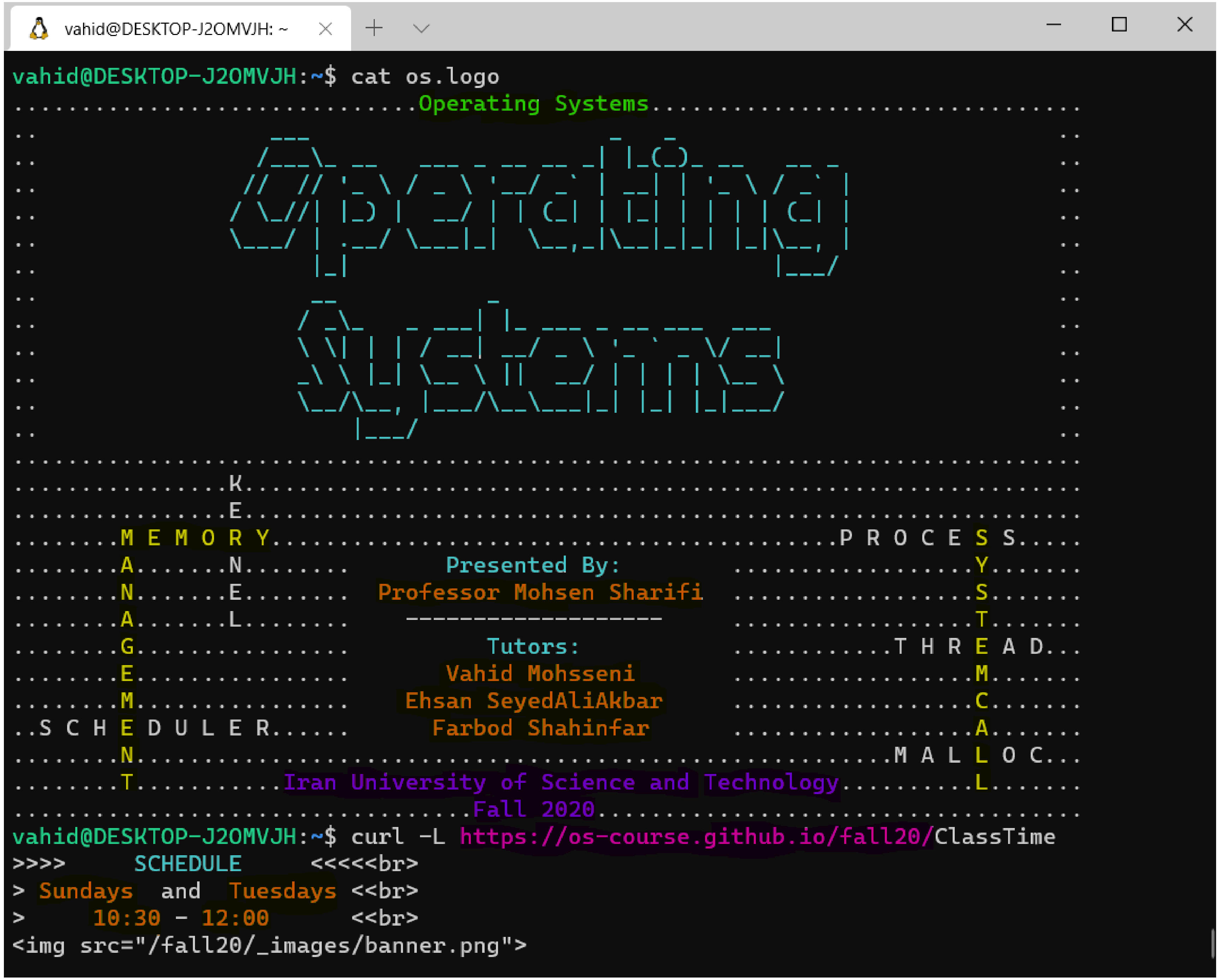
Operating Systems

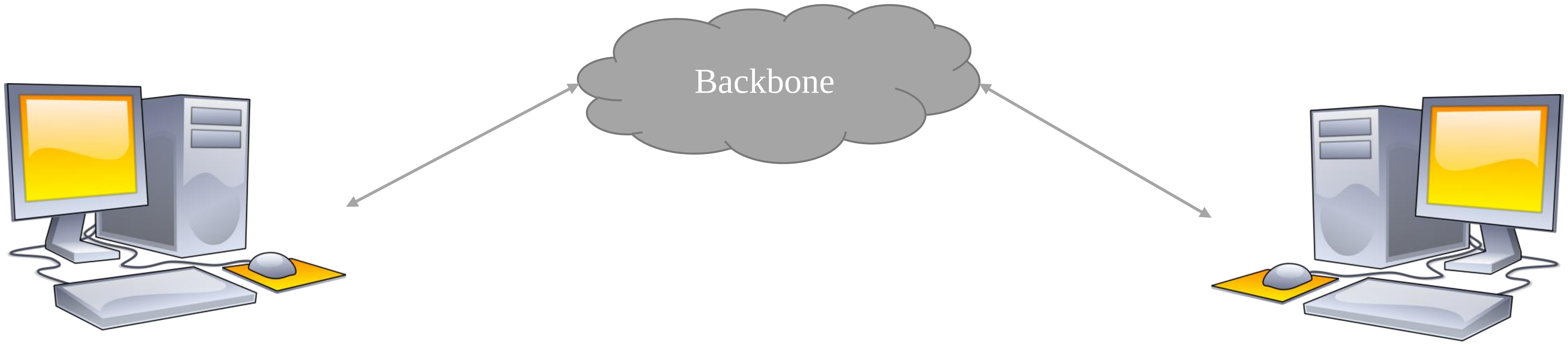
Socket Programming

Fall 2020

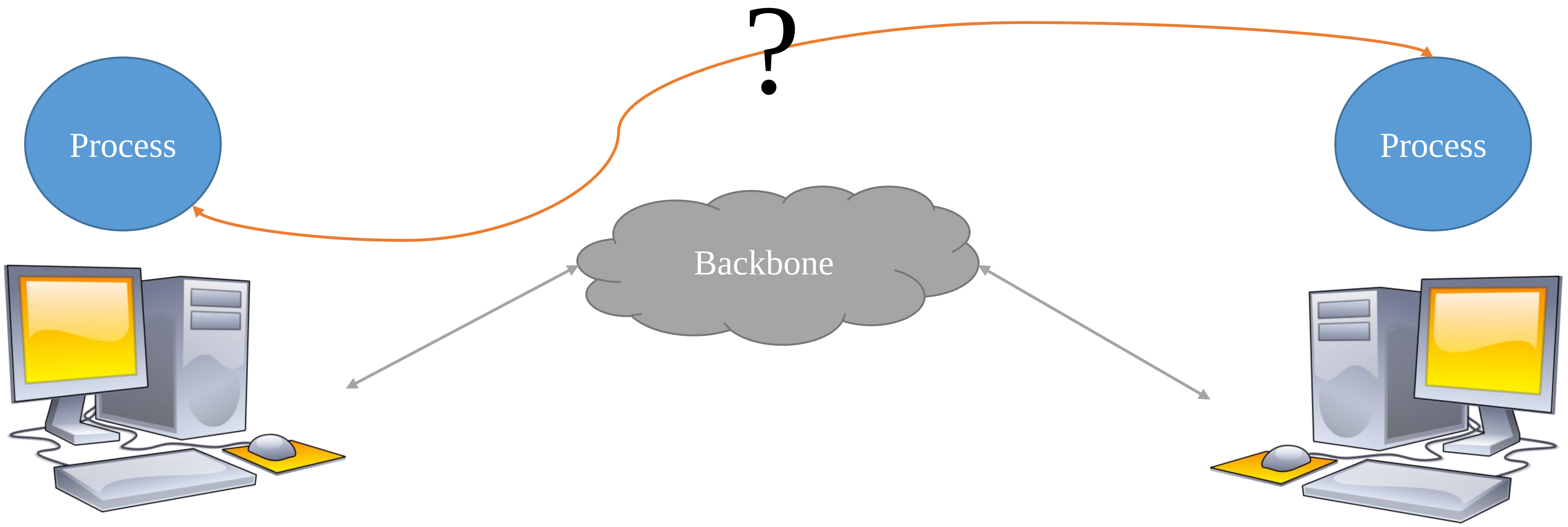


```
vahid@DESKTOP-J2OMVJH:~$ cat os.logo
..............................Operating Systems................................
..                 ___                      _   _                            ..
..                /___\_ __   ___ _ __ __ _| |_(_)_ __   __ _                ..
..               //  // '_ \ / _ \ '__/ _` | __| | '_ \ / _` |               ..
..              / \_//| |_) |  __/ | | (_| | |_| | | | | (_| |               ..
..              \___/ | .__/ \___|_|  \__,_|\__|_|_| |_|\__, |               ..
..                    |_|                               |___/                ..
..                    __           _                                         ..
..                   / _\_   _ ___| |_ ___ _ __ ___  ___                     ..
..                   \ \| | | / __| __/ _ \ '_ ` _ \/ __|                    ..
..                   _\ \ |_| \__ \ ||  __/ | | | | \__ \                    ..
..                   \__/\__, |___/\__\___|_| |_| |_|___/                    ..
..                       |___/                                               ..
...............................................................................
................K..............................................................
................E..............................................................
........M E M O R Y..........................................P R O C E S S.....
........A.......N........       Presented By:        ..................Y.......
........N.......E........  Professor Mohsen Sharifi  ..................S.......
........A.......L........    -------------------     ..................T.......
........G................          Tutors:           ............T H R E A D...
........E................       Vahid Mohsseni       ..................M.......
........M................    Ehsan SeyedAliAkbar     ..................C.......
..S C H E D U L E R......      Farbod Shahinfar      ..................A.......
........N........................................................M A L L O C...
........T...........Iran University of Science and Technology..........L.......
..................................Fall 2020....................................
vahid@DESKTOP-J2OMVJH:~$ curl -L https://os-course.github.io/fall20/ClassTime
>>>>     SCHEDULE     <<<<<br>
> Sundays  and  Tuesdays <<br>
>     10:30 - 12:00      <<br>
<img src="/fall20/_images/banner.png">
```

