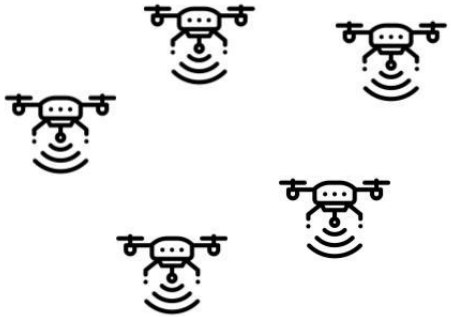
Dependable Distributed Systems
Master of Science in Engineering in Computer Science

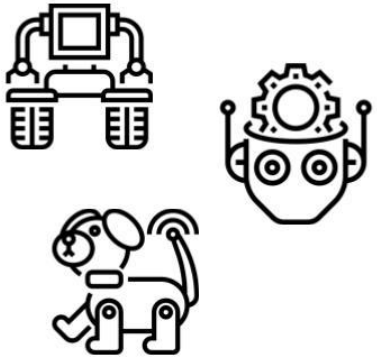
AA 2021/2022

Lecture 4: From Theory to Practice

Recall: Distributed System

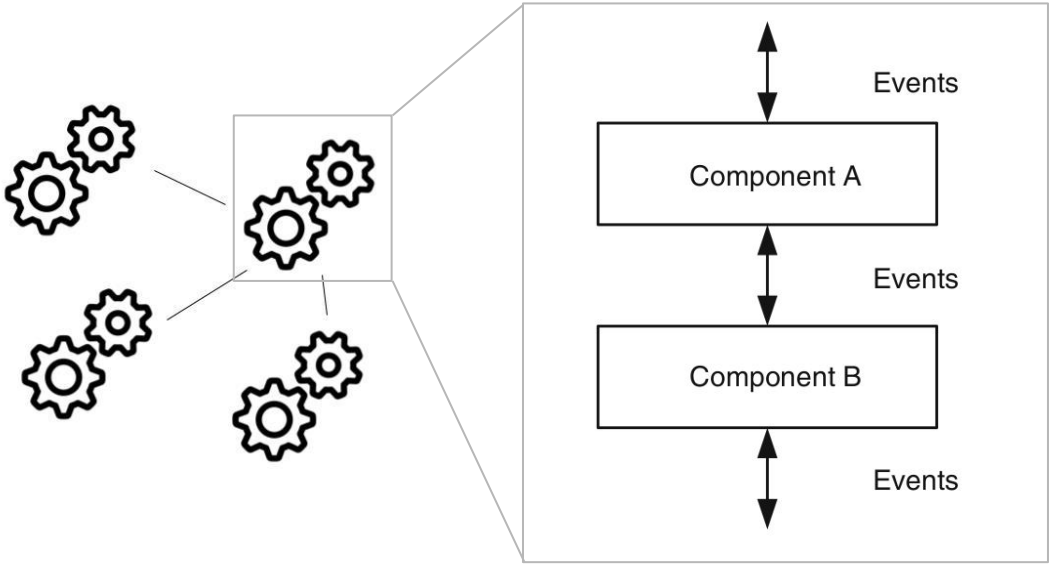
A distributed system is a **set of spatially separate entities**, each of these with a certain **computational power** that are **able to communicate and to coordinate** among themselves for reaching a **common goal**





icons taken from freeicons.io

From a Physical System to an Abstraction



Back from Theory to Practice

_> How to deploy a distributed system over Internet?

_> How to implement link abstraction?

_> How to set up the composition model?

_> ...

OSI Model

The Open Systems Interconnection model (OSI model) is a **conceptual model** that **characterises and standardises the communication functions** of a telecommunication or computing system **without regard to its underlying internal structure and technology**. Its goal is the interoperability of diverse communication systems with standard communication protocols.

