Internet-Technologie & Web Engineering
Introduction

Dr.-Ing. Matthäus Wander

Universität Duisburg-Essen

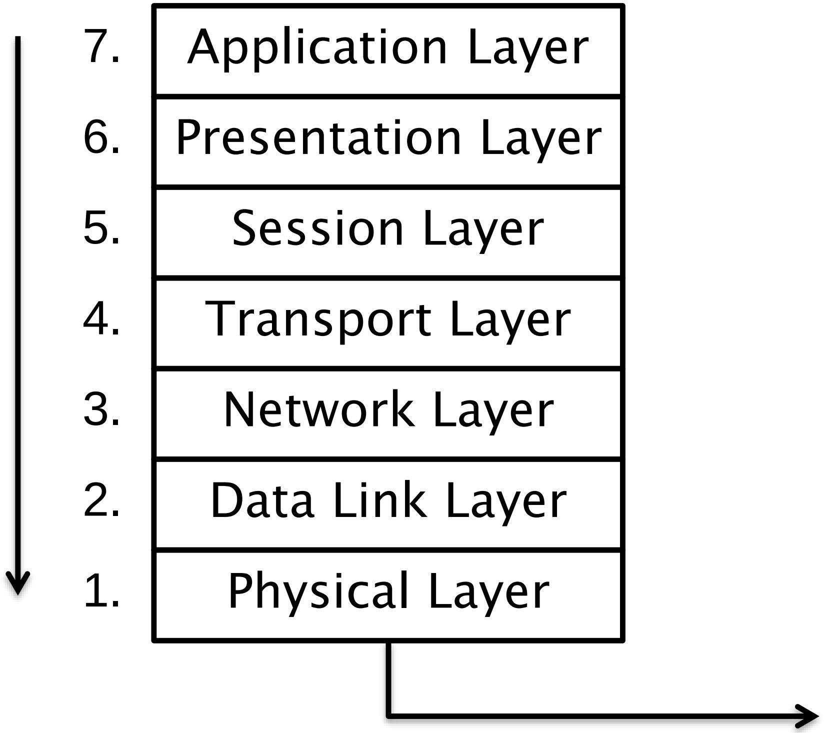
Goal

- Applications run on more than one computer
- Why?
  - Communicate with people (Facebook, Skype)
  - Share files (Dropbox, BitTorrent)
  - Access remote data/functionality (LSF, online shop)