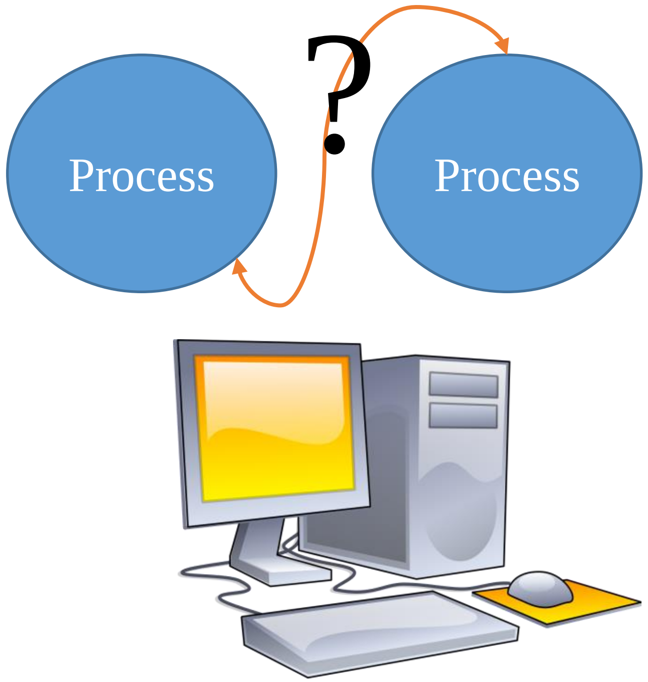
Goal



Goal



Goal



Solution

We need something to establish a connection between processes.

What is this connection?

Inter-Process Communication

One way is: using Sockets

History

Berkley Sockets

- Released on 4.2BSD Unix OS in 1983.
- Programming Interface
- All Modern OS implemented a version of Berkeley Socket interface.
- It became the standard interface for the applications running on internet.
- Written in C, other programming languages using a wrapper library on C APIs.

