Basic Python Programming

[Session 2] Network Programming

Contents

- Internet & Backgrounds
- Client-Server Model
- Socket Programming

Intro

Motivation

- **Countless connected devices**
  - Smartphones
  - Desktops
  - Laptops
  - …and what?

Weather forecasting toaster, awesome!

- **IoT(Internet of Things)**
  - "Ubiquitous" had already been achieved
  - Then, we apply network and Internet to everything

In This Course…

- **There are many concepts where the Internet is based on**
  - Network layer, protocol, packet, routing, so on…
- **We will cover the high-level concepts for networking**
- **…And apply them to our mini-project!**

Internet & Backgrounds

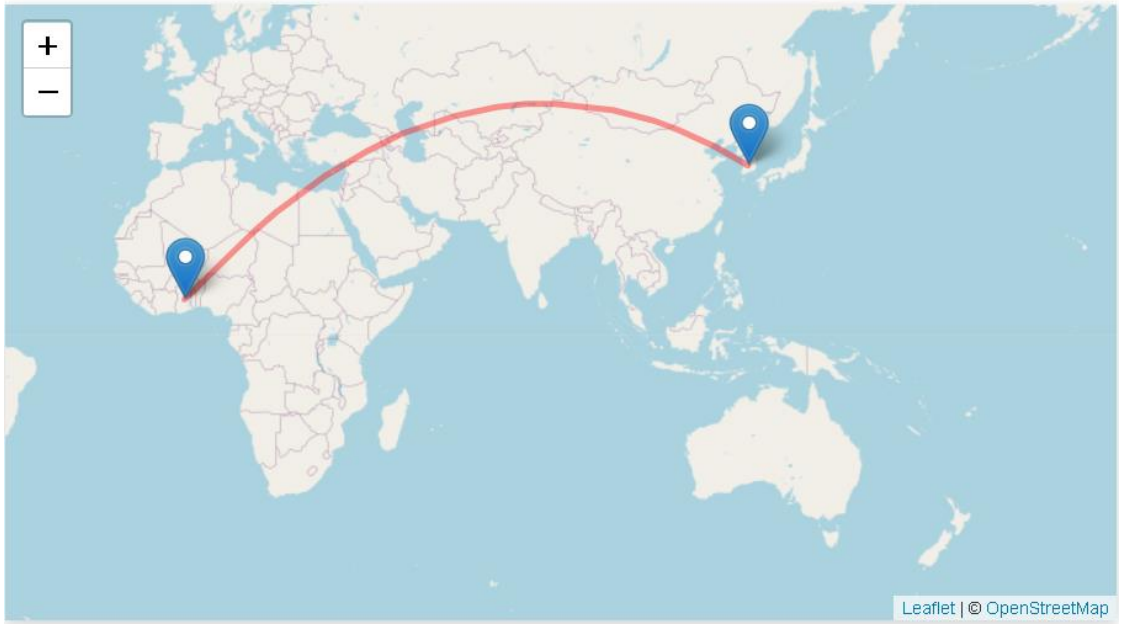
In the connected world

Network [1]

- **How far is Korea from Ghana?**
  - Let's ask Google

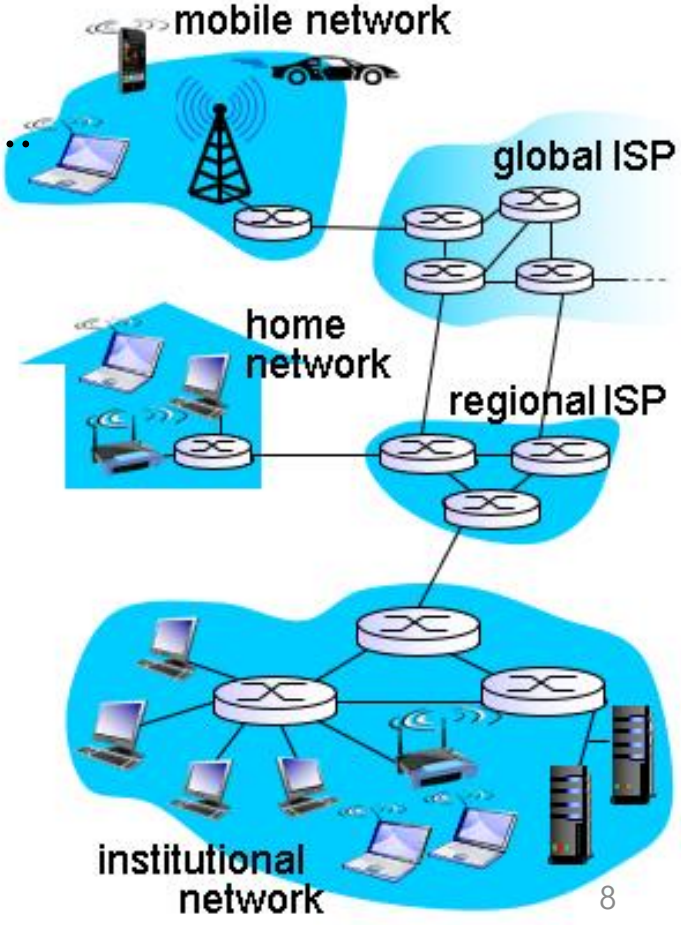
Distance from Ghana to South Korea

Distance from Ghana to South Korea is 12,794 kilometers. This air travel distance is equal to 7,950 miles.



Network [2]

- **How can we communicate with each other, despite such a long distance?**
  - Whatever you send, you may lose it on the way…
- **But we are in the connected world!**
  - We send/receive via communication links
    - Some wire?
  - And data is forwarded by packet switches
    - Router?

