Telematics I

Chapter 1
A Quick Tour

- Goals of this chapter
- Examples
- Direct connection between two devices
- Multiple devices
- Errors

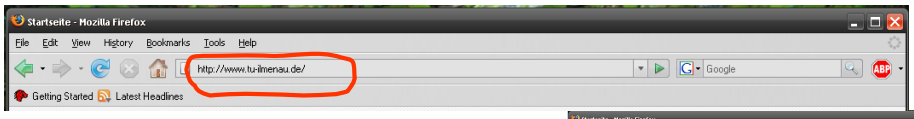
(Acknowledgement: these slides have been edited from Prof. Karl's set of slides)

Goals of This Chapter

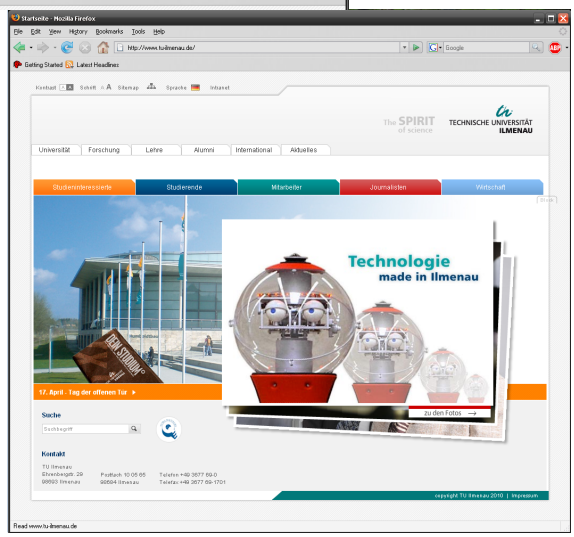
- Provide a brief, guided tour to communication networks, in particular, Internet style networks
- It starts from the simple case of two directly connected devices and generalizes to larger networks
- Goal is to provide a rough understanding on
  - *Which* typical problems exist,
  - *Why* they occur,
  - *How* a solution could look like.
- Details on solutions will be treated in the remainder of the course

Basic Example 1: World Wide Web

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

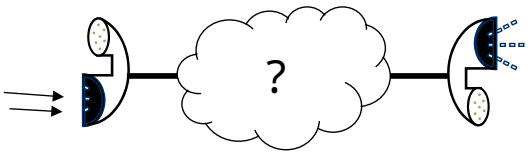


??



Basic Example 2: Telephony

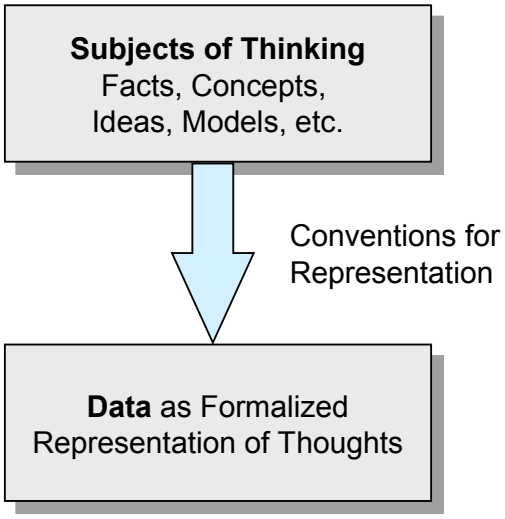
- What happens when picking up a telephone and making a phone call?
  - How to find the peer's phone? How to transmit speech?



- 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"?

Some Terminology: „Data“

- Data (universal)
  - Representation of facts, concepts, ideas and instructions in a formalized manner, that is suitable for communication, interpretation and processing through human beings and/or technical devices
  - General examples for data representations:
    - Spoken language
    - Written language
    - Signs



Model for Generation of Data by Human Beings

Some Terminology: „Information“

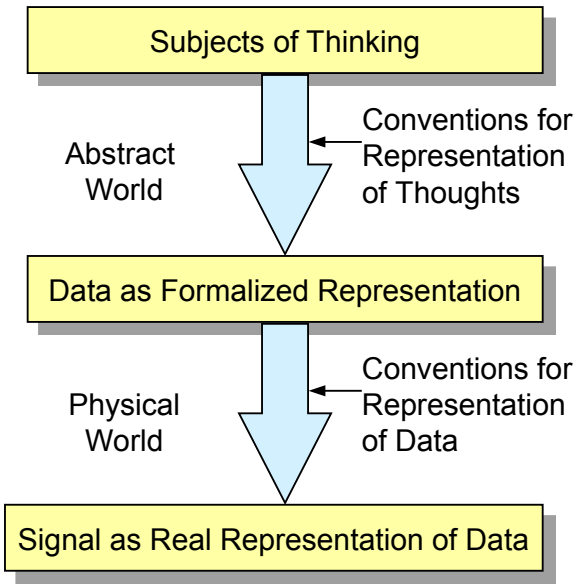
- Information:
  - The meaning, that a human being associates with data according to the conventions that were used when the data was generated
  - Attention:
    The notion of information is thus only related to human beings!
  - This is a stricter definition of the term information than is usually assumed in everyday language
  - Humans and machines can manipulate data, but only humans are able to gain information from data
  - Therefore, the term information should be avoided when talking about data communication, telecommunication, etc.

