Cornell University

How to the communication of distributed system over the Network

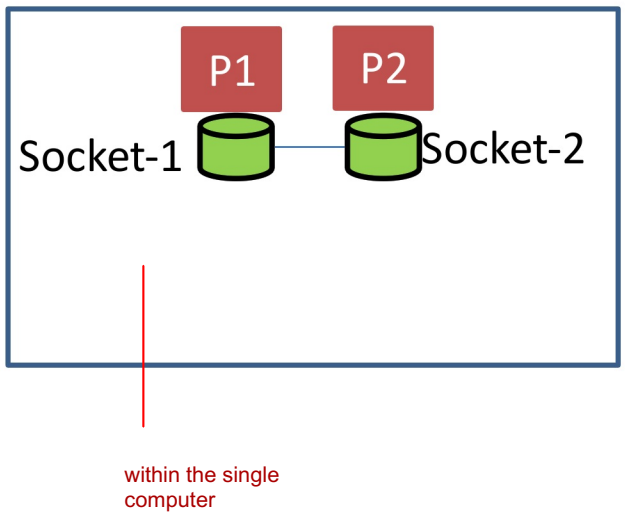
Introduction to Socket Programming

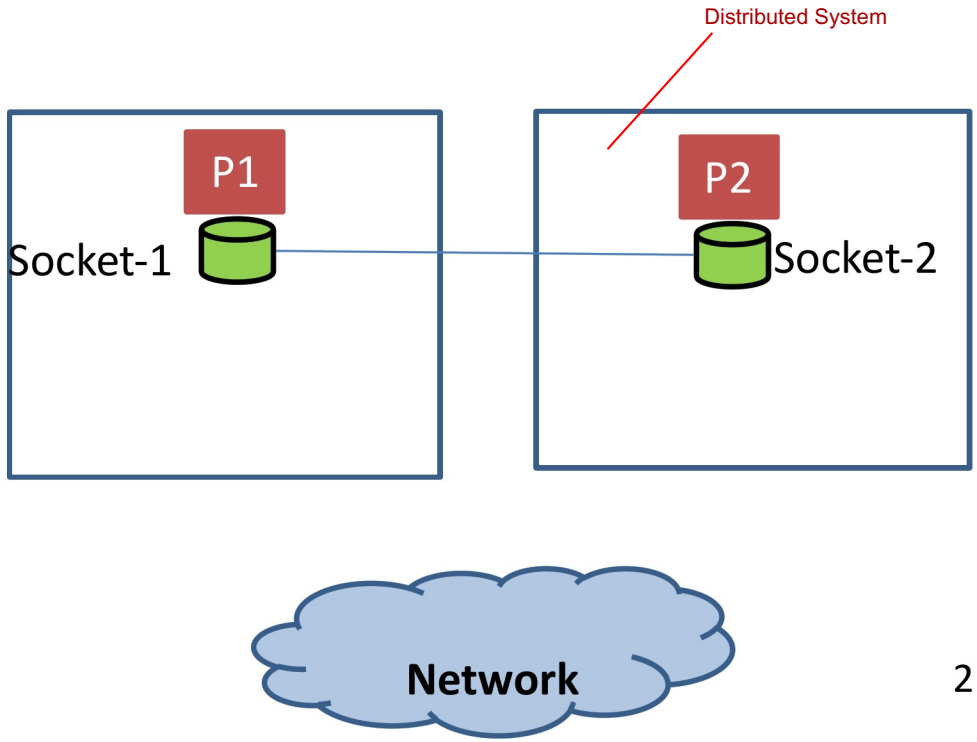
CS 4450 (Spring 2018)

Vishal Shrivastav, Cornell University

What is a Socket?