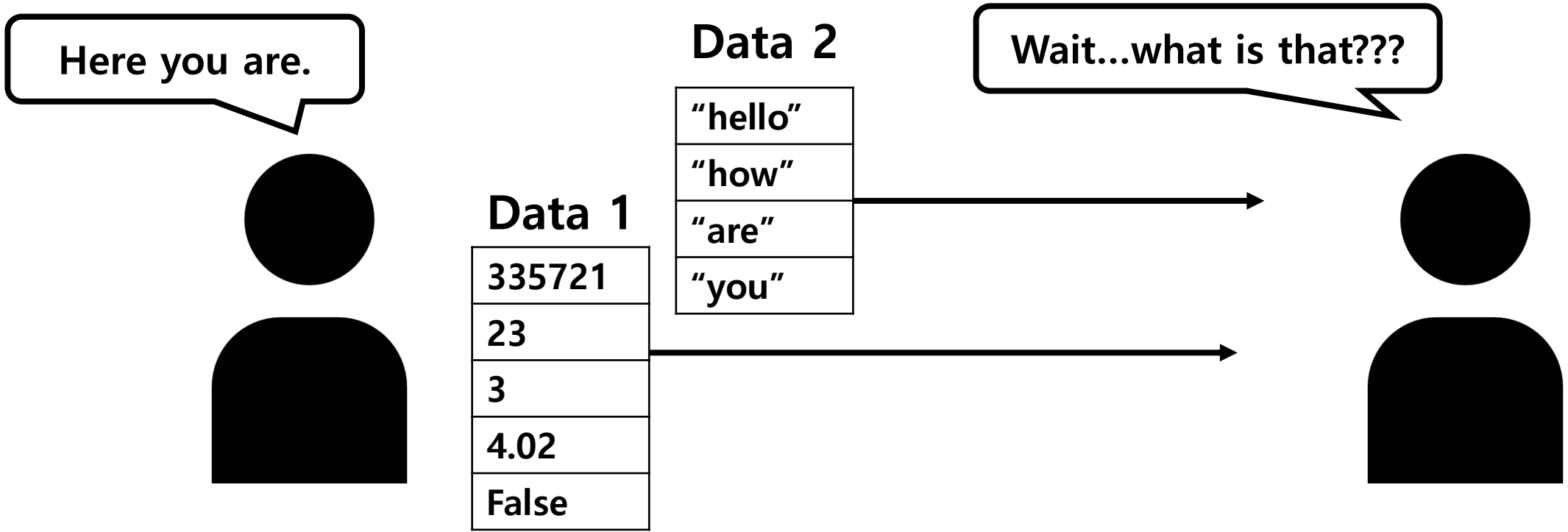


Internet

- **Internet is a global network system with Internet protocol suite**

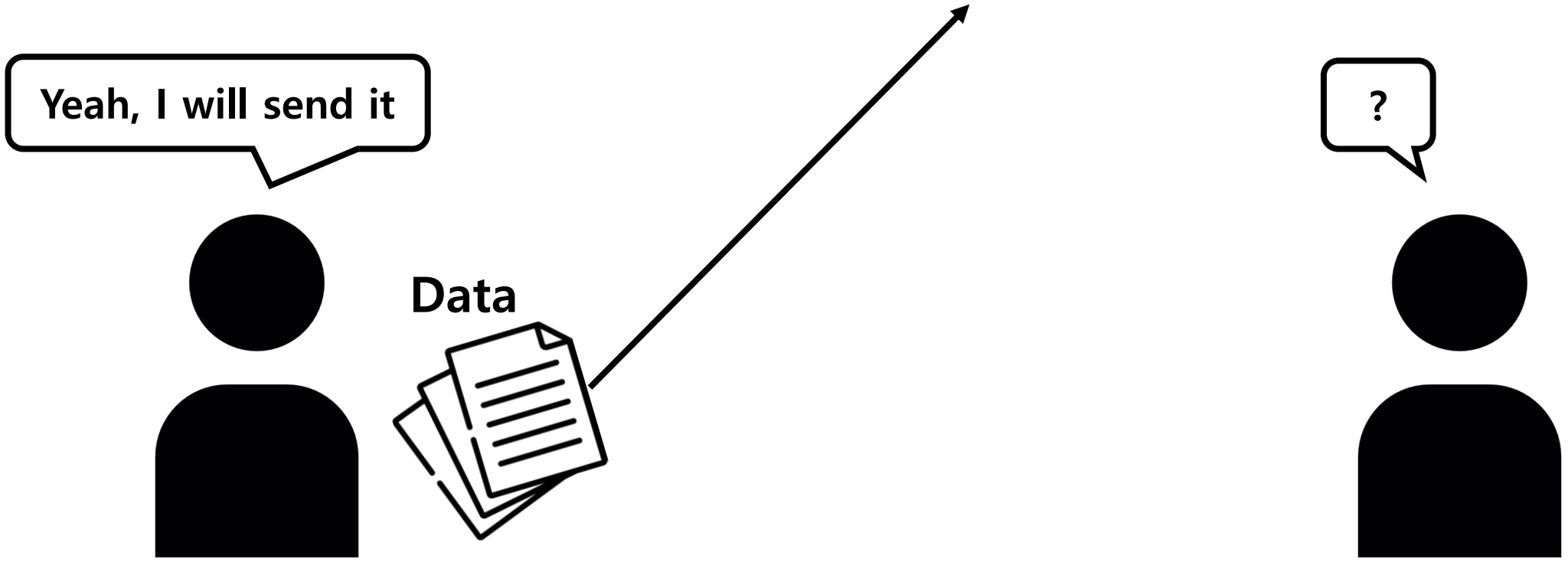
Protocols: Motivation [1]

- **What if there's no rule(protocol) in networking?**
  - How can the receiver understand the given data?



Protocols: Motivation [2]

- **What if there's no rule(protocol) in networking?**
  - If there's some problem, then how can handle it?



Protocols

- **Protocols define rules**
  - Which format of data?
  - How to establish the connection?
  - How to check its validity?
  - What the given data/field means?
  - So on…

- **Example**
  - TCP, UDP
  - IP(IPv4, IPv6)
  - HTTP, HTTPS, Telnet, DNS, …
  - You've probably heard of some of these!

Protocol Example: TCP vs UDP [1]

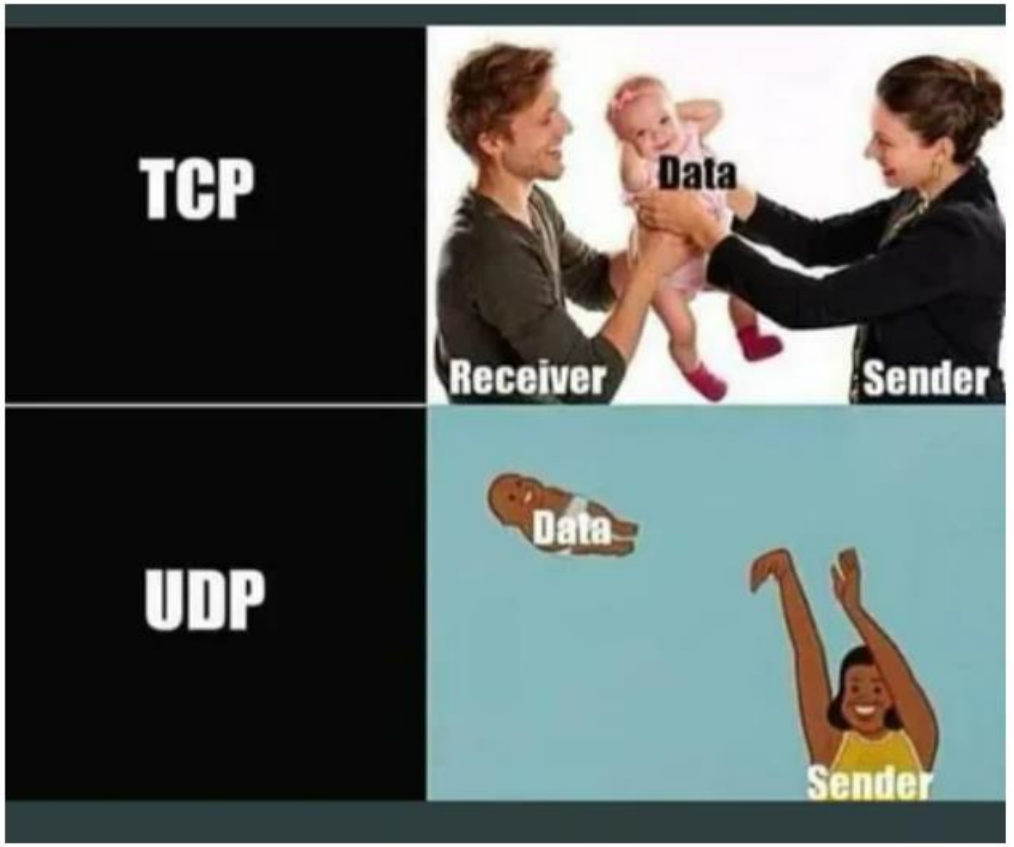
- **TCP and UDP defines a rule for sending/receiving data**
  - TCP: Transmission Control Protocol
  - UDP: User Datagram Protocol

- **How they differ?**

Protocol Example: TCP vs UDP [2]

- **TCP is reliable because it ensures:**
  - Data must be delivered to receiver (in order)
  - There's no data loss
- **...but slow**

- **UDP is faster than TCP but..**
  - Sometimes, data is lost
  - Unreliable Damn Protocol..?



Protocol Example: TCP vs UDP [3]

- **We can choose one according to our purpose / preference**
  - Let's take some examples!

Protocol Example: TCP vs UDP [4]

- **Example 1: Chat App**
  - Sometimes, a character / the order does matter

Protocol Example: TCP vs UDP [5]

- **Example 2: Streaming**
  - It must be fast!
  - Can you notice that 2~3 pixels are missing from a frame of video?