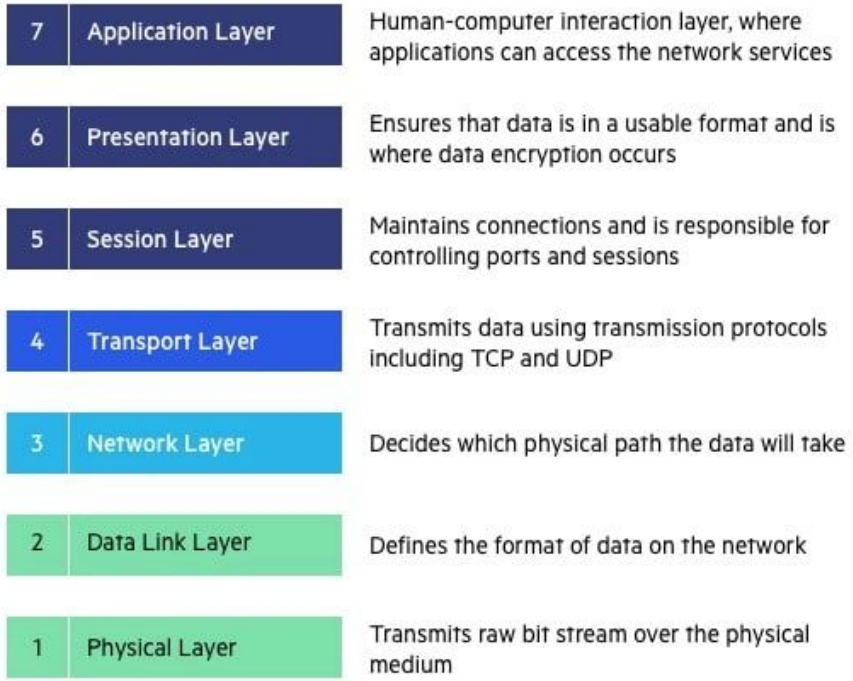


image taken from https://www.imperva.com/learn/application-security/osi-model/

IP

Internet Protocol (IP), Network Layer, has the task of **delivering packets from the source host to the destination host solely based on the IP addresses** in the packet headers.

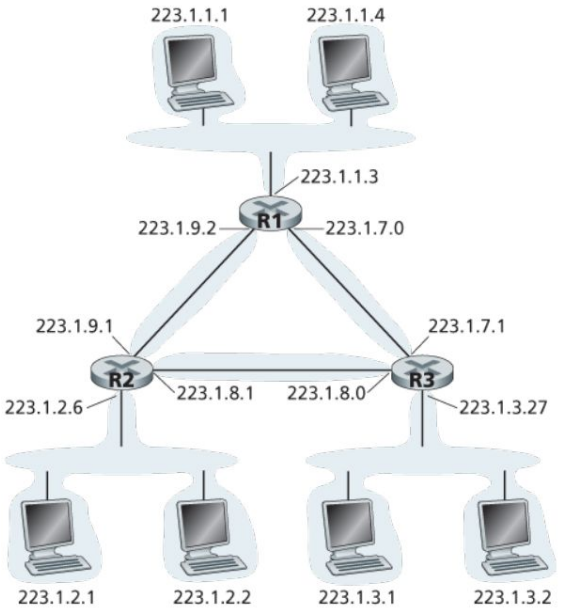


image taken from [1]

TCP

Transmission Control Protocol (TCP), Transport Layer, provides

- **“reliable”,**
- **ordered,**
- **error-checked**

delivery of a **stream** of octets (bytes) **between applications** running on hosts communicating via an IP network.

An application is identified by a **port number**

_> TCP establishes a **connection** between two applications (processes), Logical end-to-end transport

UDP

User Datagram Protocol (UDP), Transport Layer, provides

- **no guarantee on delivery**
- **no guarantee on ordering**
- **optional checksum**

delivery of a stream of octets (bytes) **between applications** running on hosts communicating via an IP network.

An application is identified by a **port number**

_> UDP directly send data to a IP-Address, port number, **connectionless**

Socket

A network socket is a **software structure** within a network node of a computer network that serves as an endpoint for sending and receiving data across the network, namely it **provides an API for the networking architecture**.

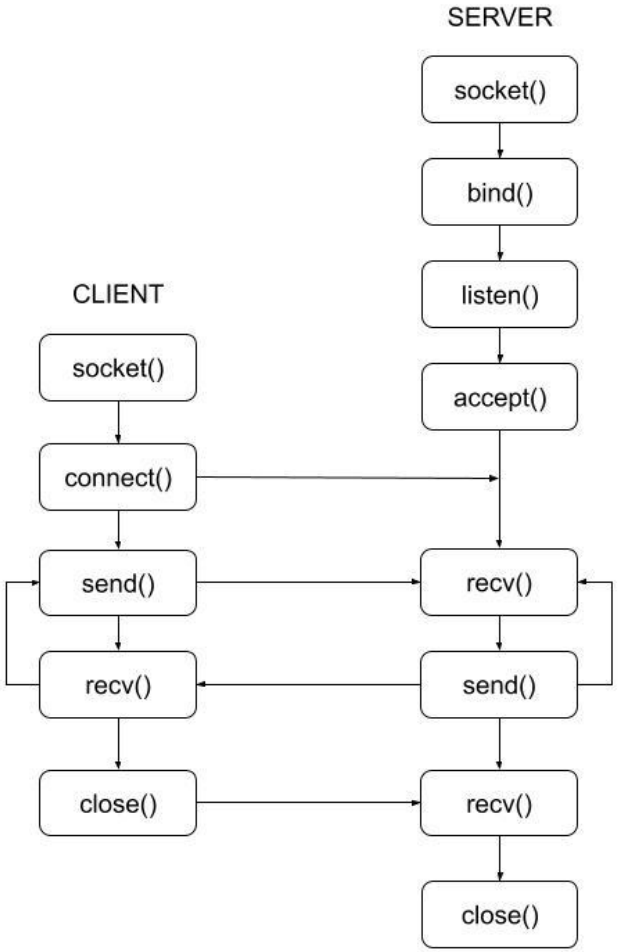
A pair of processes communicating over a network employs a **pair of sockets, one for each process**.

