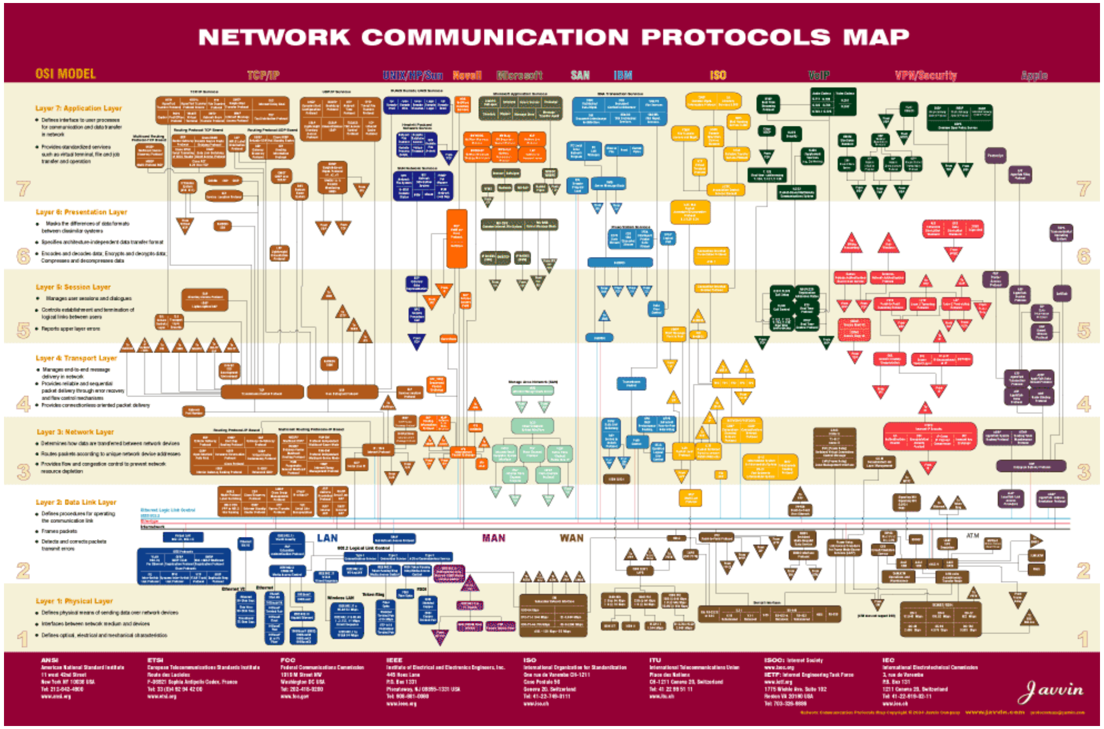
Computer Networks & Distributed Systems
Chapter 1: A Quick Tour through Networking



Prof. Dr. Max Mühlhäuser
(lecturer in most years, incl. SS 2019)

Prof. Dr.-Ing. Ralf Steinmetz
(lecturer, SS 2017)

Prof. Dr. Thorsten Strufe
(now TU Dresden → KIT,
responsible for last major revision)

Julien Gedeon
(deputy lecturer, SS 2019)

Computer Networks & Distributed Systems (CN&DS)

Leslie Lamport: „A distributed system is one in which the failure of a computer you didn't even know existed can render your own computer unusable."

What is a "distributed system" (DS)?

- **Preliminary (!) answer: A system where computers cooperate "closely"**
  - Cooperation yields **one system** (-of-systems) to a certain extent
  - This extent i.e. the service/functionality of the *one system* can be specified somehow
  - This one system would not exist without the network
- **In other words: no distributed system without computer network**
- **So, we look at systems where:**
  - There are several computers (of kinds)
  - The computers are connected by a **computer network** (**CN**)
  - …most interesting problems stem from the network & the distributed nature
- **There are various kinds of such systems – where does CN end and DS begin?**
  - Internet, IPTV, P2P, VoIP telephony, …

Motivation

- **Focus of this course:**

  Basic building blocks for networked applications

- **Networking is a fast-moving area**
  - Basic, fundamental knowledge becomes important

- **Three parts in the course**

  1. Fundamental theoretical concepts
     - Foundations of networking
  2. Networking
     - How the theory is applied in practice
  3. Some application examples
     - So that the course isn't too boring… ;-)

Outline: Subjects

- **Fundamental theory:**
  - Graph theory: helps to understand → improve on network-as-a-whole problems?
  - Queuing theory: helps to understand → improve on more detailed problems (model data packets as they travel through the networks)

- **Networking:**
  - Internetworking: How to connect physical networks and address devices
  - Routing (extension to GT): How do we find the way for messages to get from A to B?
  - Multicast: How can we send a message to a group of recipients?
  - Transport: How do we build reliable connections?

- **Distributed Systems and Applications:**
  - Web protocols (HTTP)
  - TLS
  - Name resolution (DNS)
  - …

Outline: Lecture Sequence

- **Networking:**
  - Routing:
    How messages get from A to B
  - Internetworking:
    How to connect physical networks
  - Transport:
    How do we build reliable connections?
- **Fundamental theory:**
  - Queuing theory:
    How networks provide service
- **Multicast**
  - How can we contact a group of recipients?
- **Application-oriented examples:**
  - Web protocols (HTTP)
  - TLS
  - Name resolution (DNS)
  - …

- **Chapter 2: Routing**
- **Chapter 3: Internetworking**
- **Chapter 4: Transport**
- **Chapter 5: Queueing Theory**
- **Chapter 6: Multicast**
- **Chapter 7: Applications/Distributed Systems**

About Networking

Still quite young and fast evolving discipline

- **Computer languages**
  - Well defined analytical background
  - Well structured research area
  - Gone through several phases of maturity

- **CN and DS**
  - Plethora of modeling methods
  - Somewhat disorganized research area
  - Still in infancy?

Problem as stated before: Constant flux

- Immense speed of development
  - Core Networks: From T1 (1.544Mbps) to 100Gbps Switches and beyond
  - Consumer: 1.2kbps modems to ADSL2+…: 50 → n*100 Mbps
- Changing infrastructure, devices, services, communication methods
  - Workstations/Servers → PCs → Cell Phones → Wireless Sensors
  - Unix → BSD/Linux → Win (everybody!...) → MacOS/Android/TinyOS/…

But Let‘s Do a Quick Tour first…

- **You know some of the following from other courses, but …**

…. anyways, it will

- **Provide a brief, guided tour to communication networks**

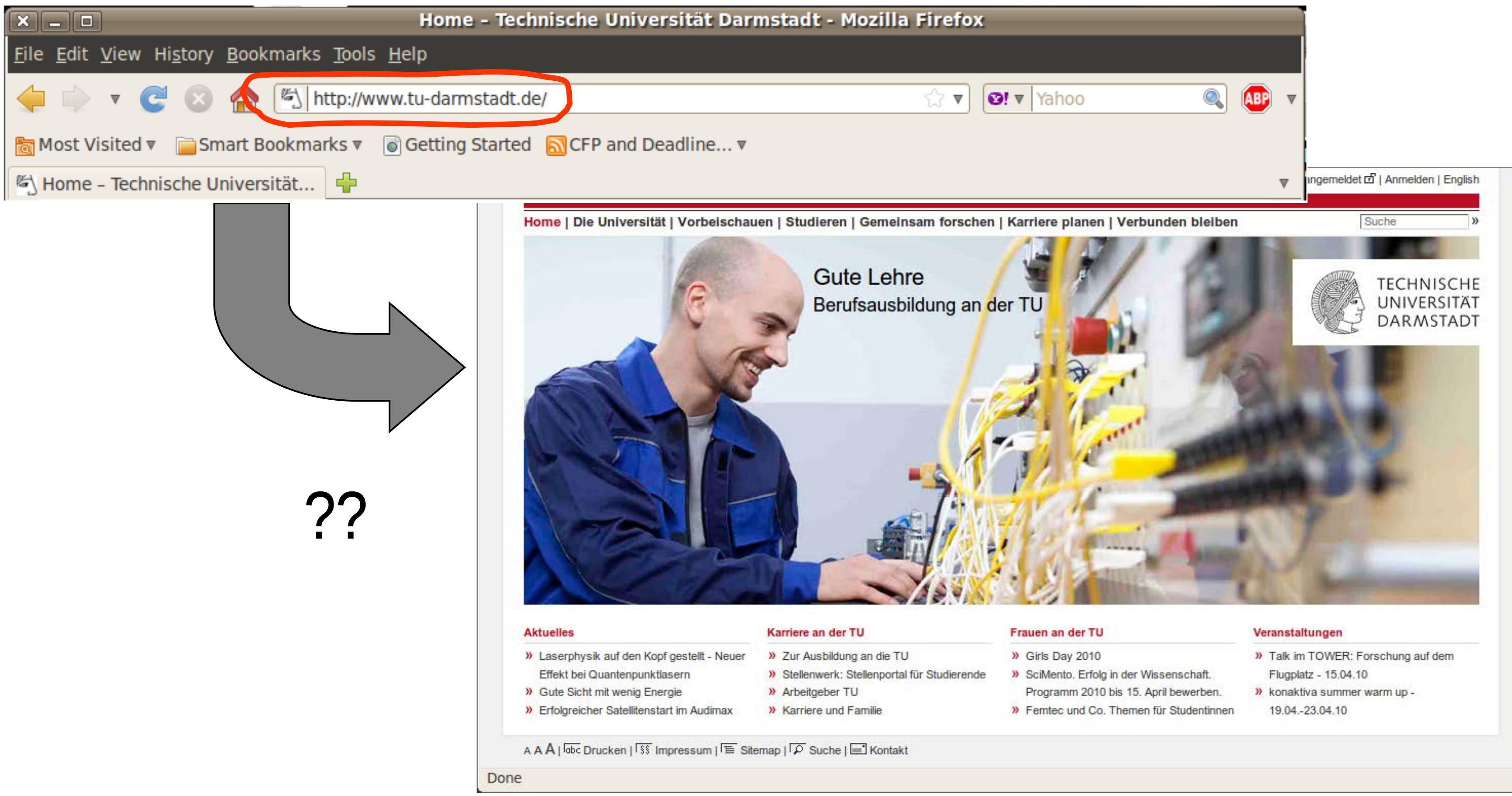
- **Starting from the simple case of two directly connected devices**
- **Generalize to larger networks**

- **Goal is to provide a rough understanding on**
  - *Which* typical problems exist,
  - *Why* they occur,
  - *How* solutions could look like.

- **Details on solutions will be treated in the remainder of CNuvS**

Basic Example 1: World Wide Web

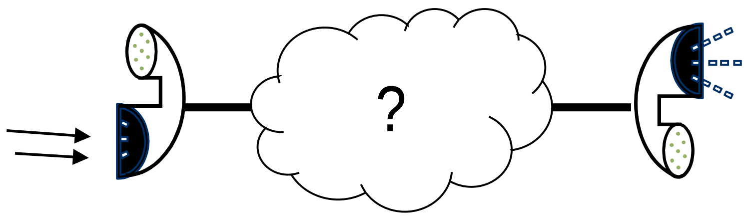
- **What happens when you enter http://www.tu-darmstadt.de into a Web browser?**



??

Basic Example 2: Telephony

- **What happens when picking up a telephone and making a phone call?**
  - How to find the peer's phone? How to transmit speech?