Note

- **Anyway, we only need to remember just one thing**

- **Communication requires a rule(protocol)!**

IP Address

- **I want to text to my friend… but how can we find him/her?**
  - We need some addresses!
  - And this is **IP address**

- **Two types of ID address:**
  - IPv4 (32bit): 192.168.10.253
  - IPv6 (128bit): 2626:28000:0220:0001:0248:1898:25C8:1946

- **Then… Everything's OK now?**

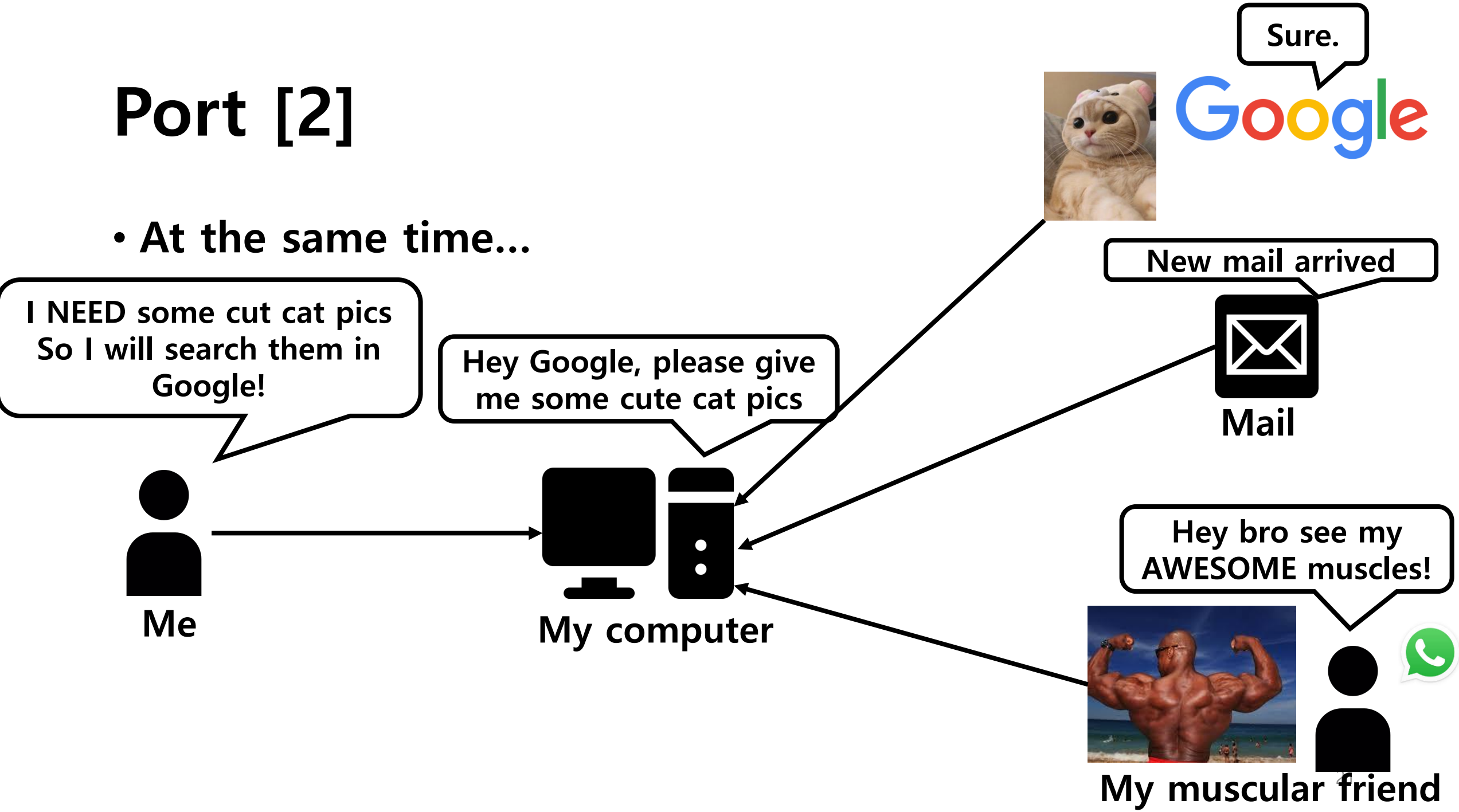
Port [1]

- **Our computer runs a lot of applications concurrently.**
  - Web browser
  - Messenger
  - Game
  - …

- **And each communicates with a different one**
  - With Google server
  - With Discord server
  - With my friend
  - With PUBG server
  - …

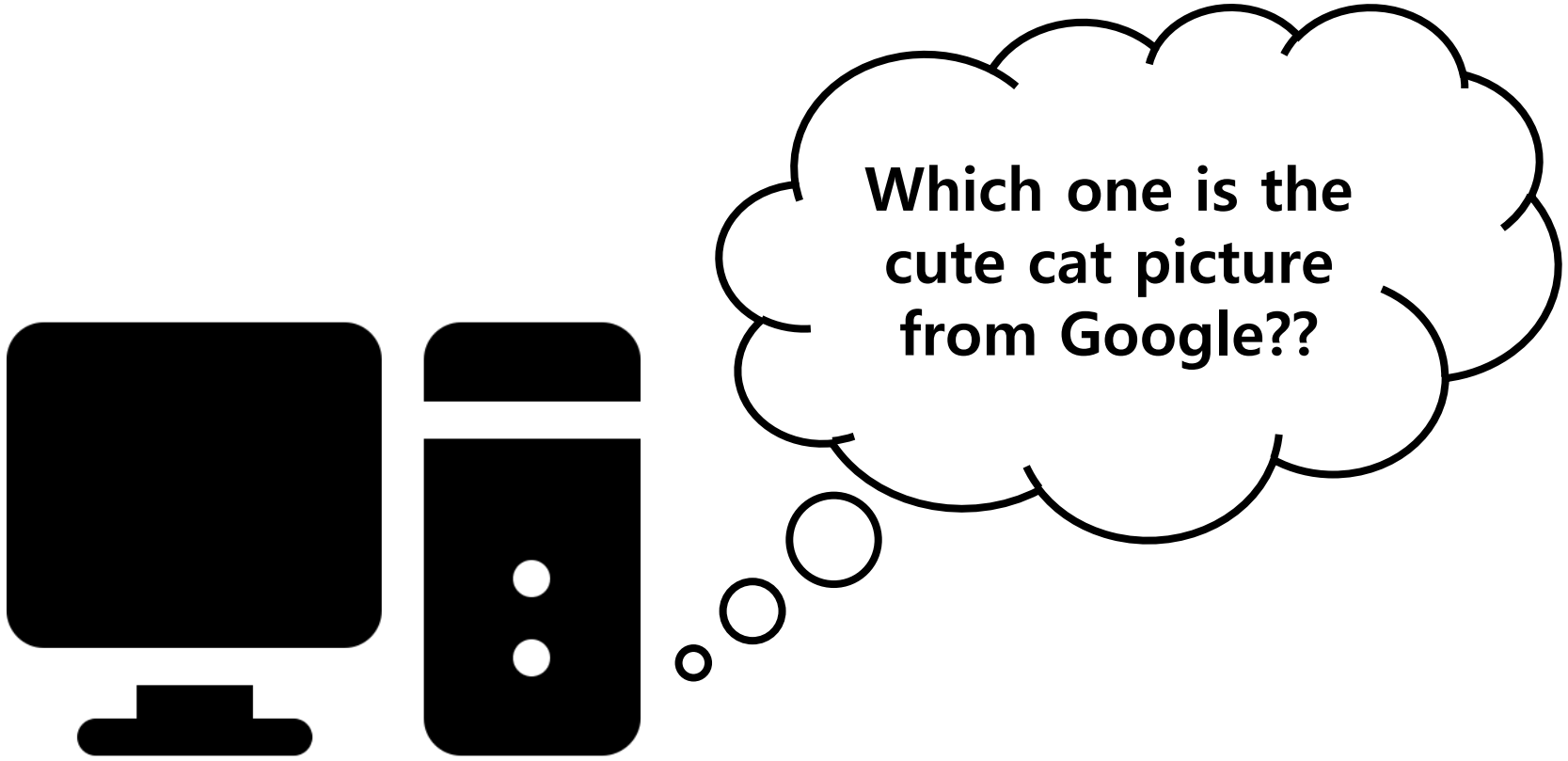
Port [2]

