Why use just one computer? (An intro to Distributed Systems)

João Mota

On the menu

- I've got a system. Is it distributed?
- How processes communicate
- Multicast
- Hands-on!
- Final thoughts

I've got a system. Is it distributed?

Symptoms of a distributed system:

- Your system has **more than one process**
- Your system's processes (can) **run on different computers**
- Your system's processes **communicate with each other** (via messages)

Distributed systems in the wild

- The web and the internet
- E-mail
- ATM networks
- P2P applications (bittorrent and others)
- Swarm robotics
- IoT
- Autonomous driving
- Blockchain
- ...

How processes communicate

In distributed systems, **communication is key**. Different computers can't really cooperate if they don't understand how to talk to each other.

Application	← Application specific protocol
Transport	← Communication between two processes
Network	← Communication between computers that **aren't** directly connected
Data Link	← Communication between two computers that **are** directly connected

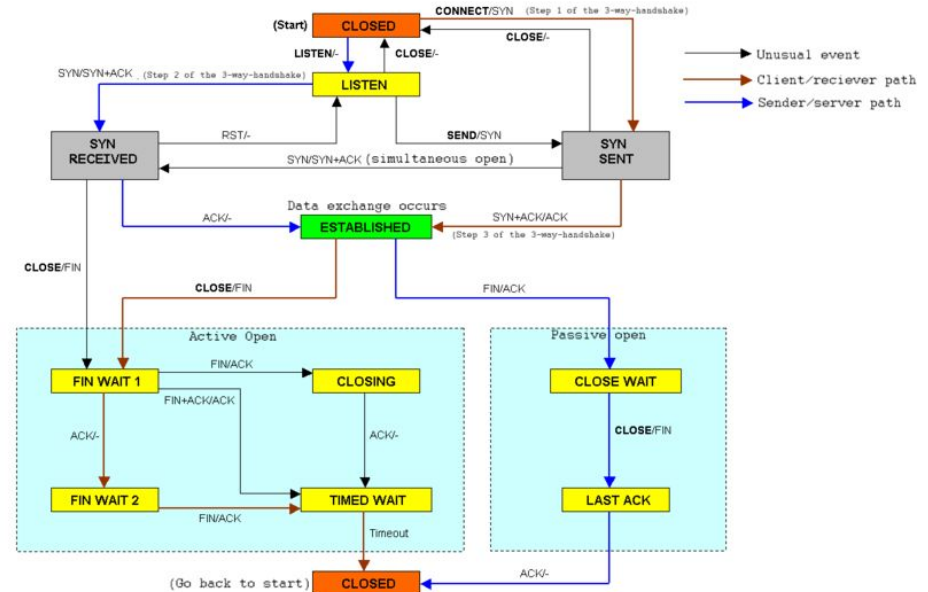# How processes communicate - Transport Layer

The two big transport layer protocols:

	UDP	TCP
Connection setup	No	Yes
Order	No	Yes
Reliability	No	Yes
Flow Control	No	Yes
Endpoints	*	1

