P2P Networks – Exercise # 1

Prof. Dr. Thorsten Strufe,

&

Ikram M. Khan,

P2P Networks Group,

TU Darmstadt

Date: Oct. 25th, 2011

Acknowledgements: some of these slides are based on material from J. Deutschmann, G. Schäfer, and K.W. Ross

Who We Are?

- Organizer
  - Ikram M. Khan, khan[at]cs.tu-darmstadt.de

- Tutors
  - Ana Barroso
  - Dominik Fischer

- Office
  - S2|02 B220
- Office hours
  - Could be arranged as per email.

Organization of Exercise Course

- Mode of exercise course
  - Theoretical exercise (poss. group of two participants)
  - Programming exercise (poss. group of max. four participants)

- Exercise cycle
  - Weekly, Tuesday, 14:25 – 16:05

- Content
  - Dicussion about previous exercises
  - Presentation of new exercise (theory & programming)
  - General questions regarding lecture

- Due date
  - Latest every Tuesday 14:25 – handover in the exercise or drop it in letter box in front of S2|02 A110

Bonus Systems

- You must pass exams without the bonus
- It can only used in the first two exams possible
- It can be used only once per student
- Exercise points >= 90 %
  - Lifting your grade two steps e.g., 1.7 → 1.0
- Exercise points >= 50 %
  - Lifting your grade one step e.g., 1.3 → 1.0
- Exercise points will be shared via webreg

Final Exam

- Exams
  - Date: to be announced
  - Hopefully oral exams (if # of participants < 32)
  - Successfully absolving exercise could buy you bonus

Course Material

- Slides will be available on the course web-site

- Literature
  - Books:
    - Coulouris et al., Distributed Systems
    - Booth et al., „The Art of Research”
    - Jorgenson, „Beej's Guide to Network Programming ”
    - Donahoo et al., „TCP/IP Sockets in C: Practical Guide for Programmers”
- Web:
  - Kademlia: A design specification
    - http://xlattice.sourceforge.net/components/protocol/kademlia/specs.html