- At the same time…



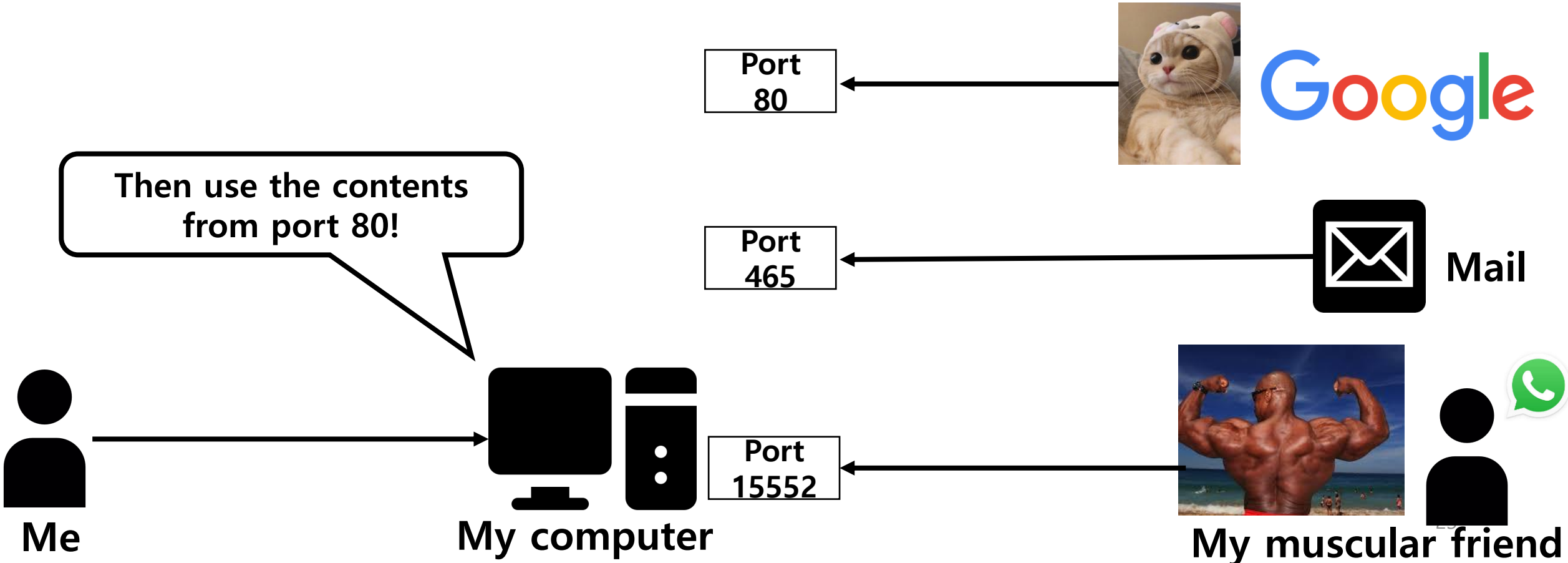
Port [3]

- **Computer may think…**



Port [4]

- We use several “ports” and communicate separately



So…

- **We can determine how do we communicate**
  - **Who?** By IP address
  - **By which application?** By port number
  - **With which rule?** By protocol

Client-Server Model

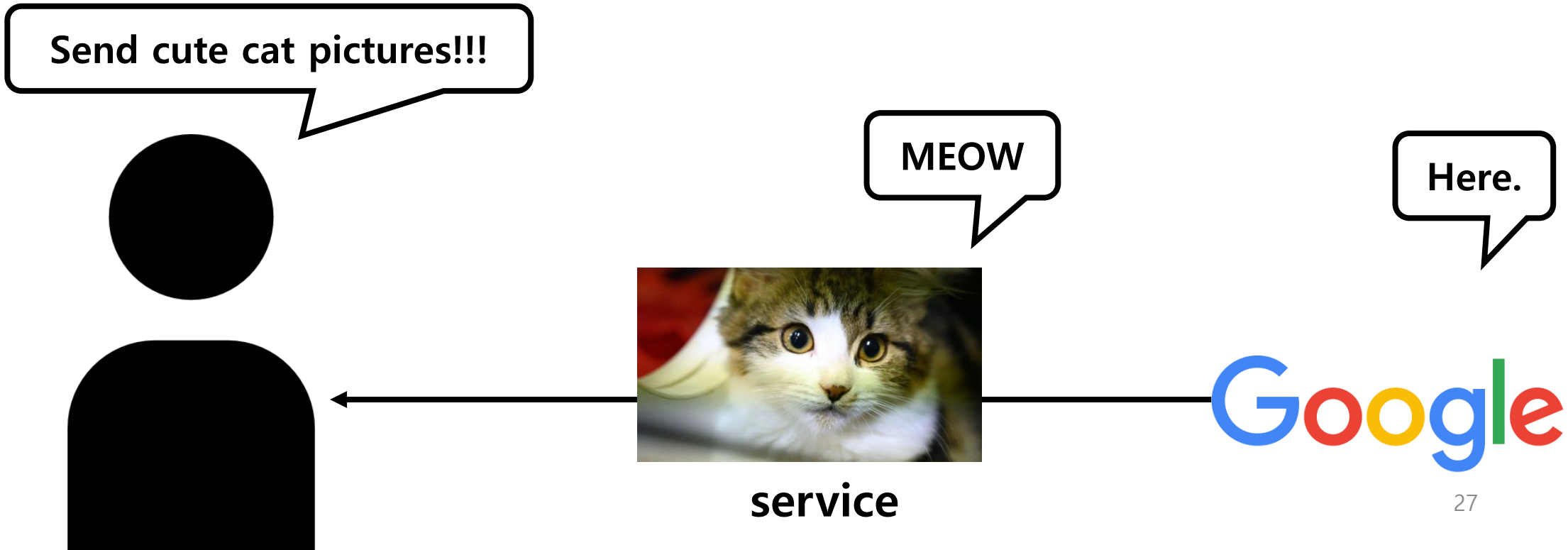
Intro

- **What is server and client?**
  - Client: requests / gets the service
  - Server: provides the service

- **The meaning is also same in networking!**

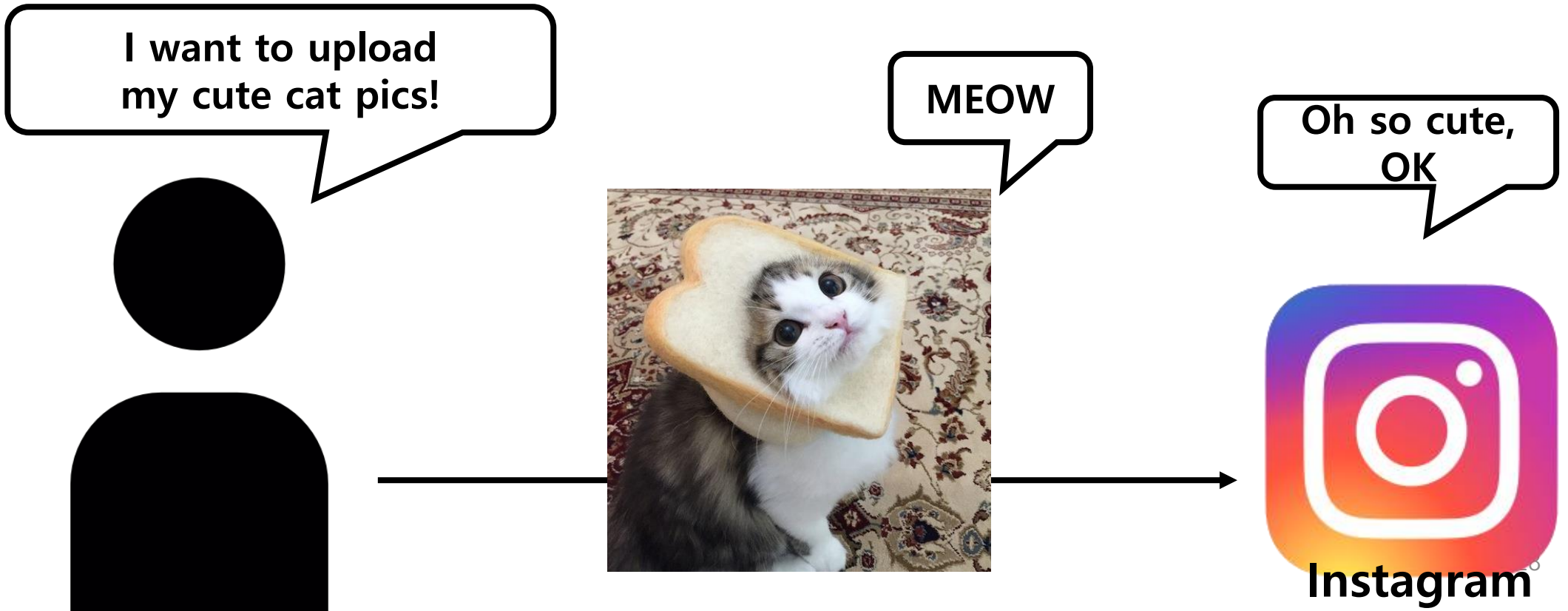
Client / Server [1]