Some Terminology: „(Data) Communication“

- Communication originally means
  - Exchange of data between human communication partners
- According to the universal meaning of the term „data“ as introduced before, this means:
  - Every concrete communication is a data communication
- In the literature and daily language, usually the following, more narrow definition is used:
  - Transmission of digital data between telecommunication devices
- In this class we use the term in the following sense:
  - Data (tele-) communication is the generic term for all data exchange over an immaterial carrier and a significant distance between human beings and/or machines
  - Immaterial carrier:
    - Flow of energy (e.g. electrical, optical, electro-magnetical wave)
    - Opposite: Data transport over material carrier (e.g. letter, magnetic tape, floppy disk, CD, DVD)

Some Terminology: „Signal“

- Signal:
  - A signal is the physical representation of data in the form of a characteristic variation in space and/or time of one or more physical quantities
  - Signals are thus the real physical representation of abstract data

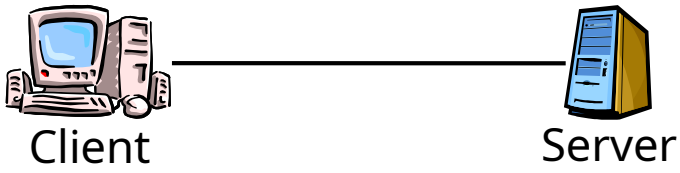


Overview

- Examples
- ***Direct connection between two devices***
  - Signals
  - Low-level communication properties
  - Duplex
- Multiple devices
- Errors

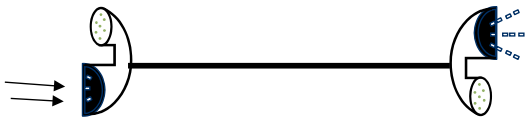
Simplest Communication: Direct Physical Connection

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



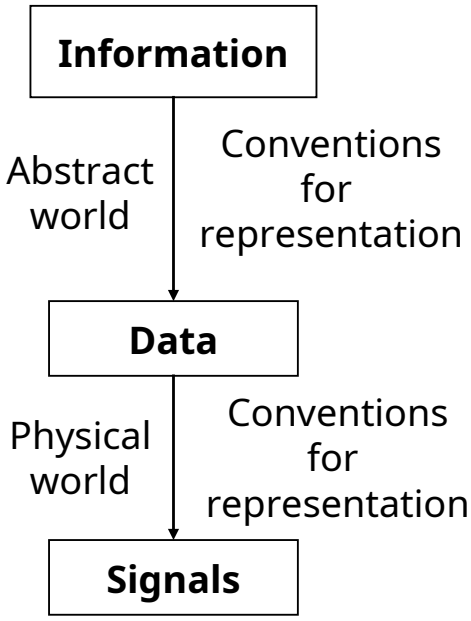
  - Server provides data, client consumes it

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



What Good is a Physical Connection? – Signals

- 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*
- *Immaterial signals in physical media enable data communication between remote senders and receivers*



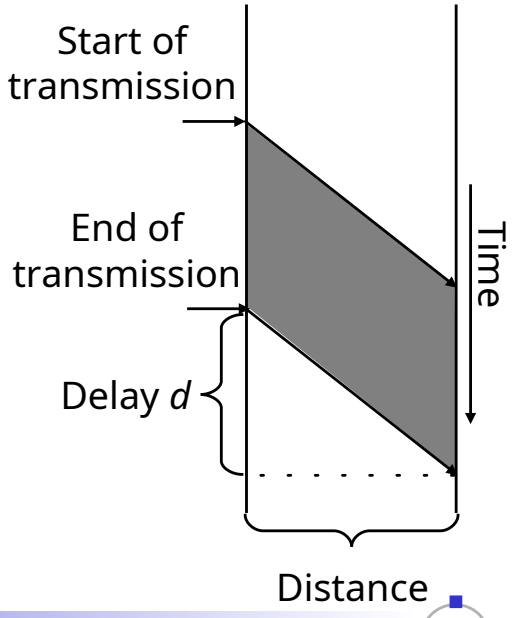
Bits and Signals

- 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 simple convention for a copper wire:
  - A “1” is represented by current
  - A “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