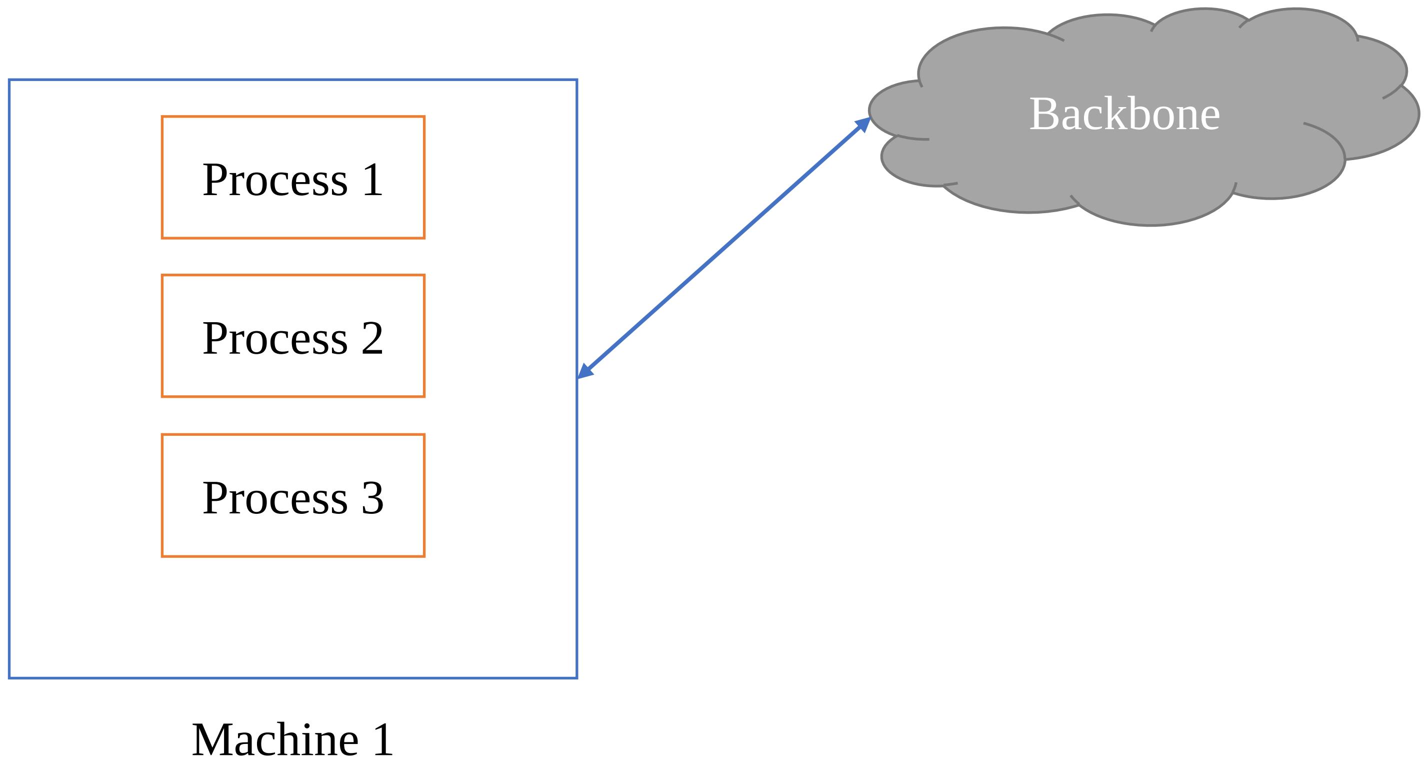
Berkley Sockets

Known as Sockets

It is an abstraction through which an application may send and receive data.