- We usually (but not always) communicate as clients



Client / Server [2]

- **Client doesn't mean just receiver**



Client / Server [3]

- The important point is that:
  - The task and role are divided into two parts: client / server

Client / Server: TCP [1]

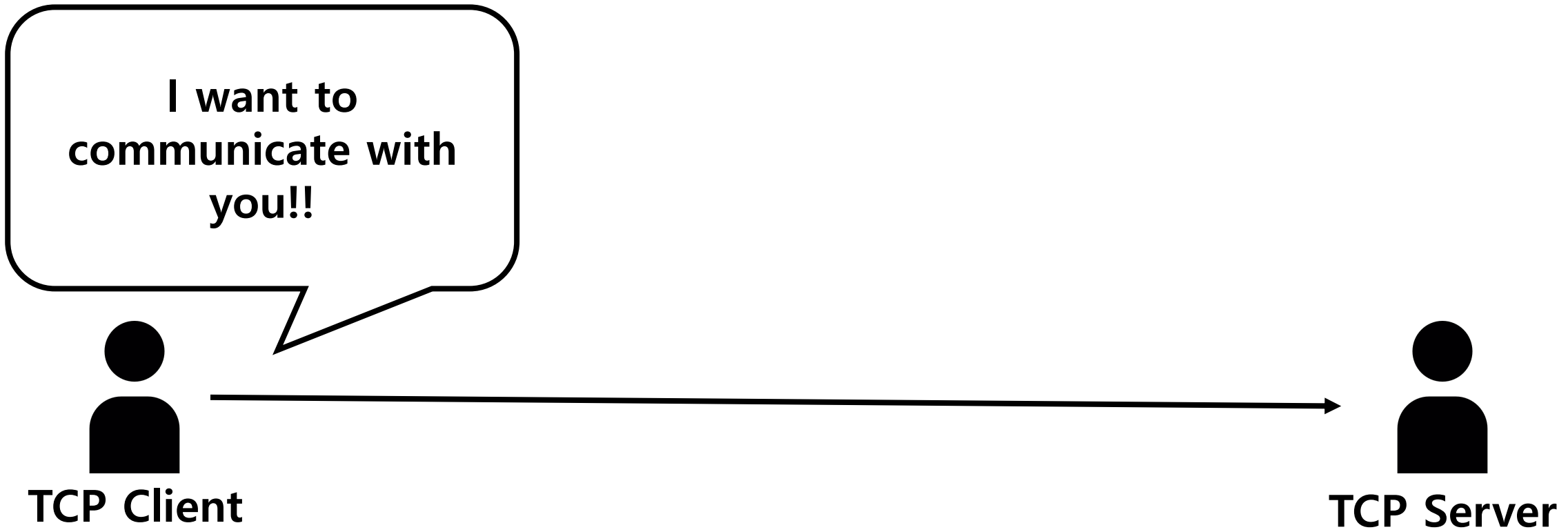
- **We will cover only TCP(reliable) client-server**
  - But UDP case works similarly
  - We explained it in the supplement

Client / Server: TCP [2]

- **Step 1: Listen**

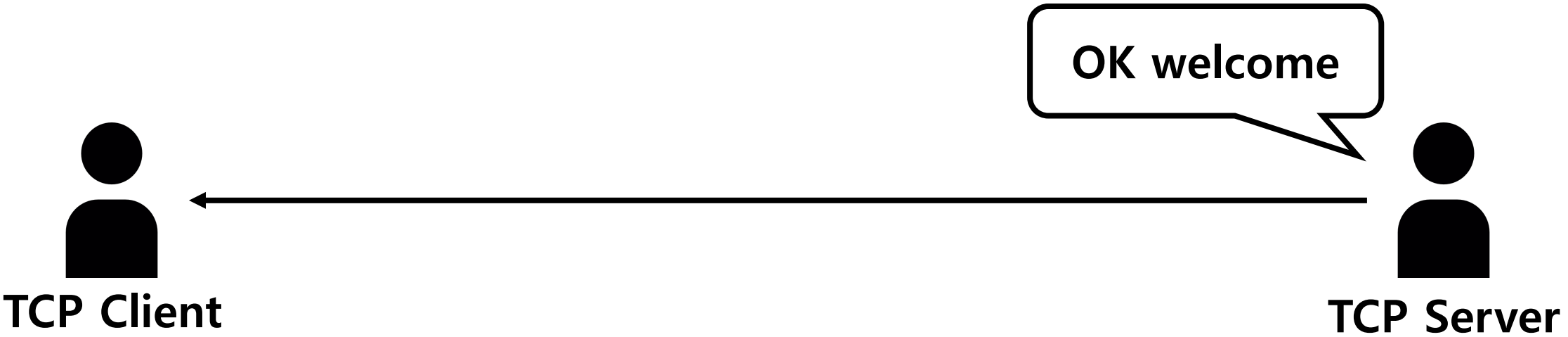
Client / Server: TCP [2]

- Step 2: Connect



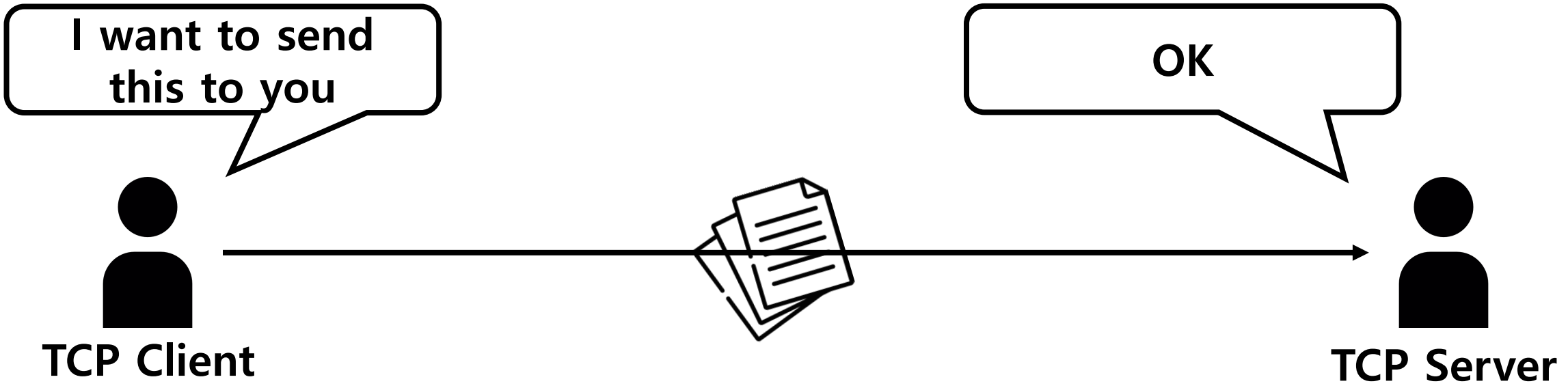
Client / Server: TCP [3]

