Distributed Systems (DSy)

Protocols 1

OST
Eastern Switzerland
University of Applied Sciences

Thomas Bocek

27.02.2023

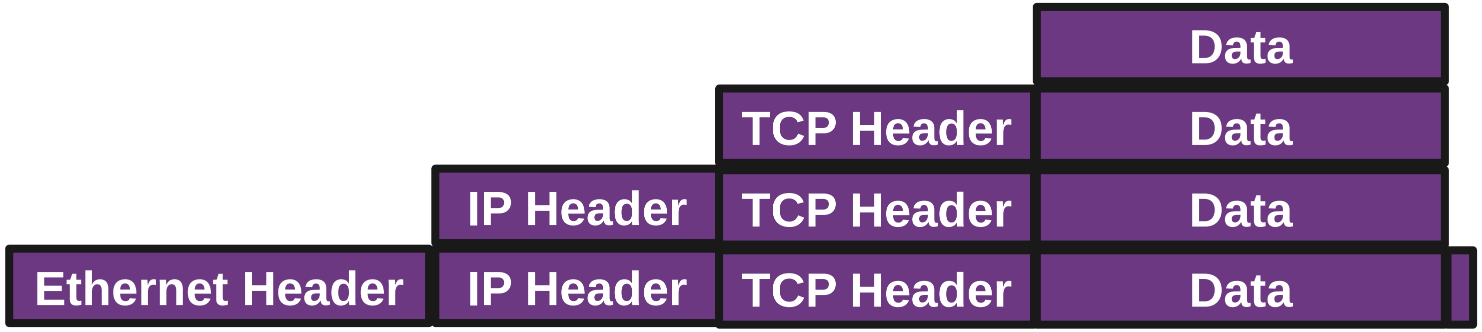
Learning Goals

- Lecture 3 (Protocols, part 1)
  - How do network layers work?
  - What are the TCP mechanisms?
  - What are the problems of TCP, and how other protocols (HTTP/3) can improve that

Networking: Layers

- Networking: Each vendor had its own proprietary solution - not compatible with another solution
  - IPX/SPX – 1983, AppleTalk 1985, DECnet 1975, XNS 1977
- Nowadays most vendors build compatible networks hardware/software from different vendors
  - Cisco, Dell, HP, Huawei, Juniper, Lenovo, Linksys, Netgear, MicroTik, Siemens, Ubiquiti, etc.
- Goal of layers: interoperability
  - 1984: ISO 7498 - The Basic Reference Model for Open Systems Interconnection

OSI model	"Internet model"
Application	Application
Presentation	
Session	
Transport	Transport
Network	Internet
Data link	Link
Pysical	



Networking: Definitions

RFC 1122, Internet STD 3 (1989)	OSI model
Four layers	Seven layers
"Internet model"	OSI model
Application	Application
	Presentation
	Session
Transport	Transport
Internet	Network
Link	Data link
	Physical