How processes communicate - Transport Layer
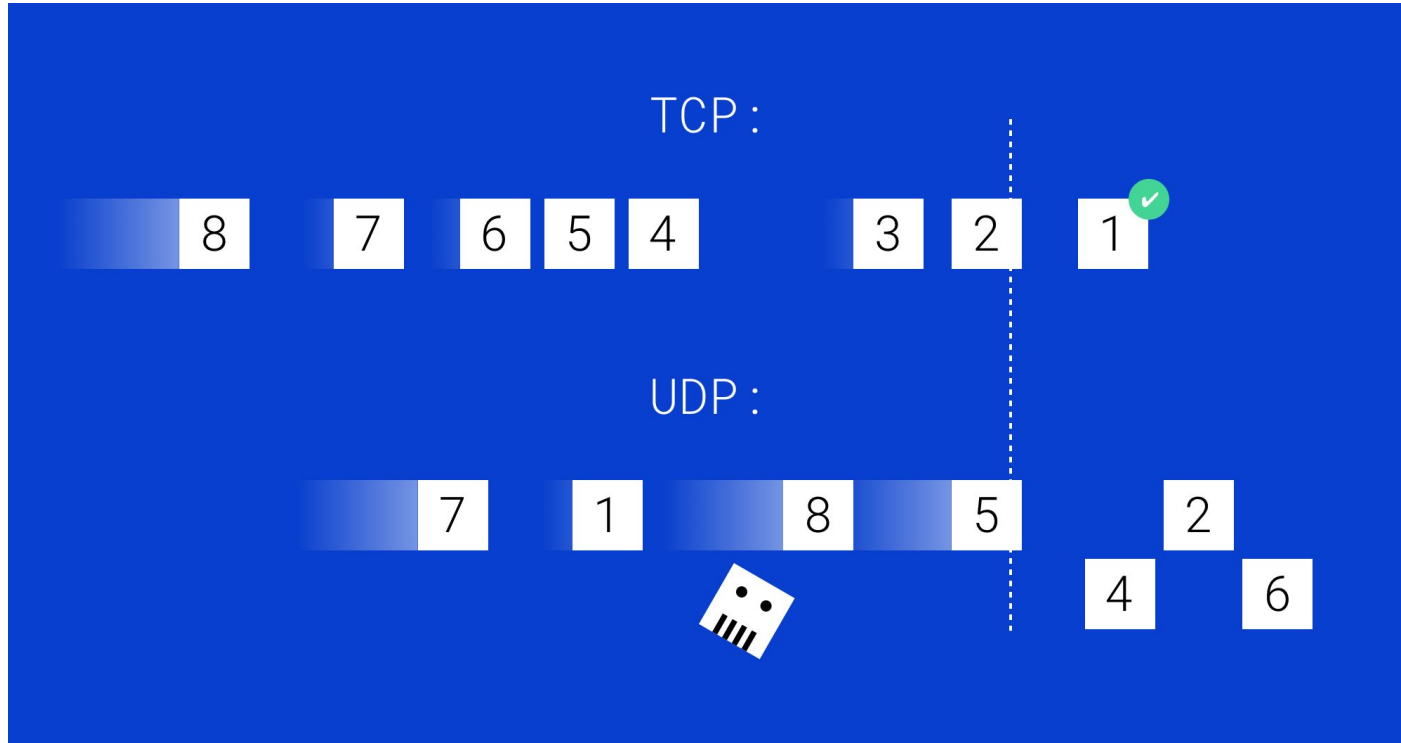
Connection setup:

How processes communicate - Transport Layer

Connection Setup - Advantages:

- More information = more control
  - Can ensure flow control
  - Can ensure order
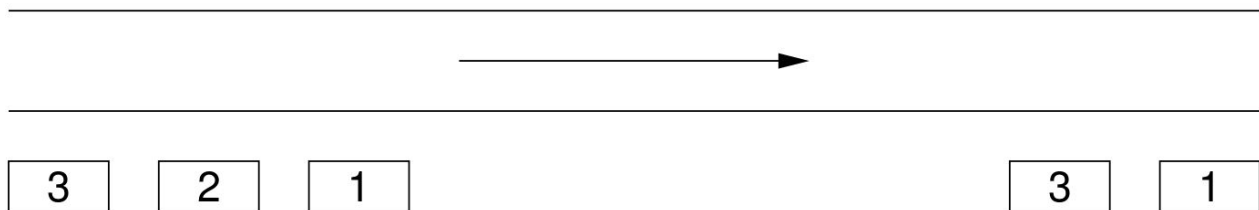 - Can ensure delivery
- Provides some security

How processes communicate - Transport Layer

Order:

TCP :
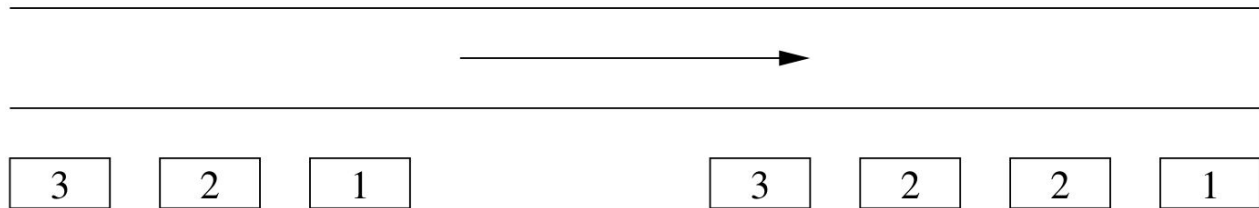
8   7   6   5   4   3   2   1 ✓

UDP :