- **Step 3: Accept**



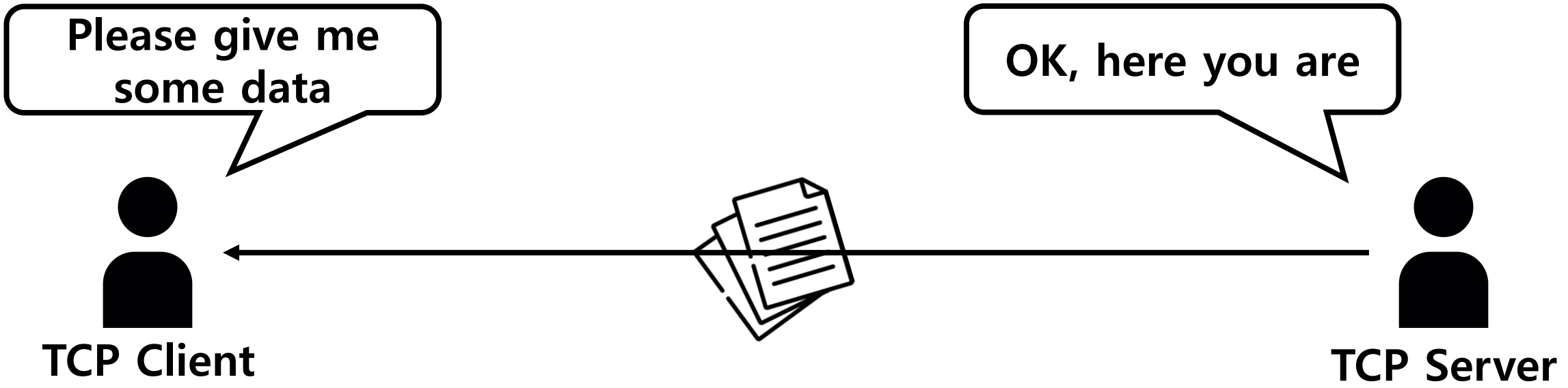
Client / Server: TCP [4]

- **Step 4: Request to send / receive data (maybe repetitively)**



Client / Server: TCP [5]

- **Step 4: Request to send / receive data (maybe repetitively)**



Client / Server: TCP [6]

- **Step 5: Close the connection**
  - Both can close the connection

Implementation

- Well… So good so far.

- But how can we implement this?

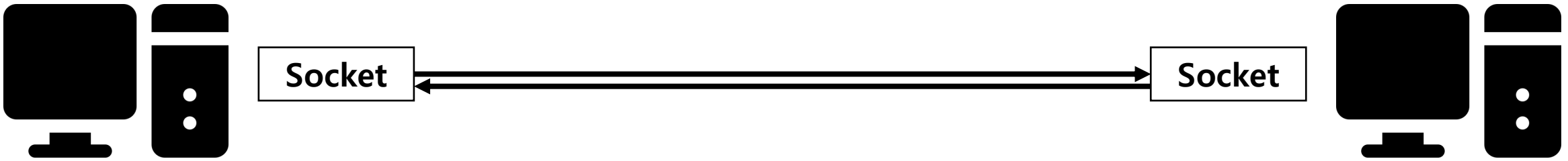
Socket Programming

We use Python, finally!

What Is Socket?

- **Socket is an interface to communicate across the network**

- **It contains some information**
  - IP address
  - Port number
  - Protocol



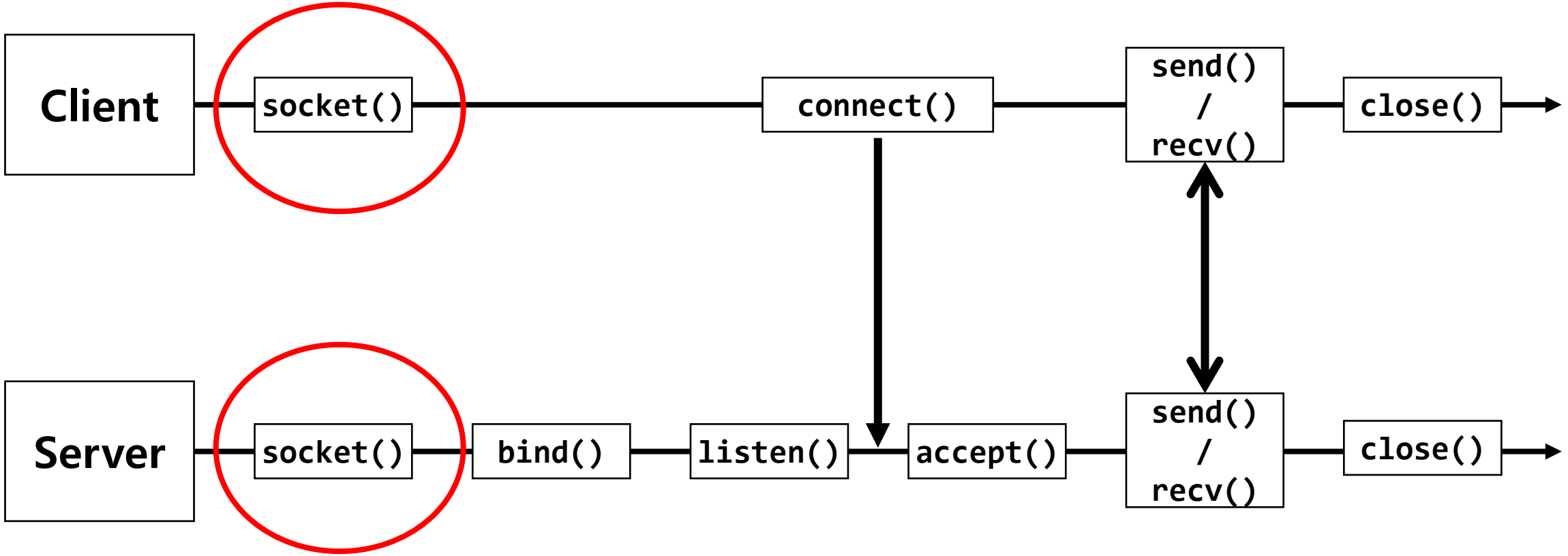
Socket Programming in Python [1]

- **Several functions for socket programming are provided**

- **The procedure is almost same with that of client-server**
  - Each step is done by calling some function