- A socket is a method for accomplishing inter-process communication (IPC)
  - Allows one process to communicate with another process on the same or different machine





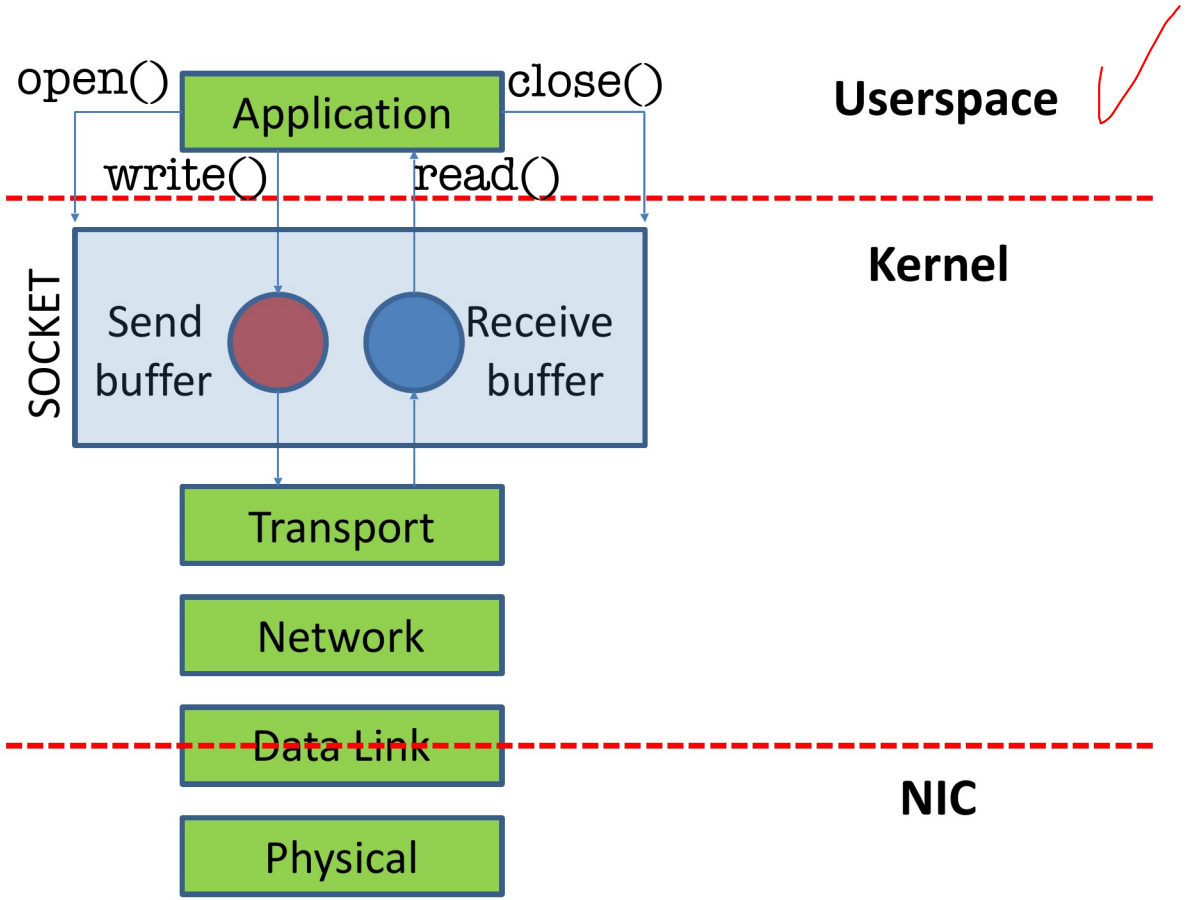
Operations on a Socket

operation Similar to the file handling but all operation with buffer inside the client and server

- Socket works very similar to a file
  - `open()` -- open a socket
  - `read()` -- read from a socket (analogous to receive data)
  - `write()` -- write to a socket (analogous to send data)
  - `close()` -- close the socket

Where does Socket fit in the Network Stack?