⇨ Distributed Application or Distributed System

- What do we need?
  - Computer network
  - Rules for network communication

ISO/OSI Network Model

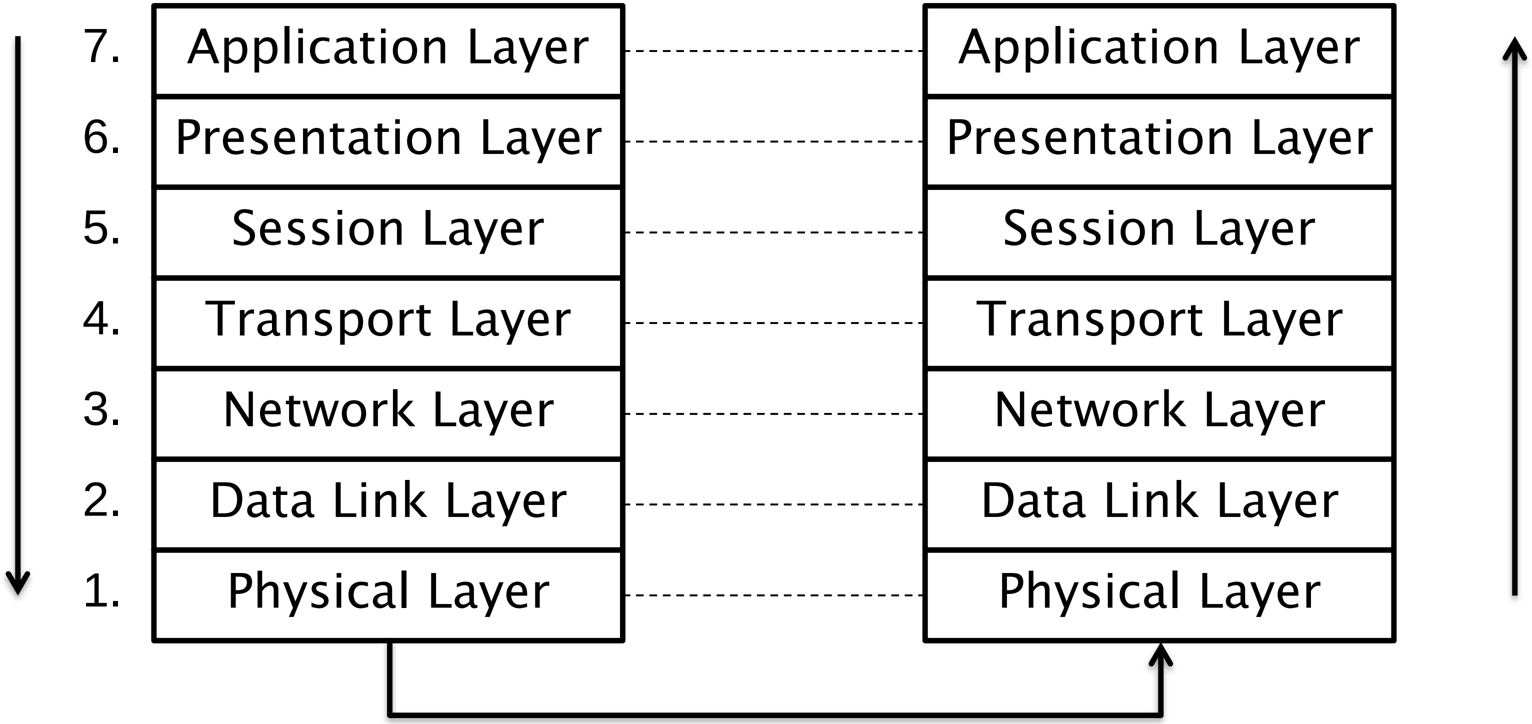


Physical medium (copper cable, fiber, radio transmission)

Sender:

- Protocol Data Unit (PDU) handed to bottom layer
- Each layer prepends PDU from upper layer with a header (or footer)

ISO/OSI Network Model

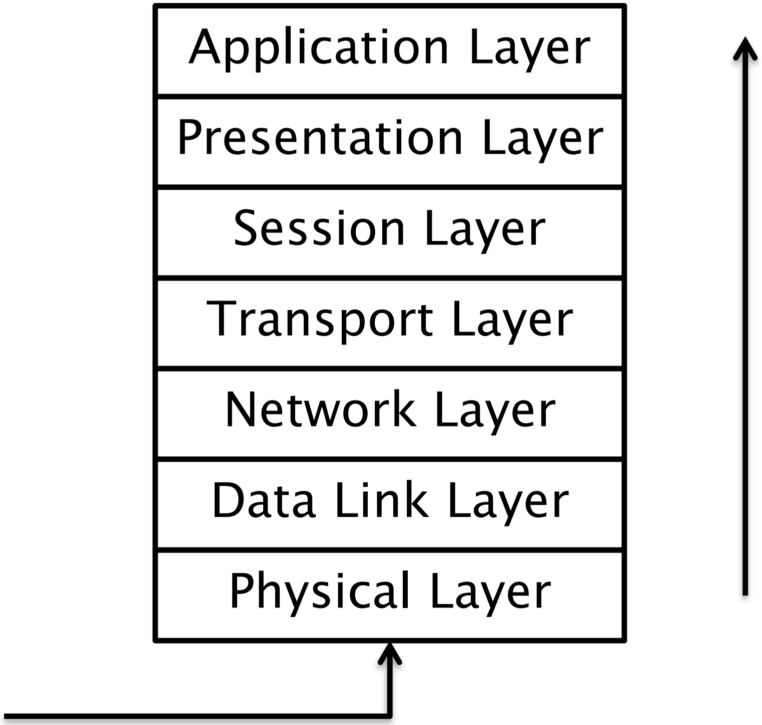


Physical medium (copper cable, fiber, radio transmission)