Purpose and Scope of This Course

Theoretical exercises

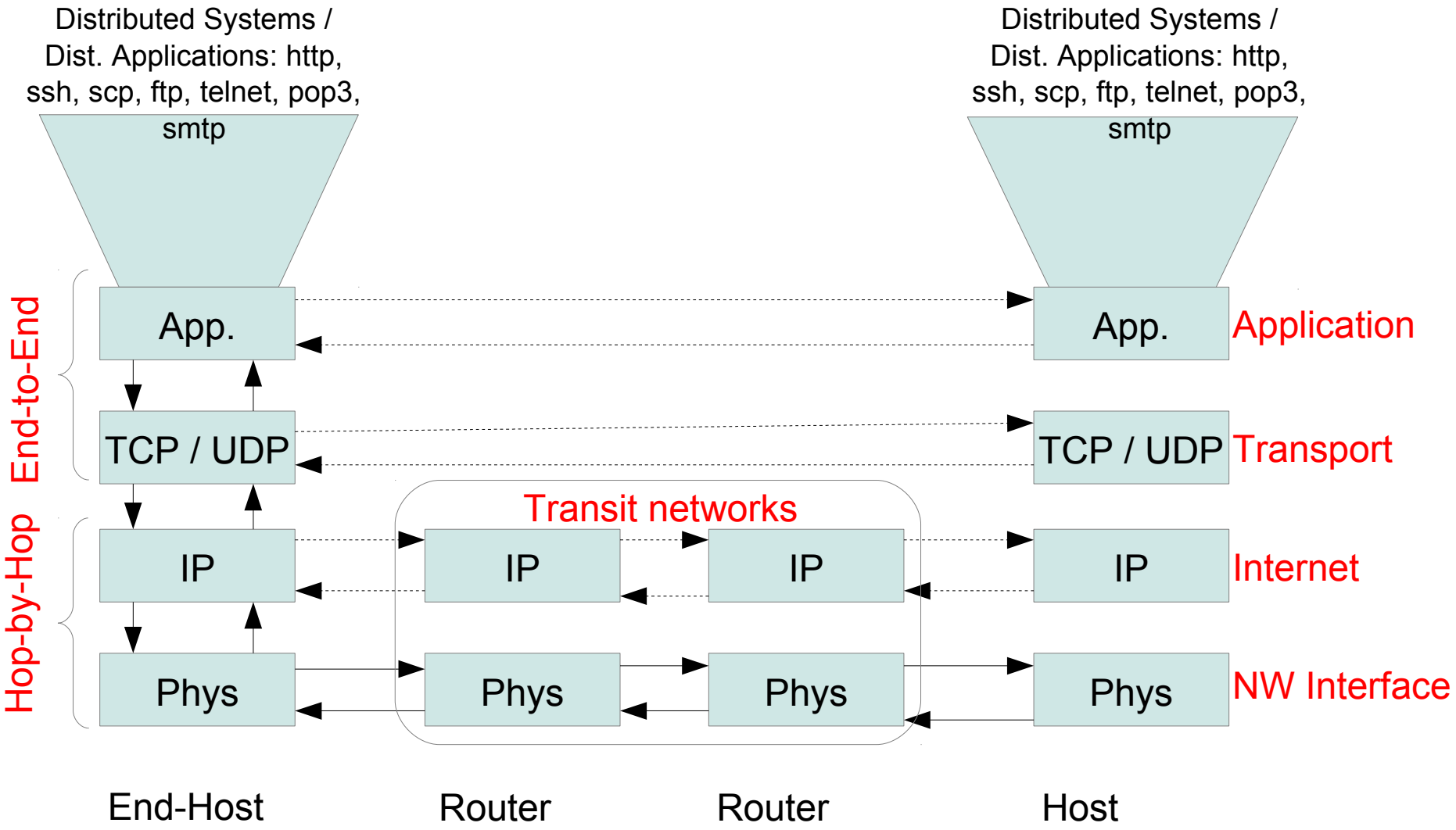
- Written exercise - solving problems

Programming exercises (C/C++, Java, Python)

- Hands-on experiences with P2P technologies.
- Programming Tasks
  - Basic network programming
  - Programming and solving graph related problems
  - Programming to under P2P technology Chord, CAN, Pastry
- Big Project – P2P-based file-sharing client

Questions?

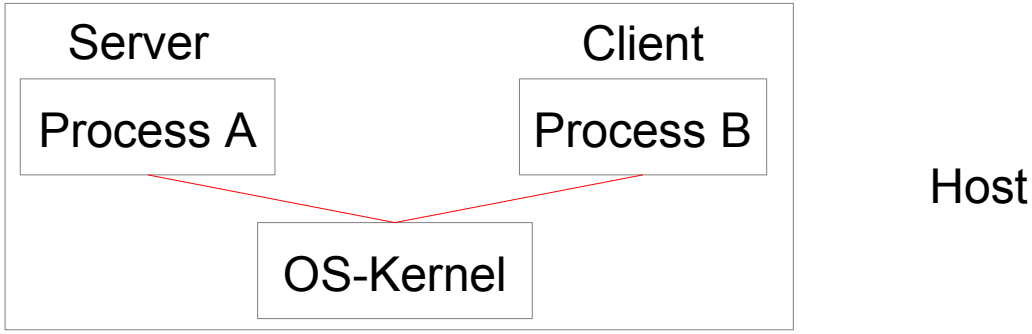
Recall: Application Layer



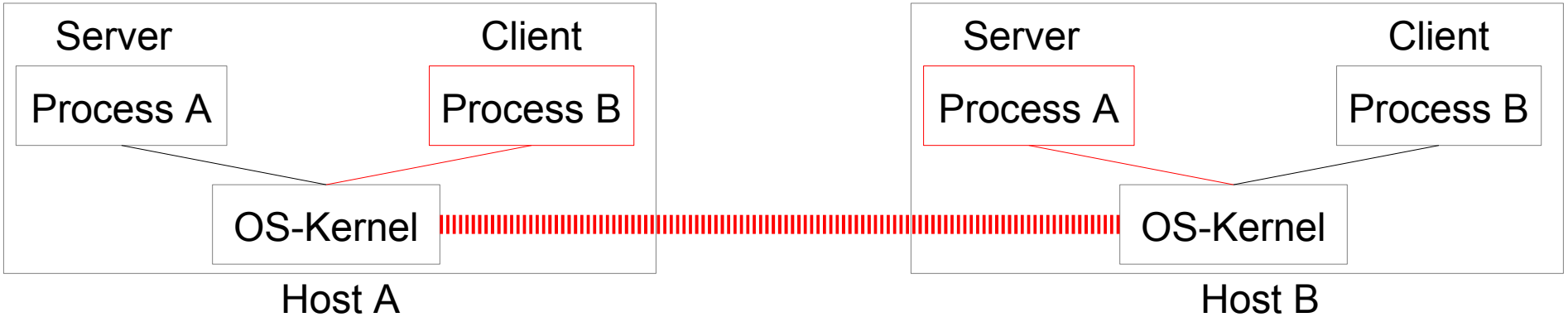
What is Application Layer?

What are distributed systems / distributed applications?

- … system of combined, communicating components.



- Inter Process Communication (IPC) over physically “long” distances

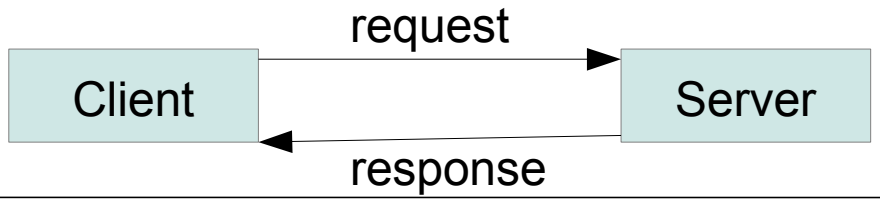


What is Distributed?