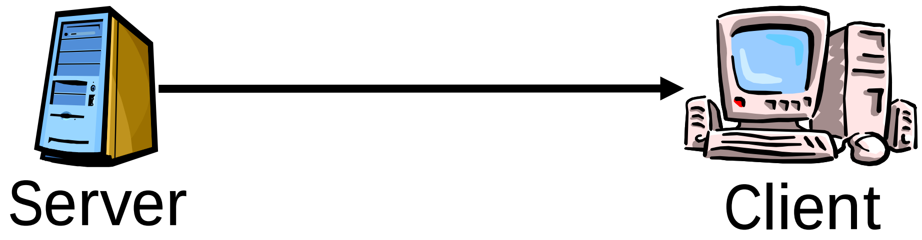


- **What are the crucial differences between transferring a Web page and a phone call?**
  - Web: Bunch of data that has to be transmitted
  - Phone: *Continuous* flow of information, must arrive *in time*

- ***By the way: what actually is "data" & "information"?***

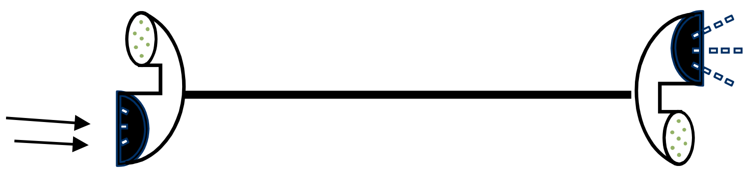
Simplest Communication: Direct Physical Connection

- **Web example: Browser=client and server**
  - Simplest case: directly connect them by a (pair of) cable



  - Server provides data, client consumes it

- **Telephony: Connect two telephones via a (pair of) cable**

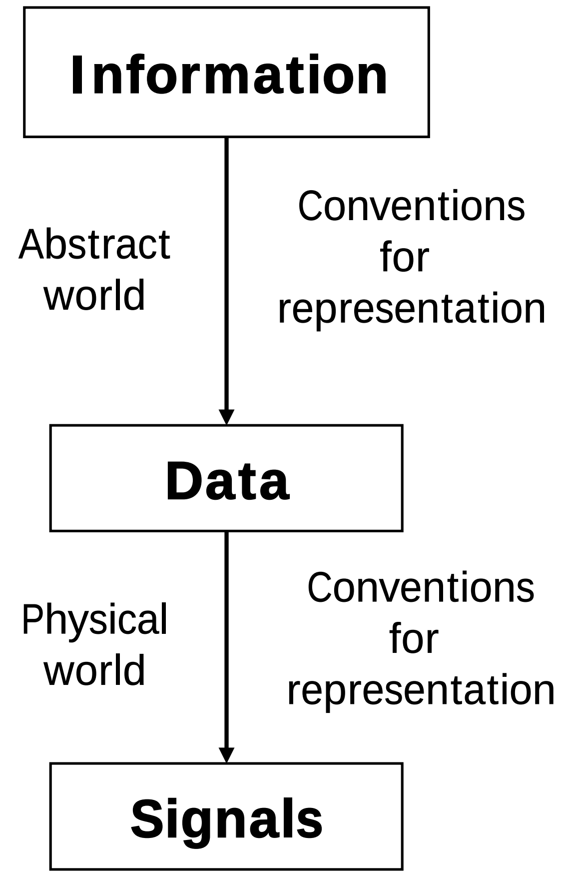


What Good is a Physical Connection? – Signals

- ***Information*** **has to be represented as** ***Data*** **has to be represented by** ***signals***

- ***Signal:*** **Representation of data by characteristic changes (in time or space) of physical variables**

- **Material example: Letters on paper**
- **Immaterial example:**
  - Acoustic waves when speaking
  - *Current or voltage in a wire*



Bits and Signals

- **What should be communicated: Data, represented as bits**
- **What can be communicated between remote entities: Signals**
- **Needed: a means to transform *bits* into *signals***
  - And: from signals back into bits at the receiver

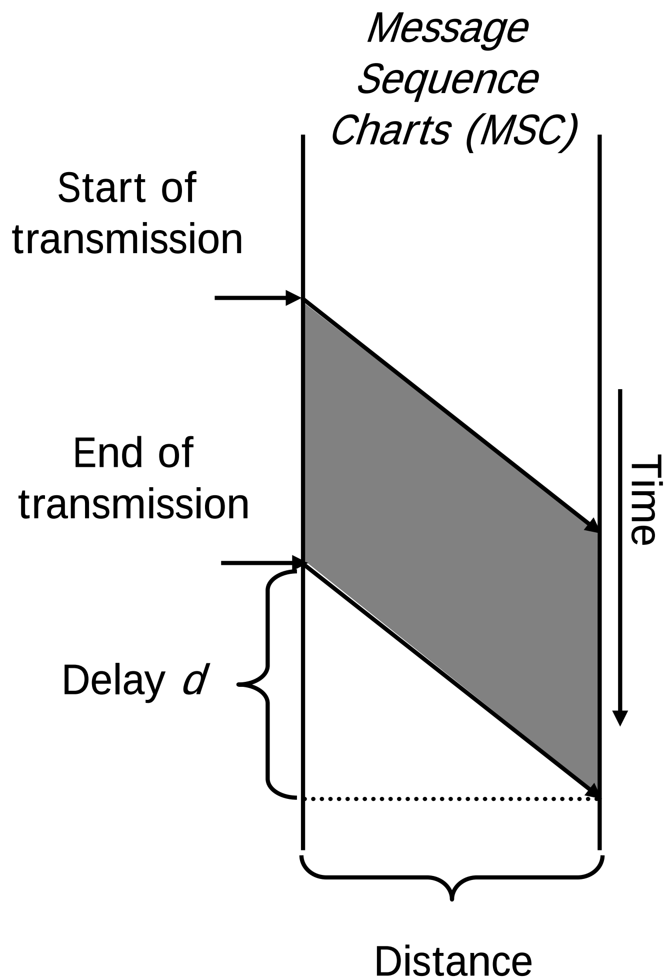
- **A (too) simple convention for a copper wire:**
  - “1” is represented by current
  - “0” is represented by no current
  - (Not practical, more sophisticated conversions necessary)

- **Questions: How to detect bits, decide on their length, handle errors?**

Low-Level Properties of Communication

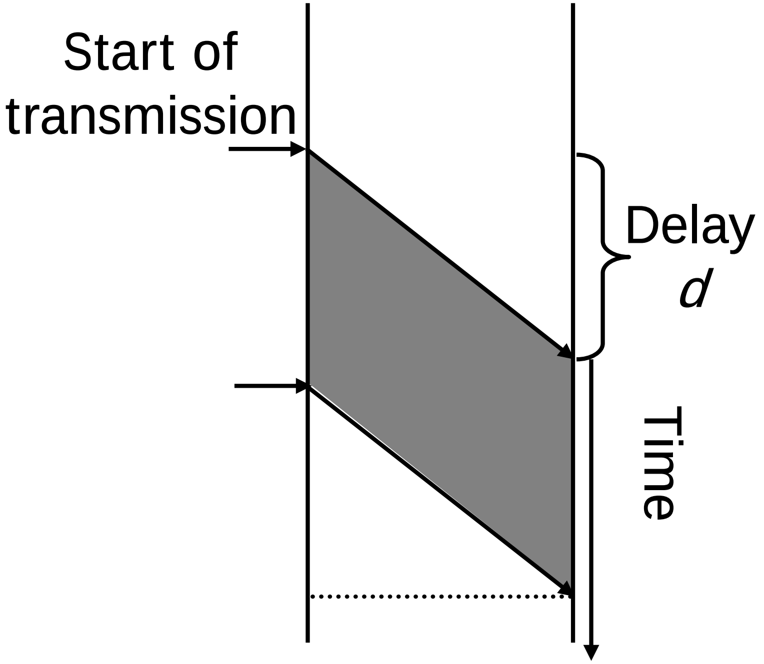
- **Some low-level properties:**
  - ***Propagation delay*** *d*: How long does it take for a signal to reach receiver?
    - Propagation speed *v*: How fast does a signal travel in the medium?
    - Electromagnetic waves in vacuum: speed of light $v=c$ (in copper $v= 2/3$ c)
    - d = distance / v
  - ***Data rate*** *r*: With which data rate (in bits/second) can a sender transmit?
    - (End of Trans. – Start of Trans.)

      = Data size / data rate
  - ***Error rate***: What is the rate of incorrect bits arriving at the receiver?
    - Also: what error *patterns*?



How Can a Wire Store Data

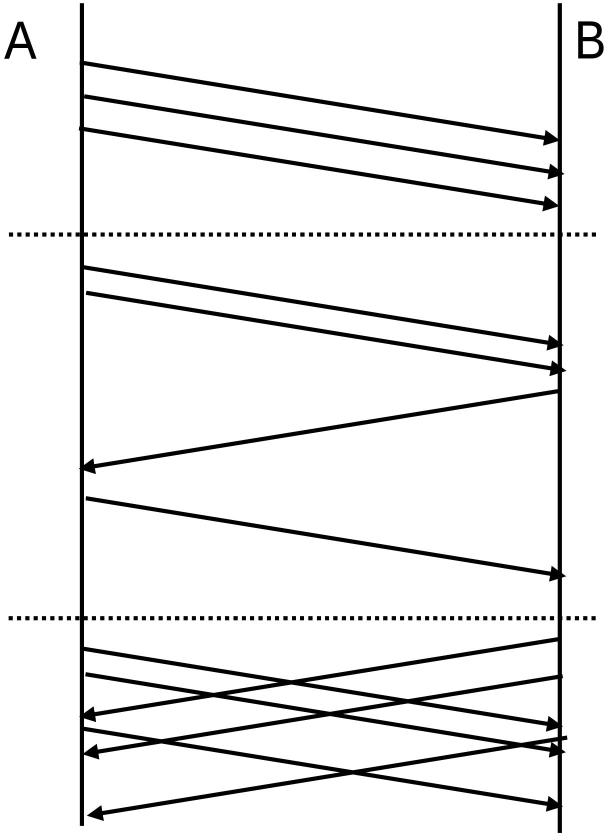
- **What happens in the first *d* seconds after transmission starts?**
  - Bits are transmitted and propagated towards the receiver
  - Sender keeps sending bits
  - First bit arrives after *d* seconds
  - In this time, sender has transmitted *d*r* bits
  - Where are they? Stored in the wire!



Two Physical Connections for Two-Way Communication?

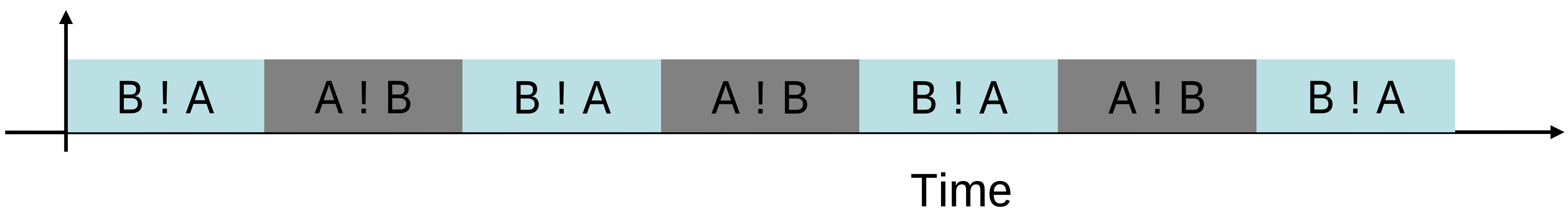
- **Two-way communication**
  - Telephony: Both parties want to say something
  - WWW: Server needs to know which webpage shall be delivered

- **Different cases possible:**
  - *Simplex:* Only one party transmits
    - Example: Radio broadcast (many recipients!)
  - *Half duplex:* Parties *alternate* as senders/receivers
    - Example: Conversation
  - *Full duplex:* Both parties (may) send all the time



How to Realize Duplexing

- **Simplex operation: trivial**
- **Half duplex**
  - Two pairs of cables, one for each direction – wasteful
  - Use one cable intelligently – participants alternatively transmit, wait their time until it is their turn
    - Both sending at the same time would not work, signals *interfere*
    - Problem: How can one node decide that the other is done sending?