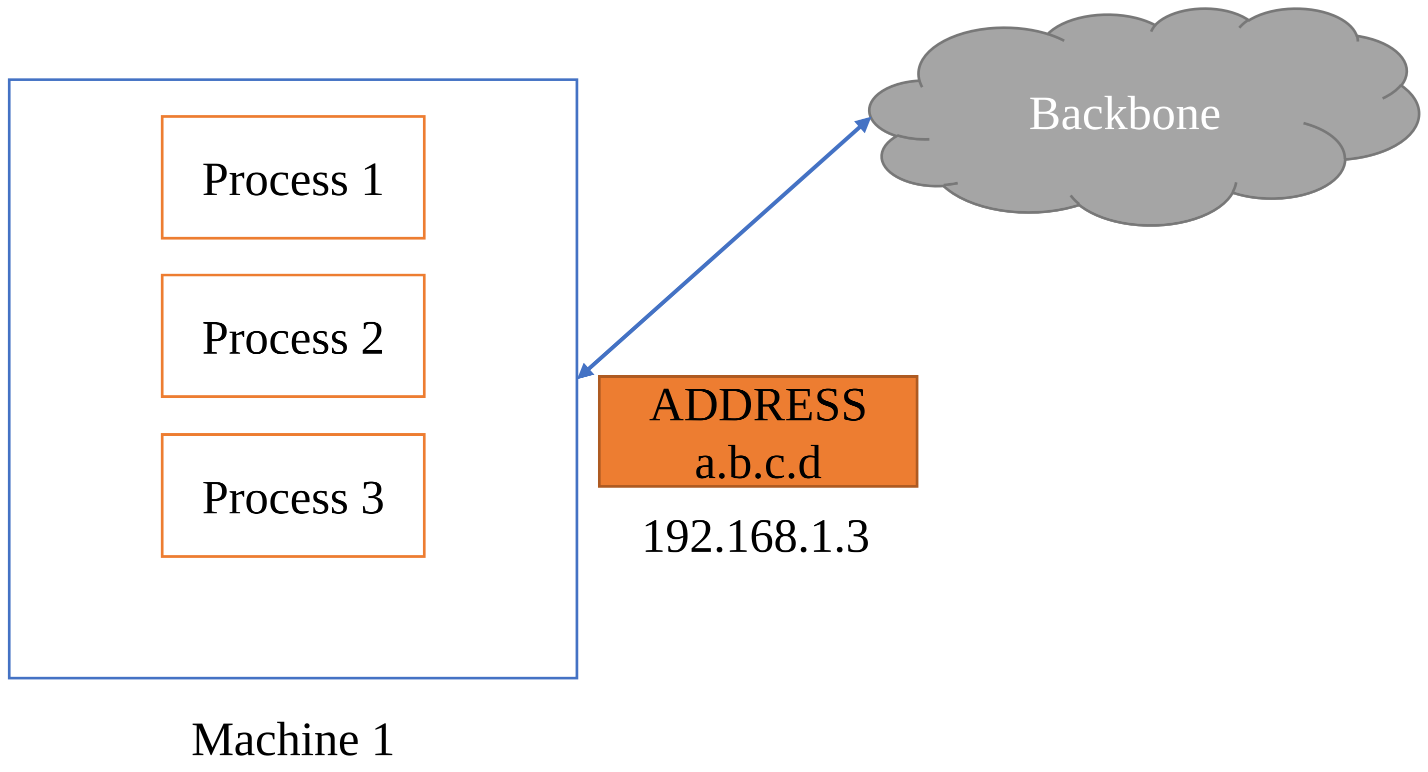
Standard API for networking

Addressing



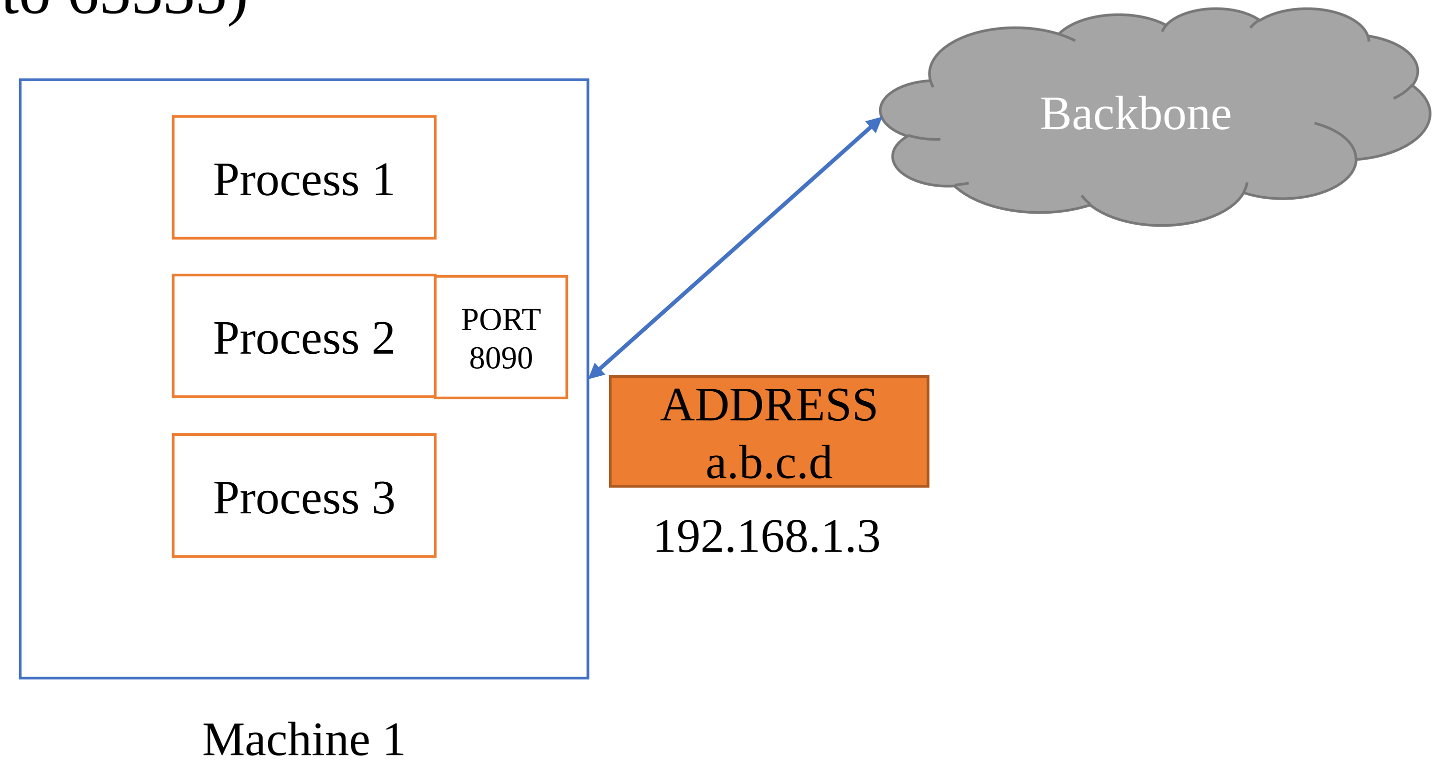
Addressing

- IP



Addressing

- PORT (0 to 65535)