- Data
  - Location of data caused by location of users, updates, data sources
  - For administrative reasons (central access, database)
- Computation
  - Parallel processing on multiple machines
  - Computation using specialized hardware
  - Access to specialized sensors / actuators
  - Access to exclusive data (services!)
- Users
  - Physically placed at different locations
  - User of different roles

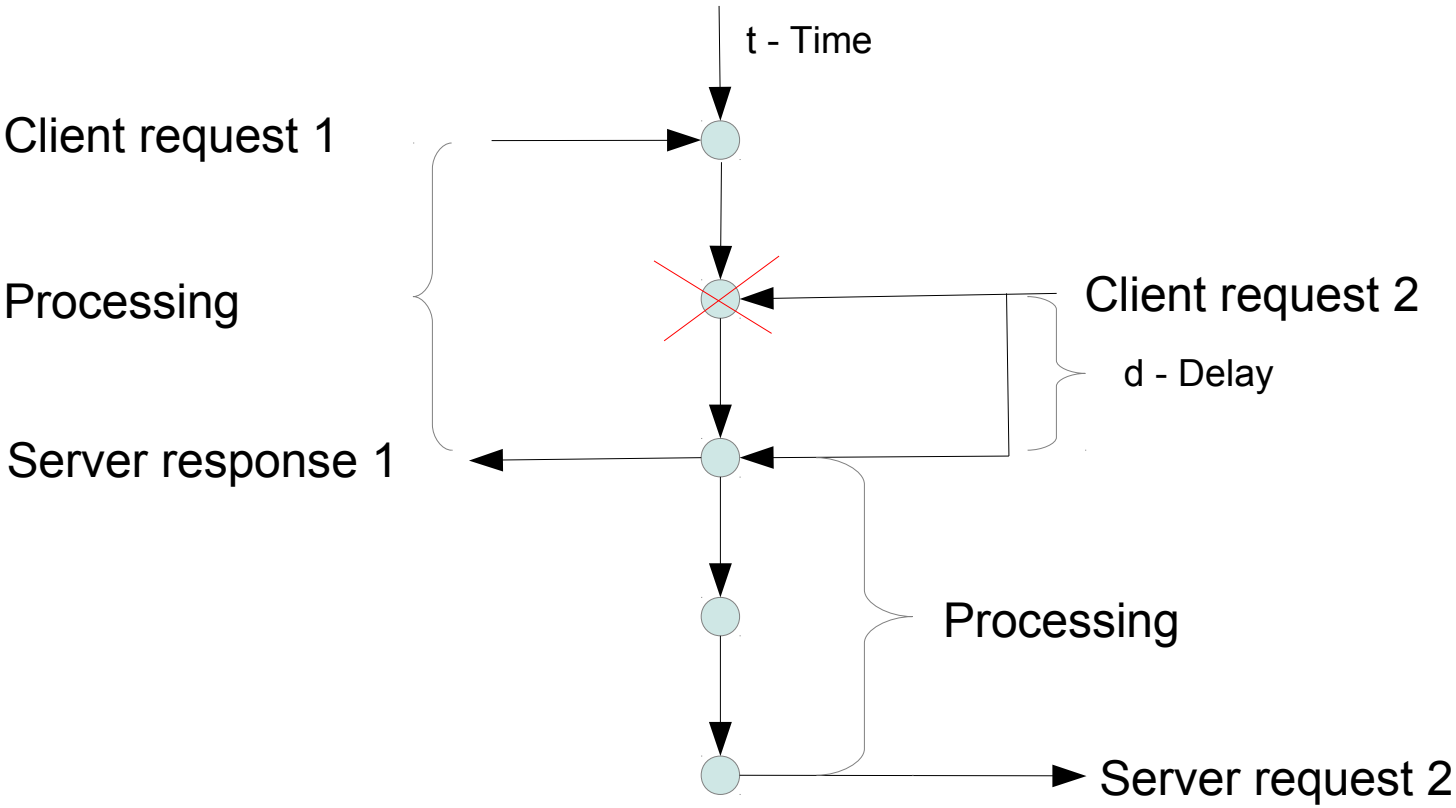
Client Server Model

- Client /server model introduces two roles (Servers, Clients)
  - Iterative
  - Concurrent
- Communication through “request response”
  - Reliable (Stream, connection-oriented, atomic request)
  - Unreliable (Datagram, connection-less, usually atomic requests)
- Communication relation is asymmetric
  - **Client** send **request** to the server
  - The **server** handles the **request**, produce the answer, and sends a **response**