ISO/OSI Network Model

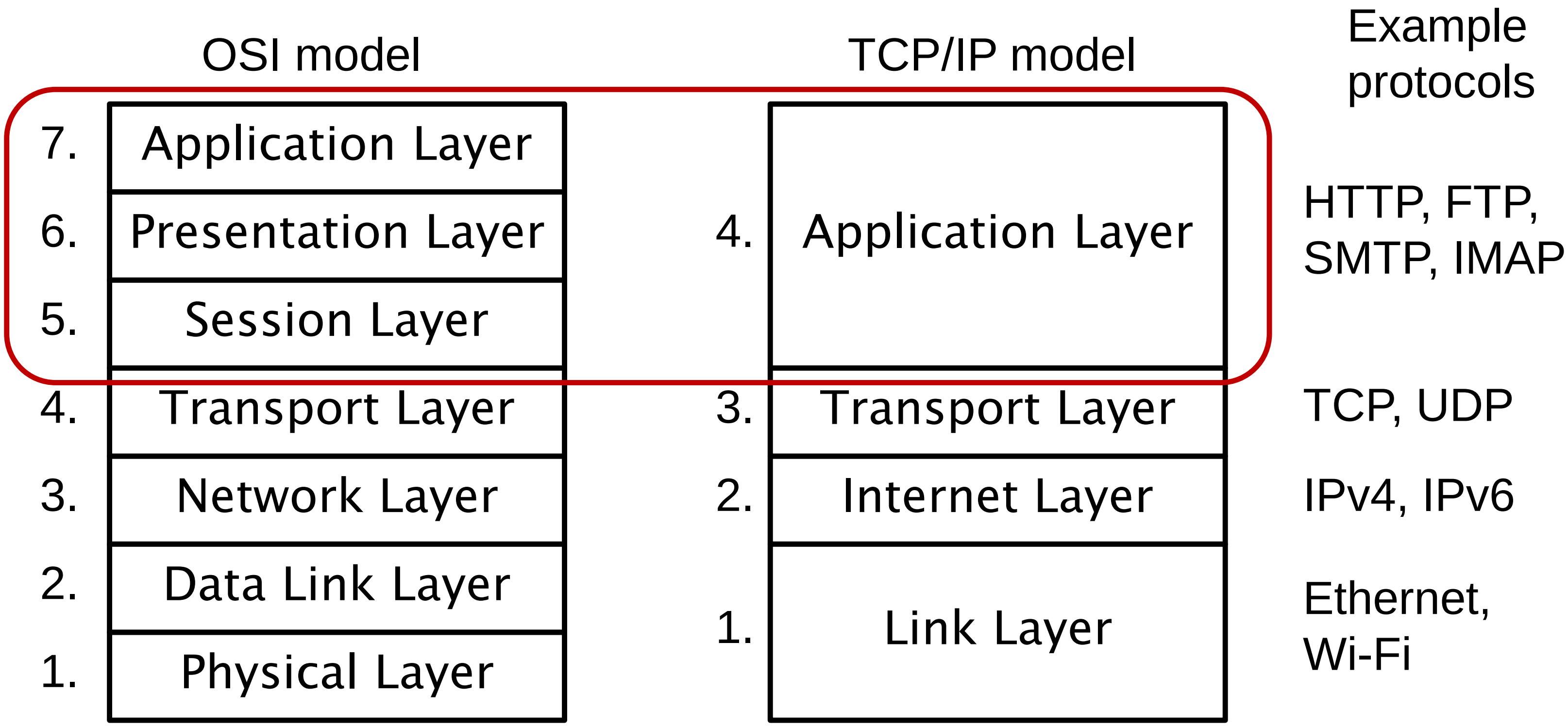
Receiver:

- Each layer strips header (or footer) from PDU, performs its function
- Gives PDU to upper layer



Physical medium (copper cable, fiber, radio transmission)