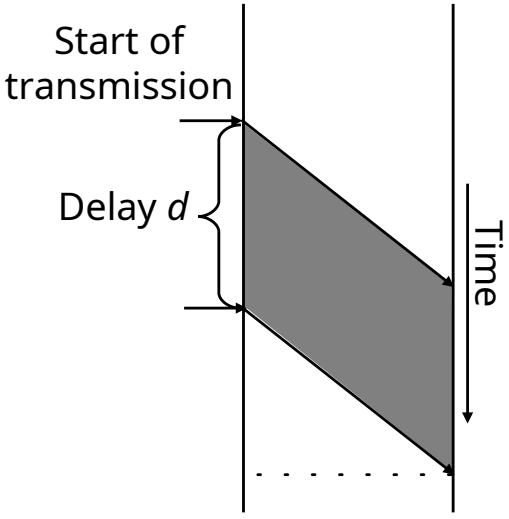
- Transmitting signals on a physical carrier results in some low-level properties
  - Propagation delay *d*: How long does it take for a signal to reach receiver?
    - Propagation speed *v*: How fast does a signal travel in the medium?
    - Electromagnetic waves in vacuum speed of light *v*=*c*, in copper *v*= 2/3 c
    - *d* = distance / *v*
  - Data rate *r*: With which data rate (in bits/second) can a sender transmit?
    - (EOT – SOT) = Data size / data rate
  - Error rate: What is the rate of incorrect bits arriving at the receiver?
    - Also: what error *patterns*?

Message Sequence Charts (MSC)



How Can a Wire Store Data – Bandwidth-Delay Product

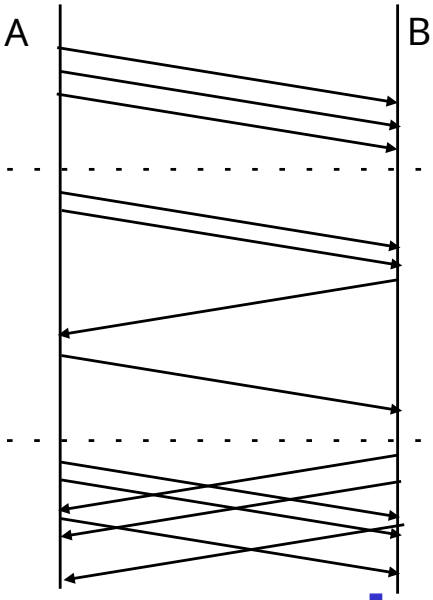
- What happens in the first *d* seconds after transmission starts?
  - Bits are transmitted and propagate towards the receiver
  - Sender keeps sending bits
  - First bit arrives after *d* seconds
  - In this time, sender has transmitted *d* * *r* bits
  - Where are they? Stored in the wire!
- *d* * *r* is the product of delay and data rate
  - Commonly called ***bandwidth/delay product***
  - Crucial property of high-performance networks



Example (transcontinental):
- Data rate 100 Mbit/s
- Delay 6000km/(2/3c) = 0.03 s
- *d* * *r* ≈ *3* Mbit

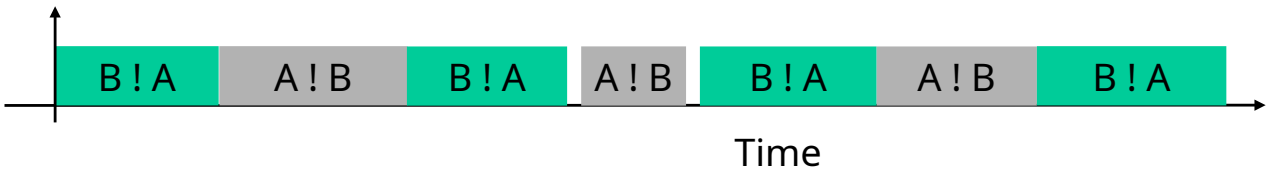
Two Physical Connections for Two-Way Communication?

- 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 at the same time
  - *Half duplex:* Parties alternatively send data
    - Example: Conversation
  - *Full duplex:* Both parties send all the time



How to Realize Duplexing

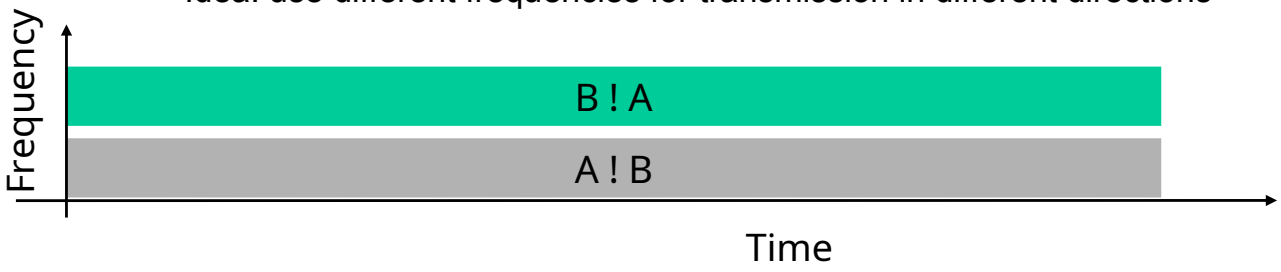
- Simplex operation: trivial
- Half duplex
  - Two pairs of cables, one for each direction – wasteful
  - Use one cable intelligently – participants alternatively transmit, wait their time until it is their turn
    - Both sending at the same time would not work, signals *interfere*
    - Problem: How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