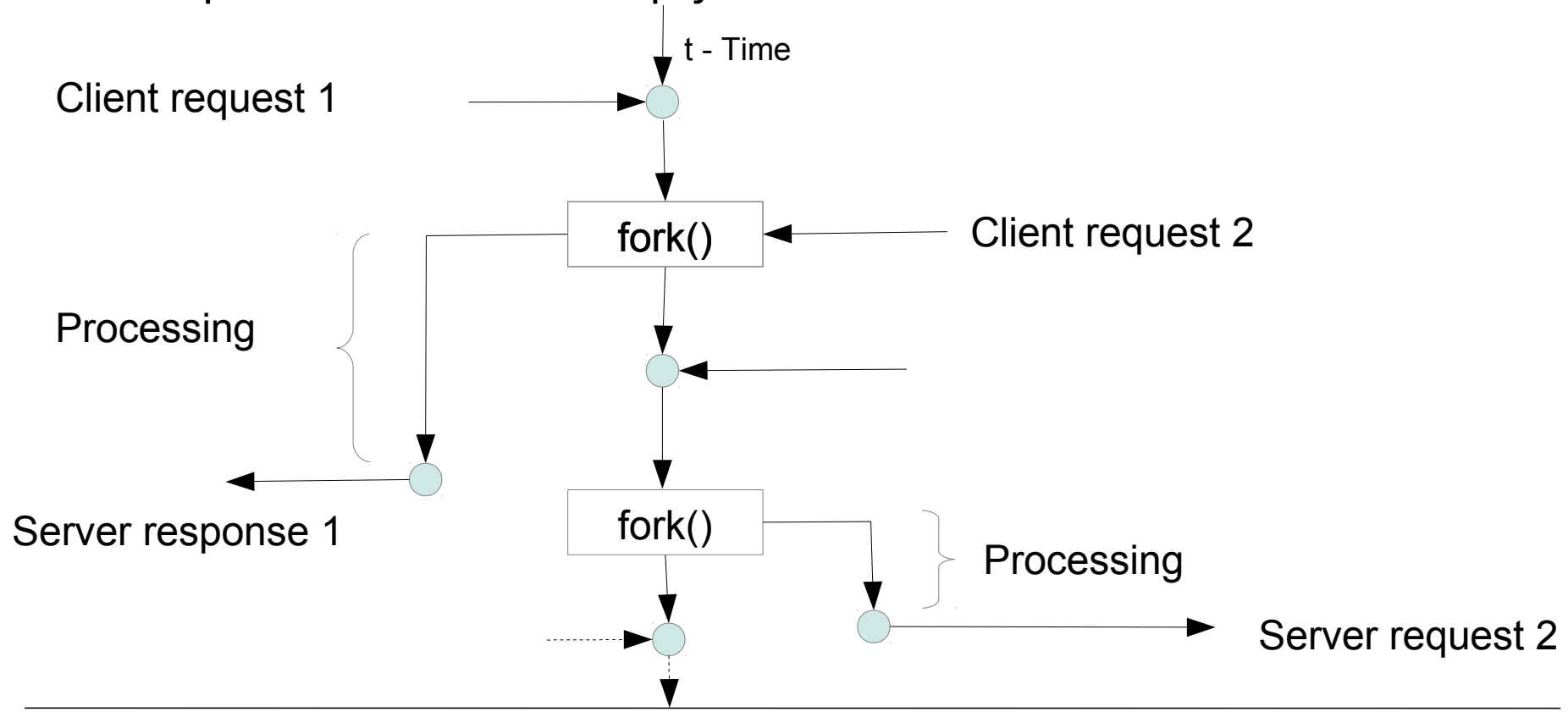
Iterative roles : Server

- Server process only a single client request at a time
- Multiple client requests are enqueued (by the OS) and processed in the order of their reception