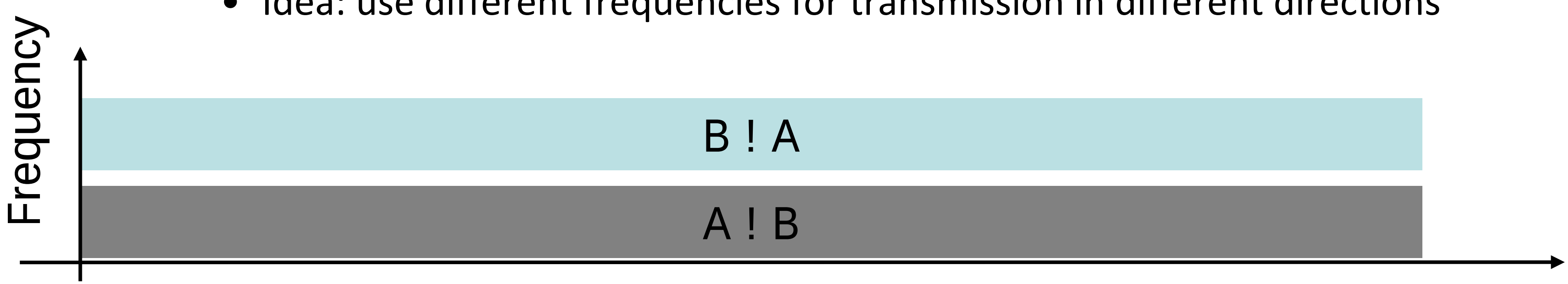


  - ***Time division duplex – TDD:*** time slots are predefined, fixed, synchronized
  - ***Alternative: on-demand –*** each sender pre-announces length of next bit sequence or sends characteristic (set of) symbols when done

    (problems: max. length defined? symbol-set must not be part of “payload”, bit-synchronization – for receiver – needed for each bit sequence, etc.; these problems must be solved for full-duplex, too – see below)

How to Realize Duplexing

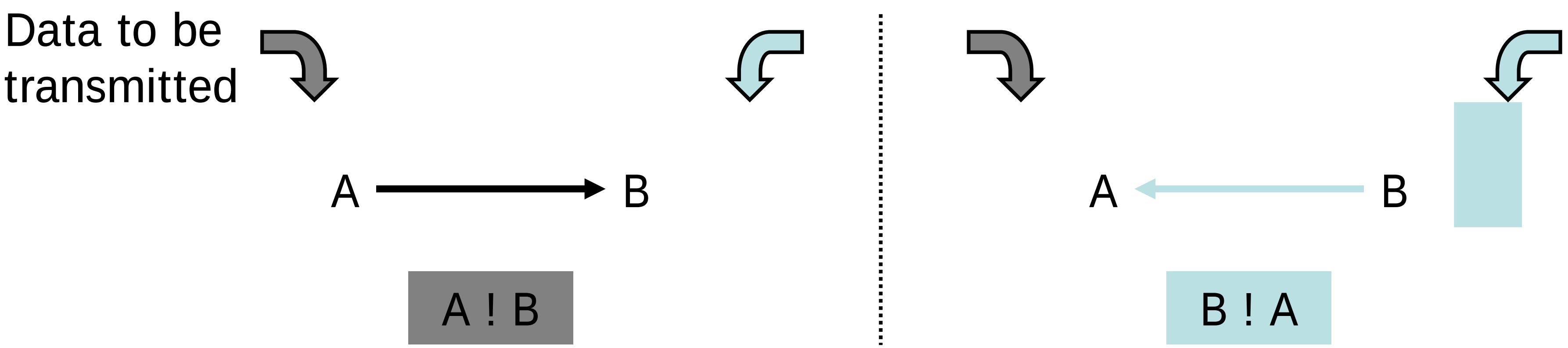
- **Full duplex --- distinguish wired / wireless comm.**
  - Two cables (copper: pairs of cables for current-loop) OK, but overhead (cost, size, maintenance)
  - Does it work with *one* cable (pair of cables), too?
    - Yes (exploit some properties of physical medium), but *short distance* only
  - Wireless transmission (and CableTV, …):
    - transmissions in different frequencies do not interfere
    - Idea: use different frequencies for transmission in different directions



- ***Frequency Division Duplex – FDD***

How to Realize Duplexing

- **Full duplex by time division duplex TDD?**
  - Sounds like a contradiction:
    - both A and B always have data to send, but have to take turns?
  - "Having data to send" corresponds to a certain *data rate* – bits per second
  - How about intermediately storing data when the other station is currently sending? Then send all stored & new data ***faster* than user data rate**!

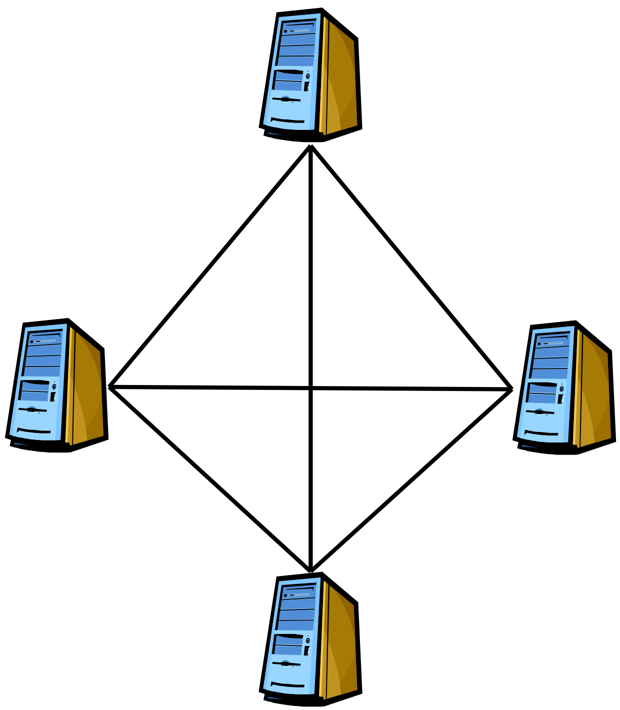


- TDD can realize full duplex if transmission over medium that is *at least twice as fast* as (unidirectional) data rate to be transmitted
- Not for analog signals (e.g., early telephony) – **why?**

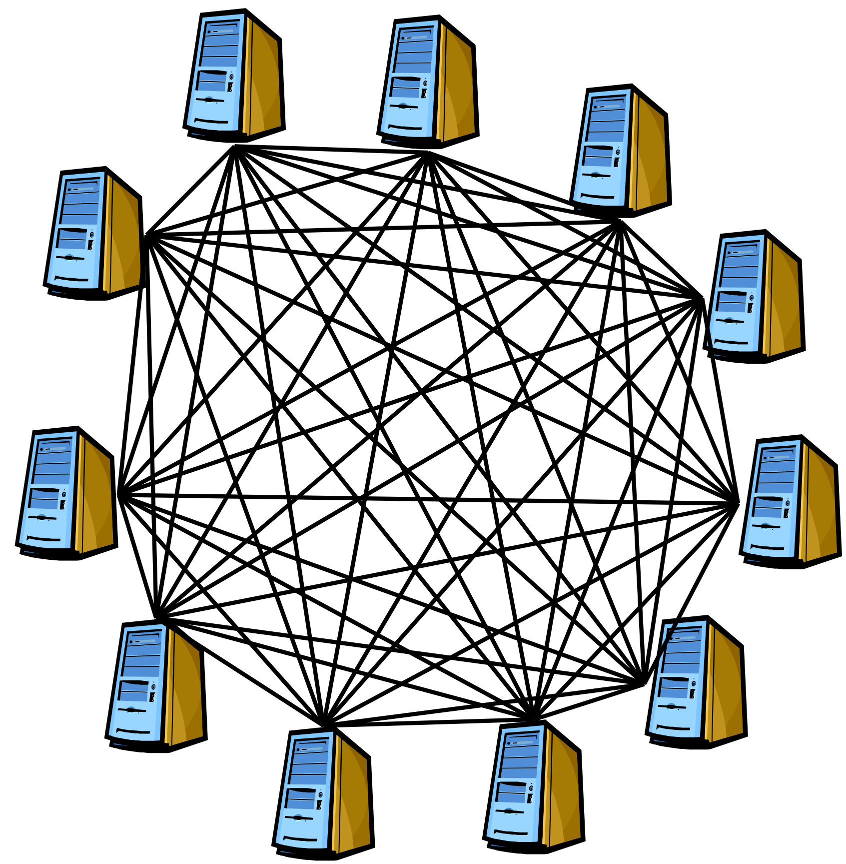
But There are More than Two Computers / Telephones

With four computers:

- **Connect each telephone / computer with each other one?**



With eleven computers:

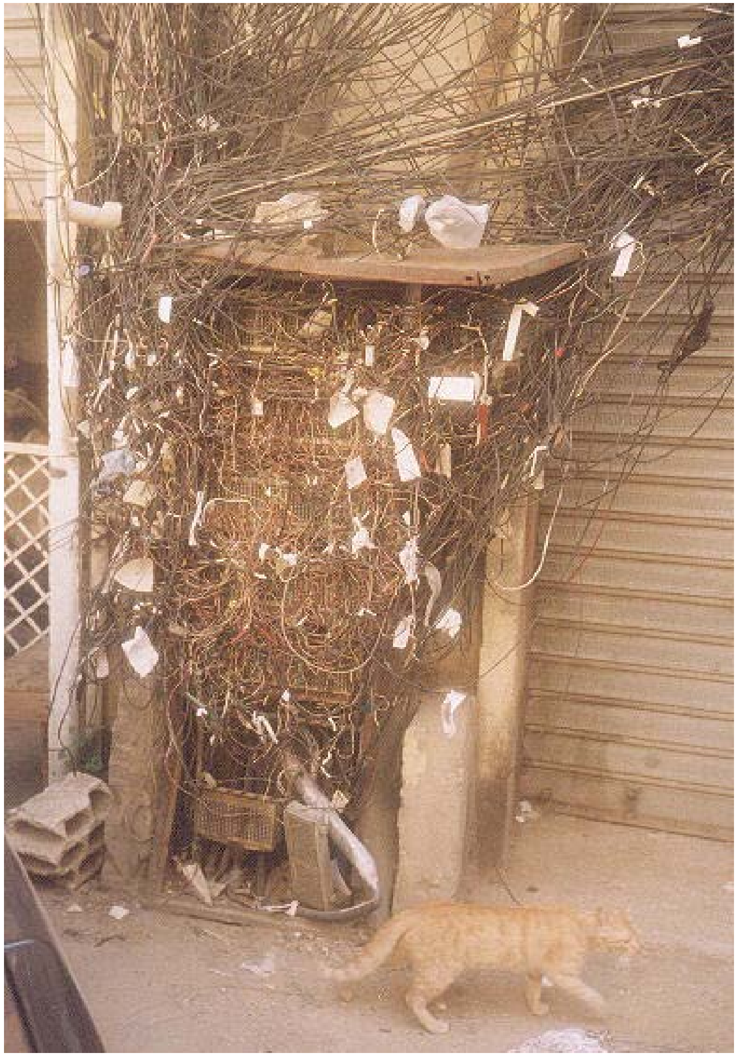


...