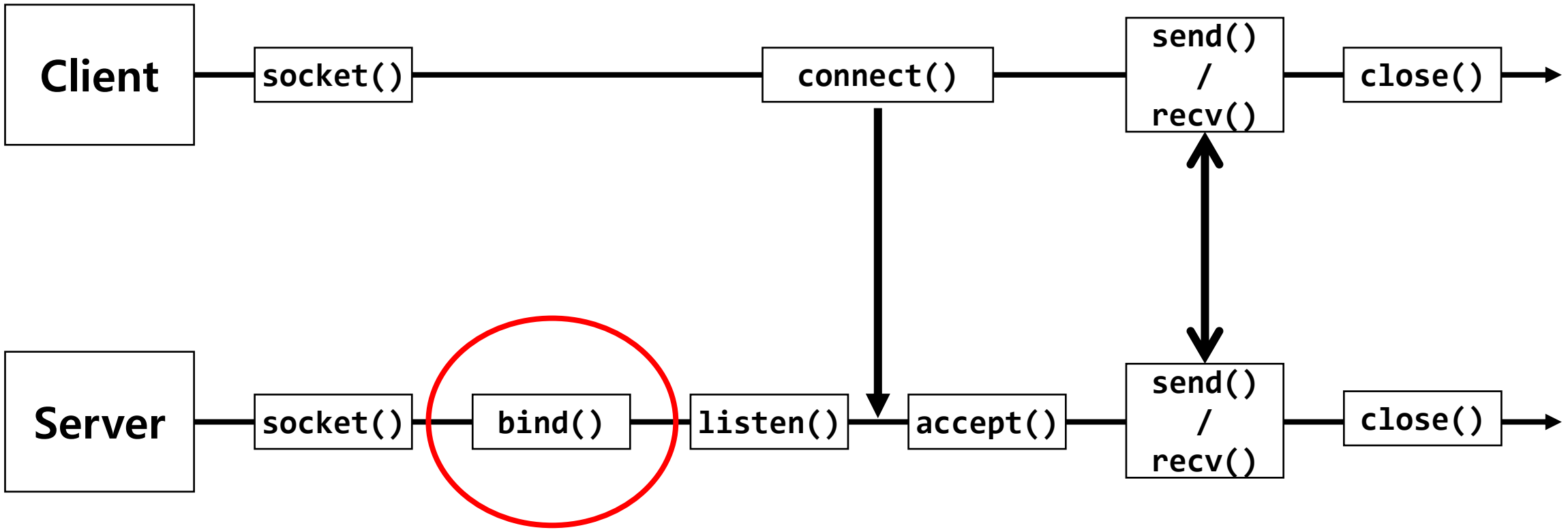
Socket Programming in Python [2]

- Step 1: Both create a socket



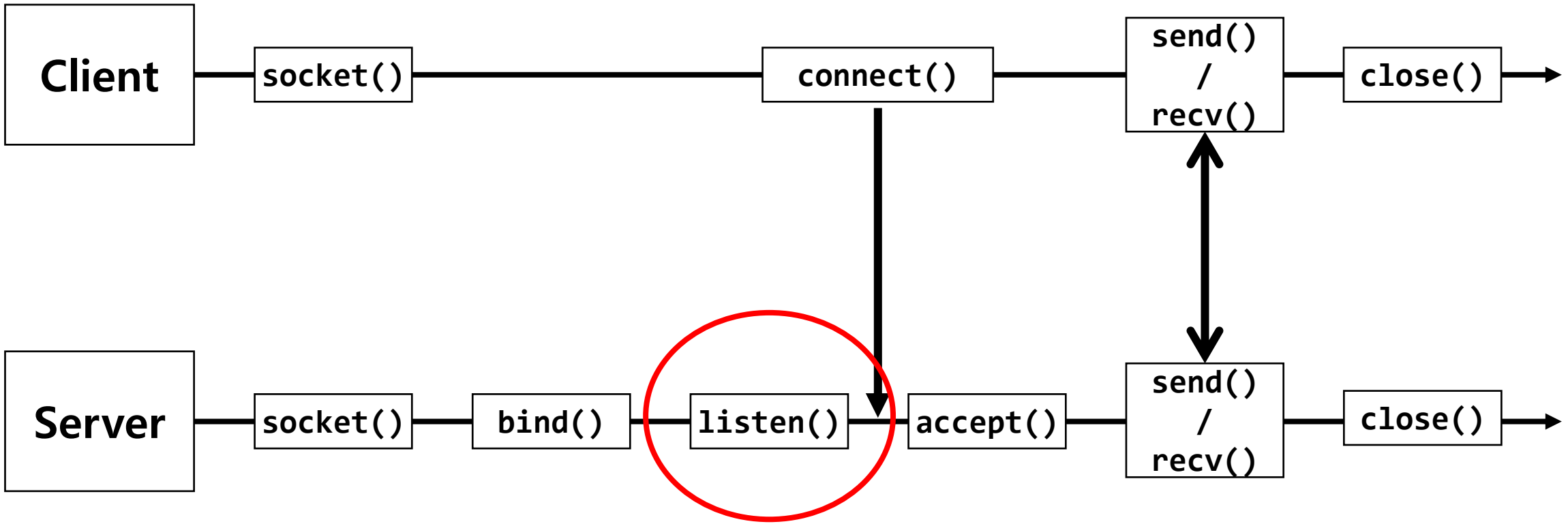
Socket Programming in Python [3]

- Step 2: Server binds socket (indicate hotename and port #)



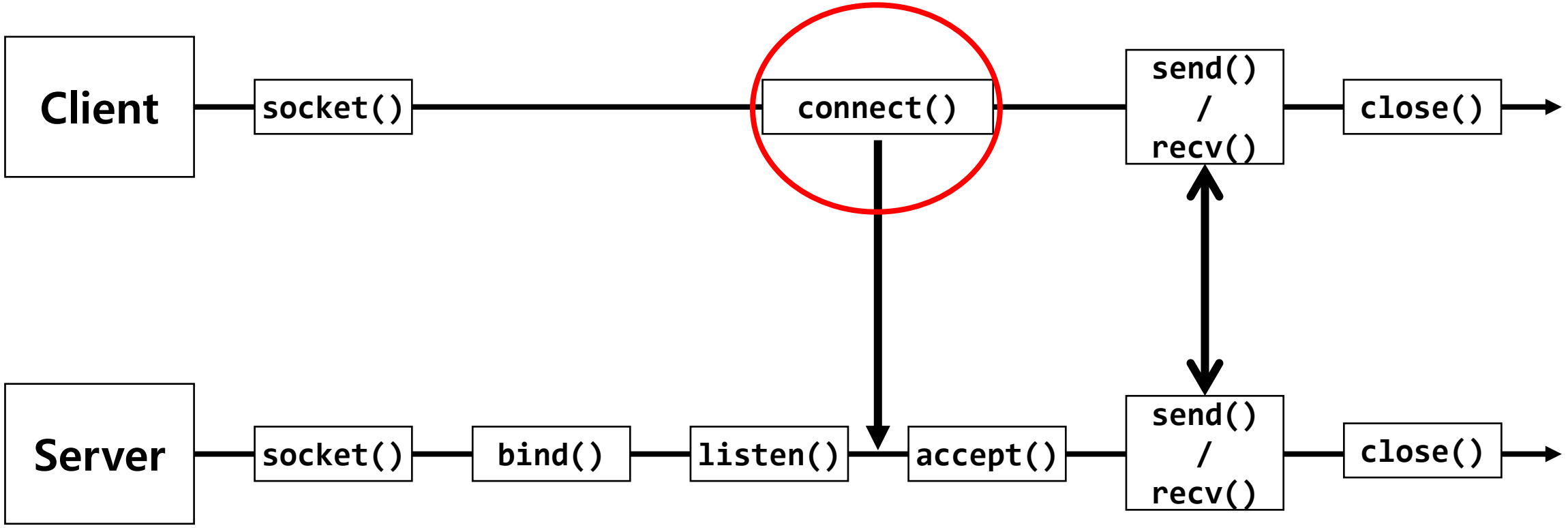
Socket Programming in Python [4]

- Step 3: Server listens (waits for client)