- ***Time division duplex – TDD***

How to Realize Duplexing

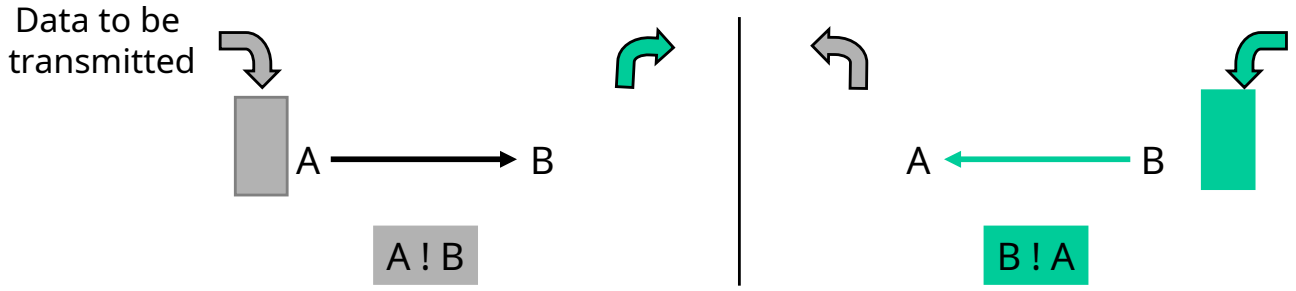
- Full duplex
  - Two pairs of cables would work, but still overhead (installation, maintenance, …) – does it work with one cable also?
  - Exploit some properties of the physical medium
    - Here: transmissions in different frequencies do not interfere
    - Idea: use different frequencies for transmission in different directions



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

How to Realize Duplexing

- Full duplex by time division duplexing?
  - 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 quickly send all stored & new data



  - TDD can realize full duplex if transmission over medium is at least twice as fast as data is to be transmitted

Lessons Learned from Duplexing

- It is useful to distinguish between
  - *Requirements* on what should be possible
  - *Rules and methods* how to implement such requirements
  - Example: Implement a “full duplex” requirement using TDD
- This distinction will become very important
  - Formalized later as *service* versus *protocol*

- Buffering is an important means to decouple different dynamics in time
  - Questions of buffer overflow have to be considered

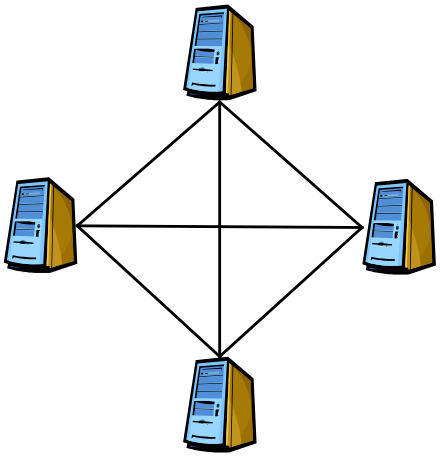
Overview

- Examples
- Direct connection between two devices
- ***Multiple devices***
  - Multi-hop connections
  - Switching
  - Forwarding
  - Multiplexing
  - Multiple access
  - Routing
- Errors

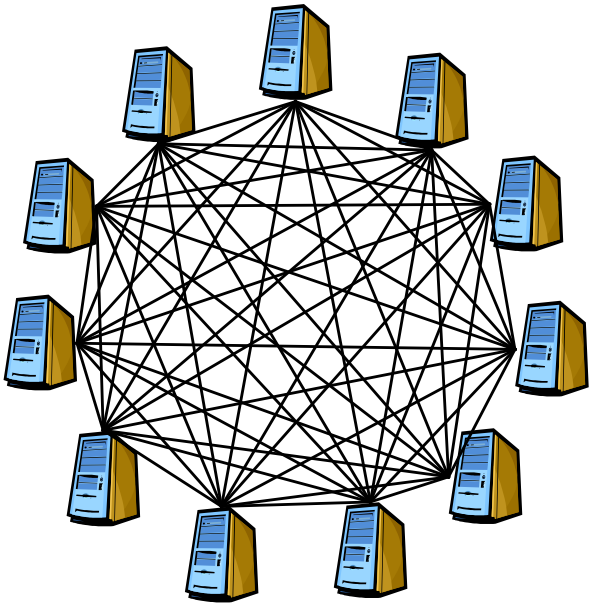
But There are More than Two Computers / Telephones

- Connect each telephone / computer with each other one?