Beirut Connections

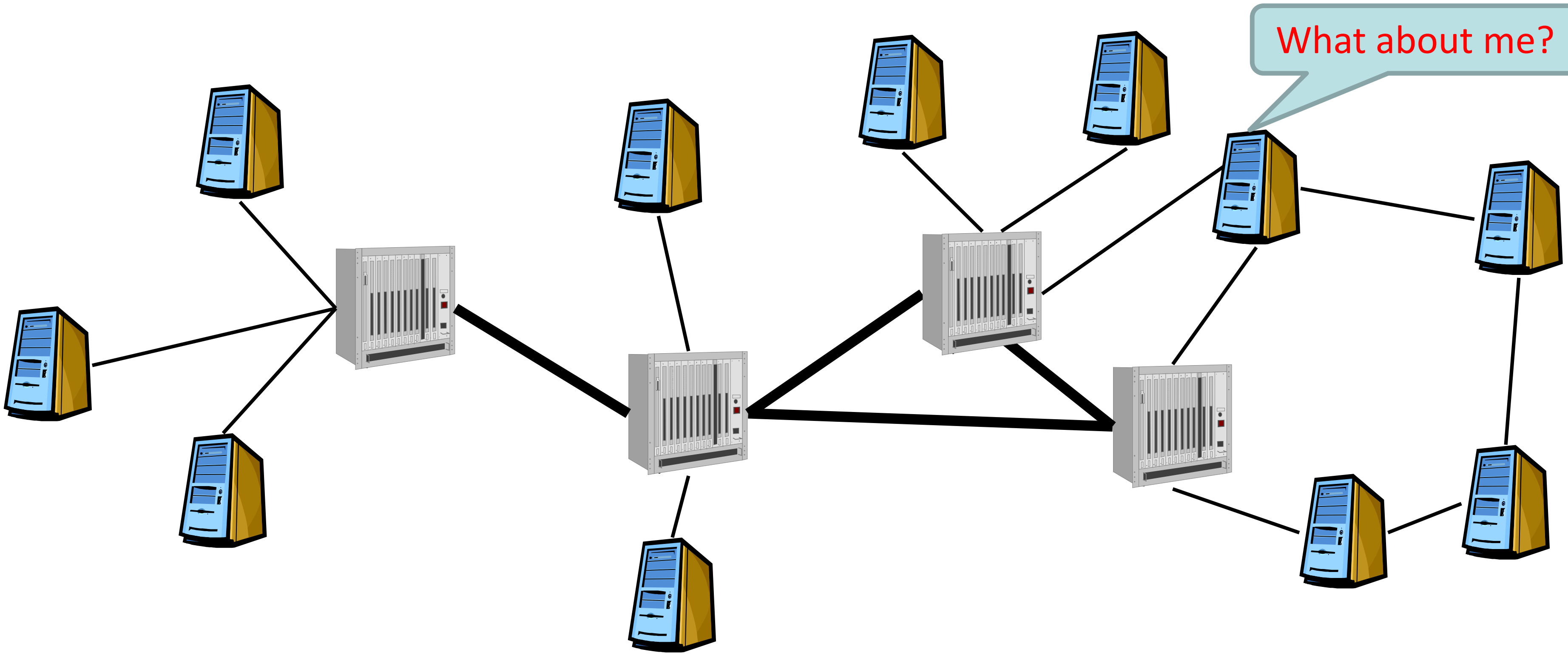
- **Connecting many phones in real life**

Beirut Connections



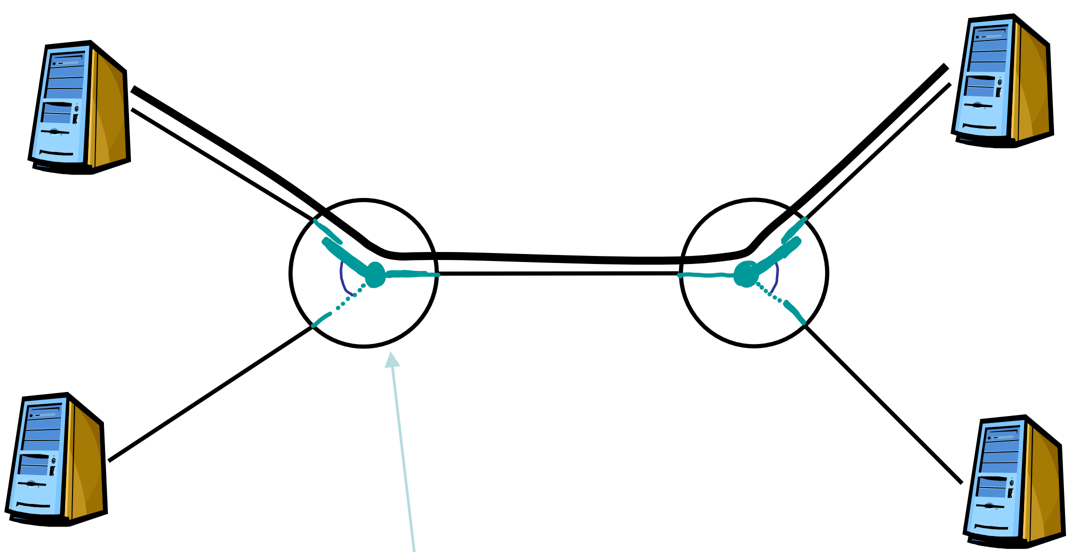
Put some Structure into a Network

- **Pair wise connection of all entities does not scale**
- **Need some structure**
  - Distinguish between "*end systems/terminals/user devices*" on one hand, and "*switching elements/routers*" on the other hand



How to Communicate Over a Switching Element

- **Using switching elements, there is no longer a direct physical connection between two terminals – How to send signals nonetheless?**
- **Option 1: Have the switching element dynamically (on demand) configure an electrical circuit between terminals**
  - Act as a real switch – "Fräulein vom Amt" → "Branch eXchange" (telephony jargon)
  - Resulting circuit lasts for duration of communication
  - ***Circuit switching*** *(Leitungsvermittlung…)*



http://www.wdrcobg.com/switchboard.html

manual ~1880 → electromechanical (pulse) ~1910 → electric -→ digital (switch) → fully digital

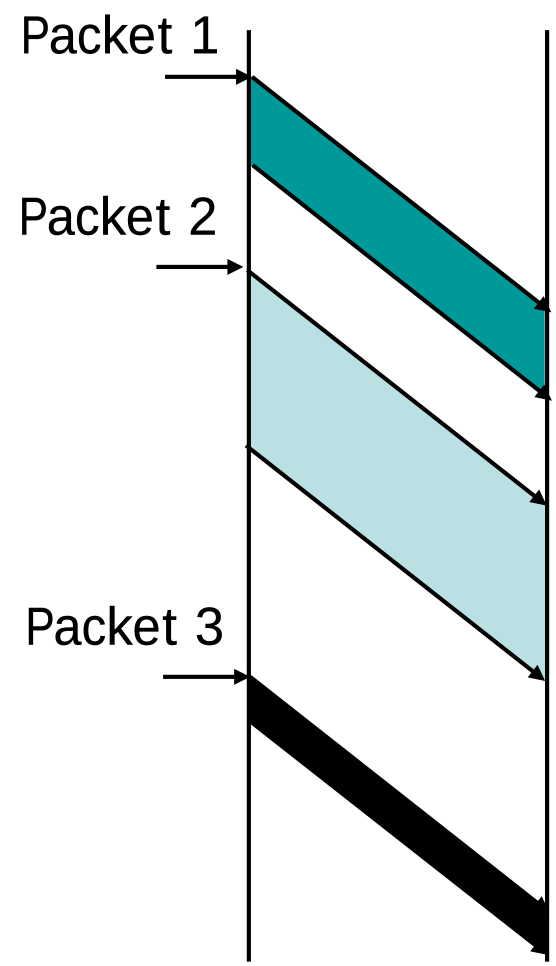
Circuit Switching – Evaluation

- **Advantages of circuit switching**
  - Simple
  - Once circuit is established, resources are guaranteed to participating terminals
  - Once circuit is established, data only has to follow the circuit (no queuing delay!!)
- **Disadvantages**
  - Resources are dedicated – what if there is a pause in the communication?
  - Circuit has to be set up before communication can commence

- **Alternatives?**

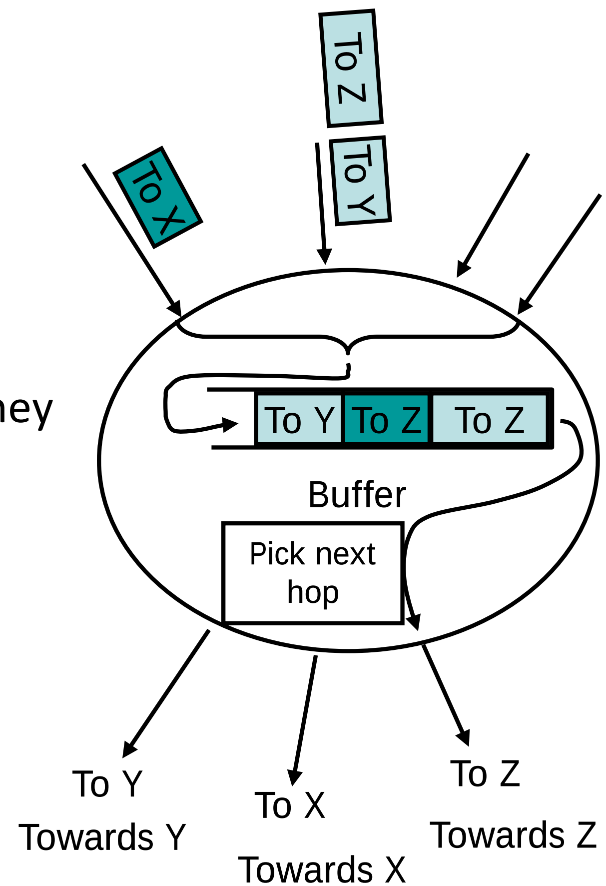
Packet Switching

- **Avoid setting up a circuit for a complete communication**
- **Instead: chop up data into *packets***
  – Packets contain some actual data
  that is to be delivered to the recipient
  (can have different size)
  – Also need administrative information
  e.g.: recipient “address”
  – Sender “occasionally” sends out a packet
  instead of continuous flow of data
- **Problems (selection):**
  – How to detect start and end of a packet
  – which information to put in a packet
  – how to deal with limited resources, congestion, …
- **Attention: typically implemented on a higher layer**
  – Abstraction level again bits, not symbols
  – Functioning transmission of sets-of symbols assumed
  (could even be based on circuit switching per-hop)



Packet Switching

- **Switches take on additional tasks**
  - Receive a complete packet
  - ***Store*** the packet in a buffer
  - Find out the packet's destination
  - Decide where the packet should be sent next (to reach its destination)
    - requires info. about the "network graph"
  - ***Forward*** the packet to this next hop of its journey

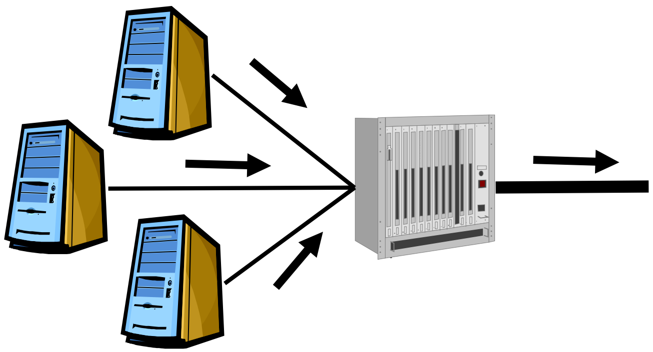


- **Also called "*store-and-forward*" switching**

Packets also need addresses for choice of port to destination!