- end-to-end transport

UDP vs. TCP

User Datagram Protocol

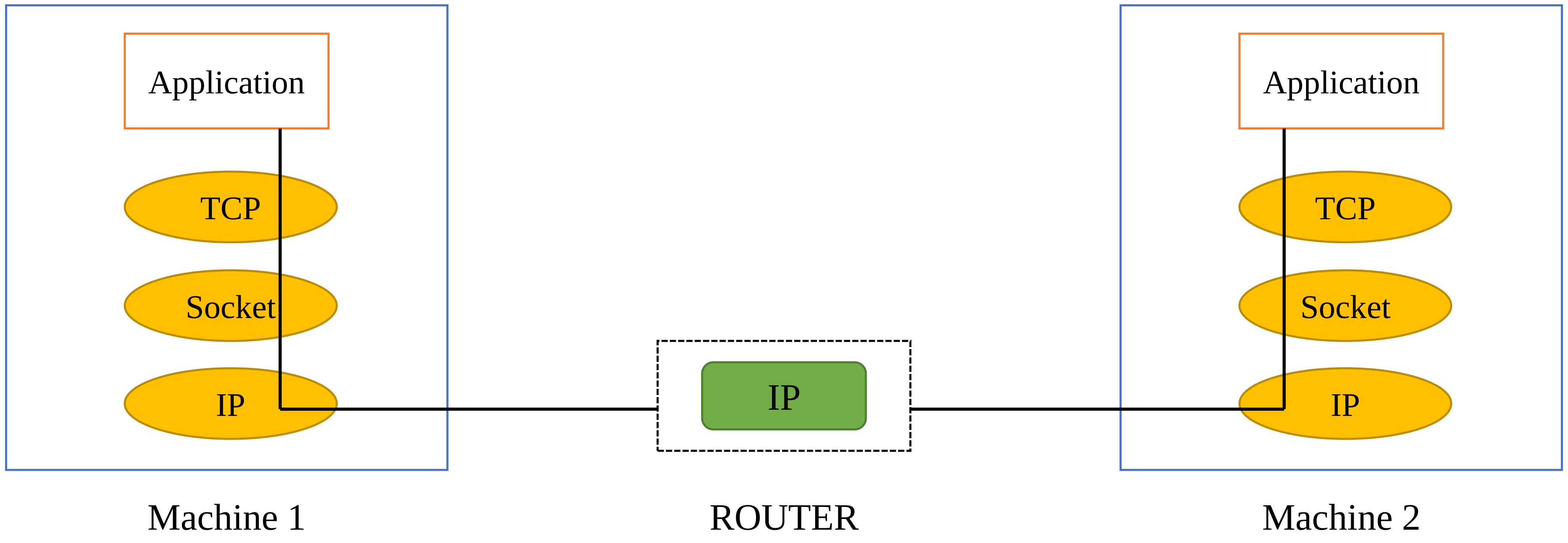
- connectionless
- out of order
- no care about if packet received or not!
- no retransmissions

Transmission Control Protocol

- reliable byte-stream channel (in-order, all arrive, no duplicate)
- flow control
- connection-oriented
- bidirectional

Addressing

- ROUTER



Primitives

Primitive	Description
Socket	Creates a new communication end point with certain type.
Bind	Attaches a local address socket.
Listen	Announces the willingness to accept connections.
Accept	Waits for a connection and accepts if one arrives.
Connect	Attempts to establish connection.
Send	Sends some data over the connection.
Receive	Receive some data over the connection.
Close	Releases the connection.

socket

- creates an endpoint and returns a file descriptor for the socket
- three arguments:
    - *domain* -> protocol family i.e. IP4, IP6
        - `AF_INET` IPv4
        - `AF_INET` IPv6
        - `AF_UNIX` local socket
    - *type*
        - `SOCK_STREAM`
        - `SOCK_DGRAM`
    - *protocol* -> explicitly specifies the protocols, if 0 passed then domain protocol will be used.

bind

- relate a socket with an address
- three arguments:
  - *sockfd* -> file descriptor of the socket
  - *my_addr* -> a pointer to `sockaddr` structure representing the address
  - *addrlen* -> a field of type `socklen_t` specifies the size of `sockaddr`

listen

- prepares socket for incoming connections.
- two arguments:
  - *sockfd* -> file descriptor of the socket
  - *backlog* -> an integer value representing the number of pending connections at any one time.