With four computers:



With eleven computers:

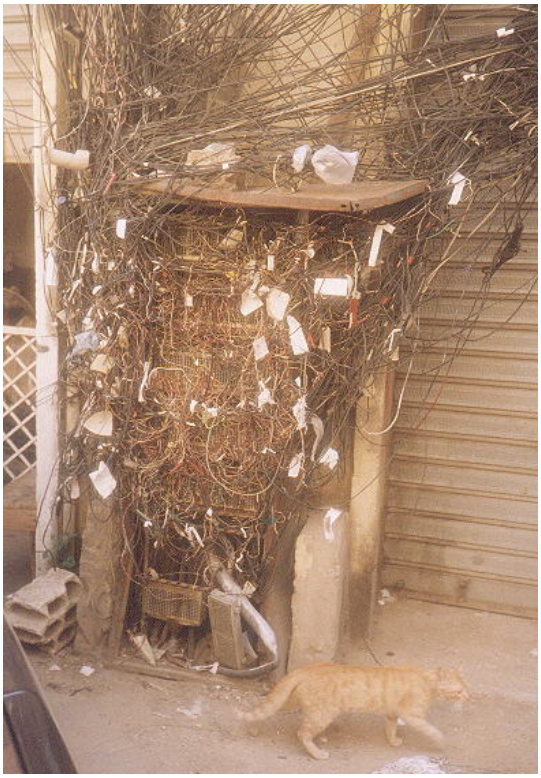


...

Beirut Connections

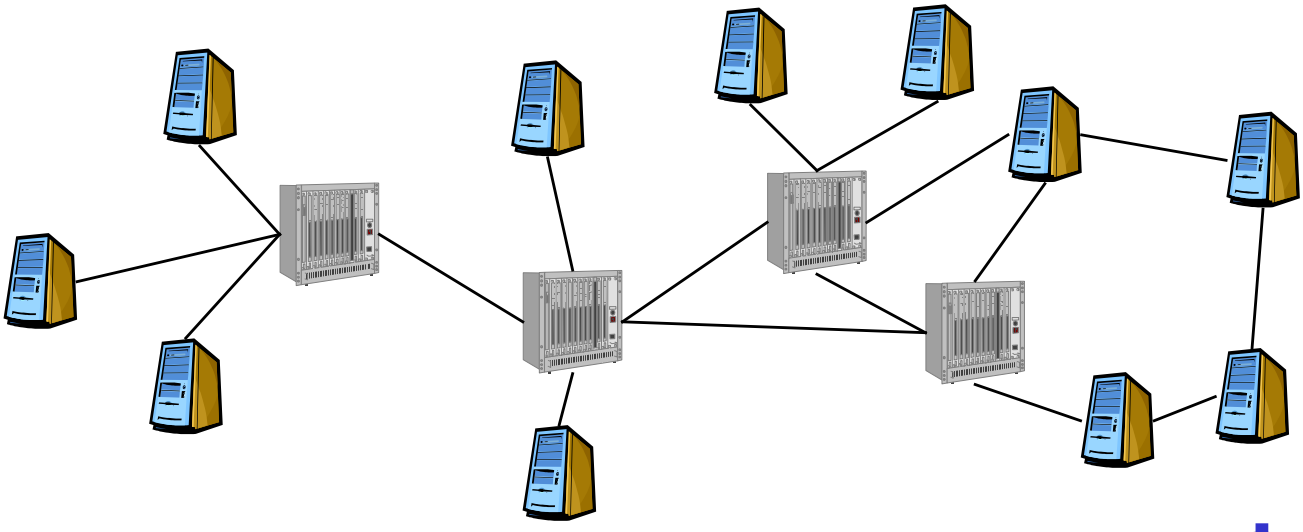
- Connecting many phones in real life

Beirut Connections



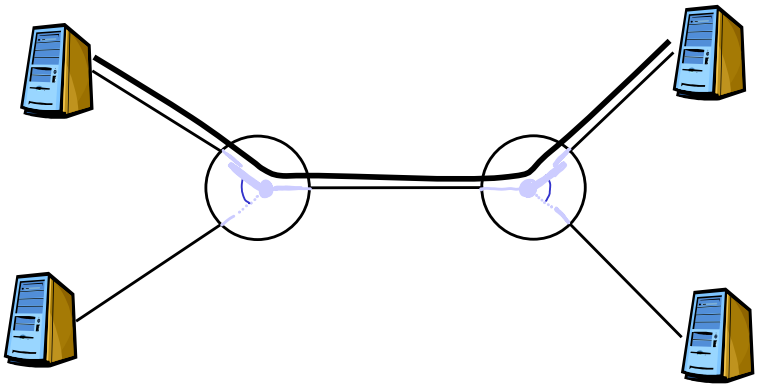
Put some Structure into a Network

- Pairwise connecting all entities does not work
- Need some structure
  - Distinguish between "*end systems/terminals/user devices*" on one hand, "*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 – compare “Fräulein vom Amt”
  - Resulting circuit lasts for duration of communication
  - ***Circuit switching***