Comparison: OSI Model vs. TCP/IP Model



	OSI model		TCP/IP model	Example protocols
7.	Application Layer	4.	Application Layer	HTTP, FTP, SMTP, IMAP
6.	Presentation Layer			
5.	Session Layer			
4.	Transport Layer	3.	Transport Layer	TCP, UDP
3.	Network Layer	2.	Internet Layer	IPv4, IPv6
2.	Data Link Layer	1.	Link Layer	Ethernet, Wi-Fi
1.	Physical Layer			

Link Layer – Physical and Data Link Layer

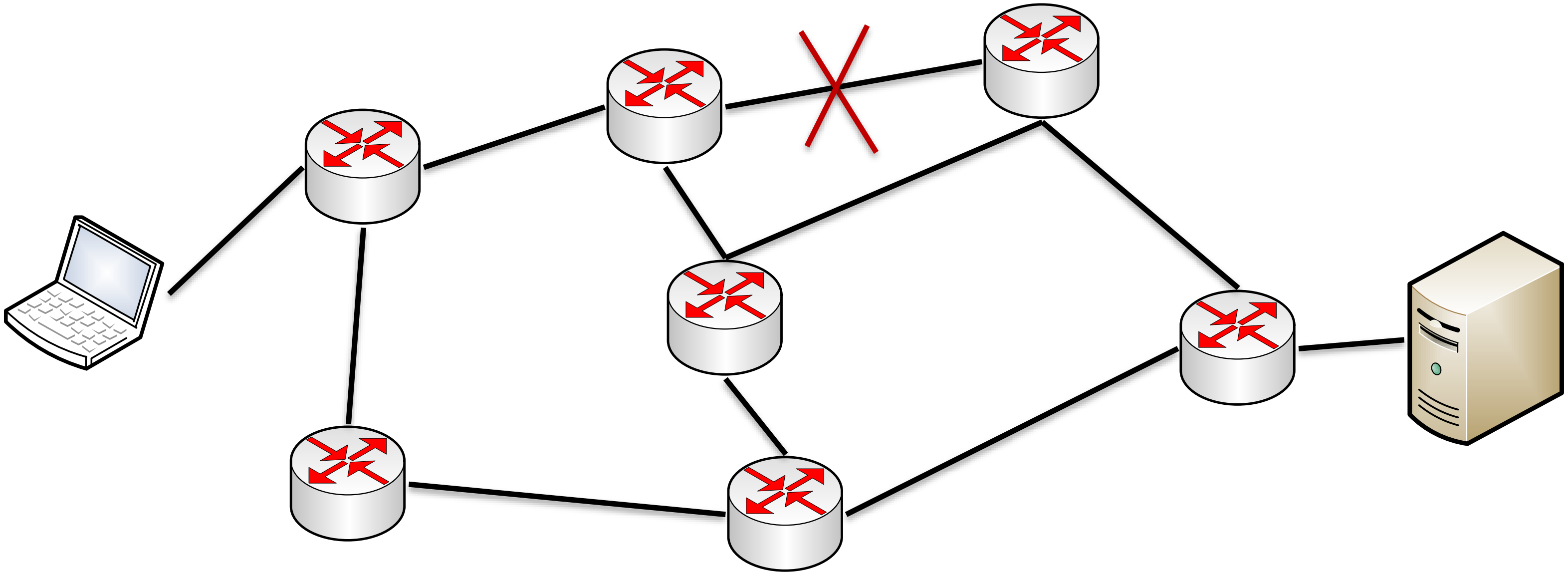
- Communication interface to the local network
  - Data transmission of directly connected computers
- Protocols: Ethernet, Wi-Fi, Bluetooth, ZigBee, …
- Data units: **frames**
- Adress type: MAC address (48/64 bit)
  - Example: 00:80:41:ae:fd:7e
- Tasks:
  - Cooperative access to (shared) network media
  - Bit encoding

Internet Layer – Network Layer

- Data transmission between computers (**hosts**)
- Protocols: IPv4, IPv6, …
- Data unit: **packets**
- Address type: IP address (32/128 bit)
    - Examples: 134.91.78.133   2001:638:501:8efc::133
- Tasks:
    - Routing: find best way for packet through network
    - Fragmentation: split large packet into smaller packets
    - Network congestion: handle overloaded network links