source: https://en.wikipedia.org/wiki/Internet_protocol_suite

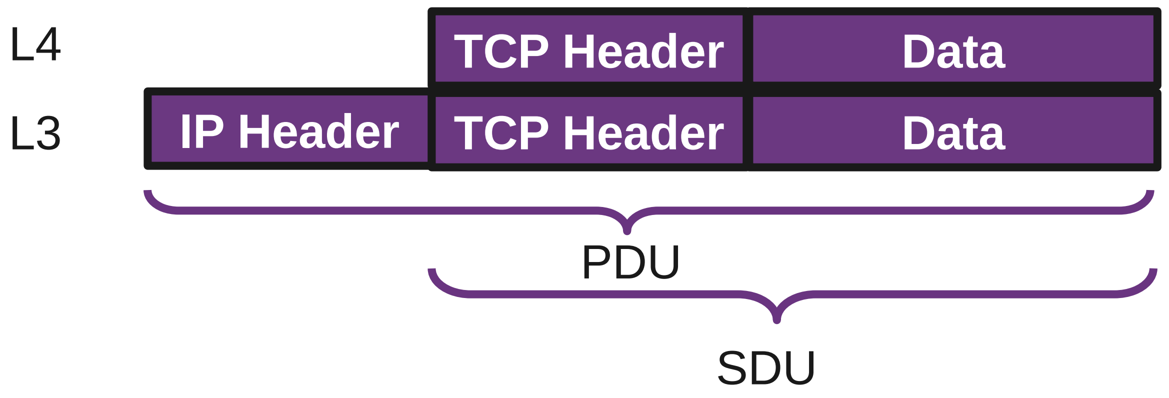
OST

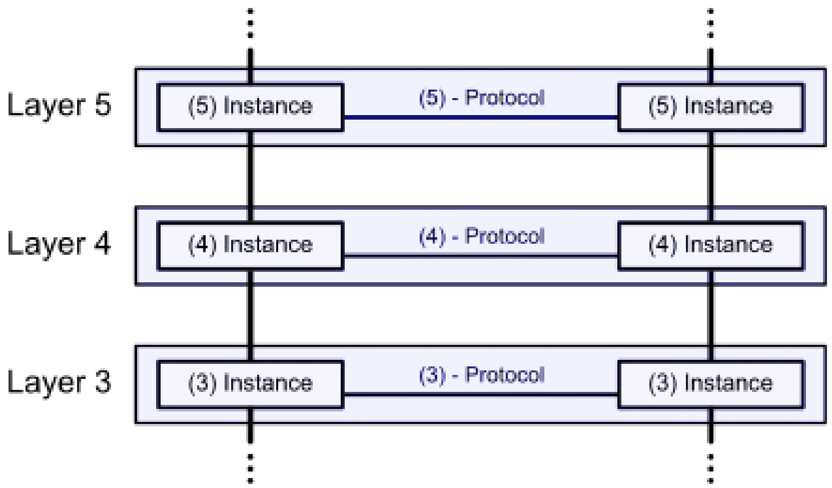
Layer Abstraction

- Protocols enable an entity/instance to interact with an entity/instance at the same layer in another host

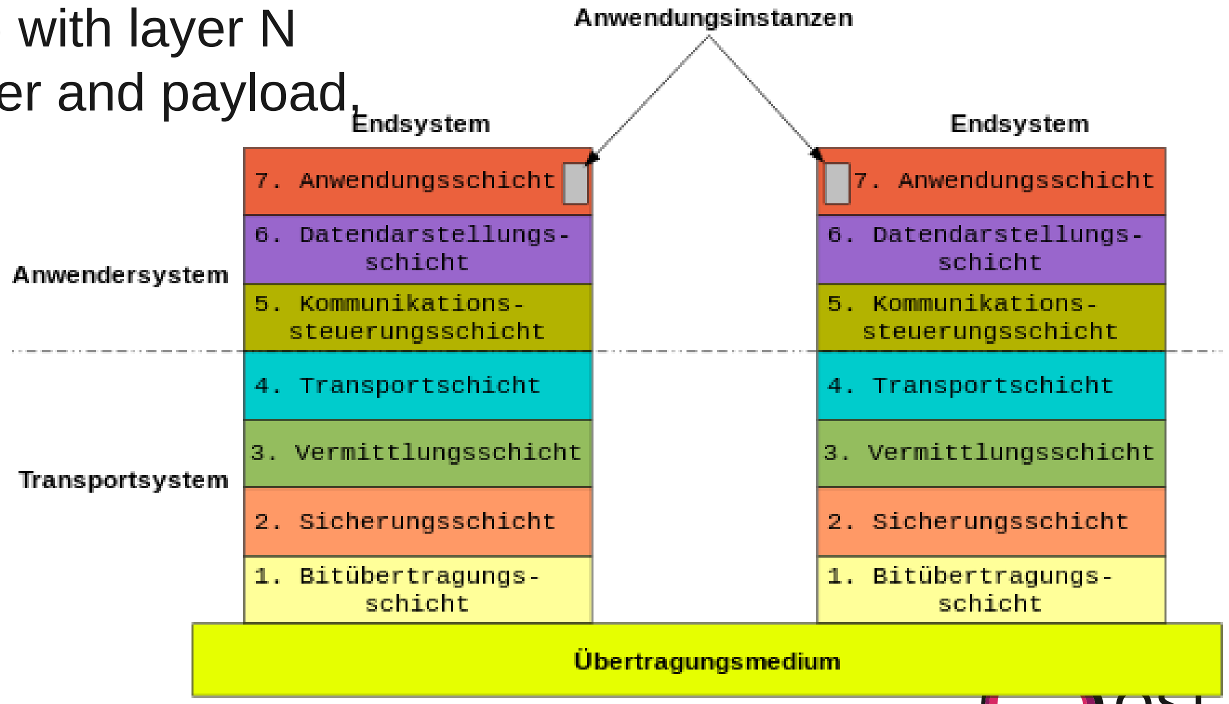
- Service definitions: provide functionality to an (N)-layer by an (N-1) layer