All the buffer on the aplication layer insde the kernal space



Blocking and Non-blocking Sockets

- By default `read()` and `write()` operations are blocking
  - Function does not return until the operation is complete

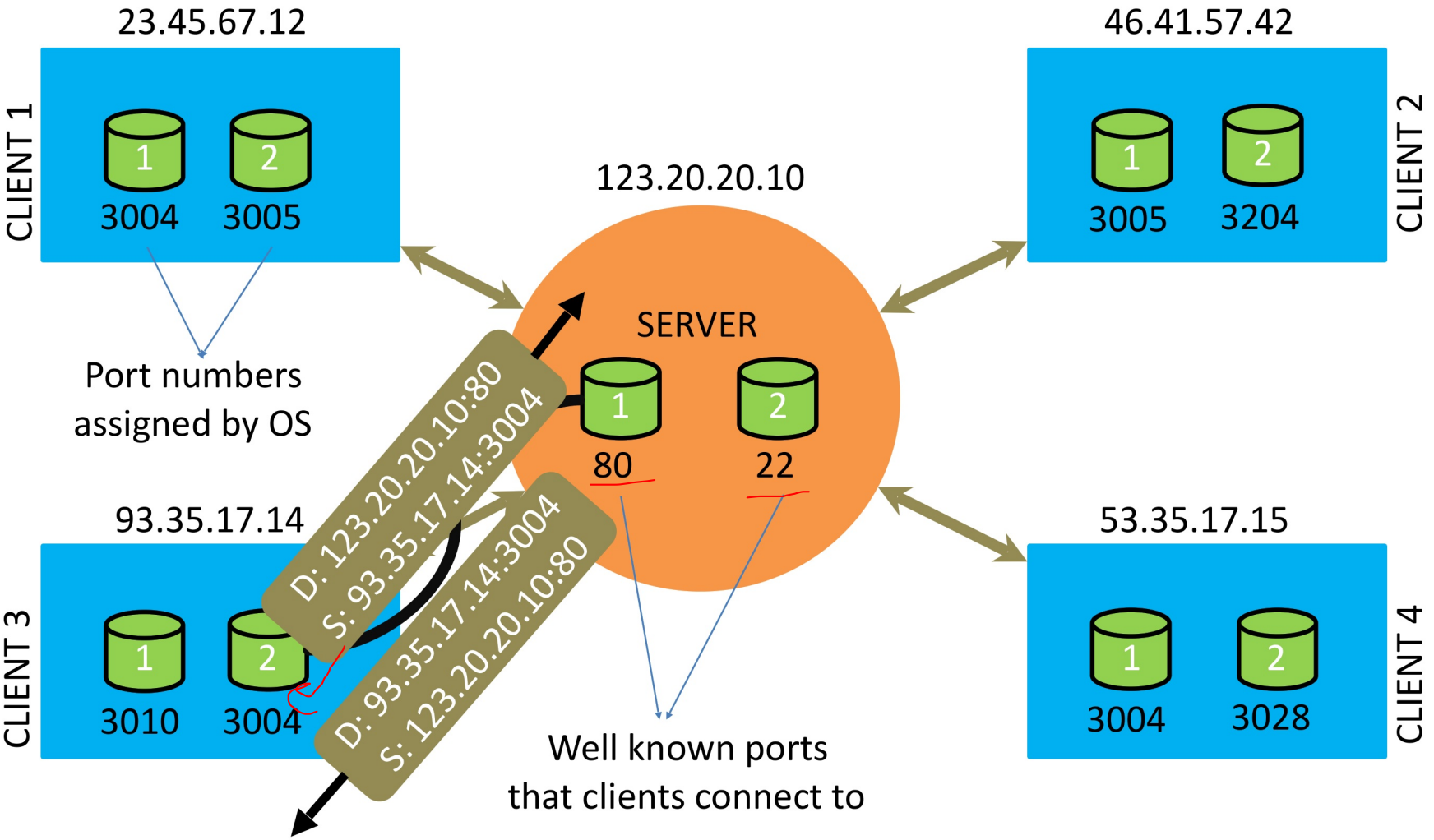
- `read()` blocks until there is some data available in the receive buffer