Multiplexing

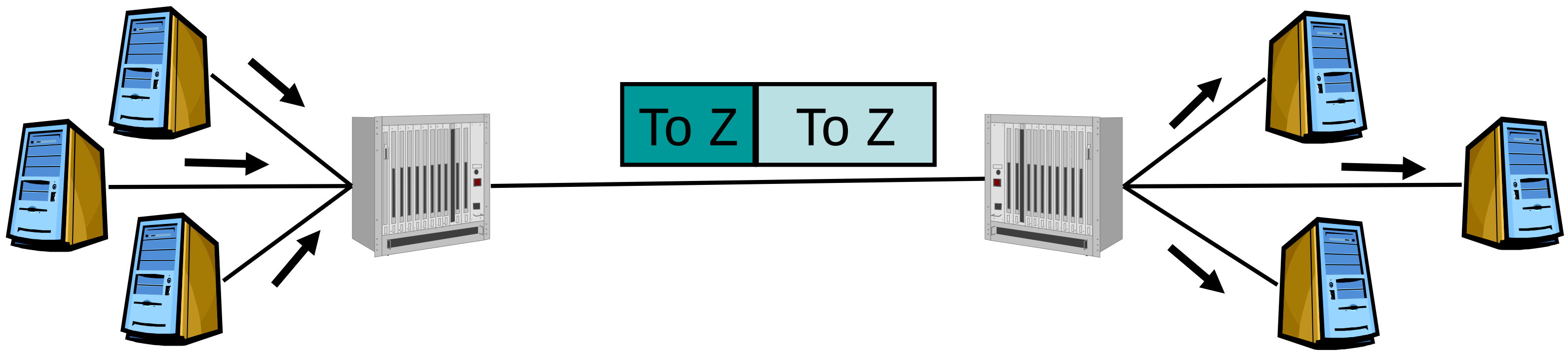
- **Previous example two packets for receiver "z"**
- *But*: **only one link to "z", different connections to same receiver**



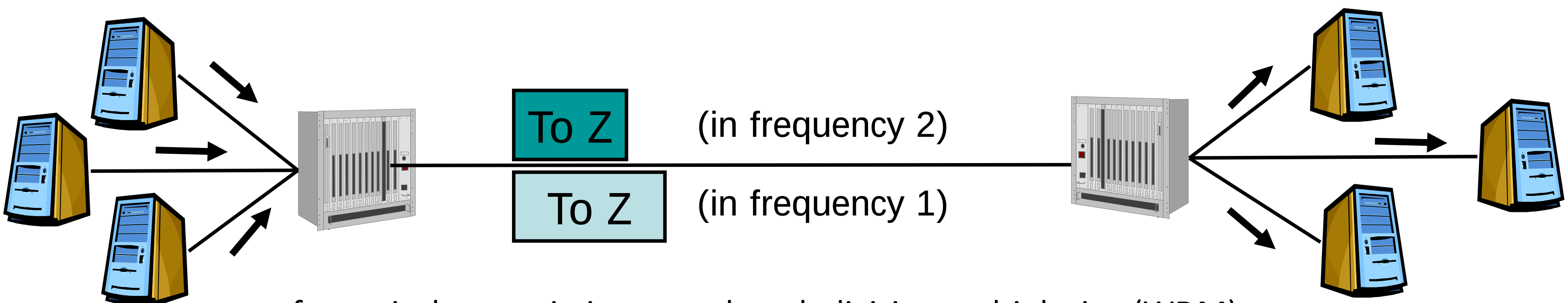
  – Organizing the forwarding of packets over such a single, shared connection is called ***multiplexing***

Multiplexing

- **Obvious option:** *Time Division Multiplexing (TDM)*
  - Serve one packet after the other; divide the use of the connection in time



- **Alternative:** *Frequency Division Multiplexing (FDM)*
  - Use different frequencies to transmit several packets at the same time



for optical transmission: wavelength division multiplexing (WDM)

Multiplexing

- **Other alternatives exist (mainly for wireless transmission)**
  - Code Division Multiplexing (CDM)
  - (Space Division Multiplexing SDM: cellular systems, handover) …

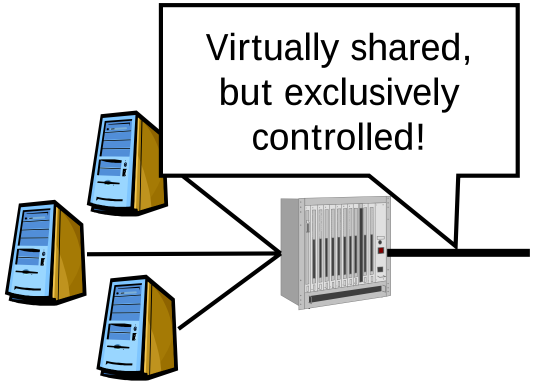
- **Interpreting multiplexing in the sense of a computer scientists' "most important occupation": *abstraction!***
  - Hides the fact that the ***physical connection*** has to be ***shared***
  - Allows entities connected to the switch to "***imagine***" they were alone (i.e. using the physical connection ***exclusively***)
  - Multiplexing virtualizes the actual, physical connection

- **However, an inverse function, the** *demultiplexer,* **is needed to restore the original flows**

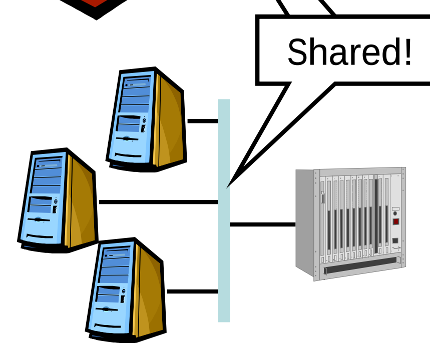
- **Note 1:** multiplexing is also possible & relevant on "higher layers", whenever many connections shall be enabled over a single "lower" connection
- **Note 2:** more precisely, the above is called "upward multiplexing"!
  "Downward multiplexing" means bundling several "*lower" (e.g., physical)* connections in support of a single (e.g., user-level) connection

Multiplexing & Shared Resources

- **(Upward) Multiplexing →**
  - regulate access to a resource shared by multiple users



- **Are there other examples of “shared resources”?**
  - Classroom, with “air” as physical medium
  - A shared copper wire, as opposed to direct connection
- **Characteristic here: a *broadcast* medium!**
  - air / electrons / … spread in every direction
  - for a single wire, this means: in *both* directions



Broadcast Medium and Multiple Access

- **Common characteristic of a broadcast medium: Only a single sender at a time!**
  - ***Exclusive access*** is necessary
  - (well, CDM on wireless is an exemption)
- **Exclusive access is simple to achieve with a multiplexer**
  - What if no multiplexer is available?
  - Exclusive access has to be ensured by all participants cooperating
- **The problem of** ***multiple access to a shared medium***
  - ***Medium access*** *(control)* for short ---- MAC (note: MAC address)
- **Rules have to be agreed upon**
  - Classroom approach: only speak when asked to (by *central instance*)
  - Party approach: try to speak only after short silence; may lead to collision → back up and try again

An Intermediate Summary

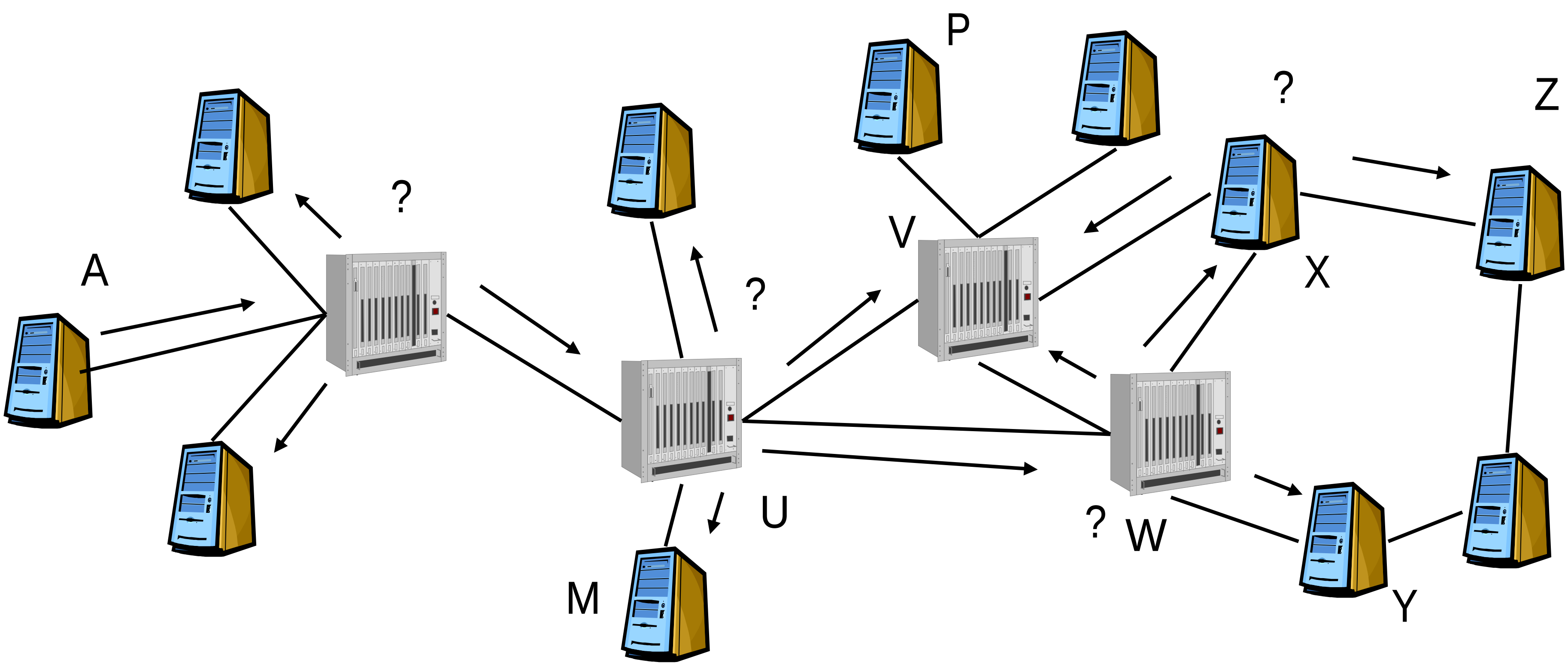
- **So far, we have a rough idea about**
  - Converting bits to signals and back again
  - Duplexing, switching, multiplexing
  - Packets as units of data transport
  - Multiple access of several entities to a shared medium

- **We know how to connect several entities into a flat or hierarchical network**

- **Missing piece for hierarchical networks: How to know *where* to send a packet**
  - For circuit switching: how to setup the circuit

Forwarding and Next Hop Selection

- **Recall: A switching element/a router *forwards* a packet onto the next hop towards its destination**
- **How does a router *know* the best possible neighbor on the path towards a given destination?**



Options for Next Hop Selection

- **Some simple options:**
  - ***Flooding*** – send to *all* neighbors
  - ***Hot potato routing*** – send to a *randomly chosen* neighbor
- ***Is this sensible? For fast-changing topologies, hardly an alternative!* otherwise, rather:**
  - Try to find good, i.e., short routes (few hops, fast lines, short queues, …)
  - Basis mostly: try to learn about the structure of the network, interpreted as a graph
- **Construct** *routing tables*
  - For each "switching element" separately
  - Separate entry for each destination
  - Contains information about the (conjectured) **shortest distance** to a given **destination** via **each neighbor**
  - Choose neighbor w/ shortest distance to destination