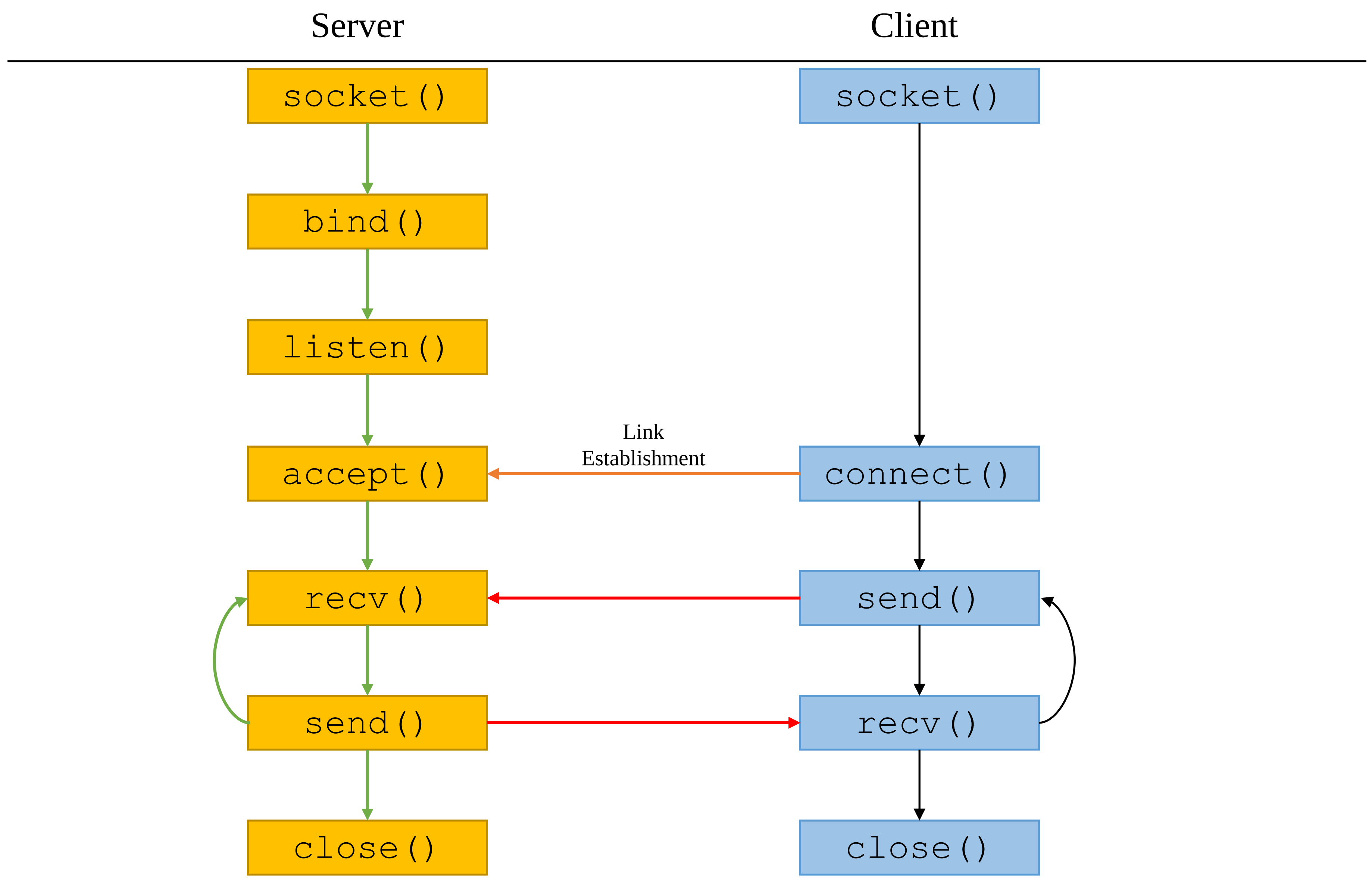
accept

- used in stream-oriented sockets.
- it creates a new socket for each new connection that arrive to host.
- returns new socket descriptor for arrival connection.
- three arguments:
  - *sockfd* -> file descriptor of the socket
  - *cliaddr* -> a pointer to a sockaddr structure to receive the client's address information.
  - *addrlen* -> a pointer to a `socklen_t` location that specifies the size of the client address structure passed to `accept()`.