- Layer N exchange protocol data units (PDUs) with layer N protocol. Each PDU contains a protocol header and payload, the service data unit (SDU). E.g. PDU of L3:





source: https://en.wikipedia.org/wiki/OSI_model



source: https://de.wikipedia.org/wiki/OSI-Modell

Layer 4 - Transport

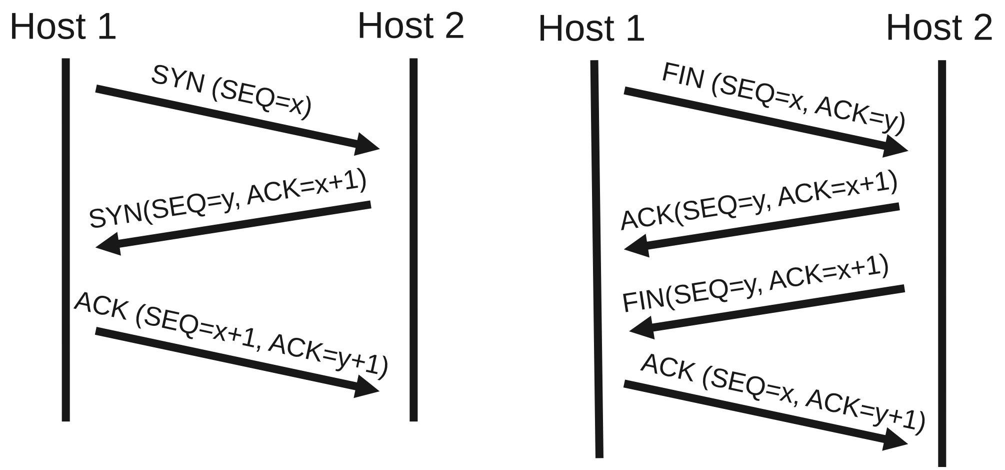
- TCP (Transmission Control Protocol)
  - Reliable (retransmission)
  - Ordered
  - Window – capacity of receiver
  - Checksum – 16bit (crc16)
  - TCP overhead: 20bytes
    - IP overhead: 20bytes
    - Ethernet frame: 18bytes (crc32)
- TCP tries to correct errors; you don't need to worry…
  - Sometimes, you need to worry…

```
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|          Source Port          |       Destination Port        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                        Sequence Number                        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                    Acknowledgment Number                      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  Data |           |U|A|P|R|S|F|                               |
| Offset| Reserved  |R|C|S|S|Y|I|            Window             |
|       |           |G|K|H|T|N|N|                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|           Checksum            |         Urgent Pointer        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                    Options                    |    Padding    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                             data                              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

source: http://freesoft.org/CIE/Course/Section4/8.htm

Layer 4 - TCP

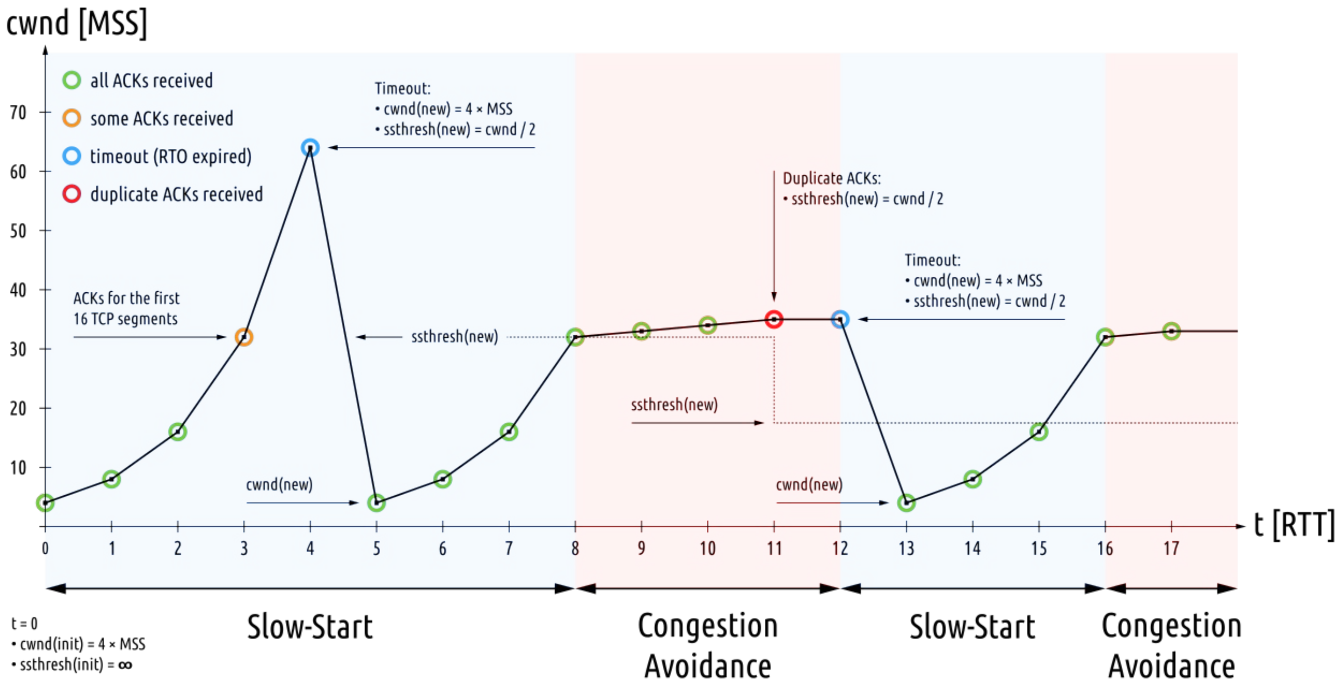
- Connection establishment
  - SYN, SYN-ACK, ACK (three way)
  - Initiates TCP session: initial sequence number is ~ random
- Connection termination
  - FIN, ACK + FIN, ACK (three/four way)
  - 3-way handshake, when host 1 sends a FIN and host 2 replies with a FIN & ACK
- Sequences and ACKs
  - Identification each byte of data
  - Order of the bytes → reconstruction
  - Detecting lost data: RTO, DupACK:



- Retransmission timeout
  - If no ACK is received aftert timout (e.g. 2xRTT), resend.
- Duplicate cumulative acknowledgements, selective ACK [link]
  - ACKs for last consecutive packets
  - 3 times same ACK → retransmit missing packets (fast retransmit)

Layer 4 - TCP

- Flow control
  - Sender is not overwhelming a receiver
  - Back pressure
  - Sliding window:
    - Receiver specifies the amount of additionally received data in bytes that can be buffered
    - Sender up to that amount of data before ACK
- Congestion control
  - slow-start
  - congestion avoidance
- Difference flow/congestion control



source:
https://upload.wikimedia.org/wikipedia/commons/thumb/2/24/TCP_Slow-Start_and_Congestion_Avoidance.svg/1280px-TCP_Slow-Start_and_Congestion_Avoidance.svg.png

TCP/IP from an Application Developer View

- Server in golang (repo)
  - git clone https://github.com/tbocek/DSy
  - Download GoLand, or others
  - go run server.go → server
- Listening on TCP port 8081
  - Return string in uppercase
- Node.js version
  - Download WebStorm, or other
- Client:
  - nc localhost 8081