Routing table of W

Destination

Neighbor	M	P	Z
U	**2**	3	4
V	3	**2**	3
X	4	3	**2**
Y	4	4	3

Routing Tables

- **Good (perfect?) estimate of real distances, freedom of loops, …**

- **Constructing routing tables**
  - Initially, typically empty – how should a new node know anything?
  - Passive: observe ongoing traffic (e.g., from hot potato routing) and try to extract information, successively improve table correctness
  - Actively exchange information between routers to try to learn network structure – ***routing protocols***

- **Problem: Size!**
  - In large networks, maintaining routing entries for *all* possible destinations quickly becomes infeasible
  - Solution: hierarchy – treat "similar" nodes identically (divide et impera) ! internetworking

So, looking at all these Problems…

- **Communication networks**
  - have to solve many problems and
  - need a lot of functionality
- **We touched just some basic problems (and an idea about their solution)**

- **Solutions will be topic of the remainder of this course!**

- **Some first faint clues:**
  - Long-lasting knowledge (“invariants”) is important for networks → for learners
  - And especially: ***abstraction*** seems to be a good idea
    (the problems stay complex enough…)

Most Important Invariant: Abstraction

- **Programming Languages:**
  - Machine code → Assembler → Low level languages → High level languages
  - OO Design, Modularization, …
- **Simplification by abstraction**
- **Same for networks: Network Layer Models**
  - ISO's OSI model (Open Systems Interconnection)
  - Internet layer model (basically, a subset of OSI model)
- **Now we can understand "Distributed Systems" (DS)!**

1. "Distributed Machine":
   set of cooperating "autonomous systems" AS
   plus "communication subsystem" CSS
2. Essential: abstracts from *some* of the problems (depending on type of DS)!
   i.e. makes them "transparent" → take burden from programmer/user ..
   - to care about: locations *or* (node/link) failures *or* heterogeneity *or* other…

– Basis: rich functionality of computer networks (CN)
  → DS "sits on top of" CN

World Model

- **ISO Basic Reference Model:**

- **Open Systems Interconnection OSI**

  – Part I: Concepts and Terms
  – Part II: 7-Layer-Model (→ 5-Layer-Model for Internet)
  – Part III: Individual protocols & services
    - Not covered in this course due to lack of time (but ***/really/*** interesting!!)

- **Part I:**

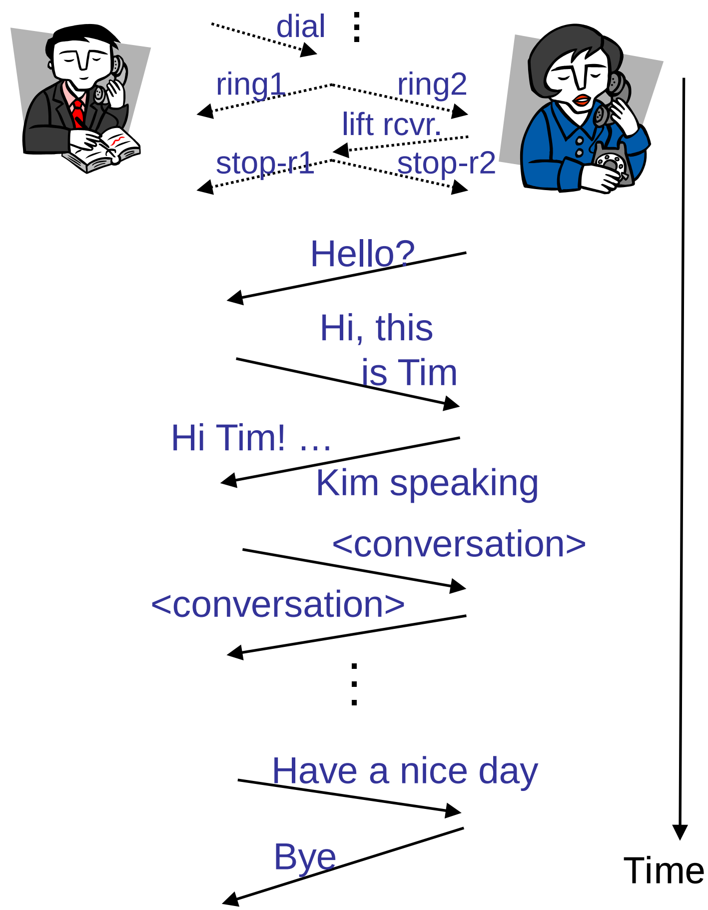
1. Protocol, Layer, Service
2. Service Primitive, Service Function, (Un-)Confirmed
3. Connections, Access Points
4. Data Units

Protocol

- **What is a protocol?**
  - What does a protocol do?
  - Example of a human protocol:
  - Analogies between computer protocols and human protocols

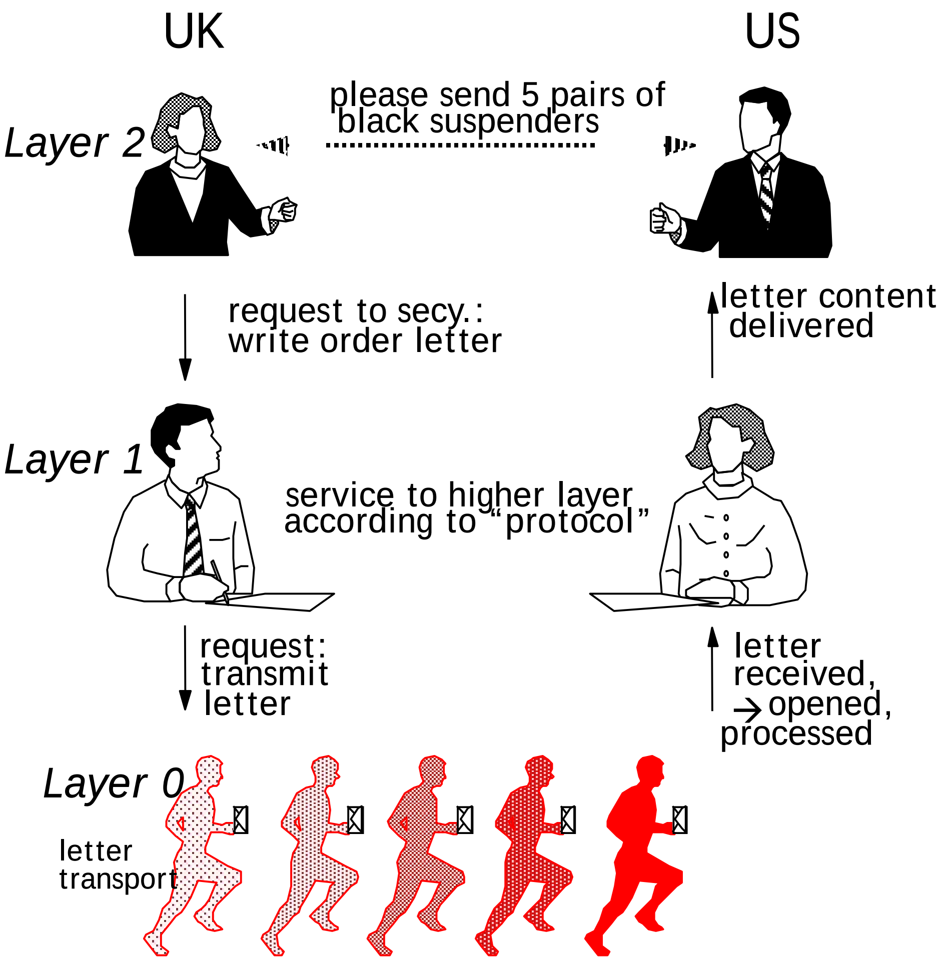
- **Definition of protocol:**
  - A **protocol** defines the ***format*** and the *order* of *messages exchanged* between two or more communicating entities, as well as the ***actions taken*** on transmission and/or reception of a message or other event

  → message formats & action rules



Above: example of a **message sequence chart - Timeline & N locations & flow of messages**

Layered Architecture

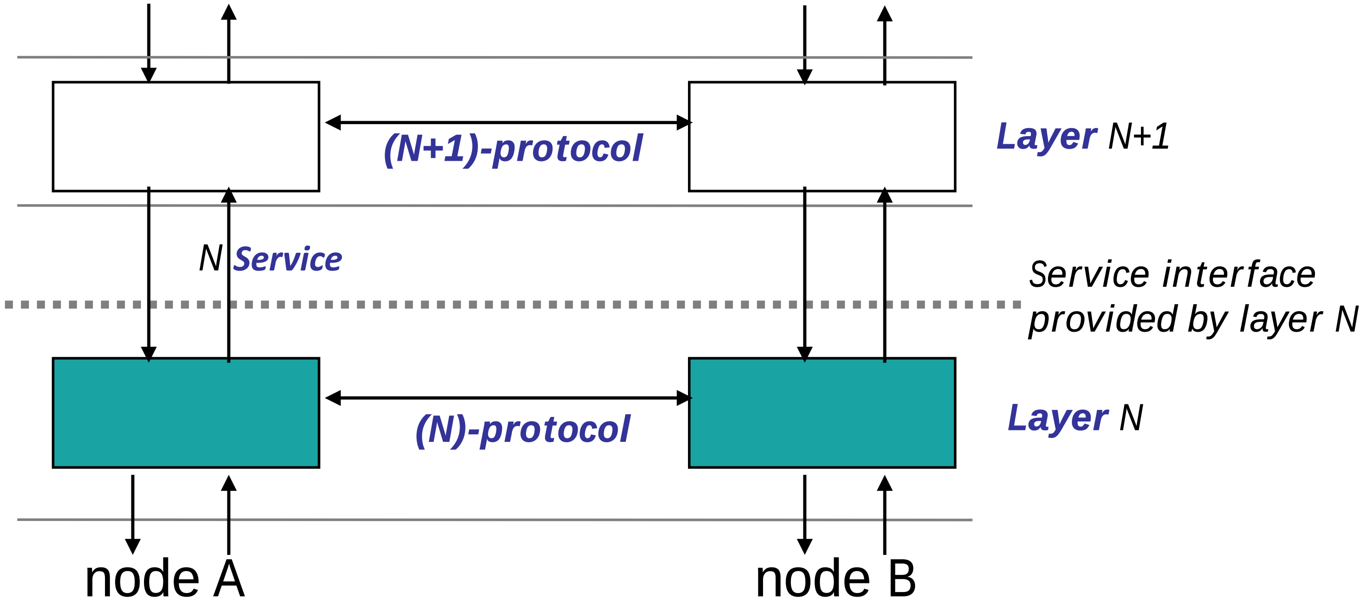


Based on lecture slides of Prof. Dr. Friedemann Mattern:
http://www.vs.inf.ethz.ch/edu/WS9900/VS/VernetSys3.pdf

- **Use protocol to ensure peer entities interoperate correctly**
- **Drawback: Overhead!**
- **Advantage: Layer content („service“) may be exchanged at any layer independently**
- **Tradeoff: overhead (SW/msg) vs. exchangeability, clarity, simplicity**

Layers in General

- **Layer-N provides (N)-service to layer-(N+1)**
  - Actual implementation is hidden in layer N (can change it as desired!)
- **(N)-entities communicate according to (N)-protocol**
  - Make use of (N -1)-service provided by underlying layer
- **Exception: lowest layer has to perform all actions itself**
  - In computer networks, lowest layer handles physical transmission