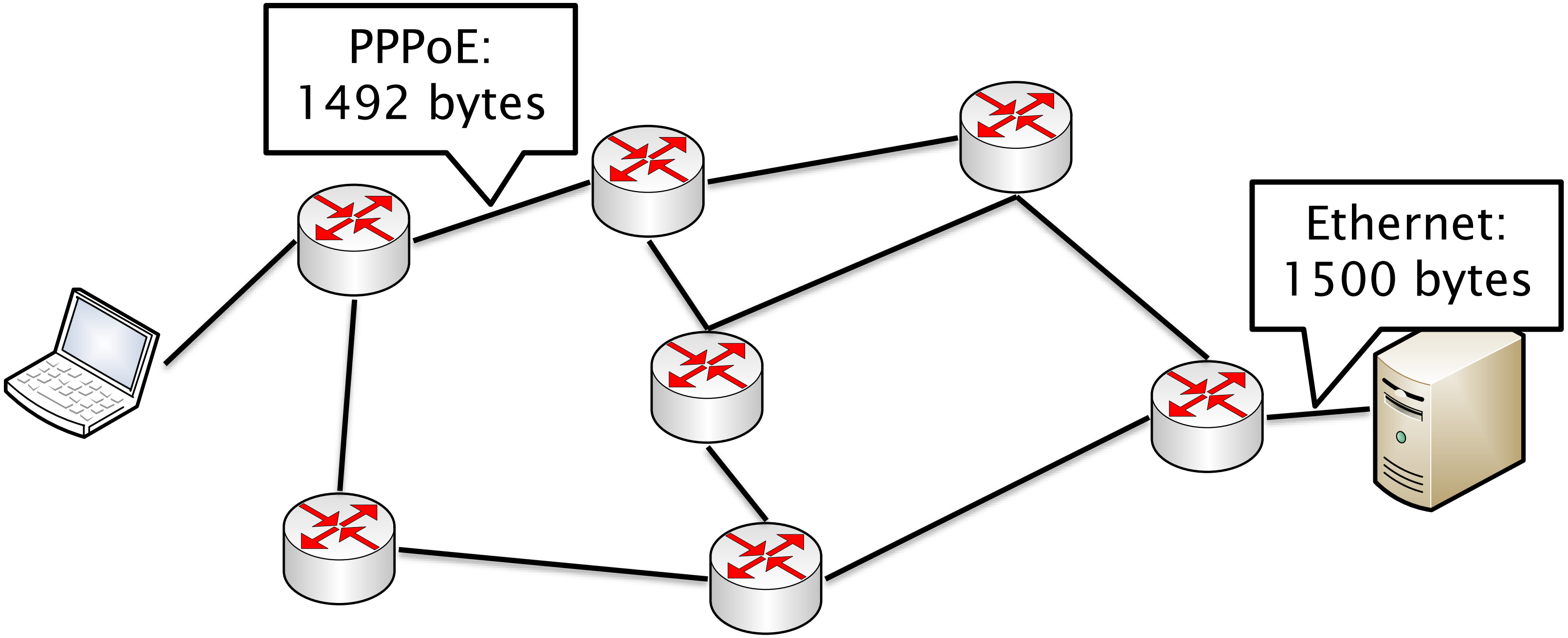
Routing

- Find best way for packet through network
  - Within a network (e.g. campus) & between networks
- Keep track of network topology when network links fail



Fragmentation

- Split large packet into smaller packets
- Network links have packet size limitations
  - **Maximum Transmission Unit** (MTU)