- When does `write()` block?
  - When the send buffer is full

Blocking and Non-blocking Sockets

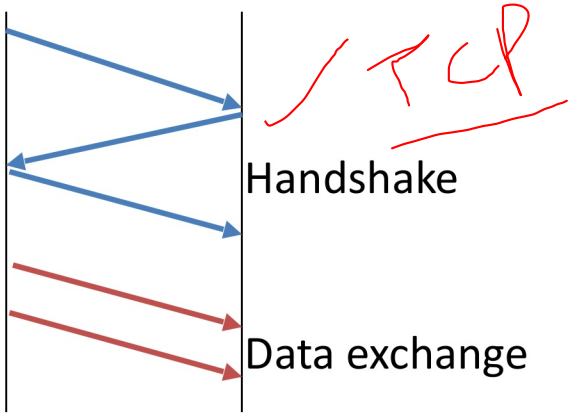
- Non-blocking `read()` and `write()` return immediately
- `read()`
  - If there is some data in receive buffer, `read()` succeeds and returns the amount of data read
  - If the receive buffer is empty, `read()` returns the ERROR code
- `write()`
  - If there is some space available in the send buffer, `write()` succeeds and returns the amount of data written
  - If the send buffer is full, `write()` returns the ERROR code

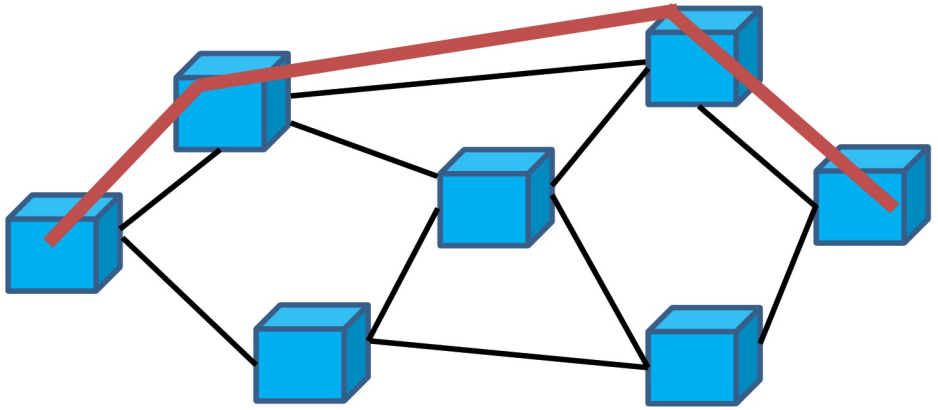
Client-Server Model



Two traditional modes of communication

- Connection-oriented Communication
  - Establish a logical or physical connection before exchanging data