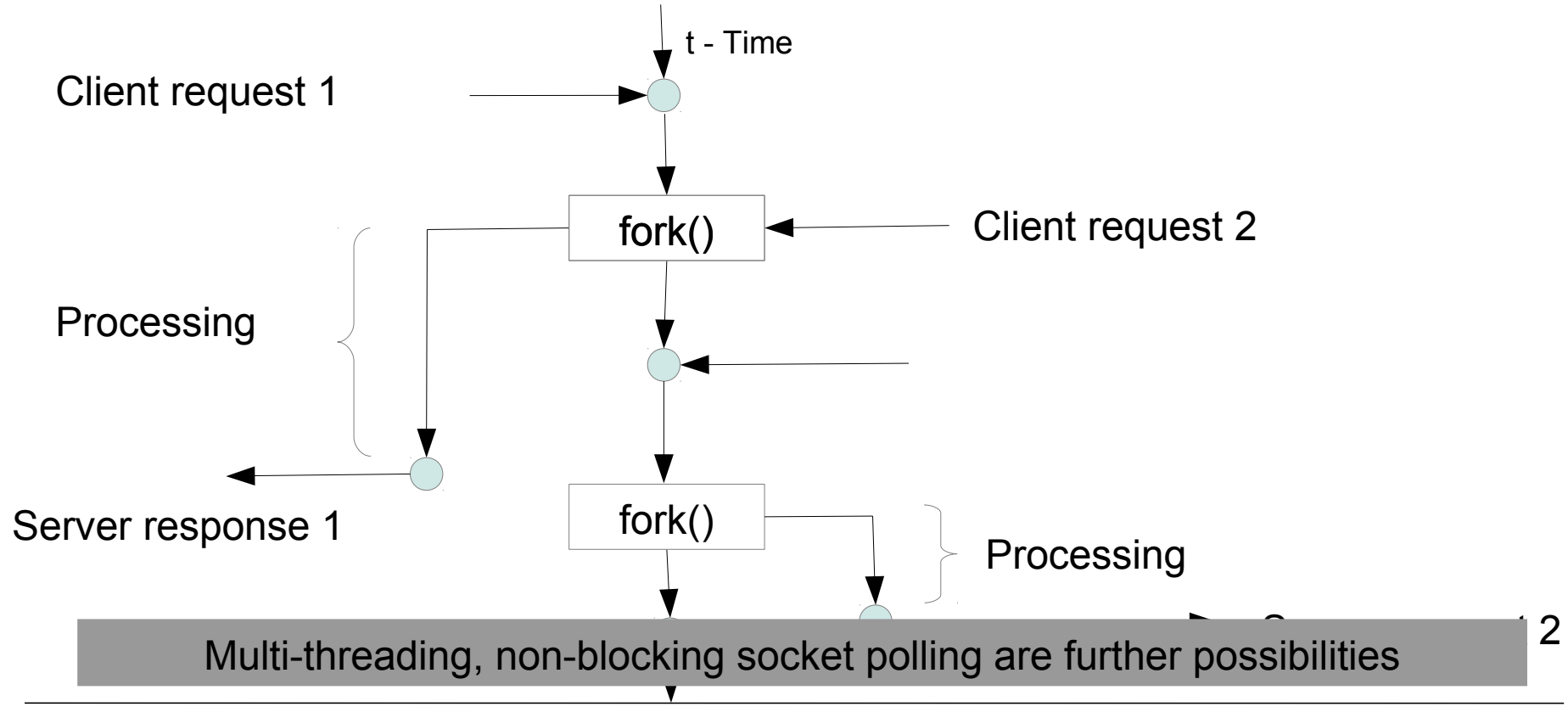
Concurrent Roles: Server

- Make task in the server accept client requests
- On reception, a new process is spawned
- Child process processes the different requests while main process remain idle/available for additional requests
- Child process send the reply and terminate