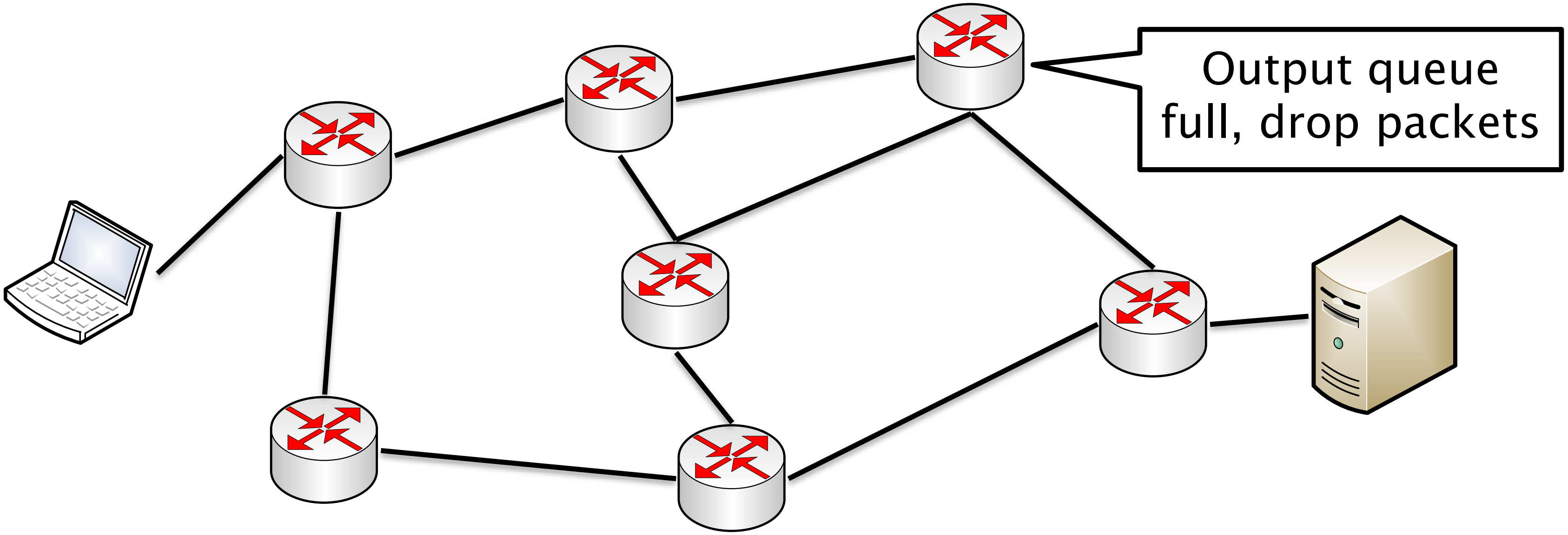


PPPoE:
1492 bytes

Ethernet:
1500 bytes

Network Congestion

- Handle overloaded network links
- Drop packet silently (other layer must detect packet loss and retransmit)
  - Or send *Explicit Congestion Notification* (ECN)



Output queue full, drop packets

Transport Layer

- Data transmission between processes
- Protocols: TCP, UDP, …
- Data unit: **segment** (TCP), **datagram** (UDP)
- Address type: port number (16 bit)
  - Examples: 80, 443, 51539
- Tasks:
  - Deliver reliable byte stream between processes (TCP)
  - Deliver individual messages with low latency (UDP)

Transmission Control Protocol (TCP)

- Connection-oriented
  - Connection establishment and termination
  - Delivers a continuous byte stream on top of packet-switched network
- Congestion control: determine transmission rate
- Detects and handles network errors
  - Lost, duplicate or out-of-order packets
  - Acknowledgement & retransmission, re-ordering
- Suitable for reliable transmissions and large data amounts

User Datagram Protocol (UDP)

- Connectionless
  - Best-effort attempt to deliver a datagram in one packet
  - Efficient due to low overhead/functionality
- Unreliable
  - Detects truncated/altered datagrams with checksum
  - No other error handling, no retransmission
- Suitable for low latency applications that handle errors themselves (e.g. VoIP, some online games)

Application Layer

- Application-specific data transmission and processing
  - Web: obtain document from web server via HTTP
  - Email: transfer email to another mailbox via SMTP
- Protocol, data unit, addressing and tasks different for each and every application
  - Web: client/server system with requests and responses
  - BitTorrent: peer-to-peer system with messages
  - VoIP: audio stream on top of messages