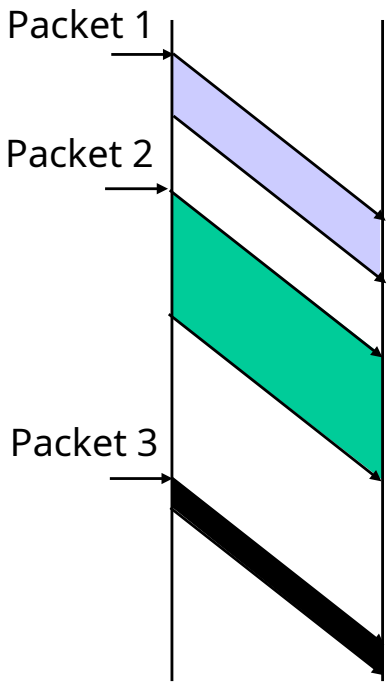
http://www.wdrcobg.com/switchboard.html

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
- Disadvantages
  - Resources are dedicated – what if there is a pause in the communication?
  - Circuit has to be set up before communication can commence
- Alternatives?

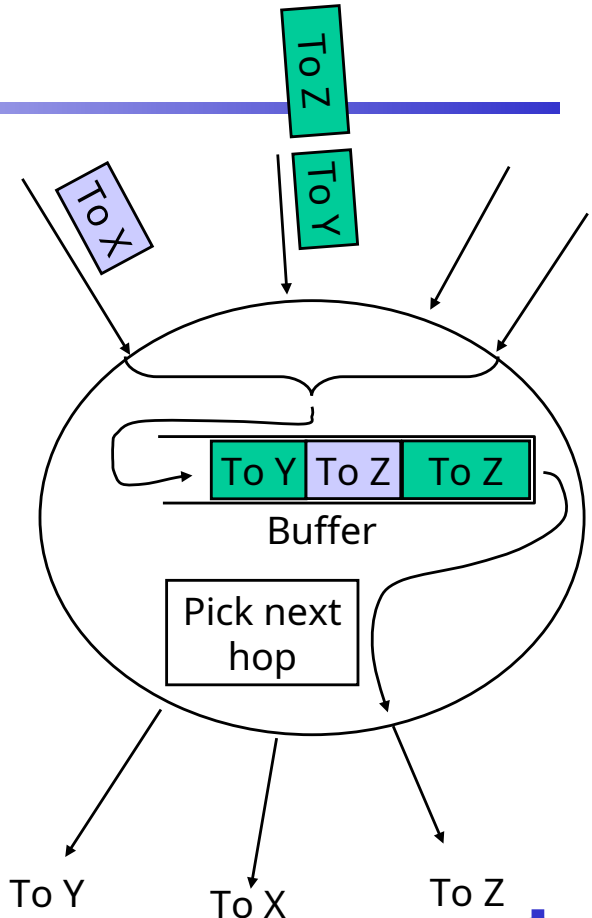
Packet Switching

- 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., who the recipient is
  - Sender then occasionally sends out a packet, instead of a continuous flow of data
- Problems: How to detect start and end of a packet, which information to put into a packet, …



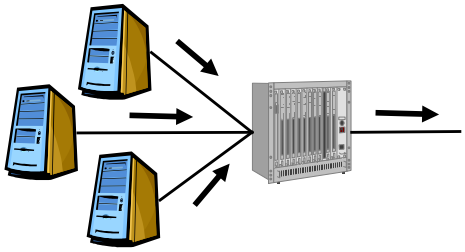
Packet Switching

- 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
    - Information about the network graph necessary
  - ***Forward*** the packet to this next hop of its journey
- Also called "***store-and-forward***" network



Multiplexing

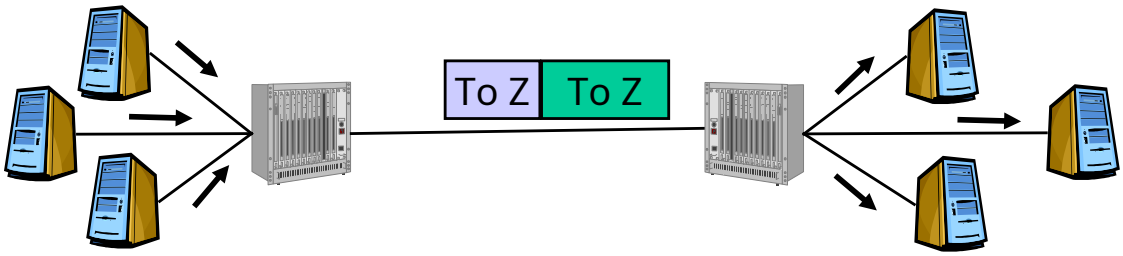
- Previous example had two packets at the head of the queue destined for terminal Z
- Similar situation: a switching element has only a single outgoing connection