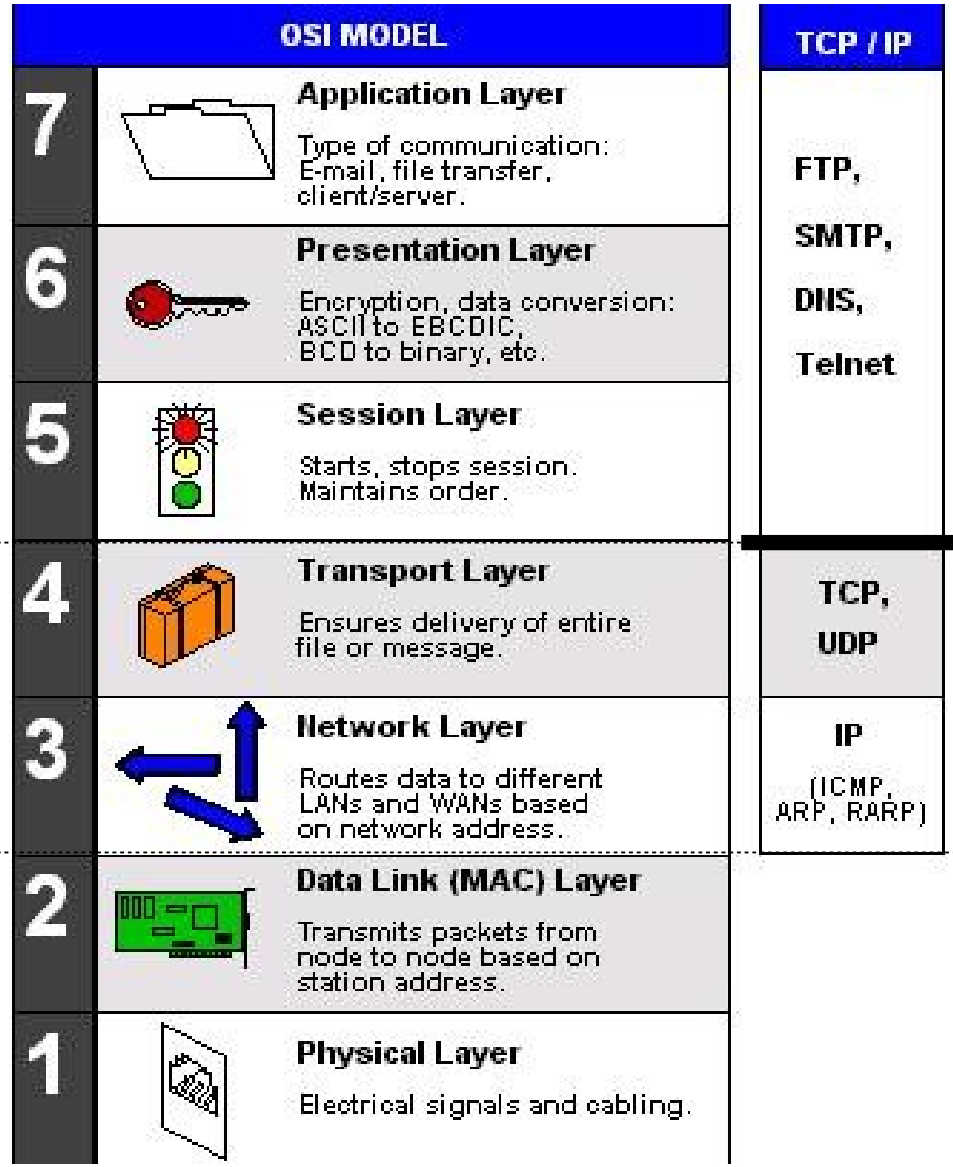


Concurrent Roles: Server

- Make task in the server accept client requests
- On reception, a new process is spawned
- Child process processes the different requests while main process remain idle/available for additional requests
- Child process send the reply and terminate