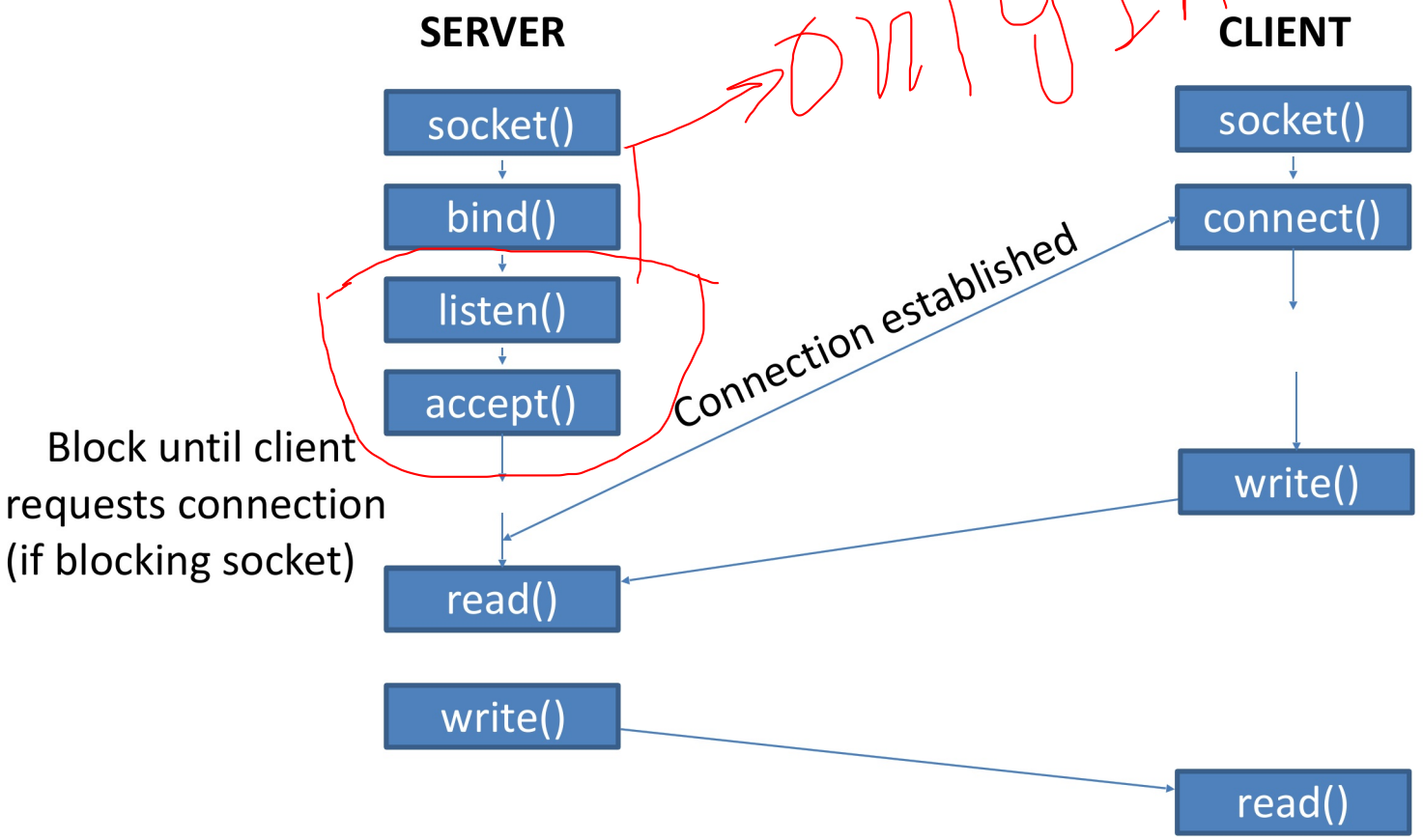




- Connectionless Communication
  - Start exchanging data without any prior arrangements between endpoints