Stream sockets = exchange messages leveraging TCP

Datagram sockets = exchange messages using UDP

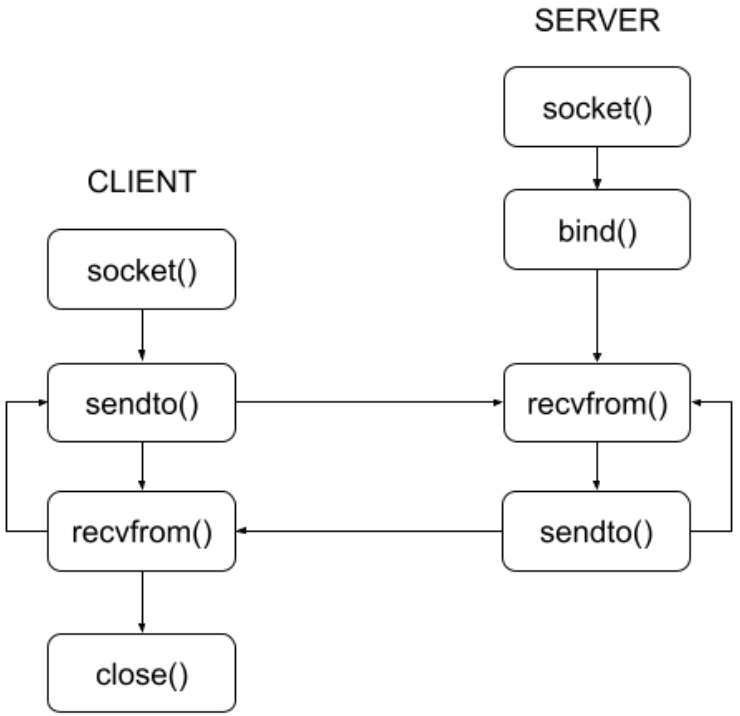
Stream (TCP) Socket

An application (process) is reachable through an **IP address** and a **port number**

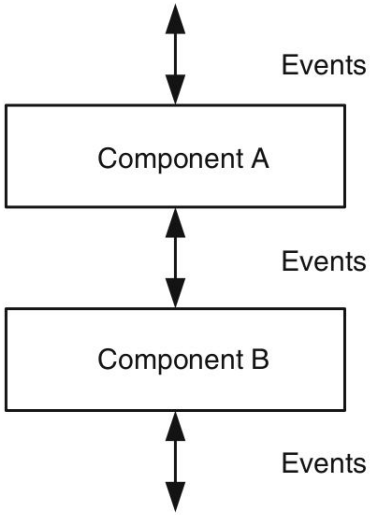


Datagram (UDP) Socket

An application (process) is reachable through an **IP address** and a **port number**



Event-Driven Programming



Event-driven programming is a programming paradigm in which the **flow** of program execution is **determined by events**.

The main routine is an event-loop that waits for an event to occur, and then invokes the appropriate event-handling routine.

Possible Implementation

_> programming language: Python

We need to **write the code to execute on every single process** (machine, virtual machine, container).

We split the code in two modules [2]:

- _> **process.py** containing the protocol executed by any process
- _> **link.py** containing the implementation of the link abstraction

Simulate a Distributed System Deployment: Mininet

Mininet [3][4] is a software creating a **virtual network**, **running real kernel, switch and application code, on a single machine** (VM, cloud or native).

_> you can code the network configuration and the protocol to execute with a Python script

_> It allows to **test correctness** of distributed system protocol,

_> It may not provide realistic performance metrics: all the virtual machines share the host machine

Note on UDP payload

The maximum safe UDP payload is 508 bytes. This is a packet size of 576 (the "minimum maximum reassembly buffer size"), minus the maximum 60-byte IP header and the 8-byte UDP header.

Any UDP payload this size or smaller is guaranteed to be deliverable over IP (though not guaranteed to be delivered). Anything larger is allowed to be outright dropped by any router for any reason. Except on an IPv6-only route, where the maximum payload is 1,212 bytes. As others have mentioned, additional protocol headers could be added in some circumstances. A more conservative value of around 300-400 bytes may be preferred instead.

The maximum possible UDP payload is 67 KB, split into 45 IP packets, adding an additional 900 bytes of overhead (IPv4, MTU 1500, minimal 20-byte IP headers).

Any UDP packet may be fragmented. But this isn't too important, because losing a fragment has the same effect as losing an unfragmented packet: the entire packet is dropped. With UDP, this is going to happen either way.

IP packets include a fragment offset field, which indicates the byte offset of the UDP fragment in relation to its UDP packet. This field is 13-bit, allowing 8,192 values, which are in 8-byte units. So the range of such offsets an IP packet can refer to is 0...65,528 bytes. Being an offset, we add 1,480 for the last UDP fragment to get 67,008. Minus the UDP header in the first fragment gives us a nice, round 67 KB.

Sources: RFC 791, RFC 1122, RFC 2460

source: https://stackoverflow.com/questions/1098897/what-is-the-largest-safe-udp-packet-size-on-the-internet

References

[1] Computer Networking: A Top-Down Approach. J. F. Kurose, K. Ross

[2] Code Repository
https://github.com/giovannifarina/DDS_primitives_and_protocols.git

[3] http://mininet.org/

[4] https://github.com/mininet/mininet/wiki/Introduction-to-Mininet