Recall: OSI Model and TCP/IP



User and System Programs

Berkeley Socket API

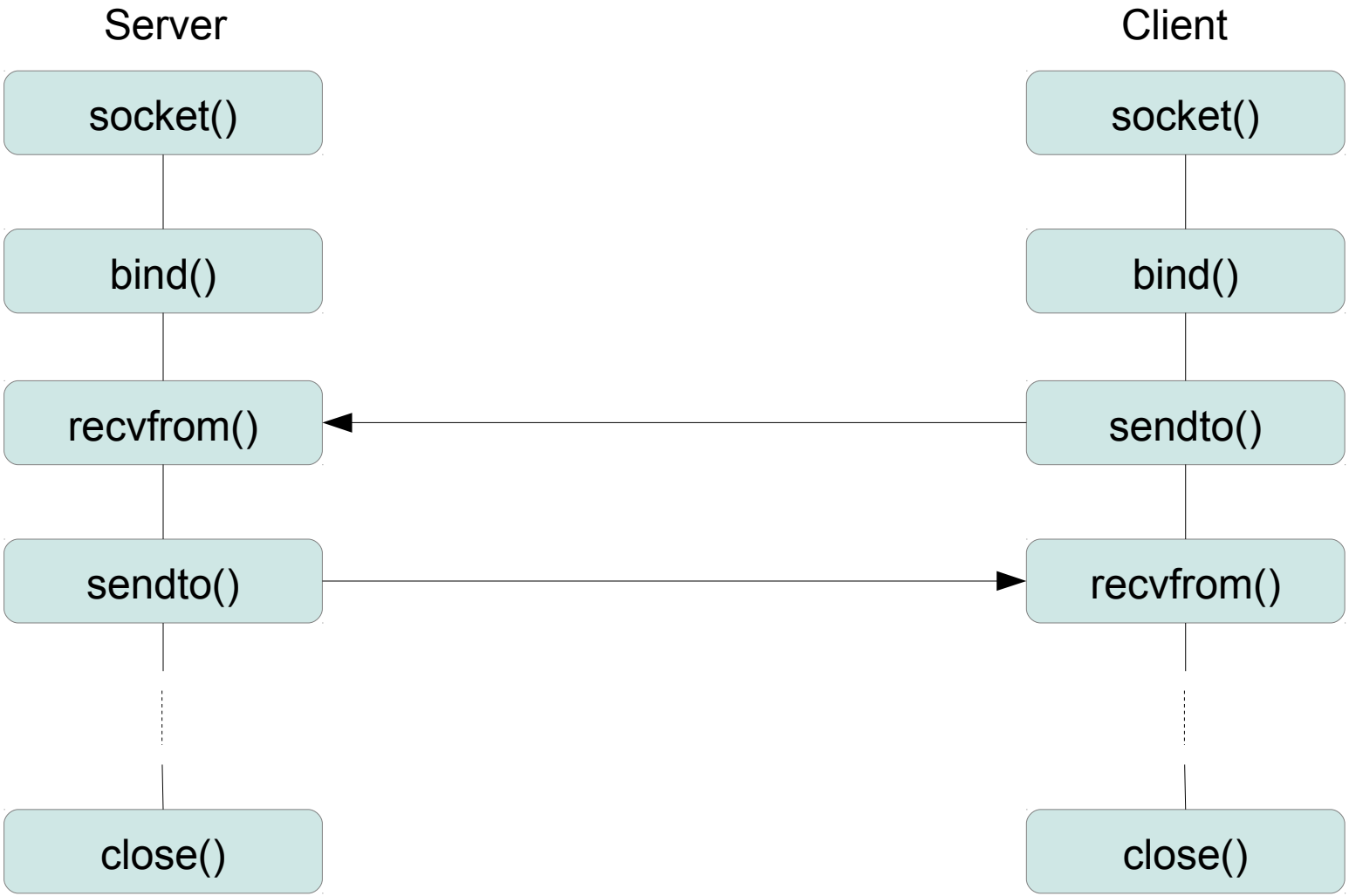
Kernel Support

Hardware

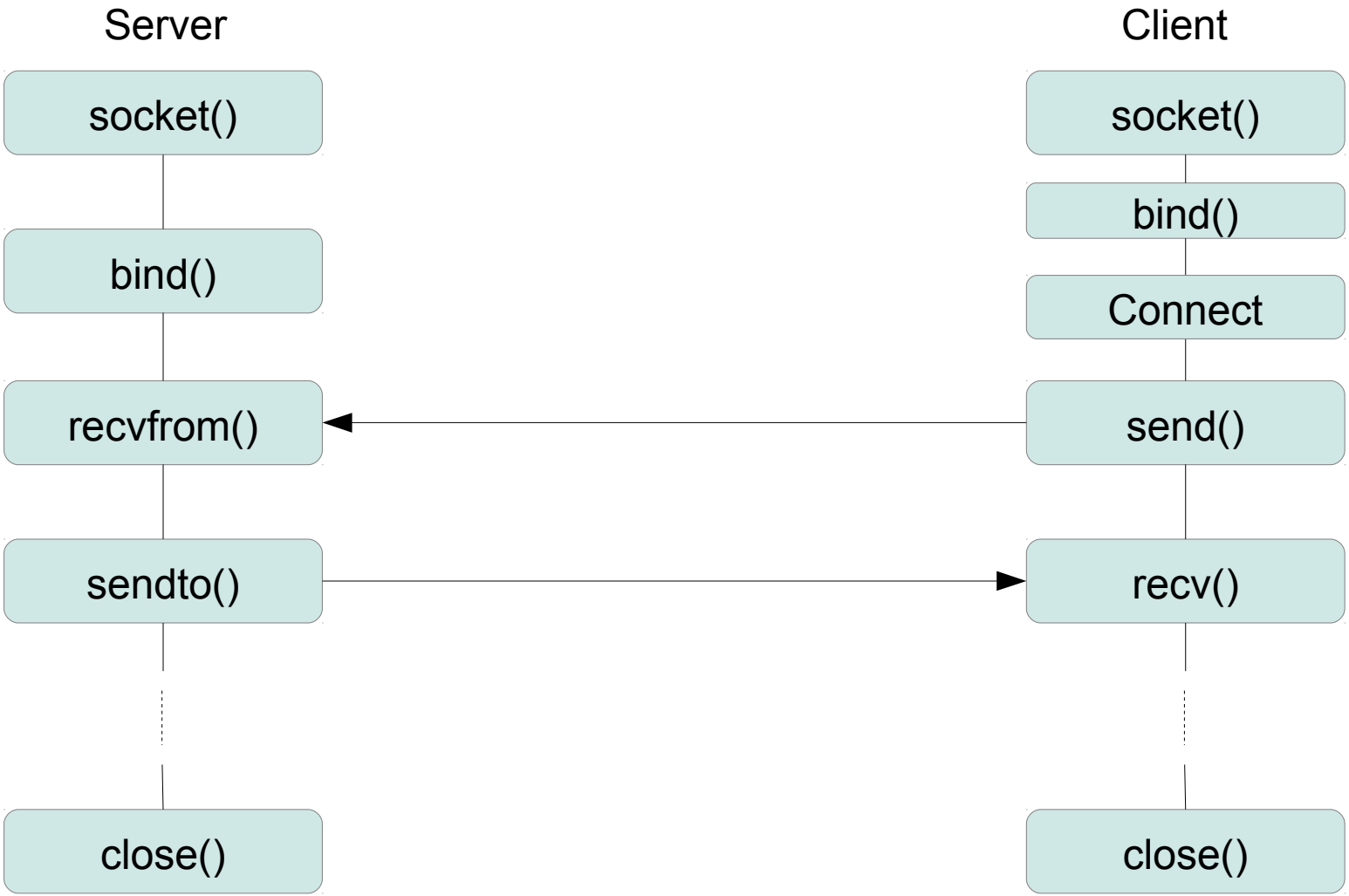
What is socket API?