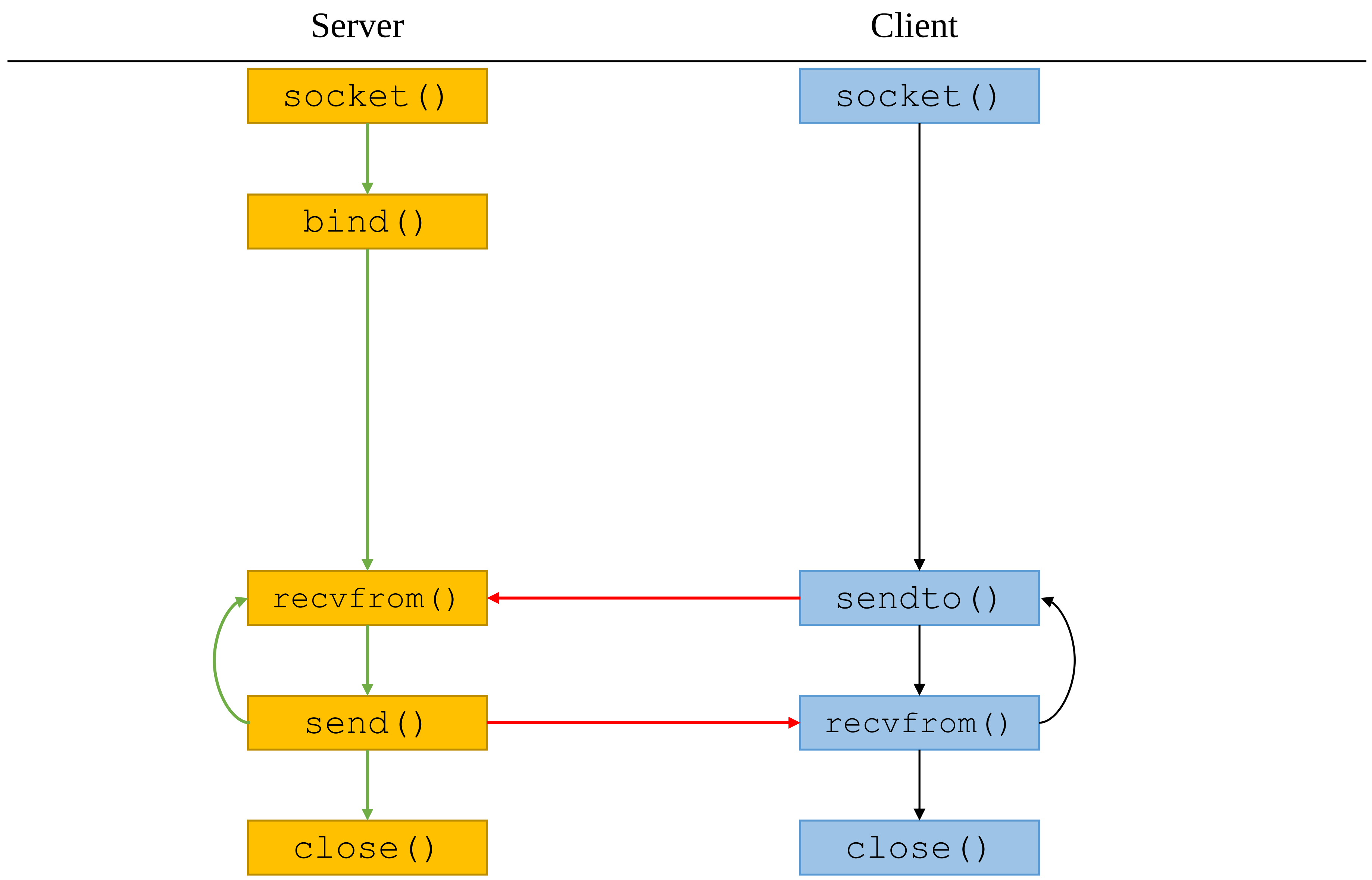
connect

- establishes a direct communication link to a remote host.
- three arguments:
  - *sockfd* -> file descriptor of the socket.
  - *sockaddr* -> a pointer to a `sockaddr` structure to receive the host's address information.
  - *addrlen* -> a pointer to a `socklen_t` location that specifies the size of the host address structure passed to `connect()`.

Client-Server Model - TCP



Client-Server Model - UDP



Let’s see examples

go to our repository…

if already cloned before, just `git pull` now.

https://github.com/os-course/iustfall20/tree/master/08_socket_example

Questions?

?