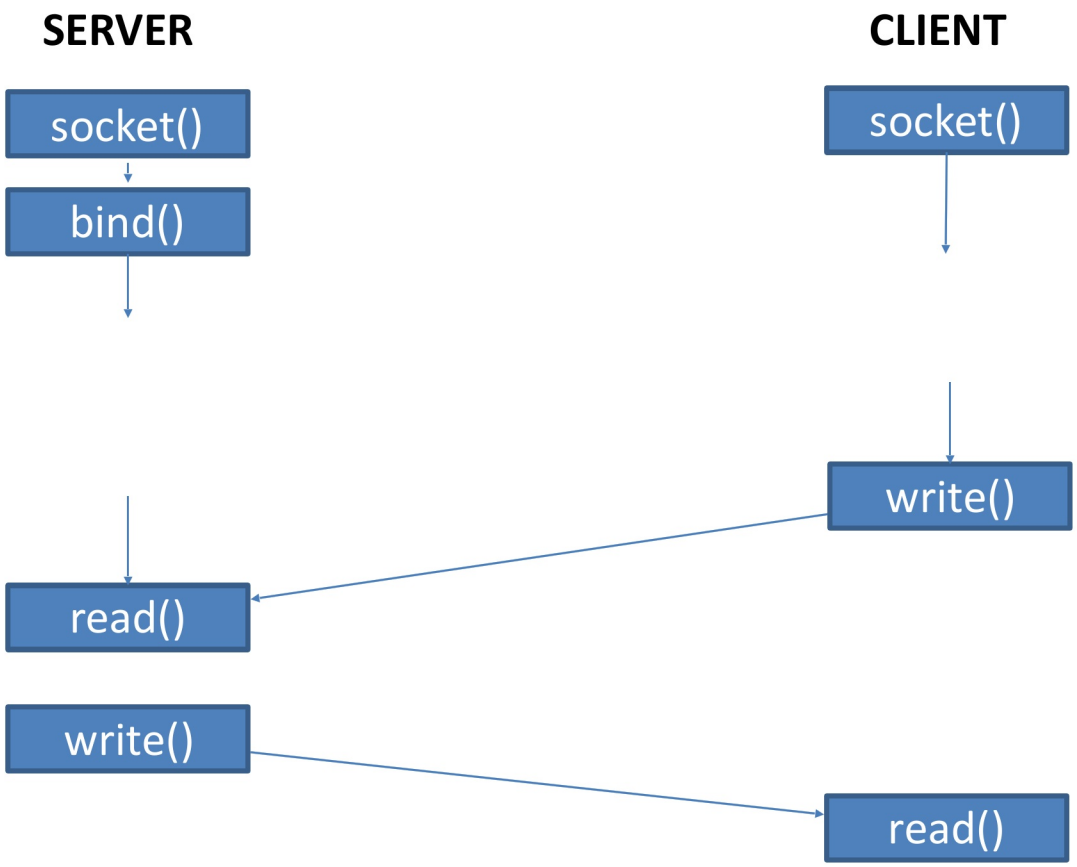
Client-Server Model - APIs

- Connection-oriented protocol (TCP-suite)



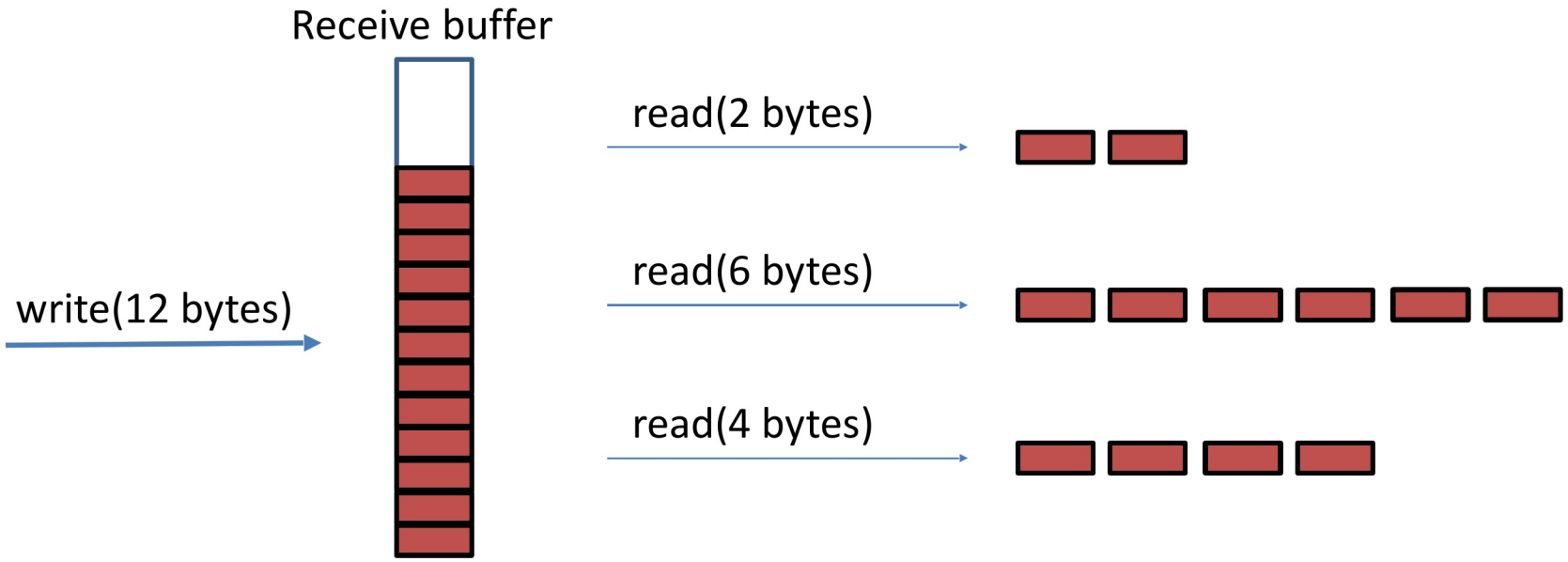
Client-Server Model - APIs

- Connectionless protocol (UDP-suite)