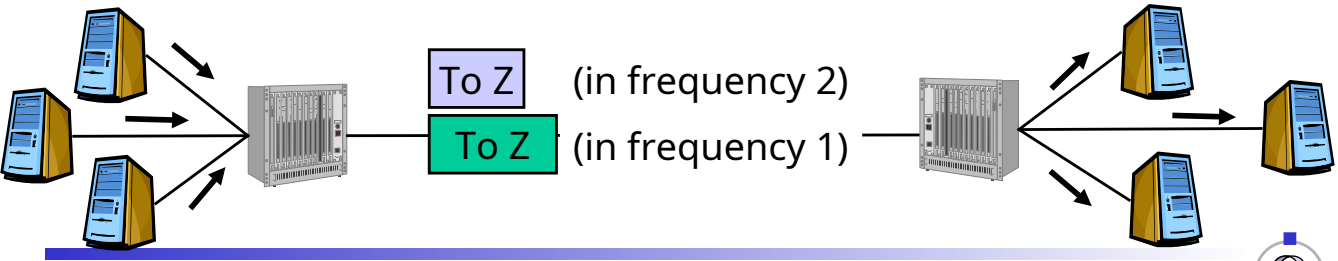
  - Such a special case is called a ***multiplexer***
  - Organizing the forwarding of packets over such a single, shared connection is called ***multiplexing***
  - Multiplexers in general need buffer space as well

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



Multiplexing

- Some other alternatives exist
  - Code Division Multiplexing (CDM), Space Division Multiplexing (SDM)
  - Mostly relevant in wireless communication

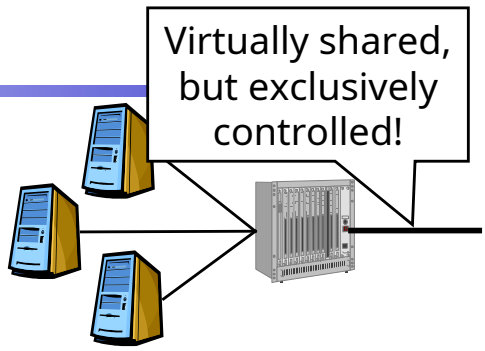
- Obvious parallels/relations to duplex operation!
  - Multiplexing describes how to operate *several pairs* of communicating entities
  - Duplexing describes how *a given pair* of communicating entities can exchange data
  - Question: How to combine different duplex and multiplexing schemes?

Mutliplexing & Virtualization

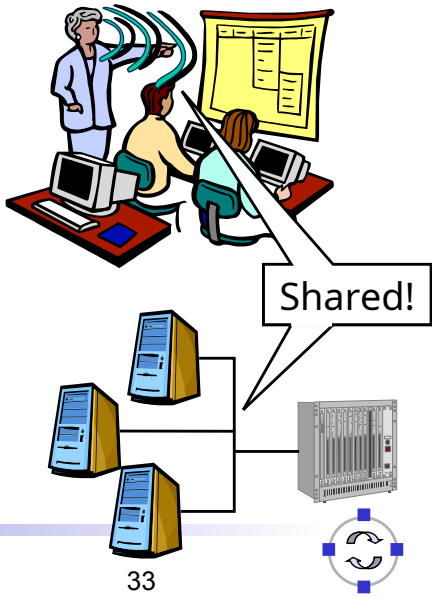
- In essence, multiplexing serves one particular purpose:
  - It *abstracts away* that a physical connection has to be shared with other contending entities, each sending a logical flow
  - It allows the entities connected to the switch to “imagine” that they alone are using the physical connection
    - At somewhat lesser properties
  - Multiplexing virtualizes the actual, physical connection
  - It needs a peer entity, a ***demultiplexer,*** to restore the original flows

- This concept of ***virtualizing*** and ***enriching*** the properties of a simpler subsystem is a pivotal technique for communication networks
  - As it is in software engineering, operating systems, etc.
  - It allows us to think in terms of increasingly more complex communication systems, build one atop the other

Multiplexing & Shared Resources



- Multiplexing can be viewed as a means to regulate the access to a resource that is shared by multiple users
  - The switching element/its outgoing line
  - With the switching element as the controller
- Are there other examples of “shared resources”?
  - Classroom, with “air” as physical medium
  - A shared copper wire, as opposed to direct connection
- Characteristic: a ***broadcast*** medium!



