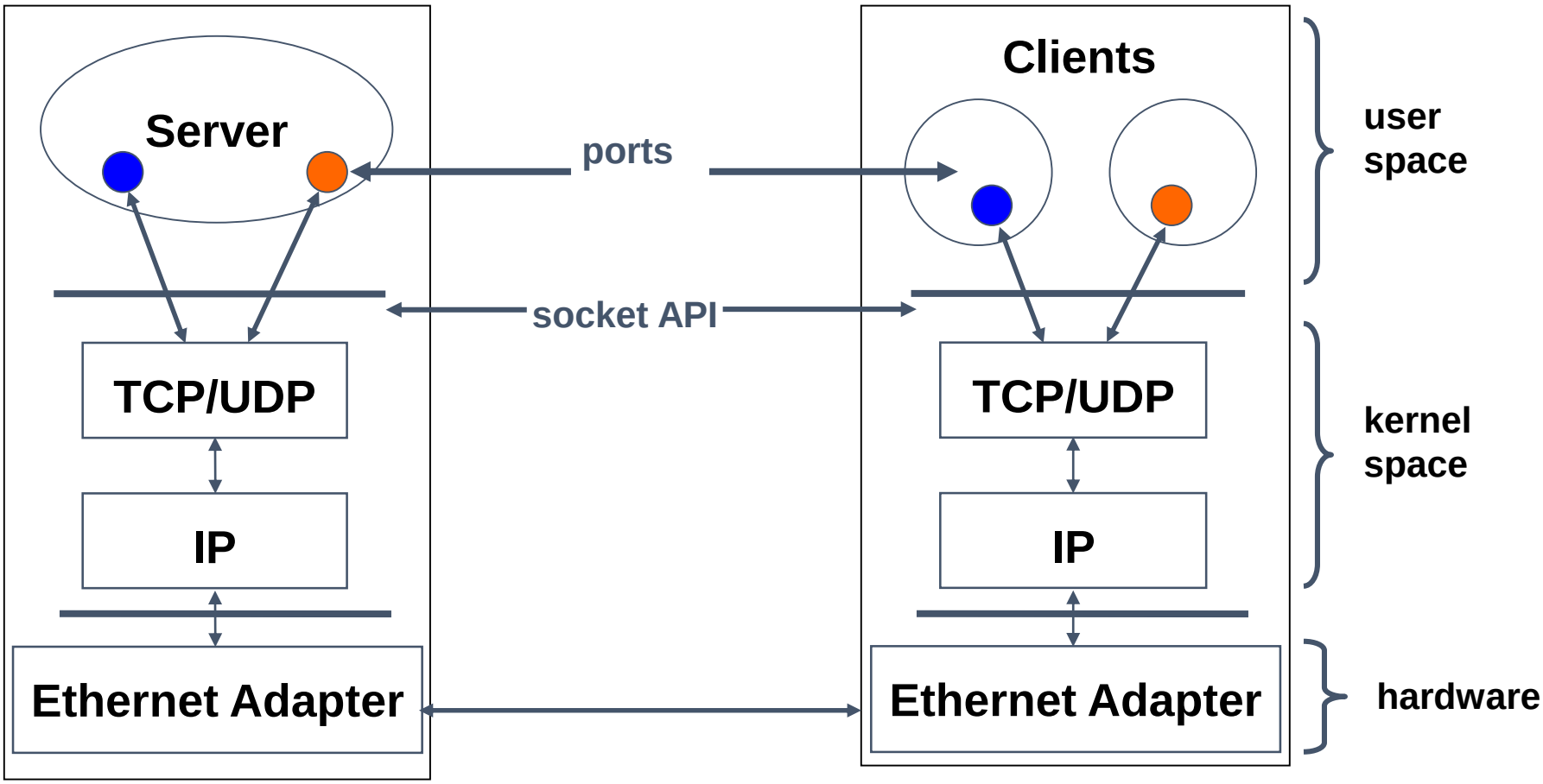


- **raw socket:**
  - **some protocols (ICMP) directly use the service of IP**
  - **raw sockets are used in some applications for performance reasons.**

Message Passing: Sockets (4)