Stream vs Datagram

- Stream based protocols (such as TCP) work on *bytes* granularity

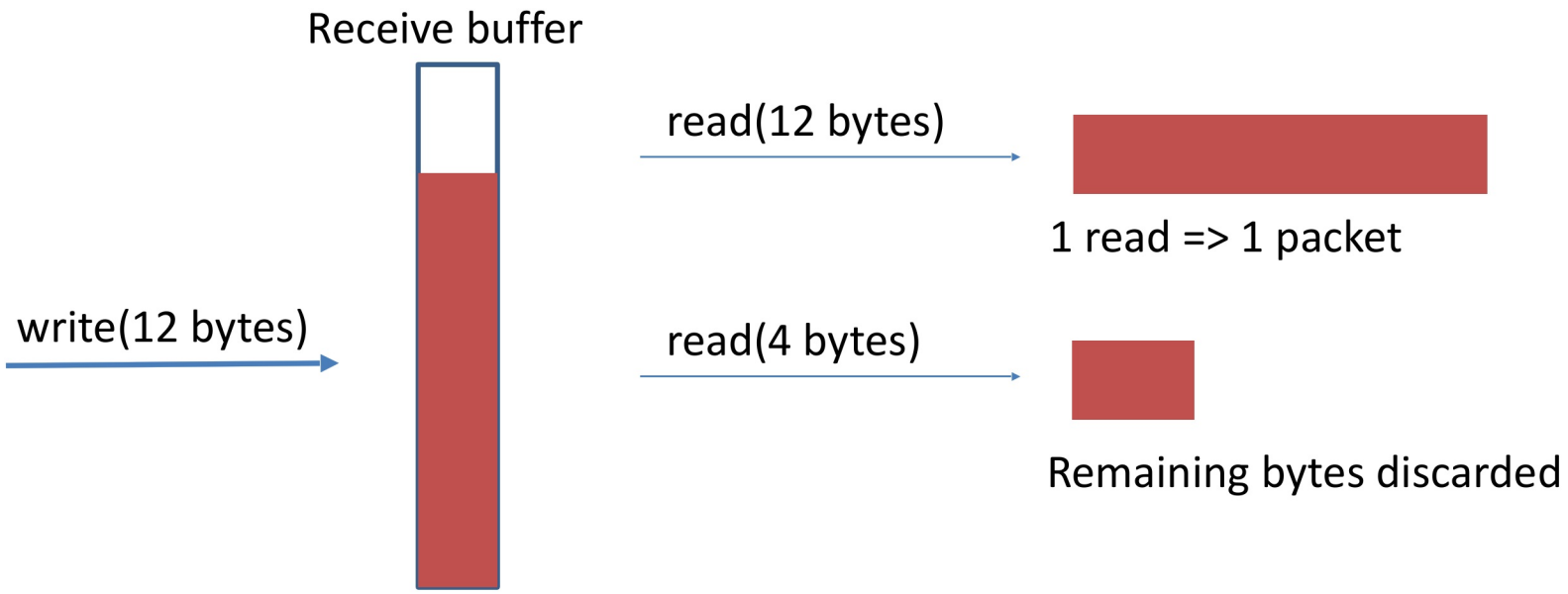


- When to stop reading?

Stream vs Datagram

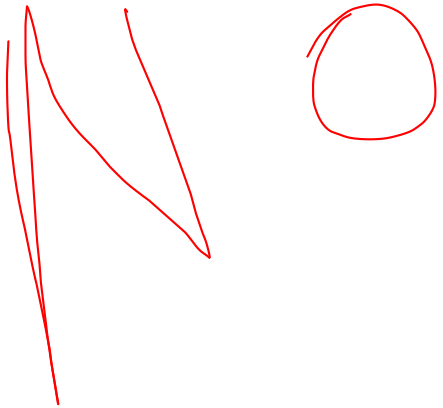
Maximum
Transmition Uniti is
5000 byte

- Datagram based protocols (such as UDP) work on *packet* granularity



- What is the right length value for read()?

Questions?



Sir showed the live Demo on his own laptop and output is draw the image ont he terminal when server recieved the Request from the client

Demos

Thank you!