Broadcast Medium and Multiple Access

- Common characteristic of a broadcast medium:
  Only a single sender at a time!
  - ***Exclusive access*** is necessary
  - There are some exceptions using CDMA, but that’s advanced material
- Exclusive access is simple to achieve with a multiplexer
  - What if no multiplexer is available?
  - Exclusive access has to be ensured by all participants working together
- The problem of ***multiple access to a shared medium***
  - ***Medium access*** for short
- Rules have to be agreed upon
  - Classroom approach: only speak when asked to, central instance

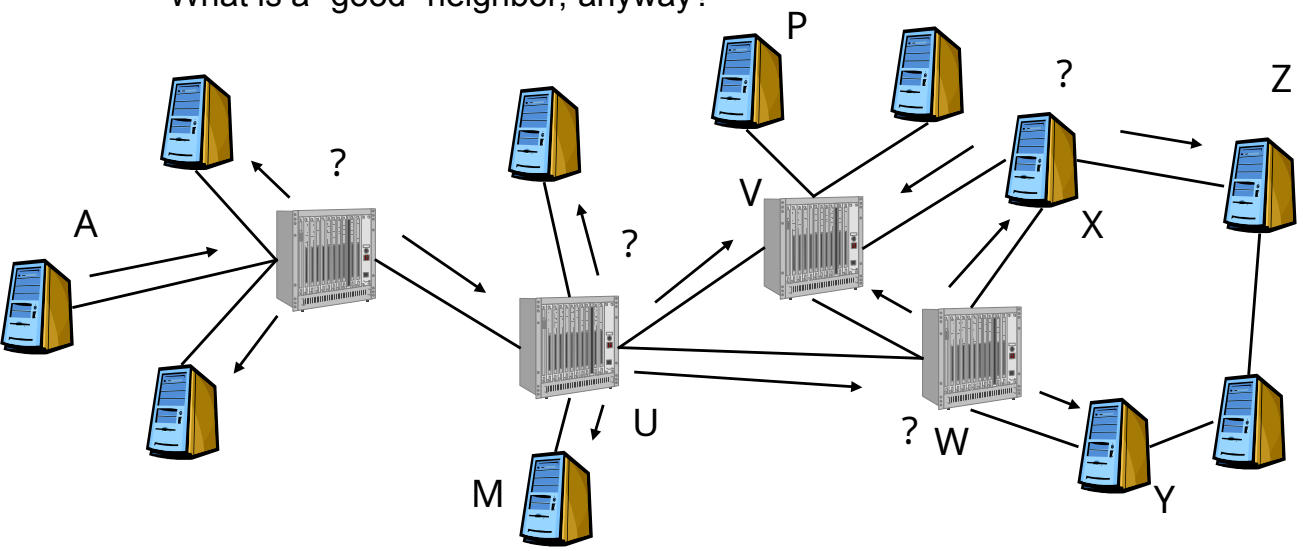
An Intermediate Summary

- So far, we have a rough idea about
  - Converting bits to signals and back again
  - Duplexing, switching, multiplexing
  - Packets as unit of data transport
  - Multiple access of several entities to a shared medium
- In essence, 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* which of its neighbors is the best possible one towards a given destination?
  - What is a “good” neighbor, anyway?



Options for Next Hop Selection

- Some simple options:
  - ***Flooding*** – send to *all* neighbors
  - ***Hot potato routing*** – send to a *randomly chosen* neighbor
- Simple options not convincing
  - Try to find good, i.e., short routes – few hops
  - 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

Routing table of W

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

Routing Tables

- Criteria
  - 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 imperara) → Internetworking

Overview

- Examples
- Direct connection between two devices
- Multiple devices
- ***Errors***

Handling Errors, Overload, …

- So we are done building a network – unless something goes wrong!
- Source of errors/abnormal situations
  - Conversion from signals to bits can fail
  - Access to a shared medium might not work
  - Packets can be lost, e.g., because buffers overflow
  - Packets can be misrouted (because of incorrect routing tables), delayed, reordered
  - Receiver might not be able to keep up with incoming stream of packets
  - Routers can fail, resulting in incorrect routing tables
  - … and many more

Handling Errors, Overload, …

- ***Error control*** at various abstraction levels needed
  - Between two direct neighbors, over a given connection
  - Between end systems, to compensate for errors not detected locally – e.g., incorrect order of packets
- ***Overload control***
  - Protect the network against buffer overflows, regulate the number of packets injected into the network – ***Congestion control***
  - Protect end system against too many packets coming in – ***Flow control***
- ***Where*** and ***how*** to implement error and overload control is a principal architectural decision
  - Main options: in the end system or in the network
  - Big difference between telephony system and Internet
  - Telephony carriers are (traditionally) interested in network-based solutions to be able to charge for it

Intermediate Summary: Basic Required Functions

- Bit-to-signal and signal-to-bit conversion
- Grouping bits into packets
- Accessing a shared medium
- Switching, duplexing, multiplexing
- Controlling errors on a connection between two systems
- Forwarding incoming packets, consulting routing tables
- Constructing routing tables, maintaining them
- Controlling errors not detectable between two neighboring systems
- Protecting the network against overload
- Protecting end systems against overload
- Ensuring correct order and possibly timeliness of packets
- Making these functions accessible from application programs
- Controlling the actual hardware that connects a wire to a computer
- … *and more!*

Conclusions

- Communication networks have to solve many problems and need a lot of functionality
- The most basic of these problems, and an idea about their solution, should have become clear

- How to group these functions, how to solve these problems, will be the topic of the remainder of this course