Service Function, Connection

- **Many services grouped into 3 phases**
  - Phase = essential service functions:
  - Connection Establishment CON
  - Data Exchange DAT
  - Connection Release (Disconnect) DIS
- **Such services are called connection-oriented services**

- **Other services are called connectionless services**
  - Connectionless services have only Data Exchange phase
  - No connection establishment or release

- **See below for more details on connection-oriented vs. connectionless**

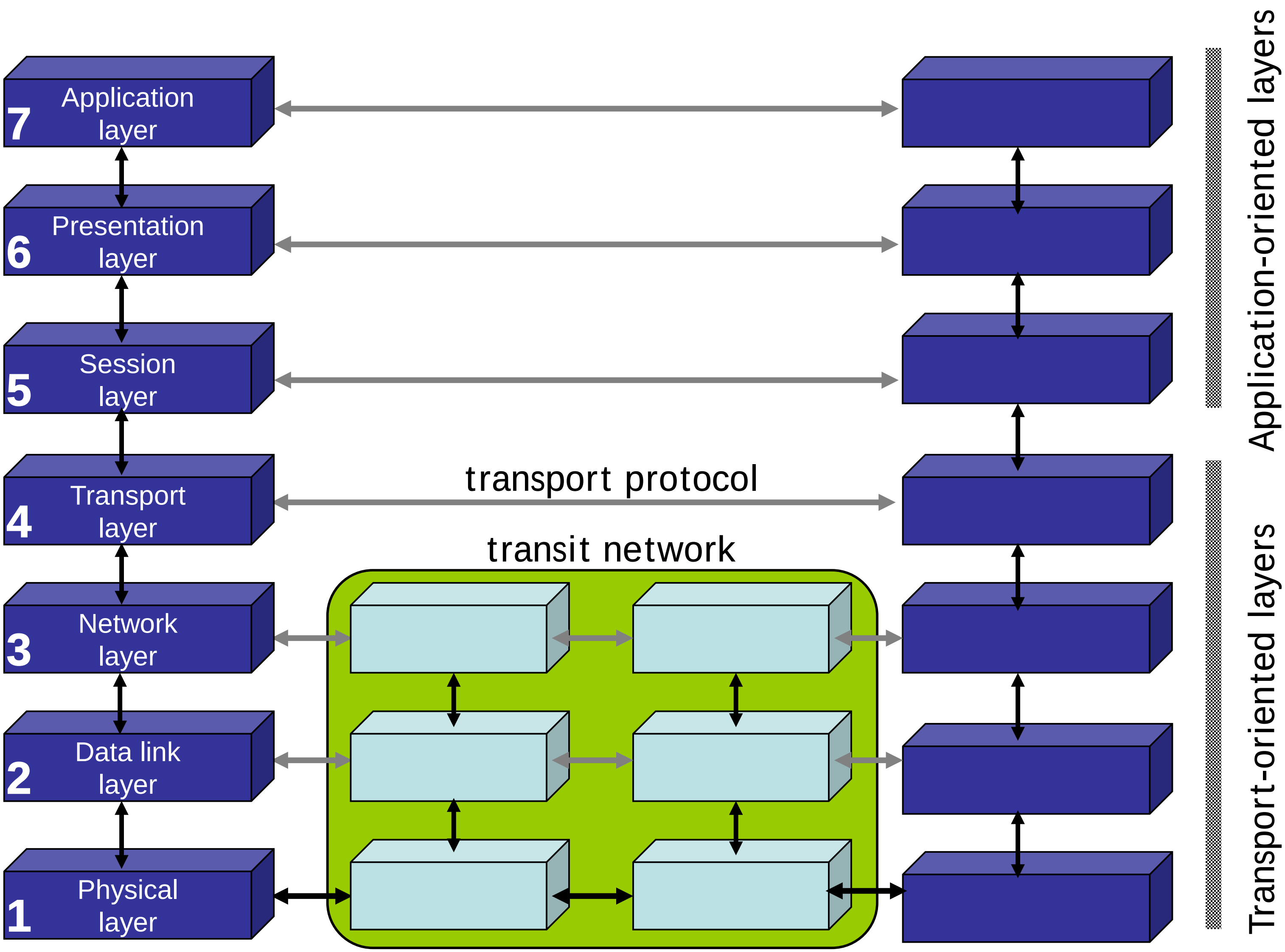
OSI Model: Data Units

- **Application level “messages” are processed as data units.**

- **Following notions for data units have become common:**
  - packet: “unit of transportation” (may contain fragments)
  - datagram: instead of packet if sent individually (connectionless)
  - frame: with final envelope, ready to send (next to lowest layer)
  - cell: small packet of fixed size

- **OSI terminology: „message“ is a PDU**
  - PDU: Protocol Data Unit
    - (N)-PDU: semantics understood by peer entities of (N)-service
    - (N)-PDU = (N)-PCI plus (N)-SDU; (N)-SDU = (N+1)-PCI plus (N+1)-SDU
  - PCI: Protocol Control Information: only used by peers
  - SDU: Service Data Unit = payload - optionally carried in PDU for user

OSI Part II: 7-Layer Architecture



OSI 7-Layer Architecture 1. and 2. Layer

1. Physical Layer PH

Non-secure bit stream between adjacent systems

- Signal representation of bits, (de-)activation of lines
- Standards for plugs, cables
- Transmission time of a bit
- Protocol example: RS232-C = ITU-T V.24; other: ITU-T X.21

2. Data Link Layer D

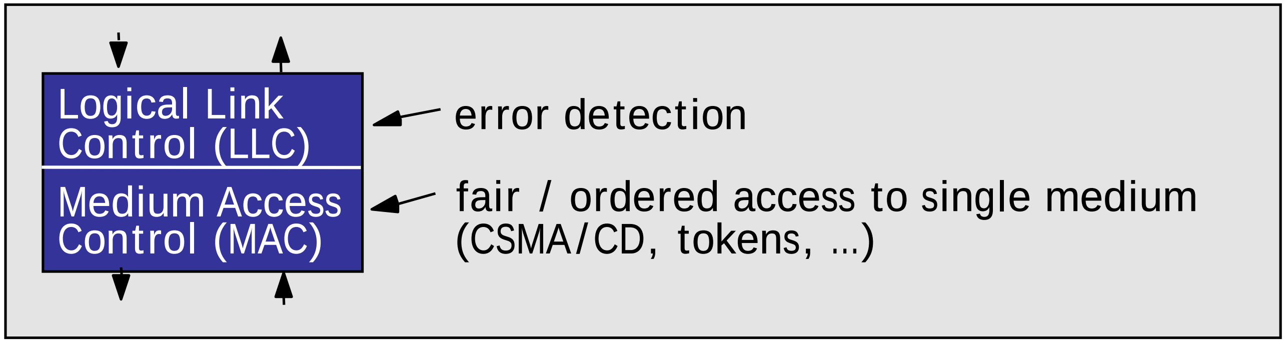
Error-recovering stream of frames between adjacent systems

- Packets as frames: boundaries detected
- Recognizes, recovers transmission errors
  - (sequence no's; checksums
    → acknowledgment or timeout + retransmission)
  - Spectrum:
    - » From detection only
    - » to forward error correction
- Residual & "severe" errors deferred to higher layers

OSI 7-Layer Architecture 2. Layer

- **Layer 2 may already include some flow control**
  - Goal: protect slow receiver
  - Flow control can be sophisticated (sliding window protocol),
    - For example, avoid slow stop-and-go for satellite connections

- **Broadcast networks (LAN) often with two sublayers**
  - Logical Link Control (LLC)
  - Medium Access Control (MAC)

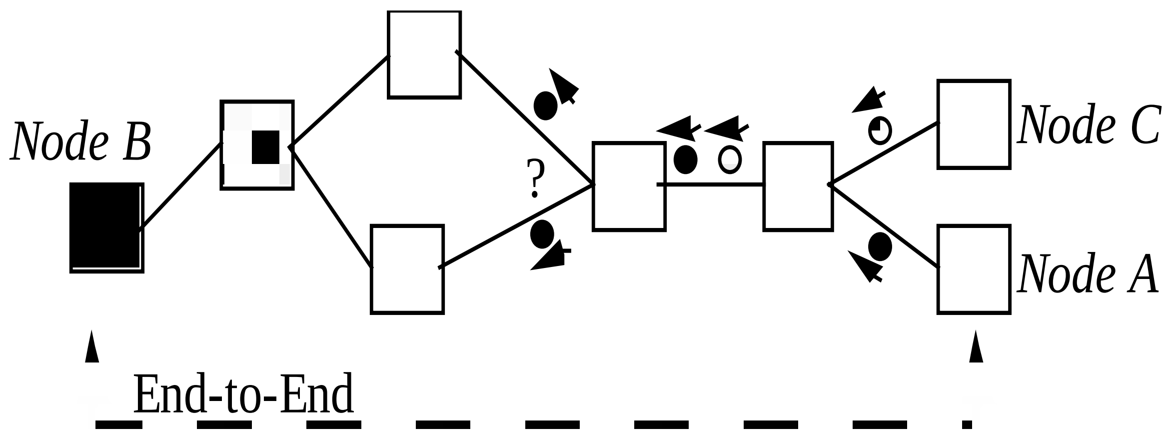


Based on lecture slides of Prof. Dr. Friedemann Mattern:
http://www.vs.inf.ethz.ch/edu/WS9900/VS/VernetSys3.pdf

OSI 7-Layer Architecture 3. Layer

3. Network Layer N

- Packet stream between end systems
  - Routing: find paths, forward packets
  - Upward & downward multiplexing of connections possible
  - Error correction & flow control between end systems, in particular, if hops on path have different "quality"
  - Congestion control (protect network)

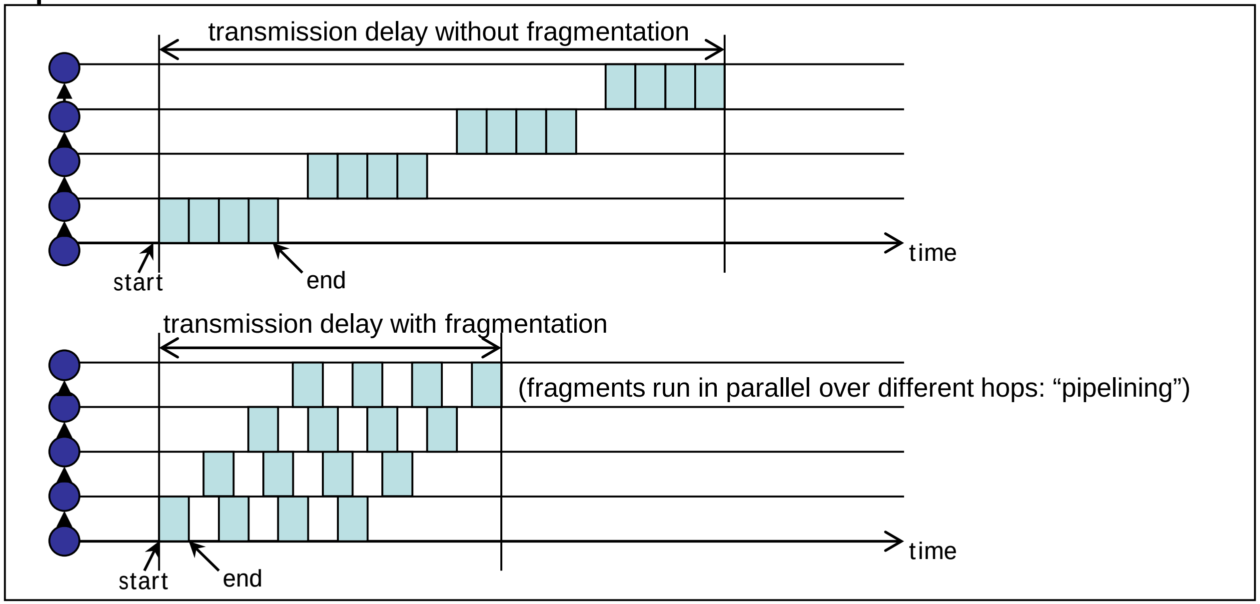


  - **Store&forward switching** (computer networks) vs. **circuit switching**
- Protocol examples: IP (connectionless), X.25 (connection-oriented)