7   1   8   5   2   4   6

How processes communicate - Transport Layer

Reliability - Packet loss:

| 3 | 2 | 1 | | 3 | 1 |

Reliability - Packet duplication:

| 3 | 2 | 1 | | 3 | 2 | 2 | 1 |

How processes communicate - Transport Layer

Flow control:
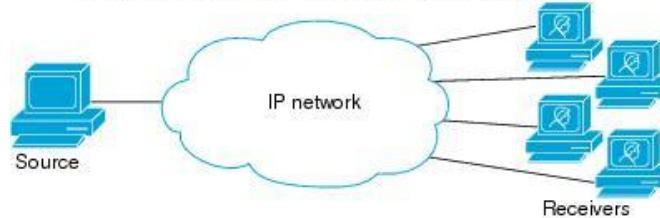

A Simple Example Of Flow Control
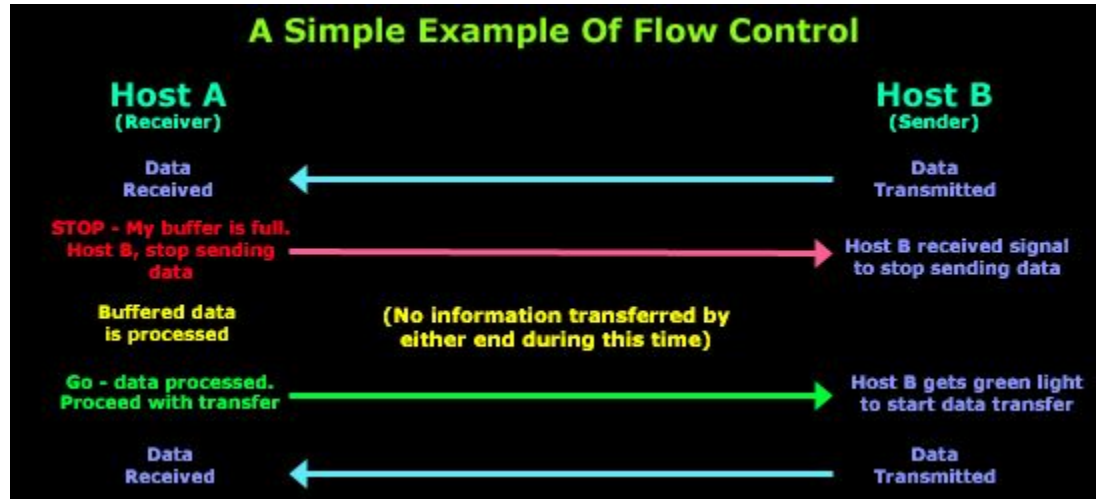
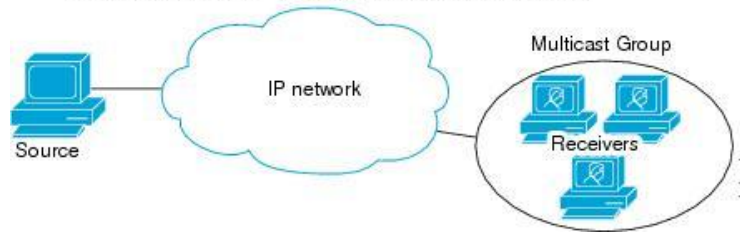# How processes communicate - Transport Layer

Endpoints:

Unicast transmission—One host sends and the other receives.

Broadcast transmission—One sender to all receivers.

Multicast transmission—One sender to a group of receivers.