- An abstraction that is provided to application programmer to implement distributed systems on the application layer.
- Support connection oriented and connection-less communication
- Allow network programming like file access:
  - User descriptors handles for remote process
  - I/O: `create, open, read, write, close`
- A socket specifies addresses and protocols
  - Address: Internet → Host/Port
  - Protocols: UDP, TCP
- A connection is defined by a socket pair, consist of:
  - (protocol, local-addr., remote-addr., local-process, foreign-process)
- Socket API offers Additional library, parameters, and managment

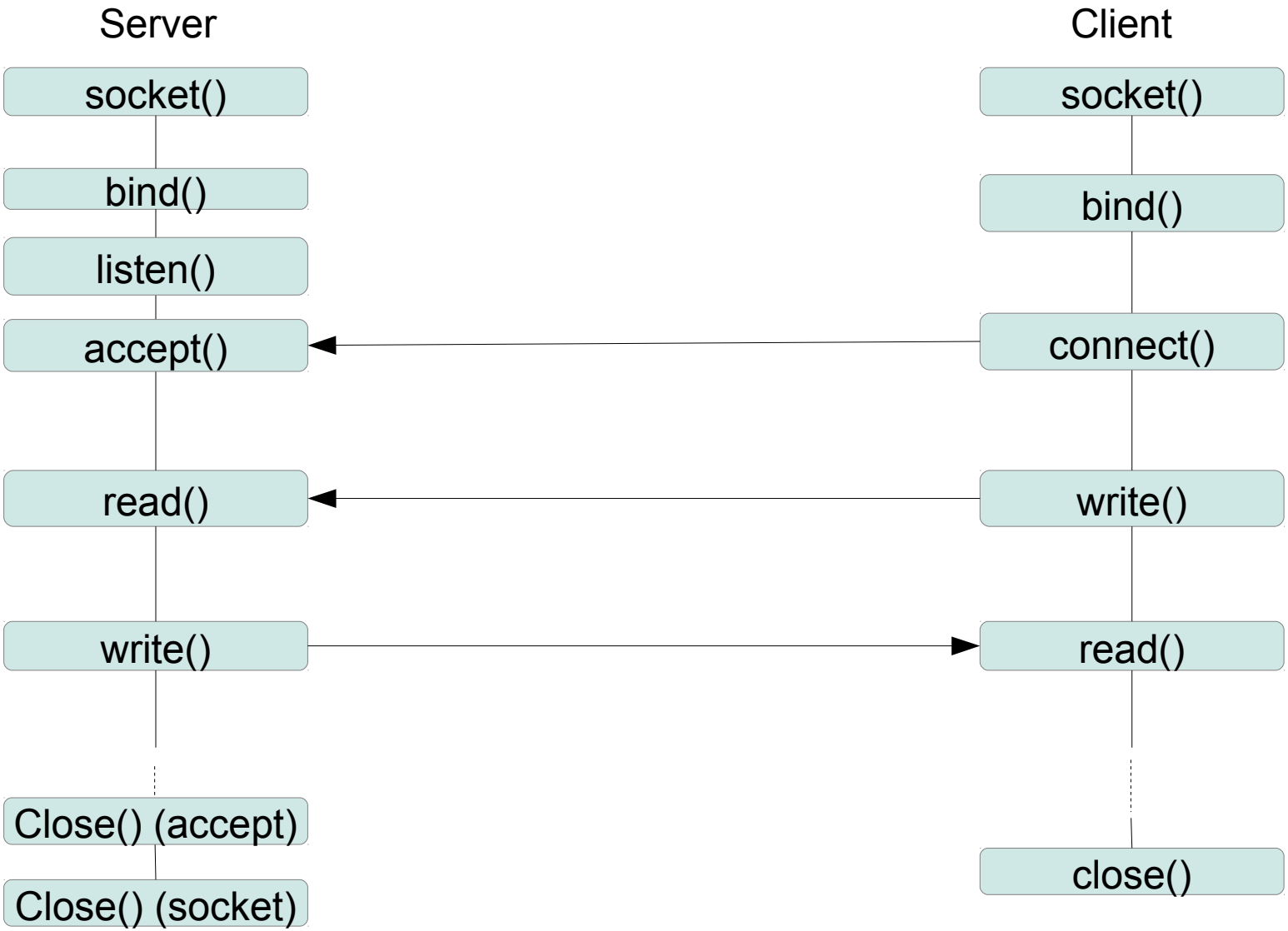
Socket Calls (Connection-less Protocol)



Socket Calls (Connection-less Protocol, “Connect”)



Socket Calls (Connection-Oriented Protocol)



Important Links

- Course webpage
  - http://www.p2p.tu-darmstadt.de/teaching/winter-term-20102011/p2p-networks-lecture/
- Mailing list
  - https://mail.rbg.informatik.tu-darmstadt.de/mailman/listinfo.cgi/p2p-ws11
- Forum
  - http://www.d120.de/forum/viewforum.php?f=229&start=0
- Webreg
  - https://www.dekanat.informatik.tu-darmstadt.de/webreg/index.php?page=login&lva=20.0117.4&semester=10W