```
const net = require('net');
const server = new net.Server();
server.listen(8081, function() {
  console.log('Launching server...');
});

server.on('connection', function(socket) {
  socket.on('data', function(chunk) {
    console.log(`Data received from client: ${chunk.toString()}`);

socket.write(chunk.toString().toUpperCase() + "\n");
  });
});
```

```
package main
import ("bufio"
    "fmt"
    "net"
    "strings")
func main() {
    fmt.Println("Launching server...")
    ln, _ := net.Listen("tcp", ":8081") // listen on all interfaces
    for {
        conn, _ := ln.Accept() // accept connection on port
        message, _ := bufio.NewReader(conn).ReadString('\n') //read line
        fmt.Print("Message Received:", string(message))
        newMessage := strings.ToUpper(message) //change to upper
        conn.Write([]byte(newMessage + "\n")) //send upper string back
    }
}
```

TCP Considerations

- Fallacy 2: Latency is zero
  - Nürnberg data center: 15ms, Australia: 300ms
  - Ping ftp.au.debian.org , Starklink + 50ms
- Problem: TCP handshake is not flexible
  - You need a handshake (1RT)
    1) If you want to make sure the other side accepts packets (and not drop it) - ensure both sides are ready to transmit and receive data
    2) If you want to exchange public / private keys
  - TCP supports 1) but not 2)
    - Use another security layer for 2), but a security layer needs at least 1 RT
    - TCP + Security = at least 2 RT
      - Nürnberg + Starlink:2 x (15 + 50ms) = 130ms
      - Australian: (2 x 300ms) = 600ms

- TCP + Security at least 2 RT
  - DNS query may be required too: 3 RT
  - Old security protocols add RT: 4RT
- Starklink, Kuiper, OneWeb in lower orbit
  - Viasat is 631 ms (higher orbit)
- Worst case: Starlink/Australia/DNS/TCP/old sec: 1.4s before data can be sent → new protocols on the way (HTTP/3)
- Wireshark