Network Protocol

- Challenge: **interoperability**
  - Different computer hardware and operating systems
  - Different software implementations and vendors
  - Different feature sets and extensions
- **Network protocol**: rules for interaction
  - Defines what to send when, and what it means
  - Syntax: message format (which bytes to send)
  - Semantics: meaning of messages and bytes
  - Clarifies the communication, but not application design choices (e.g. user interface, local file storage)

Internet Standardization

- Who publishes Internet standards?
- IEEE: communication technology
  - Ethernet, Wi-Fi, Bluetooth, …
- IETF: Internet protocols
  - IP, TCP, HTTP, DNS, FTP, SMTP, …
- W3C: World Wide Web standards
  - HTML, CSS, XML, SVG, …
- And various others, e.g. ECMA (JavaScript, JSON), ISO/IEC (JPEG, MP3), ITU (H.323)

Internet Engineering Task Force (IETF)

“We believe in: rough consensus and running code.”

– David D. Clark, 1992

- IETF is an open standardization organization
  - Run by volunteers (usually with jobs in industry)
  - Anyone can participate (no membership required)
- Coordination of protocol engineering
  - Working groups, mailing lists, international meetings
  - Public resources: https://www.ietf.org

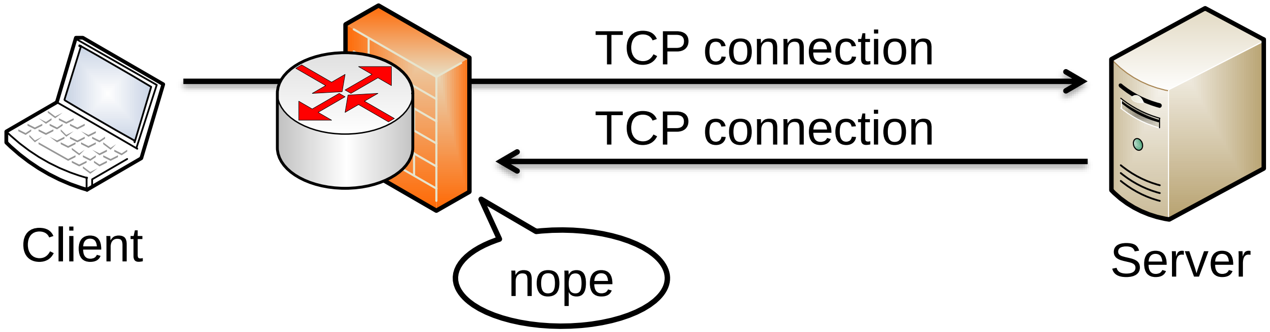
IETF Publications

- Request for Comments (RFC)
  - e.g. RFC 768: *User Datagram Protocol*
  - Various types of RFCs: (Proposed) Standard, Informational, Experimental, Obsolete
- Internet Standard (STD)
  - Well-known, mature and stable specification
  - e.g. STD 6: *User Datagram Protocol* (same as RFC 768)
  - Process: Internet Draft (I-D) ⇨ Proposed Standard (RFC) ⇨ Internet Standard (STD)
  - Very few protocols become STD (e.g. HTTP is not)

IETF Publications

- Best Current Practice (BCP)
  - Usually operational advice (how to run networks)
- IETF publications use specific terminology
  - e.g. MUST, MUST NOT, SHOULD, SHOULD NOT, MAY
- RFCs never change
  - Corrections published separately as Errata
  - New RFCs may update or obsolete old RFCs
- RFCs do not always reflect state of the art
  - New publications take time and effort

Model of Internet Communication



- Client host connected to the Internet
  - Can create outgoing connections
  - Local network does not allow incoming connections
  - Port forwarding necessary (manually, or via UPnP)
- Server host allows connections in and out

Further Challenges of Internet Applications

- Parallel activities
  - Autonomous components executing concurrent tasks
  - One server deals with >1 clients (do not block!)
- Communication via message passing
  - No shared memory, but network delays
- No knowledge of global state or global clock
  - Each client/server has their limited point of view
- No absolute trust in the other side
  - Assume buggy software or a malicious attacker