How processes communicate - Transport Layer
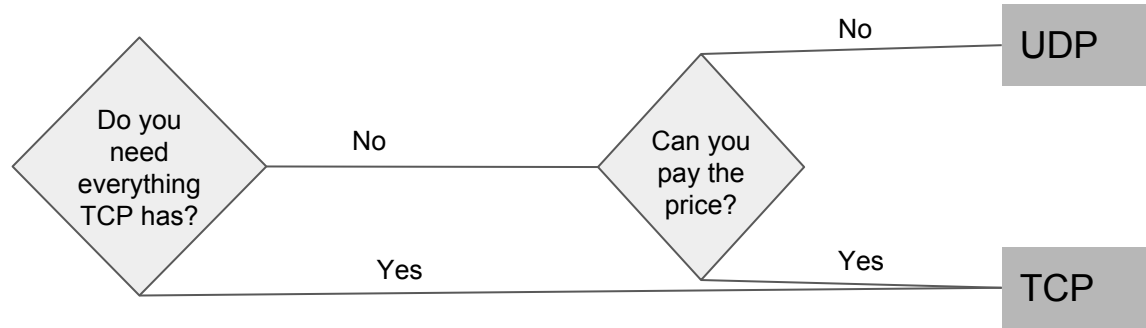
Extra: TCP Congestion Control

- Avoid, detect, react to congestion in a given medium
- Adapts to variable bandwidth
- Three main mechanisms:
  - Slow Start
  - Congestion Avoidance
  - Fast Retransmit/Recovery
- Provides """"fair"""" network bandwidth allocation
- End-to-end

How processes communicate - Transport Layer

How to choose the transport protocol for your application:

How processes communicate - Application Layer

Here is where you can really define your system's way of communicating. You can design your protocol any way you want, but there are some useful frameworks:

- Binary
- Text
- JSON
- XML

Multicast

- IP Multicast
- Application-level Multicast

Multicast

Multicast: A naïve implementation

- N senders
- M receivers
- N*M single-ended channels
- M messages must be sent for each message
- Same message will probably traverse the same link multiple times
- The sender needs to know all of its receivers

Multicast

IP Multicast

- Use a spanning tree containing the sender, all receivers and all nodes in between them
- The sender does everything as if it were doing unicast
- Receivers **subscribe** to the multicast group to be added to the spanning tree
- They can also **unsubscribe** to be removed
- IP multicast can only be used within the same network and using UDP
  - Reliable multicast is **hard**
  - ISPs don't provide multicast cross-networks
  - IPv6 supposedly does!

Multicast

Application-level Multicast

- Build an overlay network
- Can use spanning tree
- Alternatively use epidemic algorithms

Hands-on!

goo.gl/QLrdEk

Final thoughts

So, why use just one computer?

- It's simpler
- You don't need your processes to be on different computers
- You don't even have multiple processes
- [Distributed systems are a pain in the a**!](#)

Multicast

Why are distributed systems even a thing?

- They enable communication (potentially) without spatial boundaries
- They enable collaborative work across multiple systems
- They allow for bigger systems
- Spreading functionality may be useful!
  - Redundancy
  - Sharding
  - Divide-and-conquer algorithms
  - Partial anonymity

What now?

This workshop was just the tip of the iceberg.

- Remote Method Invocation (RMI)
- Cryptography
- Name resolution
- Serverless features
- Synchronization
- Fault Tolerance
- Consensus
- Atomic commitment

DISTRIBUTED SYSTEMS
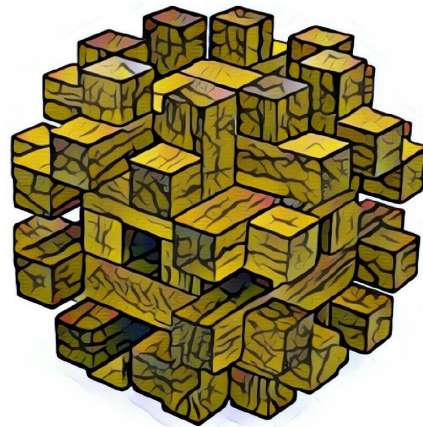
Maarten van Steen
Andrew S. Tanenbaum

THIRD EDITION - VERSION 01