Based on lecture slides of Prof. Dr. Friedemann Mattern: http://www.vs.inf.ethz.ch/edu/WS9900/VS/VernetSys3.pdf

OSI 7-Layer Architecture 4. Layer

- **4. Transport Layer T**
  - Logical connections between individual processes (ports, sockets)
    - Not only connections between end systems (computers)
  - Multiplexing
  - Packet assembly / disassembly for N-layer
  - Address handling, error correction, flow control
  - Hide network details, quality differences
  - Protocol example: TCP



OSI 7-Layer Architecture 5. and 6. Layer

- **5. Session Layer S**
  - Helps users to span and structure consecutive connections
  - Token: right to request set of service functions (diff. kinds of)
  - Communicate about checkpoints (used f. recovery / rollback?)
  - Hardly used (SNA, PTT telematic services such as Teletext)

- **6. Presentation Layer P**
  - Harmonize data representations in different end systems
  - OSI proposed & standardized
    - ASN.1 ("Abstract Syntax Notation"): syntax of user data types
      - Compare: C structs, Pascal records
    - BER ("Basis Encoding Rules"): common representation of data between all end systems (everybody converts from / to BER)
  - May also carry encryption
  - Internet: carry out in Application layer, use different, more pragmatic standards (cf. IDLs in remote procedure calls, Google's *protobuf, …*)

OSI 7-Layer Architecture 7. Layer

- **7. Application Layer A: „service elements" (SE) in OSI**
  - SEs for "common" tasks, e.g.:
    - „association control": take care of connections
    - CCR (Commitment, Concurrency and Recovery): recovery for transactions
    - ROSE (Remote Operation Service Element): reliable operations
    - And many others... - died out, almost forgotten
  - SEs for application domains.:
    - File transfer (e.g., ftp - "file transfer protocol")
    - Electronic mail (e.g., X400 **M**essage **H**andling **S**ystem)
    - And many others...
  - Huge OSI standards, pragmatic Internet standards

OSI 7-Layer Architecture Summary

1. Physical Layer PH: insecure bit stream between adjacent systems
2. Data Link Layer D: error-recovering frame stream, adjacent sys.
   - LANs: realized as
     - L.2b: Logical Link Control;
     - L.2a: Media Access Control
3. Network Layer N: packet stream between end systems
4. Transport Layer T: end2end msg. stream betw. individual processes
5. Session Layer S: structured dialogue
6. Presentation Layer P: exchange of data (semantics!)
7. Application Layer A: cooperating entities

Note:

- **Many service functions carried out in several layers / services !**
- **→ Overhead, even reversal in part due to net homogeneity**
- **→ Internet: Layers 5 and 6 integrated with layer 7**

Types of Networks

- **Recall: Two types of network**
  - Connection-oriented (CO)
  - Connectionless (CL)
- **Recall: Three phases in connection-oriented:**
  - Connection setup
  - Data transfer
  - Connection teardown
- **Connectionless has just data transfer phase**

- **Fundamental distinction between connection-oriented and connectionless**

- **Rough analogies:**
- **Connection-oriented ≈ telephone service**
- **Connectionless ≈ postal service**

Source: www.porticus.org/bell/oldphotos_10.html

…how many calls can she handle?

…what kind of guarantee do you get?

Connection-Oriented Networks

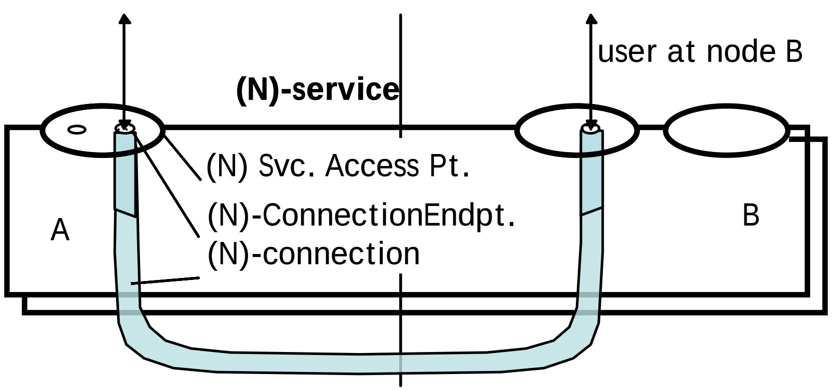
- **First phase of communication in connection-oriented network is connection establishment**
  - Also called handshake, similar to human interaction
  - Prepares for actual communication

- **After handshake, communication partners are connected**
  - “Connected” in a very loose sense
  - Only communication partners know they are connected
    - Intermediate systems might not know anything
    - In Internet, intermediate routers are unaware of “connection”

Connection-Oriented Networks (2)

- **Note: "Connection-oriented" does not imply any additional properties of the connection**
  - No reliability, flow control, or congestion control is required (nor guaranteed!)
  - TCP (Internet's connection-oriented protocol) implements them

- **BUT: "Connections" are nothing but a distributed "state", held at both end points**
  - this is the *basis* for almost all additional properties
  - … and for overhead (IP golden role: *no* state in routers)



Connectionless Networks

- **Connectionless networks have no connection establishment phase**
  – Go straight to data transfer phase
  – Also, no connection teardown phase

- **No need to maintain connection state**

- **Communication partner might not be ready for receiving**

- **Connectionless networks do not implement:**
  – Reliability, flow control, congestion control
  – Applies also to UDP (Internet's connectionless protocol)

So, Which Is Better?

- **Overhead of handshake in connection-oriented**
  - Can be significant in short communications
  - Insignificant in long communications

- **What happens when network is congested:**
  - Connection-oriented: Busy, no connection
  - Connectionless: Can communicate, but may be stalled

- ***Possible to build connection-oriented service on top of a connectionless service***

- **Pro CO: Better service (at cost of state)**
- **Pro CL: Stateless → Scalable**
- **Applications for both types of networks**

Example: Layers in Action

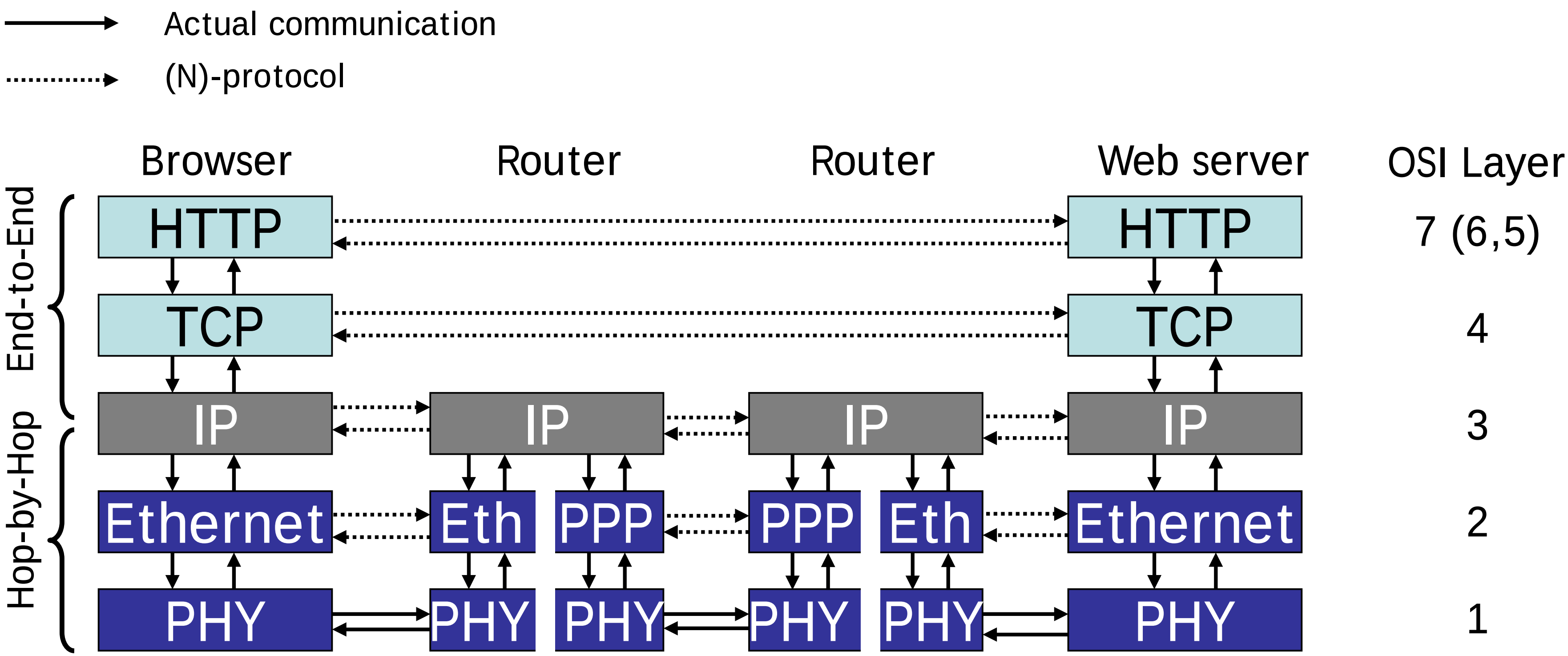
- **What happens in different layers when you use your browser to access a website?**

- **Remember**: **Internet has only 5 layers**
  - Layers 5, 6, and 7 implemented in a single application layer

- **In Internet, layers 3 and 4 are somewhat confused**
  - Transport protocol TCP (or UDP) and network protocol IP
  - Sometimes hard to draw a clear line where TCP ends and IP begins
  - **But:** Basic functionality is clearly separated

- **So, what happens?**

Layers in Action



- **Request goes down on layers at browser**
- **Physical layer handles actual sending of message to next (neighbor) node**
- **Network protocol (IP) takes care of routing message to destination**
  - Possibly several hops from one router to another
  - At each router, message goes up to IP-layer for processing
- **Transport and application layers converse end-to-end**

Functionality Recap

- **Layer 5,6,7**
  - Create HTTP request
  - Invoke layer 4 (= TCP)
  - Process reply (= web page)
- **Layer 4**
  - Open reliable connection to web server
  - Make sure data arrives in the order it was sent
  - Do not saturate network
    - Congestion control
- **Layer 3**
  - Route message from client to web server
  - Message passed from router to router
  - Layer 3 provides end-to-end service through hop-by-hop actions
- **Layer 2**
  - Put data from layer 3 in frames
  - Send frames to immediate neighbor
- **Layer 1**
  - Actual transmission of a frame as a bitstream
- **Each layer performs some critical function**
- **Layering not always "clean"**
  - Who handles congestion control or reliability?

Chapter Summary

- **Introduction to course**
- **Quick tour through networking**
- **Basic terminology and concepts**
  - Protocol
  - Service
  - Layer
- **OSI and Internet layer models**
- **Connection-oriented and connectionless networks**
- **Example of layers in practice**