Message Passing:
Socket: Client-Server Communication

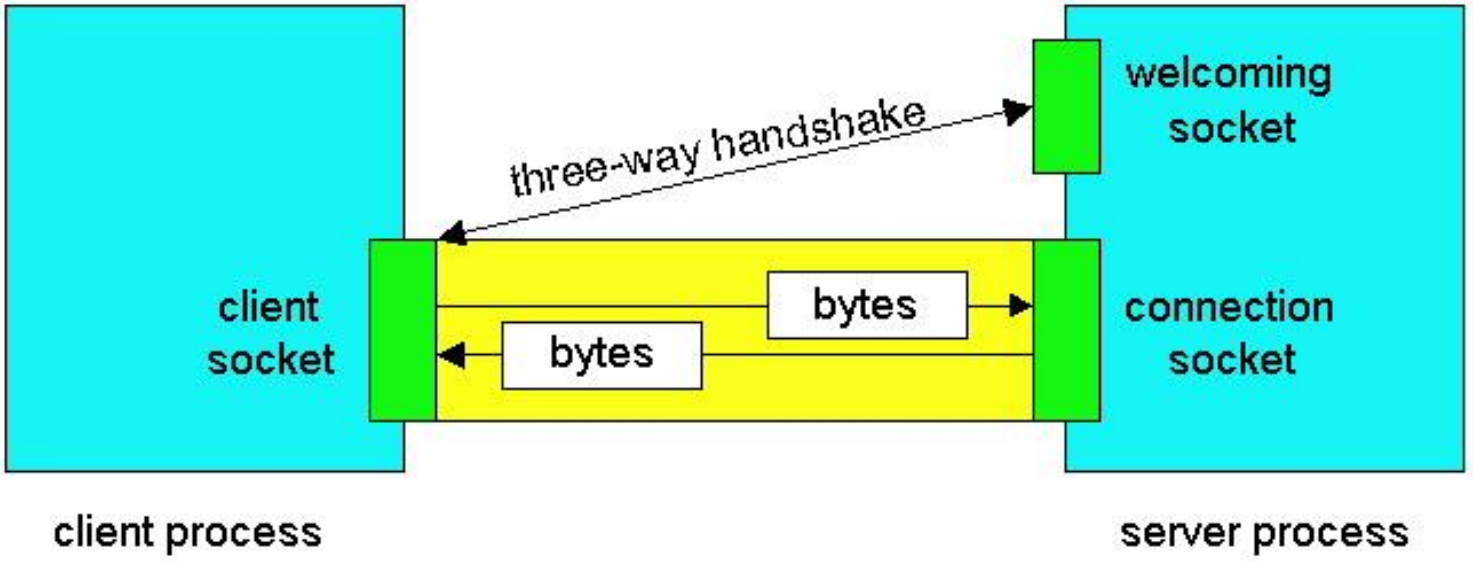


Message Passing: Socket Operations

Client operations	Server operations
Create a socket	Create a socket
Setup the server address	Bind the socket
Connect to the server	Listen for connections
	Accept new client connections
Read/write data	Read/write to client connections
Shutdown connection	Shutdown connection

Message Passing: Berkeley Socket API

socket ():	**create a socket**
bind():	**bind a socket to a local IP address and port #**
listen():	**passively waiting for connections**
connect():	**initiating connection to another socket**
accept():	**accept a new connection**
write():	**write data to a socket**
read():	**read data from a socket**
sendto():	**send a datagram to another UDP socket**
recvfrom():	**read a datagram from a UDP socket**
close():	**close a socket (tear down the connection)**



Message Passing: Berkeley Socket API

socket() creates a new socket of a certain socket type, identified by an integer number, and allocates system resources to it.

bind() is typically used on the server side, and associates a socket with a socket address structure, i.e. a specified local port number and IP address.

listen() is used on the server side, and causes a bound TCP socket to enter listening state.

connect() is used on the client side, and assigns a free local port number to a socket. In case of a TCP socket, it causes an attempt to establish a new TCP connection.

Message Passing: Berkeley Socket API

accept() is used on the server side. It accepts a received incoming attempt to create a new TCP connection from the remote client, and creates a new socket associated with the socket address pair of this connection.

send() and **recv()**, or **write()** and **read()**, or **sendto()** and **recvfrom()**, are used for sending and receiving data to/from a remote socket.

close() causes the system to release resources allocated to a socket. In case of TCP, the connection is terminated.

Message Passing: Berkeley Socket API

gethostbyname() and **gethostbyaddr()** are used to resolve host names and addresses. IPv4 only.

select() is used to suspend, waiting for one or more of a provided list of sockets to be ready to read, ready to write, or that have errors.

poll() is used to check on the state of a socket in a set of sockets. The set can be tested to see if any socket can be written to, read from or if an error occurred.

getsockopt() is used to retrieve the current value of a particular socket option for the specified socket.

setsockopt() is used to set a particular socket option for the specified socket.

Message Passing: TCP Client-Server Communication (2)