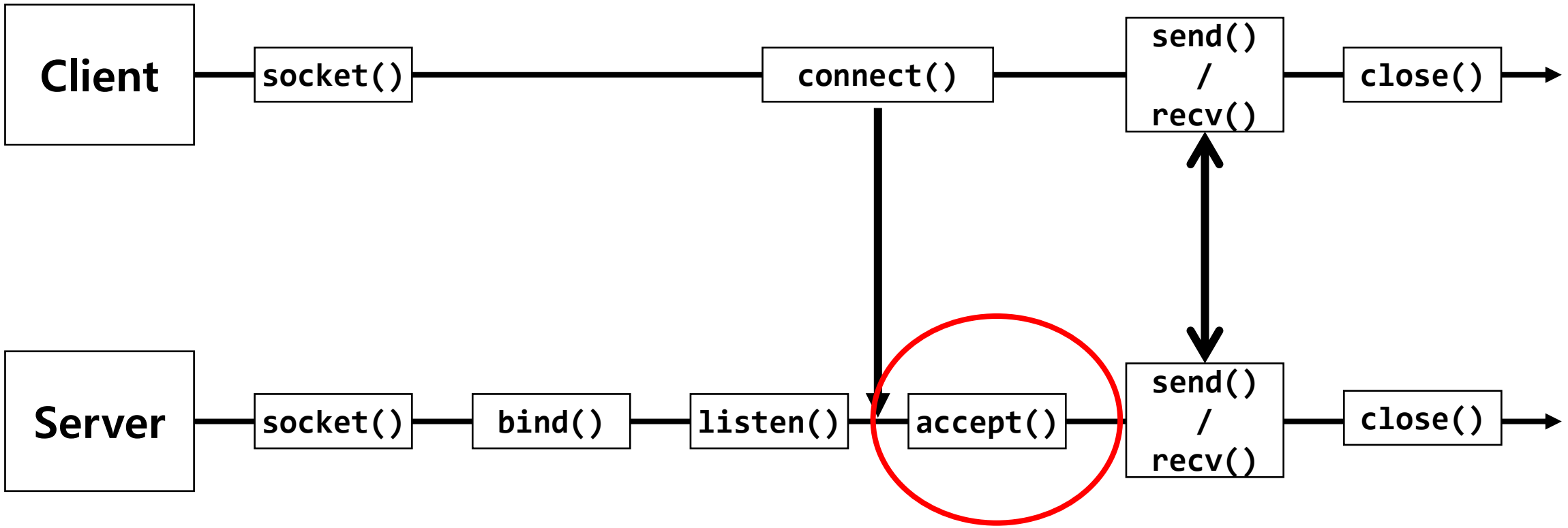
Socket Programming in Python [5]

- Step 4: Client connects to server



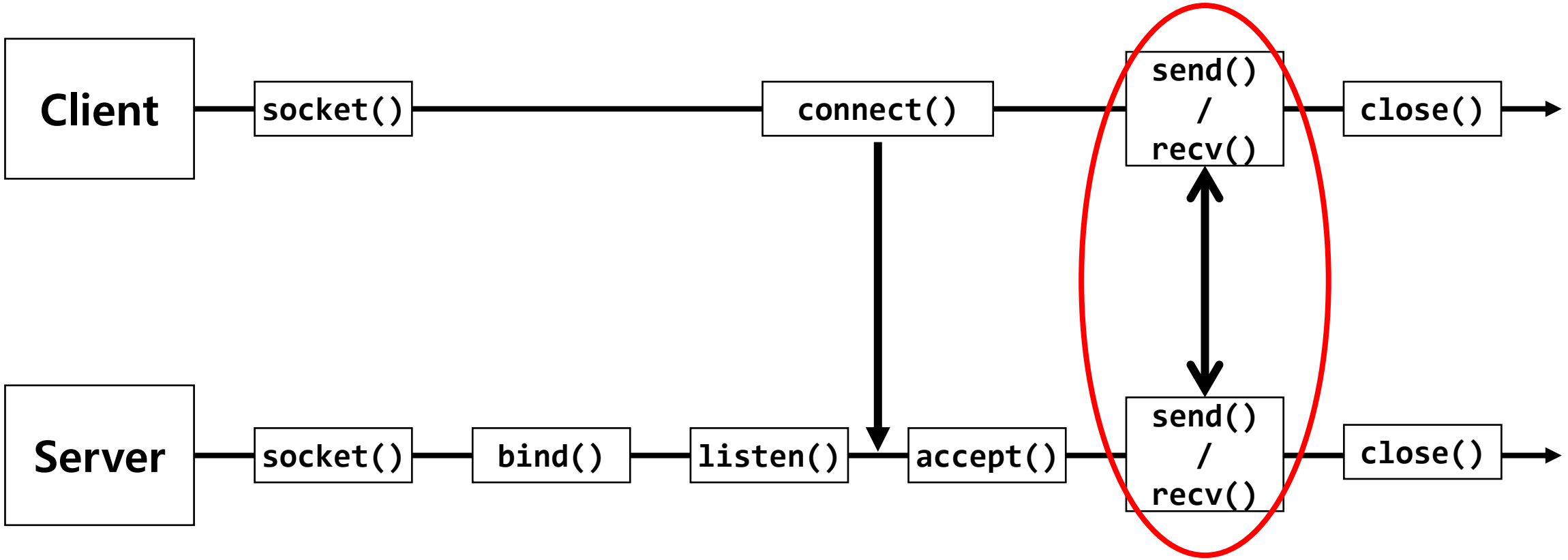
Socket Programming in Python [6]

- Step 5: Server accepts



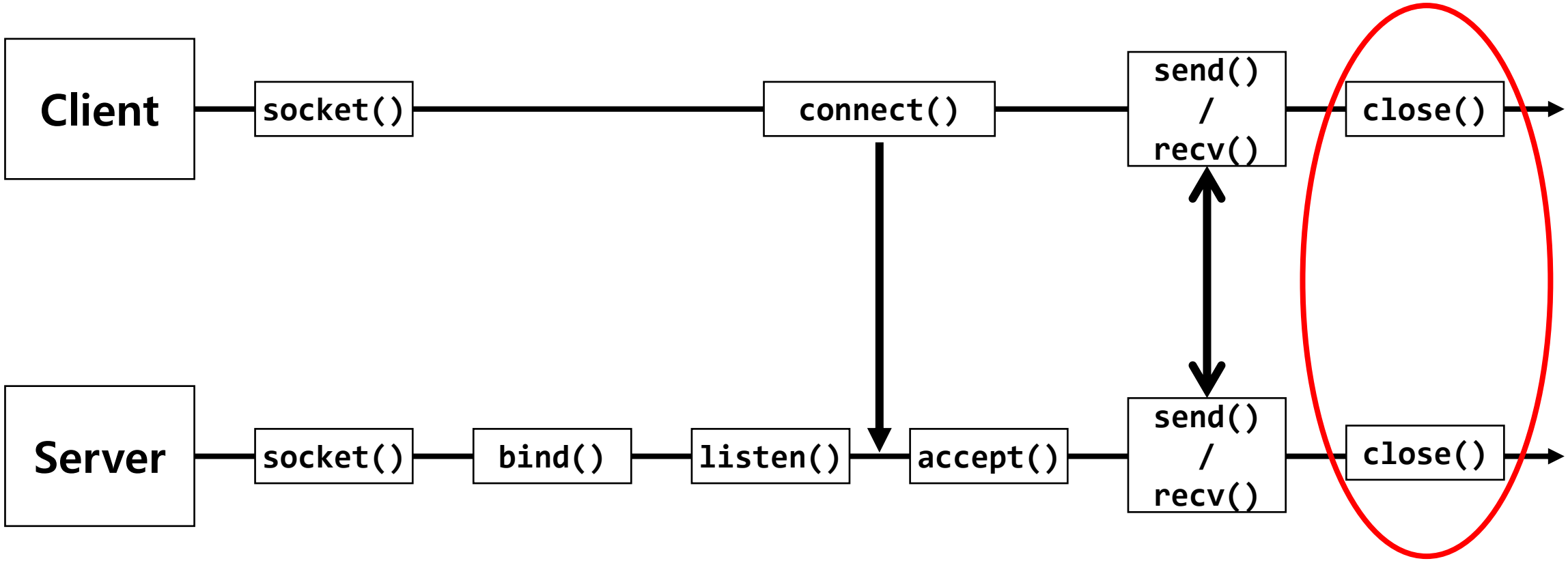
Socket Programming in Python [7]

- Step 6: Then they communicate



Socket Programming in Python [8]

- Step 7: Someone terminates the connection



Notes

- **As we can see, the task are different between client and server**
  - So we need two applications client / server
- **We will make both client apps and server apps**
- **You may be able to fully understand the concepts by making apps**
  - In lab session

References

- **Computer Networking: A Top-Down Approach 6th Ed. Jim Kurose, Keith Ross, Addison-Wesly, March 2012**

Thank you